CentraleSupélec

On The Universality of Visual and Multimodal Representations

Thèse de doctorat de l'Université Paris-Saclay
préparée à CentraleSupélec

École doctorale n°573 INTERFACES
Spécialité de doctorat : Ingénierie des systèmes complexes

Thèse présentée et soutenue à Palaiseau, le 1er Juin 2018, par

YOUSSEF TAMAAZOUSTI

Composition du Jury :

Pr. Mathieu Cord Professeur des Universités, Université Pierre et Marie Curie (UMPC)	Président
Dr. Florent Perronnin Directeur de Recherche, Naver Labs Europe	Rapporteur
Pr. Philippe-Henri Gosselin Professeur des Universités, École nationale supérieure de l'électronique et de ses applications (ENSEA)	Rapporteur
Pr. Iasonas Kokkinos Professeur des Universités, University College London & Facebook	Examinateur
Dr. Pablo Piantanida Associate Professor, CentraleSupélec (L2S)	Examinateur
Pr. Céline Hudelot Professeur des Universités, CentraleSupélec (MICS)	Directeur de thèse
Dr. Hervé Le Borgne Ingénieur de recherche, Commissariat à l'énergie atomique et aux énergies alternatives (CEA)	Co-directeur de thèse

Acknowledgements

I would like to begin this thesis by warmly thanking all the people who have made the completion of my thesis work possible, directly or indirectly. It was a wonderful adventure for me, both humanly and scientifically. I would therefore first of all like to thank all the people who welcomed me to their unit, notably François Gaspard, Patrick Sayd and Bertrand Delezoide. This three-year adventure ended as usual with the writing of a manuscript and a defense. For this last I had the chance to have an exceptional jury for which I would like to thank solemnly and with great respect and consideration the members who gave their precious time to the evaluation of my work. I thank, in particular, Florent Perronnin and Philippe-Henri Gosselin for having agreed to review my work and for the scientific exchanges we had. I also thank Mathieu Cord for chairing my defense as well as Iasonas Kokkinos and Pablo Piantanida for examining my work. The members of my jury also made this day particularly pleasant for me.

From my point of view, during the execution of a thesis, the supervision team undeniably influences the progress of the thesis as well as the way in which this experience is lived by the doctoral student. For my part, I was fortunate to be supervised by people who were able to motivate me on a daily basis, and who, with their respective experiences and knowledge in the field, played an important role in the progress of my thesis. I would therefore like to thank my LVIC supervisor, Hervé Le Bogne, who has been present for me on a daily basis and notably because I learned a lot from him, both scientifically and humanly. My gratitude also goes, of course, to Céline Hudelot, my thesis director, who made herself very available, both for the numerous progress meetings and for reviewing the thesis, despite a very busy schedule. Far beyond the professional aspect, it is the human qualities of Hervé and Céline that I would like to highlight by this sentence. My thanks also go to my former colleagues, in particular the trainees, doctoral students, postdocs, engineers, researchers and secretaries of the LVIC laboratory, for their professionalism, their conviviality and their communicative good humor and for having thus formed an efficient and pleasant working environment. I would particularly like to thank the members of the Multimedia team with whom I have had many enriching scientific and technical exchanges. I would especially like to thank Adrian Popescu for his technical and scientific help throughout my thesis.

I would now like to express more personal but nonetheless essential thanks. Indeed, the thesis is a professional adventure which evolves in parallel with the doctoral student's personal life and which, therefore, generally has a significant impact on the latter. For me, the support I had throughout this adventure from my family has been indispensable. More generally, I thank my parents, Chaïb and Malika, for having accepted all my choices, for having always supported me, but also for having placed at my disposal all the emotional and material resources necessary to allow me to advance in life. I would also like to thank my brothers and sisters for their support during my studies and their encouragement as well as for having learned me the sense of human values and encouraged me to always give the best of myself whatever I do in life. Last but not least, my most loving thanks go to Wiame, my wife, for the support and comfort she brings me, for the patience and understanding she always shows me and even more, for the happiness I lived with her since our marriage.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context

Artificial Intelligence is a hot topic that is on everyone's lips, in the news, in industry and even in politics, because of the key societal, economic and cultural challenges it implies [75, 234]. We even talk about an *AI race* that takes place on a national scale – *e.g.*, the member of parliament Cedric Villani was commissioned by the government to define an AI strategy for France –, continental – *e.g.*, the European Commission is funding several projects on the AI topic –, or even international. It should also be noted that because of the high stakes, AI research has been massively invested by the industry, in particular the well-known GAFAM (*i.e.*, Google, Amazon, Facebook, Apple, Microsoft). Industrial strategies are sometimes established independently of governments, which makes the geopolitical context of AI very complex. Consequently, there is a parallel desire to clarify the situation and anticipate the possible consequences of AI on society. This is conducted through multiple debates and initiatives around the ethical and legislative aspects of AI. The only point of agreement is that this extremely promising field, that started half a century ago, is still in its infancy.

In general, the application interest of AI is to develop systems to facilitate the life of humans, for example by relieving human beings of the most tedious tasks, such that they can focus on more rewarding occupations. In this vein, major advances are expected with regard to autonomous vehicle [32], industrial robots [28], household robots [29], decision support in the medical field [146], augmented reality [238] (for industry [239], marketing [78] and localization [79]), image-based search engine [81], and much more.

From a scientific perspective, there is no unanimous definition of AI, at this stage [201] and different approaches coexist corresponding to different founding schools of thought (act rationally, like a human, think rationally, like a human) [218]. Among the sub-fields of AI, Machine Perception has been impacted dramatically, in particular computer vision and speech processing, although one notes recent significant gains in performances in Natural Language Processing as well, in particular for translation. These progresses are strongly linked to those of Machine Learning these last thirty years, and in particular to the "third historical wave of artificial neural networks" [88], better known as *deep-learning*, that emerged in the mid-2000's. In conjunction with technological progresses that allowed to gather larger amounts of data and offered an availability of computing power and storage to process that data, the approaches turned progressively into solutions that, by design, derive from data. From the nineties, these machine learning-based techniques mainly concern the decision level, while the features used to represent the data were designed by hand. In computer vision, such notable low-level features included wavelets [158], SIFT [151] and HOG [50] while famous higher level features built on these were for example GIST [185], BOV [46] and Fisher Vectors [196]. At the mid-2000's, a resurgence of artificial networks known for a while (Multi-layer perceptron, CNN [140], LSTM [102]) allowed to learn the representation level of the tasks to solve, leading to end-to-end systems with significantly improved performances [20]. It resulted not only from the technological progresses cited previously as well as important contributions of some researchers to make datasets available to the community (ImageNet [53, 216], Places [297]), but also from theoretical advances [100] that allowed to learn significantly deeper networks. This has enabled a significant breakthrough in the quality of systems performance on many traditional tasks. These new performances have made the industrial use of AI possible and are therefore the main motivation for all this enthusiasm around AI.

1.2 Problematic & Goals

A long standing goal of AI is to design machines, capable of *perceiving* the world, of *interacting* with humans, and all this in an *evolutionary* way. In other words, we are trying to design human counterparts. For example, we can even mention the AI named "Sophia" which was naturalized citizen of the state of Saudi-Arabia. For this, we will sometimes draw inspirations from the human learning system [8, 214, 118] on the three aspects mentioned above (perception, interaction and evolution). Moreover, the learning technique we are interested in, namely deep neural networks, is said bio-inspired [249, 258]. The work of this Thesis is based on the personal intuition that in humans, we can explicit two different learning objectives:

- learning for the purpose of expertise, to perceive a specific environment with sometimes very technical interactions (for example, a surgeon will have to acquire very technical knowledge);
- learning for a generic purpose, that is to say, for everyday use in our perception of the environment and our interactions (for example, learning to count objects).

We propose in this thesis to categorize the work around AI in a similar way. Indeed, the two learning objectives in humans can be translated into the machine world by these two learning approaches:

- learn representations from few specific tasks. The aim is to then be able to carry out very specific tasks (specialized in a certain field) with a very good level of performance. We are talking here about *specialization*.
- learn representations from several general tasks. The aim is to perform as many tasks as possible in different contexts. We are talking here about *universality*.

Note that, the common difficulty to both approaches, is to switch from low-level data (pixels in the case of images) to the realization of high-level tasks, which are associated with human semantics (for example, recognize *cars* in images). This is well known as the *semantic-gap* [230], resulting from the need to match human semantics and low-level data, that should be considered when one design or learn an adequate representations.

While the first approach of learning (specialization) was extensively explored by the deep-learning community [20, 132, 228, 97, 2, 4, 63, 34, 98, 149, 271, 270, 224, 222, 289] (and much more), only a few implicit attempts [4, 23, 121, 210, 130, 42, 44, 43, 235] were made towards universality. Thus, in this Thesis, the goal is to *explicitly address the problem of universality* for deep neural network learning methods and for visual and textual data. More precisely, to tackle the problem of universality of a learning-based AI model (that consists in a representation extractor followed by a task-solving elements), two aspects needs to be addressed (independently or jointly), namely, *universal representation* and *universal task-solving*. In this Thesis, we are interested by the former aspect. For universal representations, it is important to state that, a priori, no representation is *purely* universal (*i.e.*, able to handle data from all the possible kinds), but each representation has a certain *level* of universality. Hence, our goal in this Thesis, is to start from a reference representation (with a high-level of universality) and try to learn a *more universal* representation (*i.e.*, with a higher level of universality).

1.3 Contributions

In this Thesis, we address this topic of universality in two different forms: through the implementation of methods to improve universality (called *universalizing methods* in the following) of a deep learning-based system; and through the design of a protocol to quantify its universality. Concerning **universalizing methods**, the general approach applied in this Thesis consists in forcing the system to learn a representation :

- containing individual detectors that are able to manage a maximum number of modalities, that is different types of data (*e.g.*, image, text, sound, etc.);
- containing individual detectors that are useful for a maximum of tasks (*e.g.*, classification, detection, segmentation, retrieval, etc.);
- containing detectors capable of managing a maximum number of visual domains (*e.g.*, photo-realistic images, cartoon images, commercial images, etc.);
- capable of handling as many semantic domains as possible, that is to say, more concepts (*e.g.*, dog, cat, horse in the domain of *animals*, car, bicycle, truck in the domain of *vehicles*, etc.).

More specifically, in this Thesis we proposed the following three technical contributions:

The **first contribution** (Chapter 3) is placed within the framework of learning methods for the *image* modality, the *classification* task and the visual domain of *natural* images. The goal is to increase the semantic domain of the representation by increasing its number of detectors. For this contribution we started from the semantic representation proposed by Ginsca *et al.* [81], which has (by construction) the advantage over other representations of literature to be directly interpretable from a semantic and evolutionary point-of-view. In particular, they proposed a representation composed of several tens of thousand concepts. The resulting representation nevertheless carries lots of redundancy between the detectors. They addressed this problem with a thresholding approach of a fixed number of concepts. We have proposed a method to select the detectors adaptively, according to the amount of information in the image. In a second version, we propose another method to select the detectors that relies on a semantic knowledge that links the concepts with respect to their semantic relationships.

The **second contribution** (Chapter 4) is placed in the same framework as the previous one and with the same objective, but addresses a different problem. Indeed, an increasing number of detectors usually requires a significant growth in the number of annotated data to learn them. We have therefore proposed an approach that increases the number of detectors without increasing the amount of annotated data. Our approach is composed of three modules: (i) automatically vary an initial set of annotated data to create new datasets, (ii) learn new detectors on new data and (iii) merge the detectors learned on the initial set and those learned on the variants of this set, to form the final representation.

The **third contribution** (Chapter 5) is placed in a framework of learning methods for the image and text modality, the cross-modal retrieval task (for a query image, find the associated texts and inversely, for a text query, find the most associated images), the visual domain of natural images and the general semantic domain (common vocabulary). The objective is to increase the universality of the representation through the learning of detectors that manages several modalities (image and text). The difficulty is to find a common space of representation between these two modalities. In the literature this is mainly done by mapping the detectors of the multiple unimodal representations (text or image). More precisely, this mapping is traditionally learned by solving an intermediate task of bi-directional ranking. The problem is that this procedure encourages the learning of detectors that are too specific to each instance and therefore less suited to universality. To solve this problem, we proposed to perform an additional classification subtask based on classes obtained automatically by clustering without adding new annotations.

Regarding the **quantification of universality** (in Chapter 4), we proposed to evaluate universalizing methods in the context of Transfer-learning (TL). This last consists in learning a representation on a task A (*source-task*), then reusing this representation to describe the data of a task B (*target-task*). This is useful, for example, if one is interested in a target-task that contains too few annotated data. We notice that the lack of data means that a network learned directly from the data associated with this target-task will be less efficient (if even possible without overfitting!) than if we first learn this network on a task for which much more annotated data are available, even if these last are very different from those of the task of interest [1, 11, 34, 209, 186, 187]. This can thus be seen as learning universal representations, since it can be reused in different problems of interest. Hence, this technical context naturally meets the principle of universality, at the core of this Thesis, and we therefore evaluated our contributions in the context of TL. To get even closer to the principle of universality, the methods were learned on *one* source-task and evaluated on *several* target-tasks. This also led us to propose a new quantitative evaluation criterion for universalizing methods.

The outline of the Thesis is organized as follow: in Chapter 2 we extensively discuss the state-of-the-art with regard to our goal of learning universal representations. In Chapters 3, 4 and 5, we respectively describe the first, second and third contributions. The metric proposed to evaluate universality is described in 4. Chapter 6 gives a general discussion on the salient results of the Thesis and draws some perspective to the work.

2

State-of-The-Art

Contents

We start by recording the most seminal works related to the field of image and multimodal representations (Section 2.1). Then, we describe and discuss the few attempts made in the literature towards universality (Section 2.2). Finally, since universality is closely related to interpretability, we discuss this aspect in the context of neural-networks (Section 2.3).

2.1 Image and Multimodal Representations

As mentioned in the introduction, learning universal data representations is fundamental for the construction of a general AI. Thus, in this section, we report and discuss the works in the literature that proposed to learn representations from data. We make a special emphasis on *learned* representations [20] and more precisely, representations obtained from pre-trained Convolutional Neural Networks (named "CNN-Features" in the following) and pre-trained bank of semantic classifiers on top of CNN-features (named "Semantic-Features" in the following). Learned representations have shown to be among the state-of-the-art representations used for image classification since their introduction [114, 291, 11, 209, 81, 115, 252, 22, 21, 186]. They also proved to be rather generic and were applied in various other vision and multimodal tasks, using a large variety of categories and domains [186, 209, 11, 291]. Due to the success of these approaches and the sheer amount of papers describing new features, modifications and improvements, we do not aim at being exhaustive but focus on the most relevant works with regard to the rest of the manuscript. Specifically, we first discuss (in Section 2.1.1) hand-crafted (*i.e.*, not learned) representations, then semantic-representations (in Section 2.1.2) and finally learned CNN-representations (in Section 2.1.3). An overview illustration of learned representations is given in Figure 2.2. Since learned representations are generally used in a Transfer-Learning scheme, we describe it in Section 2.1.5.

2.1.1 Hand-Crafted Features

Up to the early 2000's, images were usually represented by a vector of global features reflecting the colors, textures or shapes present in the image [230]. A higher level of abstraction was provided through the popular Bag-of-Visual-Words (BoVW) representation [229, 46], that consists in extracting a set of local patch descriptors, encoding them into a high dimensional vector and then pooling them into an image-level signature. It was inspired by the traditional Bag-of-Words (BoW) method proposed by [221] to represent textual data. In practice, the local patches were usually described by local features such as the SIFT descriptors [151]. The usual pipeline was composed of the following three main steps:

Codebook learning. From the images in a training dataset, local features are extracted, then one learns a codebook from this collection, using for instance a clustering algorithm such as k-means. Each cluster is then represented by a unique visual word (or codeword), that could be its center of gravity (or its closest local feature in the training set).

Coding. For each image to describe, local features extracted from this image are mapped to visual words into compact descriptors. In the original BoVW model, *hard-coding* [46] maps a local feature to its nearest visual word. However, this coding often introduces large quantization errors. [259] proposed the *soft-coding* method that assigns a local feature to all codewords, according to the similarity between this local feature and each codeword. While reducing the quantization errors, coding on the entire codebook may not be optimal. Wang *et al.* [269] proposed an efficient *locality constrained linear coding (LLC)* that maps each local feature to its L-nearest codewords. This linear local approximation of the manifold supporting the codewords yields to good performances for image classification [269, 147].

Pooling. In the pooling step, the local descriptors are aggregated into a unique image-level representation using a pooling function. The latter can be the average, the sum [135] or the maximum function [282] of all local descriptors (component by component) of an image. An extension named BossaNova [9] was proposed to preserve important information about the distribution of the local descriptors around each codeword, by taking into account the distribution of the distance between local descriptors and the codeword.

Many refinement have been proposed to improve this core scheme. One of the most successful, named spatial pyramid matching (SPM) [135], consists in computing a BoVW in several subparts of the image then (weighting and) concatenating the resulting BoVWs into a unique representative vector. Such a scheme allows to take into account the distribution of the local feature at a global scale of the image. Other works focused on the local spatial consistency of the local features to improve the coding [225].

The Fisher Vector (FV) [196] extends the BoVW by going beyond counting, *i.e* 0-order statistics, to encode second order statistics. The FV representation is computed by characterizing local descriptors by their corresponding deviations from an "universal" generative Gaussian Mixture Model (GMM) of the log-likelihood of the problem. The GMM model can thus be seen as a "probabilistic visual vocabulary". The deviation is measured by computing the gradient of the sample log-likelihood with respect to the model parameters. Originally designed for classification, [197] further greatly improved the retrieval performance of FV by applying a set of normalization strategies *e.g.*, L2 or power normalization to FV and combining this representation with the spatial pyramids.

Jegou *et al.* [113] proposed a simple efficient way of aggregating local image descriptors into a vector of limited dimension for large-scale applications, called Vector of Locally Aggregated Descriptors (VLAD). VLAD is often presented as a simplified version of the Fisher Vector representation, that uses a (quite small) codebook in place of the GMM of the log-likelihood to represent the "universal vocabulary", and replaces the gradient by a simple point-wise difference of the local vector and the codewords. VLAD has been further extended by aggregating tensor products of local descriptors, leading to the Vector of Locally Aggregated Tensors (VLAT) [198, 199] representation, exhibiting good performances both in image retrieval and classification.

2.1.2 Semantic-Features Learned with Explicit Supervision

A drawback of hand-crafted features is their lack of semantics. This has been alleviated in the semantic representations that were introduced in the 2000's by Natsev *et al.* [178] and extensively re-used and extended by [208, 266, 115, 252, 65, 62, 81] for different applications like image matching, retrieval or classification. Here, we thus first highlight three aspects that are important to describe semantic-features, namely *semantic* concept detectors; the *semantic-level* of the concept detectors; and the *explicit supervision* for learning the detectors. Then, we will formally describe semantic representations and detail the implementation used in this thesis as well as the reasons of this choice.

In this thesis, we define semantic representations as a representation learned from data and for which each dimension corresponds to the output of *semantic concept detectors* that can be associated to a humanly understandable word or n-gram (*e.g.*, *car*, *Eiffel-tower*, *ceramics collectibles cooking*, etc.). In the literature, three image representation levels are usually considered, low, mid and high-level, depending on the "quantity of semantics" the representation carries. However, such level has no formal or strict definition and can just be defined relatively. In a nutshell, the low-level corresponds to the information directly extracted from the data of interest, close from a signal, for instance the pixels of an image [111, 119, 134, 226]. At the opposite, the high level should match the (human) user need or understanding, reflecting the semantic concepts that are expressed through language. Let us nevertheless note that the information reflected by low-level features can also be named, "color", "shape" and "texture" being the most usual words used for this. The difference seems thus to lie in the complexity of that is named, and the possibility of describing a high-level semantic as a *compound* concept while it is harder for a low-level one. From this remark, the mid-level representation of a concept reflects such a part of a compound concept, that can be named as the two other levels, such as a part of an object. It worth mentioning that in a CNN, such a transition from low-level representation to high-level ones can be directly observed through the layers when the network maps pixels (as inputs) to semantic concepts (learning classes) [291]. Theoretically, each semantic-detector of a semantic representation can be at any *semantic-level* and in practice all the detectors of the representation are at the same level. Notice that, if a detector is at a particular semantic-level, it has necessarily detected lower-level entities [13]. For instance, if a detector fires on a *person* (high-level), it has necessarily detected lower level entities such as *leg*, *head*, *hand*, etc. Regarding the *semantic* terminology, it comes from their *semantic* detectors. Finally, for the last aspect (*i.e.*, explicit supervised learning of detectors), it corresponds to the way we learn the individual detectors of a semantic representation. Indeed, each dimension of the semantic representation is learned to directly classify data of a certain semantic concept. More precisely, each detector learns to fire on instances of a *known* semantic concept and for which positive and negative instances of that concept have been given to learn the detector. In that sense, the detectors are learned with explicit supervision.

Formally, a semantic-representation can be defined as a set of *pre-trained* semantic detectors. Let us consider an image I and a global image representation $\mathbf{x} \in \mathbb{R}^n$ (*i.e.*, representation of the data at a lower semantic-level than the desired one, that of the concept detector) extracted from I. A *raw* semantic-feature representation is a C-dimensional vector $\zeta^{raw}(\mathbf{x}) = [\varsigma_1^{raw}(\mathbf{x}), \ldots, \varsigma_C^{raw}(\mathbf{x})]$ where each dimension $\varsigma_c^{raw}(\mathbf{x})$ is the output (*i.e.*, probability of presence) of a *pre-trained* classifier of the concept c. Each classifier was learned to recognize a semantic concept c of a certain semantic-level. Each of these classifiers were trained using global representations $\mathbf{x}$ of the images associated to the concept as positive samples and global representations of the images associated to other concepts as negative ones. The complete set of C classifiers can be trained jointly or independently. In the former case, for example, the complete set of classifiers can be trained by applying a softmax function on top of their prediction for each training image, which results to output a real probability of presence of each class among the complete set of classes [1]. In the latter case, each classifier is trained *independently*, thus the classifiers are binary and are learned with SVMs or logistic regression. This latter outputs, for a training image and a class c, the probability of presence (score that is generally squeezed between 0 and 1 through a sigmoid function) of that class on the image, regardless the probability of presence of other concepts. The main advantage the independent training of the set of classifiers is that it allows to extend easily the set of classifiers. This property refers to the life-long learning (LLL) scheme [4, 206] recently introduced in response to the catastrophic forgetting phenomenon [76] observed in transfer-learning, which is highly related to the problem of universality of representations (as expressed in the introduction). Another important point is that the representation of the image $\mathbf{x}$ used as input of the classifiers can be pre-learned and fixed or trainable jointly in a neural network. The latter case is theoretically better than the former, since it learns a representation $\mathbf{x}$ that is designed to better classify the C classes and thus learns a better semantic representation. However, in a LLL scheme and using neural networks, it may lead to this well-known catastrophic forgetting problem [76]. Indeed, by being learned on new categories, the representation $\mathbf{x}$ will get better at representing images of the last learned categories and get worse in the previously learned ones. For these reasons, we focus on semantic features that are obtained on top of fixed (pre-trained)

[1]This case corresponds to the last layer of a CNN.

representations. An illustration of the different schemes is given in Figure 2.1. In Figure 2.2, we illustrate semantic-features extracted from a CNN and highlight its differences compared to other CNN-features (that will be described in the next section).

Most recent and effective works on semantic features [115, 252, 21] used LP-β kernel combiner [80] on many features to design the global image representation and they learned non-linear classifiers for each concept, which induces an important computational complexity. More recently, [81] proposed a lighter computationally scheme, based on linear classifiers. To reach similar or better performances they relied on more powerful global representations (*i.e.*, mid-level CNN features) on the one hand, and *linear* classifiers on the other hand. Moreover, they mostly increased significantly the number C of concept-detectors considered. Since [81] is one of the most efficient semantic-features from the literature and because of the two reasons mentioned in the previous paragraph, we describe their method more precisely. More precisely, in [81], the semantic-representation is built using ImageNet [53] concepts that have at least 100 associated images, resulting in $17,462$ concept-detectors (thus $C = 17,462$). Note that the ImageNet concepts used are at the highest semantic-level, (*i.e.*, objects or scenes) thus, the semantic-representation will only contain *high-level* semantic detectors. Note also that, in this Thesis, we only considered the high-level detectors to be comparable to state-of-the-art methods, but low or mid-level detectors could be used in the representation [191, 91] – generally called *attributes* (*e.g.*, *is it furry?*, *does it have a tail?*, *can it breathe underwater?*, *is it carnivorous?*, *is it slow moving?*, etc.) and used for zero-shot learning [191, 30] –, as long as annotated data at that level is available. To learn the detectors, Ginsca *et al* [81] use images representing the concept as positive samples and images from a diversified class for negative ones. This last contains images of ImageNet-concepts not considered to build the semantic representation. They used a ratio of $1/100$ between positive and negative samples to learn the detectors, this ratio having been determined empirically as a good trade-off between the resulting performance and the computational complexity. Each concept-detector $\varsigma_c^{raw}(.)$ is then a L_2-regularized linear Support Vector Machine (SVM). Finally, each output obtained from the pre-trained classifiers (computed on new images) is normalized by a sigmoid function (hence, $0 < \varsigma_c^{raw}(\mathbf{x}) < 1$). In their work, as well as in [112], they showed that the performances in retrieval and in video classification are significantly improved when the raw semantic-feature vector is *sparsified* for new images. To do so, they force to zero most of the dimensions and retain only a small number d of the largest ones. Formally, the semantic-representation with a "fixed" level d of sparsification (fixed to d for all images) is expressed as:

$$\varsigma_c^d(\mathbf{x}) = \begin{cases} \varsigma_c^{raw}(\mathbf{x}) & \text{if } \varsigma_c^{raw}(\mathbf{x}) \in \mathcal{H}_d(\zeta^{raw}(\mathbf{x})) \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

where $\mathcal{H}_d(A)$ is a subset of size d containing the largest values of the set $A \subset \mathbb{R}$. Once the semantic feature vector has been sparsified with a fixed level it is noted $\zeta^d(\mathbf{x}) = [\varsigma_1^d(\mathbf{x}), \ldots, \varsigma_C^d(\mathbf{x})]$. This approach has been originally introduced as a "fast encoding" method to force Bag-of-Visual-Word to be sparse [269]. Moreover, it results into a much smaller memory footprint since (assuming score and index are both coded on 4 bytes) the *raw* semantic-feature ζ^{raw} occupies $4 \times C$ bytes while the sparsified one ζ^d occupies only $8 \times d$ bytes (with $d \ll C$).

2.1.3 CNN-Features with Learned with Implicit Supervision

An alternative to the use of hand-crafted visual features is to learn the representation directly from the pixels. Early works in this direction [256, 175, 140, 182, 137, 138, 139, 84] were limited to few

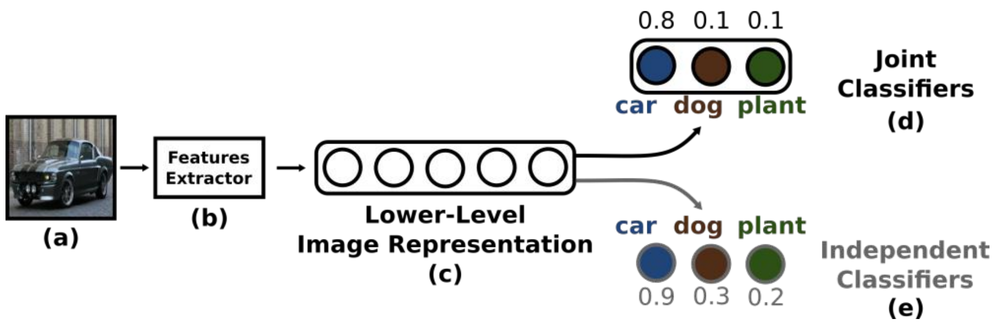

Figure 2.1: Illustration of the different schemes of semantic features. Given an input image (a), we first get an image representation (c) through a features-extractor (b) then we predict the output of a pre-trained bank of classifiers. We illustrate the jointly trained classifiers on top (d) and the independent ones at bottom (e). Best view in color.

range of domains, in particular due to the lack of available data. Following theoretical advances [100] as well as the availability of massive amount of multimedia content and computing capabilities, the CNNs allowed to provide state-of-the-art representations for many visual and multimodal tasks.

One of the most important reasons behind the success of CNNs is their ability to learn rich discriminating features (convolutional and unit-filters). The common scenario to obtain such performing features is to solve a *discriminative problem*, that consists to separate categories from one another. By solving this discriminative problem, representations are learned at all the layers of the network, and the set of features in one particular layer corresponds to the so called CNN-features. However, it is relatively rare in practice to train an entire CNN from scratch because too few data are available to estimate the millions of parameters for most practical vision problems. Instead, it is more common to use a CNN learn on the ILSVRC dataset (1.2 million images labeled among $1,000$ categories) and use it as feature extractor for the problem of interest, in accordance with the *transfer learning* scheme as detailed in Section 2.1.5.

A trained CNN truncated at any of its internal layer can thus be considered as a pre-trained detector that, when applied to images, provides *CNN-features*. Regarding the fact that the internal layers of CNN are learned in an *implicit supervised* manner, it is due to the fact that each data is annotated, but in contrast to the neurons of the last layers, those of the internal are not enforced to fire on a certain category. Simply said, each neuron of the internal layers contribute to the task of recognizing the global set of categories, but it is completely free on the pattern (and thus the category) it will learn to fire on. In that sense, it is learned with implicit supervision.

Formally, let us consider an image I_i^τ of a target-task T_τ, and a pre-trained network model $\mathcal{N}_\Theta$, with parameters Θ. A D-dimensional representation $\mathcal{R}_{i,\tau}=\Phi^K(I_i^\tau) = \{\phi_1^K(I_i^\tau), \cdots, \phi_D^K(I_i^\tau)\}$ is extracted from image I_i^τ with the CNN model $\mathcal{N}_\Theta$ truncated at layer K. Here, each ϕ can be seen as a mid-level detector (*i.e.*, detect object-parts, like faces, hands, wheels, etc.) if K is high, and as a low-level detector (that detects edges, color-blobs or even textures) if it is is low. Each ϕ is followed by a ReLU activation-function thus it outputs a zero-value if its value is negative and the value itself if it is positive. While the ReLU was not introduced to this purpose, its behavior aims to result in a sparsified representation, which is quite similar to the sparsification process in the context of semantic features. Figure 2.2 illustrates the different representations that can be extracted from the layers of CNNs and highlights their differences.

Recently, Zeiler et Fergus [291] showed that *the abstraction of features increases with the depth in the CNN* and Yosinski and its colleagues [285] put in evidence that *early layers of CNNs contain generic*

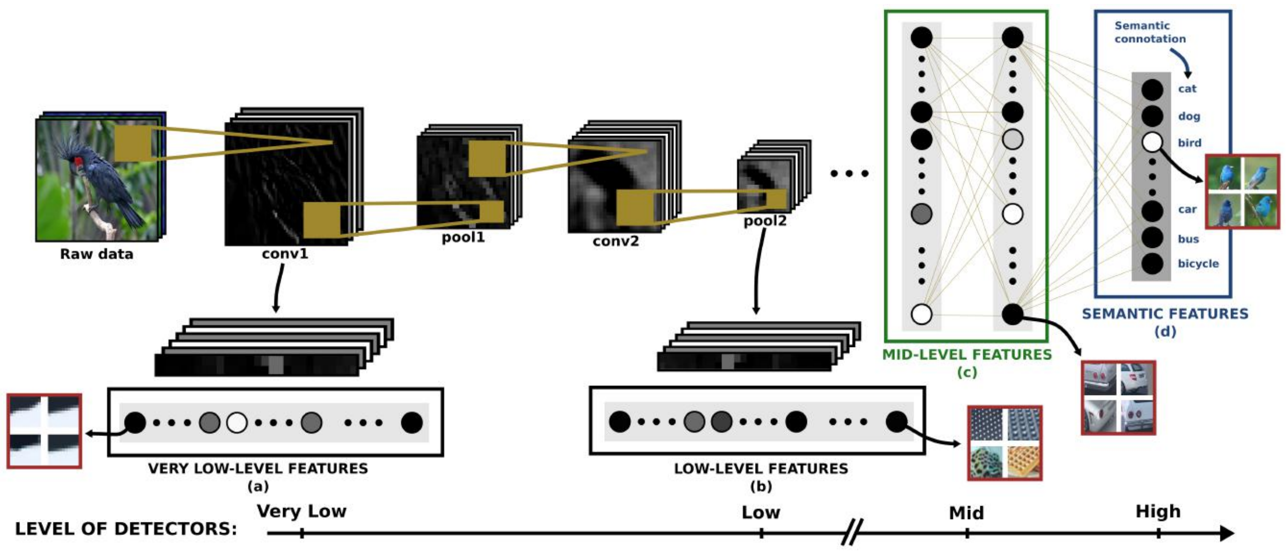

Figure 2.2: Illustration of the different learnable image representations that can be extracted from an image (raw data) through a pre-trained CNN. We can extract representations at different semantic-levels, namely very low-level (a), low-level (b), mid-level (c) and high-level (d). More precisely, in the bottom, we illustrated a scale that gives the semantic-level of detectors of each representation, and we displayed in red squares an example of classifiers that are contained in the representations of different levels (*i.e.*, each square shows four-patches that highly activate the actual classifier as depicted in Section 2.3.2). We also highlighted the fact that, the last layer of the CNN (that contains high-level detectors) also contains a *semantic connotation*, which thus outputs *high-level semantic representation*. Note also that, between very low and high-level features, there is a *continuum*, meaning that no strict semantic-level could be identified, but here we only considered those commonly identified in the literature. Best view in color.

features while last layers contain specific features, resulting in two important good practices when practicing transfer-learning, that relate to the "distance" between the target-task and the source-task (ILSVRC in most cases). If the target-task is far from the source-task (*i.e.*, images and/or labels of the target task are very different compared to those of the source-task), we should use features from the early layers of a pre-trained CNN and inversely, if the target-task is close to the source one, we should use last layers. The exact definition of the "distance" between two datasets is nevertheless an open question and, to the best of our knowledge, no method allows to determine it a priori. Thus, the choice of the layers to use to represent the target-data usually results from the *a posteriori* measure of performance on a validation set.

2.1.4 Multimodal Features

Either semantic features and CNN-features aim at mapping pixels to semantic concepts through various process. Another approach to this purpose is to extract features from each of these modalities then to learn a common space to both of them, in which they are directly comparable.

Early works on joint representation of textual and visual representation relied on topic models, in particular probabilistic latent semantic analysis and latent Dirichlet allocation [25, 171, 203, 207, 74] or cross representation of bag of words and bag of visual words [298]. Another class of approaches was proposed by Hardoon, who relied on the Canonical Correlation Analysis (CCA) and its kernelized version (KCCA) to maximize the correlation between the projections of both modalities [95]. In the following, several types of extension [86, 85, 45, 128, 253, 254, 255] have been proposed to this seminal work. In particular, Gong *et al.* [85] proposed to add to the commonly used CCA, a third view that reflects the "semantic classes" derived from the ground-truth or the keywords used to download the images. They also proposed to derive this third view from *unsupervised clustering* of the tags to avoid the use of ground truth. In the vein of reflecting semantics, [45] proposed to build semantic features into the common space, that is to say to create a signature where each dimension is a given semantic concept that is estimated by a learned binary classifier [252]. Contrary to ours, these concepts are neither meta nor abstract. However, one could image to apply the approach of [21] to get meta-concepts in the common space. Still, a major difference with our work is that each concept is obtained by supervised classification, while in our case, the *abstract* concepts deeply result from an unsupervised approach. Back to CCA, extensions to a deep learning framework have also been proposed [5, 279]. However, as pointed out in [154], a major drawback of the CCA approach it non-scalability to large amounts of data.

An alternative to CCA is to learn a joint embedding space using SGD with a ranking loss. Weston *et al.* [274] and Frome *et al.* [77] were one of the first attempt to learn *linear mapping functions* from visual and textual features to the common multimodal space, with a *single-directional* ranking loss as objective function. More precisely, the single-directional ranking objective consists in reconciling, for each image, its associated caption, and driving away all non-associated captions. Compared to the CCA approach, this deep learning approach and especially the methods based on *ranking loss* easily scale to large amounts of data with stochastic optimization in training. As a more powerful objective function, a few other works have proposed a *bi-directional* ranking loss that, in addition to ensuring that correct captions for each training image get ranked above incorrect ones, also ensures that for each sentence, the image described by that sentence gets ranked above images described by other sentences [123, 122, 127, 231].

Another very active line of research to learn multimodal representation is to rely on deep learning and for instance, techniques such as deep Boltzmann machines [233], autoencoders [179], LSTMs [56], recurrent neural networks [161, 261] and simple MLPs [40, 58, 67, 271, 270]. Methods based on deep neural networks learn *nonlinear* mapping functions and thus are able, in principle, to provide greater representational power than methods based on linear projections [77, 274, 86, 128]. From all the approaches (*i.e.*, CCA, linear mappings and non-linear mappings), the methods that provide the best results are those based on non-linear mappings through simple MLPs. Within these methods, we can identify three groups: (i) *asymmetrical methods* that consist to perform the mapping in one direction only (*i.e.*, from image features to text features [77, 33, 246] or from text features to visual ones [40, 58]); (ii) *symmetrical methods* that bi-directionally maps the features of one modality to those of the other modality [123, 122, 127, 231]; and (iii) symmetrical methods that *preserves some semantics into the learned multimodal representation* [271, 270, 67, 222]. Note that the first group of methods uses a simple MSE loss to learn the mapping functions, while the second group relies on a bi-directional ranking loss, and in the last more recent works, a multi-task objective is considered. In that case, the mapping is learned to solve both a bi-directional ranking loss (or MSE loss for [67]) and an additional loss that enforces the preservation of some semantics in multimodal representations. In particular, for the additional loss, Wang *et al.* [271, 270] uses the multiple textual captions associated to each image (through a ranking loss), Eisenschtat *et al.* [67] enforces a certain cycle-consistency, that is to say, from multimodal features, the initial unimodal should be correctly reconstructed (through a MSE loss), and Salvador *et al.* [222, 236] uses semantic categories (through a classification with softmax loss).

2.1.5 Transfer-Learning

Most learned representations (including Semantic, CNN and Multimodal) lies in a transfer learning scheme. While the concept of transfer-learning [16, 284, 133, 192] was introduced in the context of machine learning in the early 2000's, we mainly focus here on the approaches in the context of deep-learning that were introduced in 2014, by many works independently [1, 11, 114, 34, 57, 186, 187, 209, 291, 202]. In addition, they showed that CNNs work very well on *several* target datasets, even if they have not been trained on the same images and/or categories than those of the source-dataset. This latter makes them quite "universal", as discussed in details in section 2.2.3.

Transfer-Learning roughly consist in taking a machine-learning algorithm and make predictions on the future data using a statistical model that was trained on previously collected labeled or unlabeled training data. A more formal definition is given below.

Definition 1
Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer-learning aims to help improve the learning of a target-representation extractor Φ_T as well as a target-predictive function f_T in $\mathcal{D}_T$ using the knowledge acquired on a source representation-extractor Φ_S learned using a source-predictive function f_S in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.

In a context of deep-learning, the idea is to first use a large annotated auxiliary source-task [2] (*e.g.*, ImageNet classification), to learn the first layers of a neural-network (the source-representation extractor Φ_S) as well as the last layer of the network (the source-predictive function f_S), which is specific to the source-task and not directly reusable for any other task, thus removed. The first layers of the network can then be used on a small target-task, directly, as a representation-extractor (*i.e.*, $\Phi_T = \Phi_T$), instead of training extractor directly on the target domain. This latter provides a representation for each data of the target-task, that is used to train the target-predictive function f_T – could be any predictor (SVM, perceptron, detector, etc.). Two scenarios of transfer-learning can be encountered in the context of deep learning. Indeed, we can either train *only* the target predictive-function f_T on top of the *fixed* pre-trained representation-extractor (this is the *standard transfer*) or *jointly* train the target and source predictive-functions (f_S and f_T), which corresponds to *fine-tuning*. In both cases, only the data of the target-task is available and used to train the predictive-functions. In practice, the classical procedure is to remove the last layer of the pre-trained network, which was specific to the initial task, which is thus not reusable. Next, we append a new predictor specific to the target-task (*e.g.*, randomly initialized layer if perceptron) with the desired number of output units for the target task. Finally, the weights of all the network (if fine-tuning) or only the predictor (if standard transfer) are learned with classical learning strategies on the target loss function. Generally, the learning-strategy is slightly modified since the weights are updated with a smaller learning-rate, to avoid catastrophic forgetting [285]. An illustration of the transfer-learning scheme is given in Fig. 2.3.

In a transfer-learning scheme, since neural networks need many annotated data to be trained without overfitting, the source-task is generally much larger than the target-task. In practice, it is relatively rare to have a dataset of sufficient size to train the millions of parameters of a neural network for a given problem of interest. Obtaining such annotated data is a costly process, not only to collect data but also to annotate them, in particular when such annotation require a particular expertise (medical domain for example, but also for advanced manufacturing, aerial images, astronomy...). Of course, if many

[2]For simplicity, we will use the term *task* to state for both a dataset and a task.

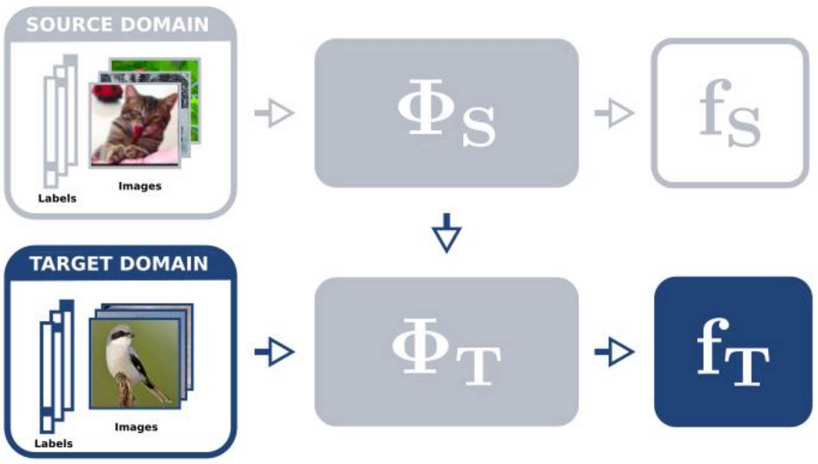

Figure 2.3: Illustration of the transfer-learning scheme. Gray blocks on top, correspond to the element learned on the **source-domain**, while blue ones correspond to elements learned on the **target-domain**. On the target-task, the source features-extractor Φ_S as well as the source predictive function f_S are learned On the source-domain. On the target-task, only the data of the target-domain are available thus the source-predictor in not reusable and is thus discarded (colored in white), then only the *trained* source features-extractor Φ_S is used and a target-predictor f_T is independently (*standard transfer*) of jointly (*fine-tuning*) trained on top of it. Best view in color.

annotated data are available for the task of interest, there is no need to work in a transfer-learning scheme, and the network should be trained directly on the task of interest.

2.2 Universality of Representations

As mentioned in the introduction, the main goal of this Thesis is to increase the universality of representations. Hence, here, we define universality concretely as well as some concepts related to it (Section 2.2.1); then we give our point-of-view on the main directions to increase universality in learned representations – semantic-representations (Section 2.2.2), CNN-representation (Section 2.2.3) and multimodal-representations (Section 2.2.4) – and present the few attempts in the literature that indirectly went towards that goal. Finally, since the universality of representations is highly related to the categories used to train them, we discuss the importance of the name of categories in Section 2.2.5.

2.2.1 Universality Definitions

In this section we especially define some concepts necessary to universality. In particular, we start by defining "universal representations", then we provide the main criteria that should be respected by a method to be qualified as an "universalizing method" (*i.e.*, a method that increases universality of data representations) and finally we define its relation to semantics.

Definition 1. A *universal model* is a model that is able to represent in a high-level, data from different nature (*i.e.*, different visual domains, different semantic domains and different modalities), while at the same time, being able to solve different tasks (*e.g.*, object recognition, detection, parts segmen-

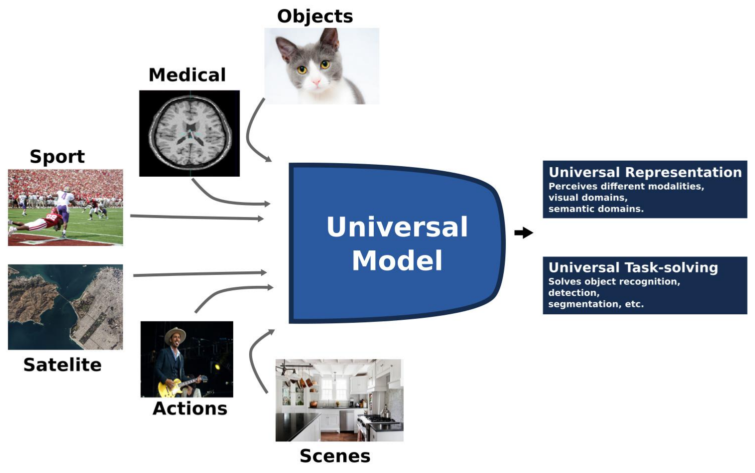
Objects
Medical
Sport
Satelite
Actions
Scenes
Universal Model
Universal Representation
Perceives different modalities, visual domains, semantic domains.
Universal Task-solving
Solves object recognition, detection, segmentation, etc.

Figure 2.4: Illustration of the definition of a universal model, representation and task-solving.

tations, etc.). The first aspect is what we call *universal representations*, while the second is called *universal task-solving*.

An illustration of the definitions of universal model, representation and task-solving is given in Figure 2.4. It should be noted that, reaching *pure* universality seems unfeasible in practice, at least today. Indeed, we do not have all the data required, neither algorithms that could handle such huge amount of data and we can even not list all the possible tasks of the universe.

As mentioned in the introduction, in this Thesis, we are especially interested by universal representations. And, a priori, no representation is purely universal (because we are not able to highly represent data of all modalities, visual domains and semantic domains). But learned representations contain a certain level of universality, because they have been widely used to solve *many tasks*, and obtained *satisfying results*. Thus as stated in the introduction our goal, is: starting from a *reference representation* with a certain level of universality (for which we know its complete learning process, that is, the training-dataset, the network-architecture, the learning-strategy, etc.), learn representations with a higher level of universality. Examples of such reference representations are thus penultimate layers of the well-know deep neural architectures (AlexNet [132], VGG [228], Inception [237], etc.) trained on large datasets as ImageNet [53] or Places [297] and which have been widely used by the community with success on many diverse tasks in the context of transfer-learning. This latter, leads us to the concept of universalizing methods that we will define after the next remark. An important aspect of our goal lies in the fact that universality is *not quantifiable* and thus, it can only be evaluated through human judgment. This point is problematic, because relying on human judgment to evaluate universality is clearly not scalable. But note that our goal considers a reference representation, and tries to increase its universality, thus the only human judgment is applied on the reference representation, and once this last is obtained, we can automatically evaluate the *increase* of universality (this point is more discussed in Section 4.2).

In the literature, several works tackle the problem of increasing the universality of representations, defining it according to two aspects: (1) the resulting representation *obtains better performances on a set of target-tasks* than the reference signature; (2) the resulting representation has more interpretable dimensions than the reference representation. Each of these two aspects worth to be precisely defined, as proposed in the definitions below.

Definition 2. A representation $\mathcal{R}_A$ is *more universal* than a representation $\mathcal{R}_B$ when its aggregated performances over a set of T target-tasks $\{\tau_1, \dots, \tau_T\}$ are superior.

Regarding the aggregation of performances, a detailed discussion is provided in Section 4.2. In [295], Zhou *et al.* consider every individual convolutional unit of a CNN and evaluate their contribution to semantically segment an image according to each element of a set of concepts. This approach can be extended to consider the contribution of an encoding unit on a subset of concepts by defining a function R that associates an encoding unit ϕ_i to a D-dimensional binary vector that is 1 at dimension j if and only if the $j-th$ concept activates the encoding unit.

Definition 3. Let us consider two representation-extractors Φ_A and Φ_B, with each of them containing a set of n detectors $\Phi = \{\phi_i^{A/B}\}_{i=1\dots n}$. The *features-extractor* Φ_A *is more semantic than* Φ_B if $\sum_{j=1}^{D}\sum_{i=1}^{n} R_j(\phi_i^A) > \sum_{j=1}^{D}\sum_{i=1}^{n} R_j(\phi_i^B)$.

As expressed in the introduction of this Thesis, our point-of-view for the ways to build universalizing methods is that it exists three main directions: learning a representation-extractor that, compared to the reference representation, results into a representation that (i) contains *more discriminative* detectors; (ii) contains individual detectors of *better quality*; and (iii) contains *more semantic* detectors, *i.e.*, detectors that can be associated with a semantic interpretation or meaning (as in definition 4).

More precisely, for the first point, if one increases the amount of detectors, it will result in a representation that is able to detect *more* concepts, and thus, to cover a larger range of recognizable concepts. In that sense, going toward the first direction increases the global scope of action of the representation, and in consequence increases the amount of potential users, which by definition, increases universality of the whole representation. Regarding the second direction, it consists in increasing the discriminability and robustness of each detector of the representation. This aims at locally increasing the scope of action of the representation (at the detector level), since it aims at firing on more different instances of a concept (*i.e.*, detects more instances of the concept through the recognition of more point-of-views, illumination settings, modalities, etc. if the concept is semantic, and other kind of unknown configuration if the concept is not semantic). This latter indirectly increases the universality of the whole representation. For instance, let us consider a representation that contains one car-detector that highly activates only when it sees wheels and consider another case when the car-detector fires when it sees wheels *and* car-windows. Hence, the second car-detector will be much more discriminative (in detecting cars) than the first one because of its ability to detect a new occlusion view (*e.g.*, when wheels can not be detected but windows can). Indeed, while the first car-detector was only successfully usable by the users who always see the car-wheels, it is now also successfully usable by those who may only see the car-windows and thus, in that sense, this way of improvement indirectly increases the universality of the representation. Finally, regarding the last point, it aims at alleviating the lack of interpretability of the representations learned with the deep-learning approach (a well known issue for representation obtained with internal layers of deep neural networks [291, 286]). We believe that adding semantics to these learned representations, in line with [18, 87]) is a good direction to improve the universality of obtained representation, in comparison with a reference one. Indeed, the resulting representation, as more interpretable, will be potentially more usable in a wider variety of applications, especially those that need interpretable decisions (which will much more the case with the introduction of the General Data Protection Regulation (GDRP)[3]). Note that, the first two directions aim at increasing universality through an increase of the performance of the representation on a set of target-tasks, while the last one aims to increase its Human interpretability. In this Thesis, we studied universalizing methods for three kind of representations, namely, *semantic-features* learned

[3]The European Parliament introduces, the GDRP, a series of measures laying down the legal framework for the protection of personal data within the European Union. It is about strengthening the rights of EU citizens and giving them more control over their personal data. In particular, users will have more information about how their data are processed. Information which must also be formulated in a clear and precise manner in the spirit of transparency.

with *explicit* supervision (described in Section 2.1.2); *CNN-Features* learned with *implicit* supervision (described in Sec. 2.1.3) and *Multimodal-features* learned on top of unimodal CNN-representations (described in Section 2.1.4). Thus, in the following sections, we discuss the works related to our contributions for these three kind of representations.

2.2.2 Universality of Semantic-Features

As said in the description of semantic features (Sec. 2.1.2), they were first introduced [115, 252] with an amount of 200 object-detectors, then [22] increased this amount to $1,500$ which was followed by an increase to $15,000$ by [21] and then to even $17,462$ and $30,000$ detectors in [81]. All these works increased the amount of detectors in the representation compared to state-of-the-art ones (that can be seen as reference representations), and thus indirectly proposed universalizing methods that go in the direction (i) of Section 2.2. In parallel to this mainstream (to increase amount of detectors), some of these works [21, 81, 112] also improved the lower-level (compared to the desired concept-level) representation used to describe images before training the set of classifiers. More precisely, [21] used more low-level features which results in better image descriptions and [81, 112] used CNN-features which results in even better semantic descriptions. Indeed, the use of better lower-level features aims to have more *discriminative* features (*i.e.*, fires more on images of the positive category and less on images of negative ones) to learn the bank of classifiers that forms the semantic representation. This latter aims at facilitating the learning of each individual classifier of the semantic representations, and thus focuses the training on hardest concepts, which makes them more performing. In summary, both of these works indirectly tackled the problem of increasing the universality of semantic representations by trying to make each individual classifier able to handle more configurations of the concepts, and thus increasing the local scope of action of the final representation (*i.e.*, goes toward direction (ii) mentioned in Section 2.2).

Since increasing the second attempt in the literature to increase universality of semantic-features (through the increase of the discriminability of the lower-level representations), is a problem specific to CNN-features, this aspect will discuss be discussed in the next subsection. Thus, here we especially focus on the first attempt that consists in increasing the amount of detectors. Indeed, we have mentioned that increasing the amount of detectors mainly increases the performances of the representation in many target-tasks, making it a universalizing method. However, a major drawback emerges, when trying to go towards this direction. Indeed, a representation containing a *large* amount of different detectors, will necessarily output, when applied on a new image, a large amount of *small* values, because simple a couple of concepts will be present in the image. We empirically observed that the presence of these small values introduces *noise* in the final semantic representation. This is problematic when learning the target-tasks estimators, since it has to consider them, making its learning harder. To fix this drawback, [81, 112] proposed to rely on the top-k sparsity process [269] (as depicted in Section 2.1.2) that consists to keep the top-K maximum values of the representation, with K fixed arbitrarily. This aims to considerably increase the performances on different target-tasks since it reduces all the noise in the final semantic-features. In that sense, the top-k sparsity process could be considered as a part of the universalizing method that consists to increase the set of different detectors. In this Thesis, we highlighted two drawbacks of this sparsity process, and proposed two contributions to fix each drawback. Both contributions will be detailed in Chapter 3.

2.2.3 Universality of CNN-Features

CNN-based representations are efficient both in end-to-end learning and in transfer-learning, making them good candidates for reference universal representations. Moreover, while not specifically designed towards to improve the universality of such representations, many works [2, 11, 101, 120, 132, 166, 176, 190, 275, 291, 297, 280, 262, 264, 297, 33, 109, 246] could be considered as universalizing methods for CNN-features. The general idea to improve the universality is to diversify and increase the feature detectors. Two families of approaches can be considered. The fist approach consist in modifying the problem on which a network is learned. The second approach relies on ensemble model, that is to say on learning several networks on different problems. In chapter 4 we will propose an unified view of these approaches, using the notion of Discriminative Problem Variation and Source Problem Variation (SPV, detailed in Section 4.1.2.1).

One Net Approaches
The methods of this family [11, 23, 120, 132, 166, 210, 246, 33, 109] usually add new categories and their annotated images to the initial problem in order to diversify it. Three kinds of categories are added: *specific* [11, 23, 210, 297] (*e.g.* rottweiler), *generic* [132, 166, 246] (*e.g.* dog) and *noisy* [262, 264, 120]. In some cases, the categories added contain data from multiple domains [23, 210]. These methods slightly increases the capacity of the network by adding parameters specific to each domain. In all cases, this approach can be quite powerful to increase the universality, but at a high cost since it requires many additive data and corresponding annotations. Another limitation of the methods that add categories of a different types than those of the source-task (*e.g.*, adding generic categories to the initial specific ones [132, 166, 246] lies in their learning methodology. Indeed, they often train jointly the network on the generic and specific data, resulting into a mix of generic and specific features in the intermediate layers. Moreover, the joint learning and especially the minimization of a softmax loss function makes the generic and specific categories mutually exclusive, which violates the actual semantics. At last, all the methods of this approach are based on a single iteration of the SPV and on the training of only one network. These observations have motivated our contribution named MulDiP-Net (Chapter 4) that is based on SPV by grouping (zero-cost or low-cost process), assumes that a clear separation between the different types of features is desirable (respect of the real-world semantics) and that introduces a set of source problems obtained by variation instead of a single one and an ensemble of networks on this set of source problems (improvement of the capacity and knowledge to acquire).

An alternative to the addition of categories is to group several of them. The categories can be grouped hierarchically [109] or using clustering [33].

EM Approach
The methods of the second approach [2, 101, 176, 190, 275, 280] give an answer to the limitation highlighted in the previous section. All the works in this approach use an ensemble-model on different source problems and a *sequential* training procedure. They train a network on an initial problem (that contains specific or generic categories) and they "fine-tune" it on another problem or on a set of small problems. As a consequence, even if the scope of the new representation is increased by this process, all the features learned on the new problems are biased toward those of the model previously learned on the initial problem. Thus, due to their sequential learning procedure, these approaches do not combine different type of knowledge (specific and generic categories) but only consider one of them, that of the last problem used for training. A consequence of the latter point is that, they need many models (*i.e.*, more than 10) to get significant diversity in the set of features, which is very costly. The MulDiP-Net method, described in details in Chapter 4 provide an answer to this limitation by carrying

independent network training (one network per problem).

From the point-of-view of the strategy used to vary the source problem, all the methods [2, 101, 190, 280] from this approach re-labels specific categories into *non-semantic* generic ones, that is to say categories that do not exist in the real world) [33] and capture the common properties among many object classes independently of an actual common semantic. These generic categories are built using hierarchical clustering on low/mid-level features (obtained from a network trained on the initial SP) of images among the initial set of categories. As a consequence, these methods are dependent to the visual low/mid-level features that can lead to irrelevant categories when low/mid-level features fail to capture the dissimilarity between different categories. We will see in Chapter 4 that for the sake of universality, it could be preferable to rely the grouping process on explicit human categorization expertise in order to reflect complex relations between categories.

2.2.4 Universality of Multimodal-Features

Universality is not limited to visual representations. Indeed, in the context of universality, the future problems to solve are *unknown*, and they may come with data of any modality (*i.e.*, visual, textual, sound, etc.). Thus, learning a representation to perceive the world through multiple senses [10] (*e.g.*, seeing, hearing, and reading), rather than only through vision, clearly goes towards our goal of universality. In practice, the most common and efficient way [95, 231, 77, 10, 270] to learn multimodal representations is to start with unimodal representations (*e.g.*, visual and textual) and to add a multimodal block on top of unimodal representations that learns to *align* them. More precisely, the multimodal block learns an embedding space where unimodal features of aligned data (*i.e.*, data from multiple modalities represent the *same* physical world), projected in the multimodal space, are close to each other, and inversely, multimodal features of non-aligned data are far from each other. With the latter approach, we hope to learn multimodal detectors that combine unimodal detectors of the same semantics (*i.e.*, multimodal detectors fire on a concept in a *modality-agnostic*) and disentangle unimodal detectors of different semantics. Back to the universality, let recall that it lies in a transfer-learning scheme, where the representation is learned on a certain problem at an initial time, then used, to represent the data in different problems, at a different time. This being said, in the multimodal community, generally, only bi-modalities are considered (*e.g.*, vision and text, vision and sound, text and sound, etc.) and end-to-end learning is considered (*i.e.*, learning and evaluating the representational power of their multimodal representation on the same problem and same visual domain). While by definition, the more modalities we consider, the more universality we get, in this work (Chapter 5), we simply followed the literature for the nature and amount of modalities. However, rather than working on one unique problem, we put ourselves in a transfer-learning scheme to go toward our goal of universality. To the best of our knowledge, it exists only a couple of works [149, 10] that evaluated the transferability of multimodal representations in a transfer-learning scheme. Indeed, [149] trained their multimodal representation on one domain (MSCOCO dataset) and evaluate it on another one (Flickr30k dataset), while [10] trained their multimodal representation using only image-sound and image-text pairs, but evaluated the transferability on sound-text retrieval. While in the former, the transferability is done on the multimodal domain, in the latter, it is done on the task with missing aligned data during the training. Nevertheless, in both cases, the authors showed that the performances were lower than directly learning on the target-problem, but still performing, since the *in-the-wild* situations (*i.e.*, universality) are challenging. In contrast, in the context of universality, we want to be highly transferable on both, different domains and missing aligned data during the training. While the last point is promising and could highly increase the capability of the learned representations, in this Thesis, we especially focused on the former point.

An important aspect in universality through the learning of multimodal representations is that, we should care to not hurt the representation power of mono-modalities. In fact, on the former case, the universality is increased, but on the latter one, it decreases. Thanks that, generally the multimodal representations are built *on top of* unimodal features (visual, textual, sound), thus does not hurt the representational power of mono-modalities, which is highly desirable. Thus, in the context of universality, it is non-desirable to *fine-tune* the unimodal representations in parallel to the multimodal one (like in [222, 72]) since, if learned on specialized domain, or general domain but with few annotated data, we could specialize even the unimodal representations, that has been already learned to be universal and thus hurt their universality.

Finally, while not extensively discussed, it should be mentioned that the problem of universal representation has seen a recent growing interest in the NLP community. Indeed, the works of Conneau *et al.* [42, 44, 43, 124] are pioneering in the problem of universality. Roughly, their goal is to find the best source-task and learning algorithm for learning the most universal representation. The work of Nie *et al.* [180] rather starts with non-annotated data and consists in finding "tricks" to get annotations. Subramanian *et al.* [235] made one step forward, by using *multiple* source-tasks, and thus proposed to find the best set of source-tasks as well as the best multi-task learning algorithm. In contrast to all these works, our goal consists to start with some annotated data and try to automatically get *more* annotations in order to diversify the source-problem, and thus learn more universal representations. Notice that, our contributions are not limited to visual and multimodal representations and could easily applied to build more universal textual ones. Another important aspect of all the works of universality in the NLP community is the evaluation protocol [41]. Indeed, while in the vision community, only end-to-end learning [23, 210] and fine-tuning [130, 211] (*i.e.*, modifying the representation on the target-problem after training it on the source-problem) schemes were considered for evaluating universality, in NLP, it was always performed in a transfer-learning without modifying the representation on the target-problem, but rather by using simple predictors on top of the representation. Let recall that as stated in [244], compared to end-to-end learning and fine-tuning, such transfer-learning scheme (w/o modifying the representation) has a better matching with cognitive studies on the visual brain [8] – that highlights the ability of humans to develop a universal and powerful internal representation of images in the early years of their development and re-use it later in life for solving any kind of problems. To the best of our knowledge, we are the first from the vision and multimodal communities to evaluate universality of representations in such a scheme.

2.2.5 Category-Names and Universality

Recently many works [54, 163, 188, 189] highlighted the importance of considering the words used to name the categories in the field of computer vision and proposed to take inspiration from cognitive studies that give some insights regarding the Human-categorization process [118, 214]. While the main goal of these works was especially to output concepts mostly used by humans rather than more specific ones, we think that the question of how to name categories is also related to the semantic encoded in the learned representations and thus universality.

Learned-representations are obtained by learning a neural-network on a training-dataset, that is formed of a set of instances organized according a set of different categories (that we call *discriminative problem* in the following). During the creation of the training-dataset, multiple annotators are asked to describe each instance according to a certain level of description. Indeed, datasets contain categories at different levels of specificity, that is to say, different level of annotators-expertise. For instance, ILSVRC [216] contains *fine-grained* categories (*i.e.*, very specific categories obtained by experts), Pascal VOC [71] contains basic-level categories (general categories obtained by persons that use the most common human language) and ImageNet [53] contains categories at several levels (*i.e.*, generic, basic, specific, etc.). The name of a category is given according to a *human* judgment, thus it is subjective. Generally, the exact choice of the word used is far from being neutral, as a large literature has shown it, both in Psychology [118, 214] and Computer Vision [54, 163, 188, 189]. More precisely, they showed the importance to consider three main levels of categories that are commonly used in the Human-categorization process:

- Basic-level concepts are the terms at which most people tend naturally to categorize objects, usually neither the most specific nor the most general available category but the one with the most distinctive attributes of the concept.
- Superordinate concepts are categories placed at the top of a semantic hierarchy and thus display a high degree of class inclusion and a high degree of generality. They include basic-level and subordinate concepts.
- Subordinate concepts are found at the bottom of a semantic hierarchy and display a low degree of class inclusion and generality. As hyponyms of basic-level concepts, subordinate categories are highly specific.

In summary, during the creation of a dataset, the discriminative-problem (DP) is the result of the way we decided to name instances and categories. As illustrated in Figure 2.5, the way we name categories, could lead to different discriminative-problems. However, datasets are associated to one discriminative-problem only. Thus, we can consider that a dataset is incomplete since it does not consider the other discriminative-problems (that would reflects other point-of-views of the instances). However, it is important to note that the semantic encoded in neural-networks highly depend on the dataset used to learn them. Indeed, learning to discriminate between $\{lemon, banana\}$ lead to different features than solving $\{lemon, strawberry\}$. This point was confirmed by [297] that showed that object-detectors emerge when learning to discriminate between scene categories (*i.e.*, a network trained on a scene-based DP learns *different* features than one trained on an object-based DP). This latter confirms the importance of the discriminative problem in the resulting features. Thus, the way we name categories has direct consequences on the learned representations. Unfortunately, as detailed in Section 2.2.3 a few works proposed to consider the discriminative-problem in the problem of learning representations, while it is highly related to universality.

2.3 Interpretability

In Section 2.2, we claimed that increasing interpretability of learned representation, increases universality. However, a first step to interpretability is to qualify and quantify what has been learned on each individual feature that forms the representation, that is to say, associate a concept to each learned detector. Thus, in this section, we will first highlight the main difficulties to interpret neural networks (Section 2.3.1), then present the different techniques in the literature that leads to associate a

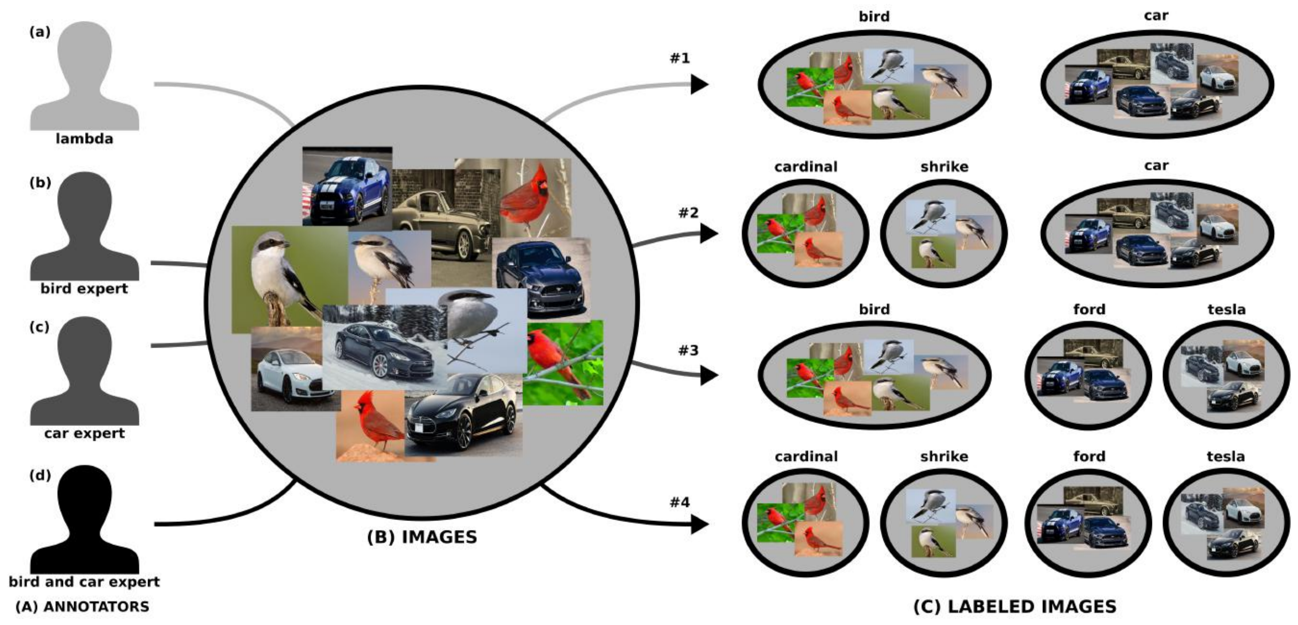

Figure 2.5: Illustration of the significant difference that can result with different discriminative problems obtained by different subjective annotators. In (A), we have four annotators – (a): lambda annotator, (b): bird-expert, (c): car-expert and (d): bird and car-expert. Each annotator has to label the images in (B) according to his(her) expertise (and thus, according to his(her) vocabulary). In (C), we report the resulting discriminative problem (DP) from each subjective annotator. We clearly see that in each case, we have very different DPs, and this could clearly lead to very different set of features after solving the DP through a CNN. Best view in color.

concept to each detector. Indeed, two approaches emerge, those that considers a concept as extension, and thus associates each detector to multiple visual instances (Section 2.3.2), and the recent ones that considers a concept as intention, and thus associates each detector to a semantic concept described through language (Section 2.3.3).

2.3.1 Interpretability and Neural-Networks

A informal definition of *interpretability* has been introduced by Alfred Tarski in 1953: "*Assume T and S are formal theories. Slightly simplified, T is said to be interpretable in S if and only if the language of T can be translated into the language of S in such a way that S proves the translation of every theorem of T. Of course, there are some natural conditions on admissible translations here, such as the necessity for a translation to preserve the logical structure of formulas*". Thus, to make neural-networks interpretable to Humans, we need to translate their "language" into the language of humans. Thus, a first step is to consider the *individual learned features* (*i.e.*, convolutional-filters, neurons) of neural-networks as the basis of their "language" and find associations (*i.e.*, translation) with the *semantic-concepts*, that are considered as the basis of the human language. Thus, for simplicity, we will use the terms *first step towards interpretability* to state for the association of semantic-concepts and learned detectors. Note that, this first step towards interpretability is highly related to universality and corresponds to the third direction to it (2.2), that is, increasing the semantics of a representation.

While the domain of semantics (and thus semantic-concepts) have been actively investigated by the community (*e.g.*, many available resources organized in the form of ontologies), it is less true for the

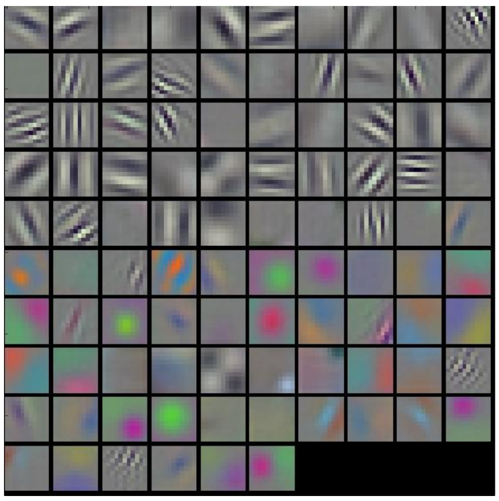

Figure 2.6: Illustration (from [132]) of the 96 filters (of spatial size 11×11) learned at the first convolutional layer ($conv1$) with the AlexNet architecture on the ILSVRC dataset. It contains edge and color-blob filters. Best view in color.

"language" of neural-networks. Indeed, the question of *what features have been learned in neural-networks?* is very difficult to answer. In particular, the main difficulty comes from the fact that neural networks are a stack of layers (*e.g.*, fully-connected, convolutional, pooling, activation, normalization and fully-connected) learned with implicit supervision, thus investigating into the understanding of what has been learned on the intermediate layers requires making sense of non-linear computations performed by millions of learned parameters. More precisely, fully-connected and convolutional layers are respectively formed by a set of individual neurons (that are represented by single values) and filters (that are represented by 3D-tensors of size $W_F \times H_F \times D_F$). For the filters, they can be directly visualized their depth is D¡4, since they correspond to gray images (if D=2) or RGB images (if D=3). However, in almost all convolutional layers of CNNs, the depth of the tensor-filter is D>3, thus the filters can be visualized as multi-spectral images, which are non-understandable by humans and thus *not-easily* interpretable. Regarding neurons, the same problem is encountered, since they output only single values that are clearly not interpretable by humans.

Nevertheless, in the actual form of CNNs, we can still visualize some elements to understand what has been learned: (i) *filters* at the first convolutional-layer ($conv1$) for all the architectures and (ii) *feature-maps* of all filters at all layers for all architectures. More precisely, filters of $conv1$ can be visualized because they have a depth of 3 and can thus be seen as RGB images. In Figure 2.6, we illustrated the 96 filters learned at $conv1$ in the AlexNet architecture. In this figure we clearly observe that the CNN has learned edges and color-blob filters. It is a good step for the understanding of CNNs, since it aims us to visualize and interpret the features learned. However, this visualization can not be performed for other layers. Indeed, for other layers, only feature maps can be visualized. And, feature-maps are visualizable since they correspond to gray images. However, a downside to visualize them is that in addition to be less interpretable, it can quickly become fastidious since each feature map is associated to *one filter* for *one image*, only. Indeed, each layer contain hundreds of filters, that reacts very differently according the images, thus many different images should be analyzed to interpret only one filter. Theses drawbacks are illustrated on Figure 2.7.

In summary, neural-networks are hard to interpret with simple techniques, thus it can be seen that there is a necessity to provide more advanced techniques, which is the scope of the next section.

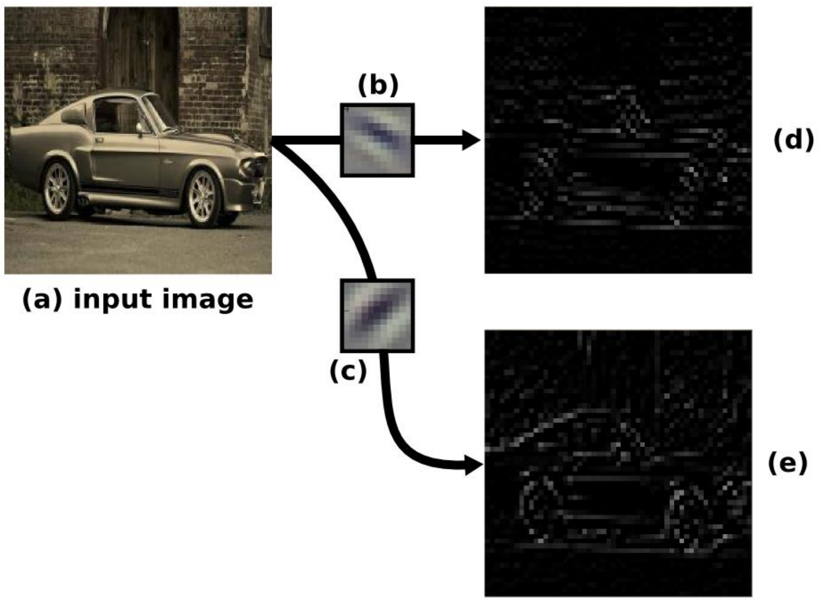

Figure 2.7: Illustration of the feature-maps (d) and (e) respectively obtained from the convolution of two filters (b) and (c) – from the 96 filters of $conv1$ of AlexNet illustrated in Figure 2.6 – with all spatial locations of an input image (a). By comparing Figure 2.6 and this one, we can easily claim that visualizing filters is much interpretable than visualizing feature-maps. Moreover, we also highlight that each feature maps is assigned to one filter but to one image only, which can become quickly fastidious if we want to interpret the features that outputs these feature maps. Best view in color.

2.3.2 Extension-Based Detector-Concept Association

During the last five years, a few works [18, 156, 227, 291, 286, 296, 295] have prompted investigation into the meanings of the learned features on intermediate layers of neural-networks. As said above, we grouped the works into two approaches: (i) those that considers a concept as *extension*, and thus associates each detector to multiple visual instances; and (ii) those that considers a concept as *intention*, and thus associates each detector to a semantic concept described through language. In this section we will especially describe the works in the first approach.

A first satisfying attempt was the work of Girshick *et al.* [82], in which they proposed to find for each convolutional filter of a CNN, the local regions (from images of a large database) that highly activate them. However, [82] was initially a contribution that consisted of an object detection method based on CNNs (*i.e.*, R-CNN). Hence, they only applied this technique on some features of the last pooling layer ($pool5$) of the AlexNet architecture. Moreover, their visualization was to highlight (through white-border windows), the *local region* in the images that highly activates the $conv5$ features (after the pooling layer). An illustration of their visualizations (from their original paper) is given in Figure 2.8. A few months later, this technique was extended by [291] (won the best ECCV paper) that applied it on *all the features* at *all the layers* of AlexNet. They also proposed a slightly more visualizable version, *i.e.*, they displayed the *top-nine* max-*patches* (rather than windows highlighting the max-region in the max-images) and in *three by three blocks* (rather than all maximums in the *same line*). In the following, we call this technique *Top-k max-patches*. An illustration of their visualizations is given in Figure 2.9. This more visualizable version was then commonly used [143, 286, 296] to visually highlight the semantics learned by features of a CNN.

Since this *top-k max-patches* technique is the basis of almost all others (and since we also used it

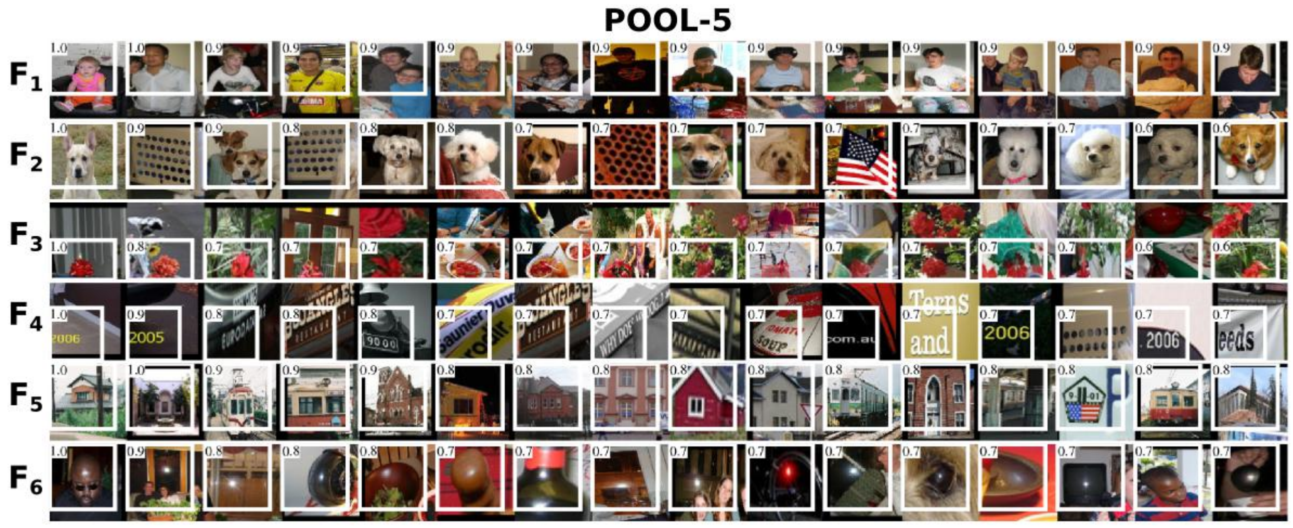

Figure 2.8: Illustration of the visualizations reported in [82]. They displayed, a set of 96 images, where each line represents the max-16 images that highly activate a filter. In each max-image, they highlighted (in a white window) the region that highly activate the actual filter. In their paper, they only visualized 6 filters of the last pooling layer ($pool - 5$) of AlexNet, which thus corresponds to filters of the las convolutional layer ($conv - 5$). Best view in color.

in one of our work (Chapter 5)), we describe it more precisely. Let consider a set of N images $\mathcal{I} = \{I_1, \cdots, I_N\}$ and the j^{th} filter $F_j^{(l)}$ of layer l of a CNN. The top-k max-patches technique, consists to, first extract from the whole set of images $\mathcal{I}$, a set of M *patches* per image – *i.e.*, local regions per image. We will use the notation R_i^j to denote the i^{th} patch extracted from the j^{th} image I_j of set $\mathcal{I}$. The set of all extracted patches will be denoted $P^{\mathcal{I}} = \{R_1^1, \cdots, R_M^1, \cdots, R_1^N, \cdots, R_M^N\}$. This set $P^{\mathcal{I}}$ is of size $M \times N$ since it contains M regions per image for a set of N images. In the literature, these patches (local regions) are called "receptive fields" and their spatial size depends on the size of the filter $F_j^{(l)}$ as well as the eventual operations computed by the previous layers. Hence, the *maximum* amount of patches that can be extracted per image depends on (i) the size of the filter $F_j^{(l)}$; (ii) the eventual operations computed by the previous layers; and (iii) the size of the input image, thus here, we will just suppose that this number is the same for all images and it corresponds to the number that we denoted M. This being said, the second step of the top-k max-patches technique consists to get the highest K patches $\{P_{max,1}^{\mathcal{I}}, \cdots, P_{max,K}^{\mathcal{I}}\}$ that satisfies, for a filter $F_j^{(l)}$, the following equation:

$$P_{max,k}^{\mathcal{I}} = \arg\max_{\mathcal{P}_i^{\mathcal{I}_k}} (\mathcal{P}_i^{\mathcal{I}_k} * \mathcal{F}_j^{(l)}), \tag{2.2}$$

where, $\mathcal{I}_k = \mathcal{I} \setminus \{P_{max,1}^{\mathcal{I}}, \cdots, P_{max,k-1}^{\mathcal{I}}\}$ is the set of patches that does not contain the previously obtained max-patches (for the actual filter $\mathcal{F}_j^{(l)}$) and $*$ is the convolution operator. Note that, while not performed in [291], the same technique could be performed for neurons as done in [286]. Indeed, the principle remain the same, except that no local patches are considered but only the full images, because neurons have a receptive field of the size of the image.

This technique was used to represent the learned features of AlexNet and aims to highlight very interesting properties of CNNs which bring, to the community, a better understanding of them. More precisely, in the work of [291], it has been highlighted that CNNs (at least, the AlexNet architecture), learn a *hierarchy of features*, that is to say, in $conv1$ the filters represent edge and color-blob patches,

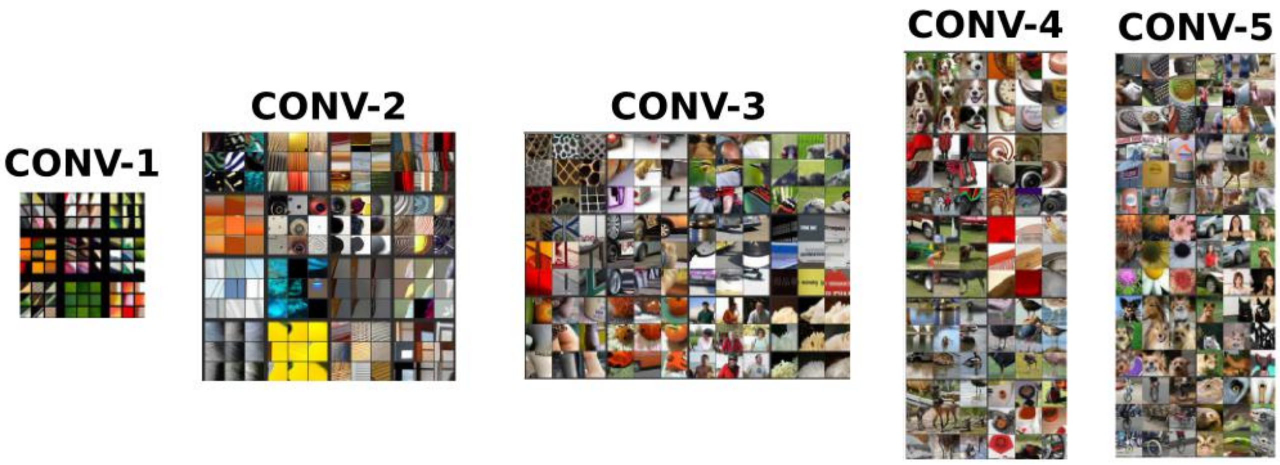

Figure 2.9: Illustration of the visualizations reported in [291]. They displayed, five blocks of visualizations, with each block corresponding to one layer of the CNN (AlexNet). In each block they displayed many blocks of three by three patches, with each of the blocks corresponding to one filter of that layer. For clarity reasons, they only displayed a few set of blocks (*i.e.*, one block contains nine-patches that highly activate one filter) per layer. We can see that, the more we go deeper, the more the filters highly activate on entities of high abstraction (*e.g.*, *edge*-detectors at $conv1$, *texture*-detectors at $conv-2$ and *object-part* detectors at $conv5$.). Best view in color.

in $conv2$ and $conv3$ *texture*-detectors have been learned, in $conv4$ there are more complex $textures$, in $conv5$, there are *object-part* detectors and finally in the last layer $fc8$, there are complete *object*-detectors. Simply said, the more we go deeper in a CNN, the more its features detect entities with higher *abstraction*.

After the successful results of these visualization techniques, some more advanced methods have been suggested in the literature. More precisely, a *backpropagation-based* approach has been investigated. It consists to use variants of backpropagation to identify or generate salient image/patch-features [156, 227, 291]. Note that, the *gradient-based* approach has been initiated by [69], in 2009 (*i.e.*, three years before the breakthrough of deep-learning). This visualization technique brings some insights to the deep-learning field but not as much as the *top-k max-patches* technique [82, 291], that was seen (at that time) as a breakthrough in the field of deep-learning. Another interesting method was proposed by Zhou *et al.* [296]. Indeed, while all other methods used the theoretical receptive field (RF) of each convolutional layer, they proposed to rather focus on the *empirical* size of the RFs, and showed that it is surprisingly much smaller than the theoretical one. In practice, to estimate the learned RF of each unit in each layer, they proposed a data-driven approach on top of the top-k max-patches technique. Indeed, for each top-k image, they replicate each image many times (5000 at all spatial locations) with small random patches, which results in 5000 occluded images per original image. Then all these occluded images are feed into the same network and record the change in activation as compared to using the original image. If there is a large discrepancy, the given patch is important and vice versa. This allows them to build a discrepancy map for each image, and highlight more precisely the region that provokes the activation of the filter. An illustration is given on Figure 2.10. Otherwise, Yosinski *et al.* [286] highlighted that some very discriminative detectors (*i.e.*, text, face or even t-shirt detectors) that were not asked to be learned (since they do not correspond to the labels of the training-database ILSVRC), were surprisingly learned by the network.

In summary, these visualization techniques leads to interpret learned detectors of neural networks and highlights amazing properties of them. However, all these works considers the concepts as extension,

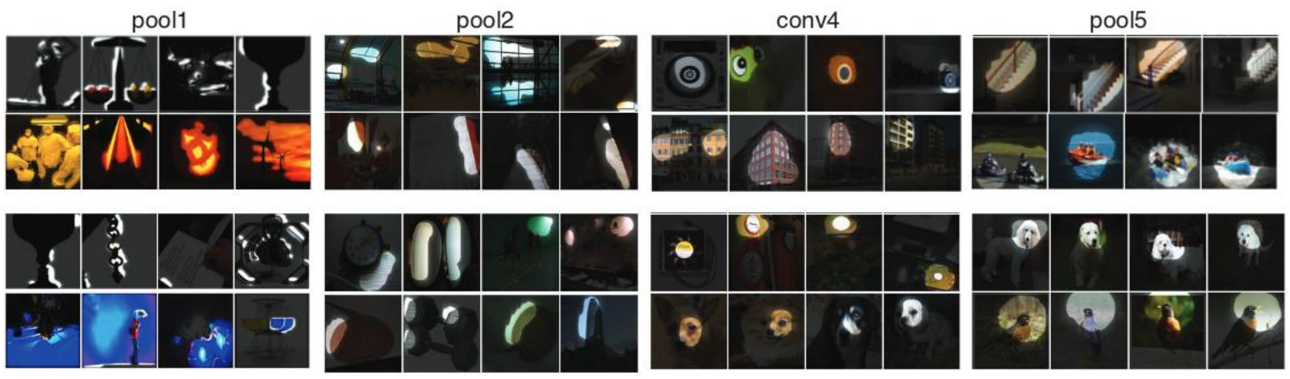

Figure 2.10: Illustration of the visualizations reported in [296]. For each layer, the display the top-4 images obtained (for a certain unit filter) with their "Bubble-vizualisation" method. On top, those learned on the Place [297] dataset and at bottom those learned on ImageNet [217]. Best view in color.

and thus aim at associating the individual learned detector to *multiple instances* (*e.g.*, top-k images or more), which can be less interpretable than associating them to semantic-concepts. This latter, consist in considering concepts as intention and is discussed in the next section.

2.3.3 Intention-Based Detector-Concept Association

Very recently, [18, 295] suggested a *semantic-based* approach (called "Network-Dissection") to directly and automatically match features (here, convolutional filters and neurons) of internal layers of CNNs with *labeled* interpretations (*i.e.*, categories from the most common *colors*, *textures*, *materials*, *object-parts*, *objects* and *scenes*). They applied their technique on state-of-the-art network architectures to investigate their effects on interpretability (at least, with their definition of interpretability). Indeed, such an approach seems quite promising because it is able to characterize a whole complex network with a single bar chart which aims to interpret a network very quickly. In this bar chart, each bar represents a layer, and each of them is separated through different colors, in which each color represents the amount of features of a particular type (*e.g.*, amount of texture-filters, amount of object-part filters, amount of material-filters, etc.). An illustration of their bar chart is given in Figure 2.12. Moreover, among the bar chart, their method is able to associate to each individual detector a certain element from a large bank of semantic-concepts (*i.e.*, colors, textures, materials, object-parts, objects and scenes). An illustration is given in Figure 2.11.

In summary, many visualization techniques have been introduced in the literature and each of them has its own advantages and limitations. Anyway, putting them all together [18, 69, 82, 156, 227, 143, 286, 291] leads to a better understanding of CNNs by the highlight of important properties and behaviors. More importantly, these works make neural-networks more interpretable than before. In particular, the work of [18, 295] was a first attempt to the association of concepts-detectors with an intention-based definition of concepts and should thus be considered as a reference work for the interpretability of neural-networks.

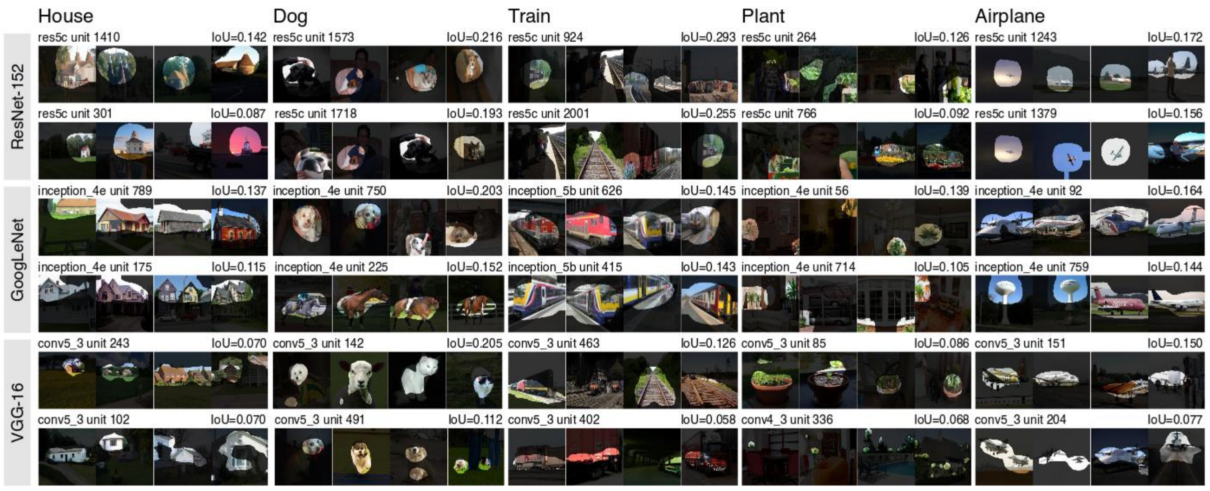
House
Dog
Train
Plant
Airplane
ResNet-152
GoogLeNet
VGG-16
res5c unit 1410 IoU=0.142
res5c unit 1573 IoU=0.216
res5c unit 924 IoU=0.293
res5c unit 264 IoU=0.126
res5c unit 1243 IoU=0.172
res5c unit 301 IoU=0.087
res5c unit 1718 IoU=0.193
res5c unit 2001 IoU=0.255
res5c unit 766 IoU=0.092
res5c unit 1379 IoU=0.156
inception_4e unit 789 IoU=0.137
inception_4e unit 750 IoU=0.203
inception_5b unit 626 IoU=0.145
inception_4e unit 56 IoU=0.139
inception_4e unit 92 IoU=0.164
inception_4e unit 175 IoU=0.115
inception_4e unit 225 IoU=0.152
inception_5b unit 415 IoU=0.143
inception_4e unit 714 IoU=0.105
inception_4e unit 759 IoU=0.144
conv5_3 unit 243 IoU=0.070
conv5_3 unit 142 IoU=0.205
conv5_3 unit 463 IoU=0.126
conv5_3 unit 85 IoU=0.086
conv5_3 unit 151 IoU=0.150
conv5_3 unit 102 IoU=0.070
conv5_3 unit 491 IoU=0.112
conv5_3 unit 402 IoU=0.058
conv4_3 unit 336 IoU=0.068
conv5_3 unit 204 IoU=0.077

Figure 2.11: Illustration of the visualizations reported in [18, 295] for the association of concepts to each detectors learned. We can observe that each unit of a certain layer is associated to multiple visual instances and more interestingly, to a single semantic-concept (*e.g.*, *house*, *dog*, *train*, etc.). They reported the results for multiple network-architectures. Best view in color.

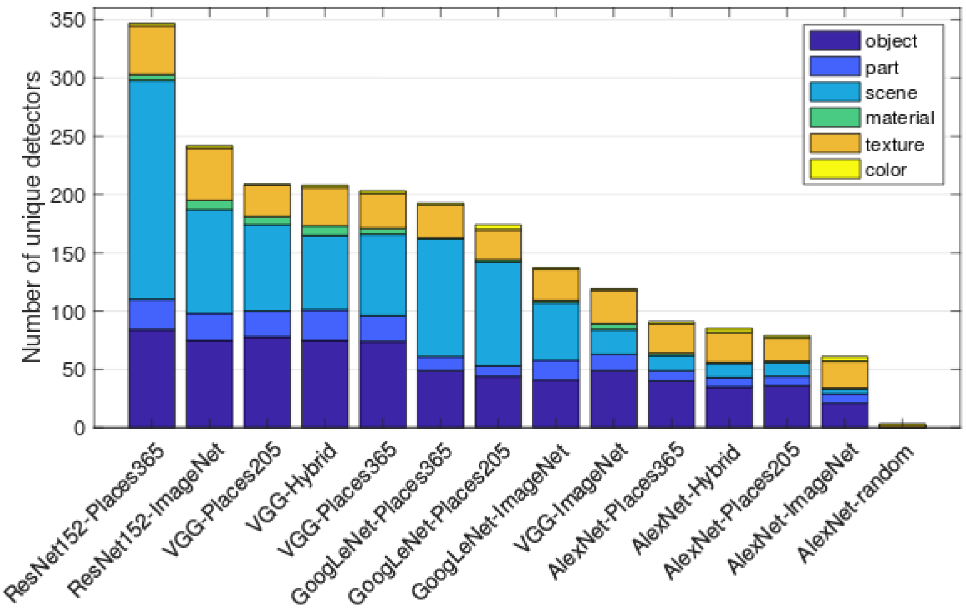

Figure 2.12: Illustration of the visualizations reported in [18, 295]. They compare different architectures and training database CNNs (x-axis) through their amount of unique detectors (y-axis) in a particular kind (object, part, scene material, texture and color). Each bar represents the last convolutional layer of one network. Best view in color.

3

Improving Universality of Semantic Representations using Structured Sparsity

Contents

Semantic-features are good candidates to learn more universal representations thanks to their ability to extent the representation without having access to the data used to build the previous features. Indeed, increasing the amount of features was the most (indirectly) investigated way in the literature to increase universality of the whole semantic representation. The state of the art showed that applying a *fixed* level of sparsity to these semantic representations improves them significantly. In this chapter, we highlight two major drawbacks of this approach and propose a specific solution to each of them. The first contribution, that consists in computing an *adaptive* sparsity according to the content of each image, is presented in Sec. 3.1 and the second contribution that consists in computing a sparsity according to the human-categorization process is presented in Sec. 3.2. We summarize in Table 3.1 the notations used in this Chapter.

Symbol	Description
I	Image
$\mathbf{x}_I$	Mid-level representation extracted from an image I.
$\mathcal{C}$	Set of considered categories, $\mathcal{C} = \{c_1, ..., c_C\}$.
C	Dimensionality of the semantic representation, $C = \|\mathcal{C}\|$.
$\zeta(\mathbf{x}_I)$	Semantic representation of an image I computed on its mid-level feature $\mathbf{x}_I$.
$\zeta^m(\mathbf{x}_I)$	Semantic representation with a sparsification of type m – *i.e.,* if $m = raw$, no sparsification; if $m = d$ with $d < C$, the sparsification level is d; if $m =$ CBS the sparsification level depends on the image and if $m =$ DCL the sparsification depends on the concept-level.
$\zeta_i^m(\mathbf{x}_I)$	A semantic representation, i^{th} dimension of any semantic representation of type m, (*i.e.*, $\forall m \in \{raw, d, \text{CBS}, \text{DCL}\}$).
$\phi_{c_i}^m(\mathbf{x}_I)$	Output of a visual binary classifier with a sparsification of type m with $m \in \{raw, d, \text{CBS}, \text{DCL}\}$
$\phi_{c_i}^V(\mathbf{x}_I)$	Output value of the visual detector of class c_i and i^{th} dimension of $\zeta(\mathbf{x}_I)$.
$\phi_{c_i}^S(\mathbf{x}_I)$	Output value of the semantic detector of class c_i.
$\mathcal{H}_d(A)$	Set of the d largest values of a set A of C values ($A \subset \mathbb{R}^C$).
$\max_\zeta$	Maximum value of any semantic feature, $\max_\zeta = \arg\max_{\phi_{c_i}}(\zeta^{raw}(\mathbf{x}_I))$.
$H(\mathbf{x}_I)$	Entropy of the random variable of the semantic representation of an image I.
$\Gamma(\mathbf{x}_I)$	Sparsification threshold for a semantic representation of an image I.
α	Normalizing parameter for the computation of $\Gamma(\mathbf{x}_I)$.
$\varsigma(c_i)$	Subsumption function that outputs the set of concepts that are subsumed by c_i.
$\mathcal{H}$	Semantic hierarchy (with "is-a" relations) with all nodes being concepts.
$\mathcal{N}$	Set of all concepts in a semantic hierarchy.
$\mathcal{P}$	Set of *superordinate* concepts from a whole set of C concepts in $\mathcal{N}$.
$\mathcal{BL}$	Set of *basic-level* concepts from a whole set of C concepts in $\mathcal{N}$
$\mathcal{B}$	Set of *subordinate* concepts from a whole set of C concepts in $\mathcal{N}$
$\mathcal{B}^K$	Set of most-salient subordinate concepts of an image I in $\mathcal{N}$
D^d	Target-dataset that contains d categories.
$\mathcal{BL}^d$	Set of basic-level concepts of the target-dataset D^d.
$\mathcal{P}^d$	Set of superordinate concepts from a whole set of C concepts, but adapted to the target-dataset D^d.
$\mathcal{B}^d$	Set of subordinate concepts from a whole set of C concepts, but adapted to the target-dataset D^d.

Table 3.1: Notations used in Chapter 3 of this Thesis.

3.1 Constrained Local Semantic Features

3.1.1 Introduction

As described with details in Section 2.1.2, a semantic representation of image I is a C-dimensional vector $\zeta(\mathbf{x_I}) = [\varsigma_1(\mathbf{x_I}), \ldots, \varsigma_C(\mathbf{x_I})]$ where each dimension $\varsigma_c(\mathbf{x_I})$ is the output (*i.e.*, probability of presence) of a *pre-trained* classifier of the concept c applied on $\mathbf{x_I}$, a mid-level representation of image I. Semantic representations usually exploit all classifier outputs of the set of considered categories $\mathcal{C}$ [252], leading to a dense representation of images. However, [81, 112] showed that forcing the lowest values to zero improves the performance of these representations in image retrieval. Such a sparsification process is also interesting for image classification, since the compactness of the representations improves the computational efficiency at testing time. However, the setting of this exact amount of sparsity remains an open problem.

We propose a method to adapt the level of sparsity of these semantic representations according to the actual content of *each* image. We consider that a given concept can be retained only if we are confident enough on its detection and if it has a significant contribution in the amount of information carried by the semantic features. From these hypotheses, we derive a method named "Content-Based Sparsity" (CBS) that automatically determines an appropriate level of sparsity for each image independently, taking into account its actual visual content. CBS is presented in details in Sec. 3.1.2.1.

Secondly, we also address the problem of analyzing the content of the images at a local scale for semantic features. Former works such as Object Bank [115] encoded the spatial location of objects using a pyramid representation. More precisely they concatenated the base classifier outputs computed at three different scales. Such an approach is feasible when one uses a small number of detectors (200 in [115]) but becomes intractable when tens of thousand detectors are used [81, 21]. We propose an alternative solution, named "Constrained Local Semantic Features" (CLSF), that is inspired by a popular scheme in the domain of bag-of-features [92, 135] and CNNs [34, 96], namely pooling of local regions. While not new in itself, it is, to the best of our knowledge, the first attempt to use it in the context of semantic features. An advantage in comparison to the concatenation-based approaches is that the pooling does not change the size of the representation. Moreover, the use of a pooling in the following of our CBS scheme is a convenient way to focus only on important information. In fact, while, in a global scheme, CBS has the ability to sparsify the semantic representation regarding the content of each image, in a local scheme, it has the ability to neglect non-informative regions and consider highly informative ones, which is a desirable property, since it avoids to introduce noisy information in the final representation. CLSF is presented in details in Sec. 3.1.2.2.

We validate our work (in Sec. 3.2.3) in the context of mono-modal and multi-modal documents. First, we focus on images only and conduct experiments on four publicly available benchmarks, focusing on a scene classification task with the MIT Indoor benchmark and multi-class object classification with Pascal VOC 2007, Pascal VOC 2012 and Nus-Wide Object datasets. Results show that the proposed CLSF approach achieves state-of-the-art performances on three benchmarks (VOC 2007, Nus-Wide Object and MIT Indoor) and obtains competitive results on Pascal VOC 2012 compared to the best approaches of the literature.

3.1.2 Proposed Method

3.1.2.1 Content-Based Sparsity

If one retains a fixed number of concepts in a semantic representation, it can lead to an incomplete description for an image representing a lot of objects (Fig. 3.1, left image). In the same vein, this description is likely to be over-complete for images that contain too few objects (Fig. 3.1, right image). Thus, in this section, we propose a new sparsification strategy for semantic representations that is based on two hypotheses regarding the sparsity-level, (i) it should be adapted to the number of objects contained in the image and (ii) it should also depend on the confidence one has on the concept

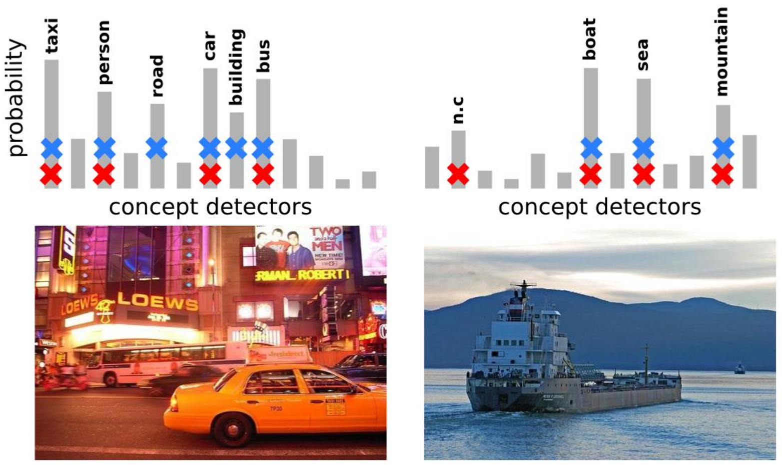

Figure 3.1: Illustration of the CBS method and of the interest to adapt the sparsity-level of semantic representations to the content of each image. Two different images are presented here, the left one contains a lot of objects while the right one contains much less. On top of the images, we illustrate their associated semantic features with gray bars representing the activation of the concepts on the image. Concepts selected by fixed sparsity-level (with a level $d = 4$) are marked with a red cross and those selected by our CBS method are marked with a blue cross. We clearly observe that the fixed sparsity-level scheme selects noisy-concepts (n.c) for the right image and misses useful concepts (*road* and *building*) for the left one. More properly, the proposed CBS scheme adapts the number of selected concepts and keeps an adequate large number for the left image and a lower one for the right image. Best view in color.

detection itself, reflecting its potential quality. These hypotheses are illustrated on Figure 3.2 in the context of our proposal. It represents four images and their "semantic feature profile" that is their semantic features with the values sorted in decreasing order. Such a profile allows to better visualize the relative values of a semantic representation.

The first hypothesis insures that images containing lot of objects (left image of Fig. 3.1) have a semantic representation with more non-zero values than that of images with few objects (right image of Fig. 3.1). The second one insures to keep only visual concept detection with high quality. For instance, for a low confidence on concept detection (bottom of Fig. 3.2), only few concepts would probably be relevant and should be retained. At the opposite, when the confidence on detection is high (top-right of Fig. 3.2), one can assume that a larger number of concepts would be beneficial to represent the image and thus we should retain a large number of them. Sparse coding of features has been a lot investigated in the literature, but it is usually posed as a desirable property in itself. Our approach is new in the sense that sparsification is a *consequence* of an adaptation of the semantic representation to the (assumed) interesting properties of the actual content of the each image.

When the two hypotheses are combined, one can consider that four semantic feature profiles remain. An illustration of the profiles is depicted in Figure 3.2. The profiles on the left correspond to images with few visual concepts. They are typically sparse profiles, characterized by a small number of dominating values with high confidence. On the contrary, the right hand side of the figure exhibits flatter *distributed* profiles, resulting from the presence of many visual concepts in the image. In addition, the value of the largest concept (the most left hand value of each profile in Figure 3.2) is a piece of information with regards to the confidence that can be attributed to the quality of the detection. The confidence is high in top profiles (due to the presence of well-recognizable objects)

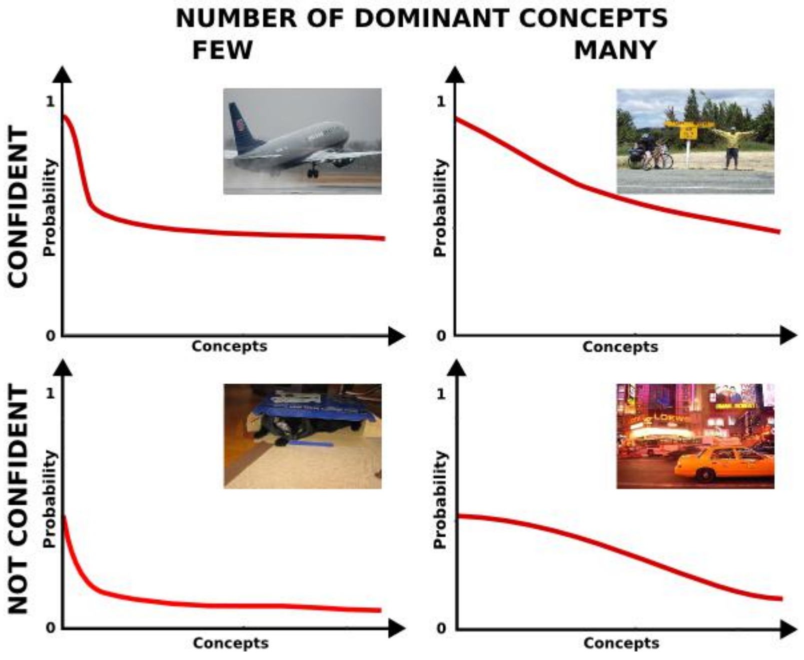

Figure 3.2: Illustration of the four prototypical semantic representation profiles, from which each configuration illustrates an image with its associated raw semantic representation. Note that the dimensions $\phi_c^{raw}(\mathbf{x})$ are sorted by decreasing values. The top profiles illustrate high confidence detection while the bottom ones low confidence ones. The left graphics correspond to the schemes containing few dominant concepts, while the right ones, depict schemes with a lot of dominant concepts.

and is low in bottom ones (where objects are occluded or in the middle of a cluttered background). If the two hypotheses above are considered, we should retain an adequate large number of concepts only for the top-right profile of Figure 3.2, when there are many dominant concepts and we are confident on their detection, while the other three cases should lead to retain few concepts only.

Let $\zeta^{raw}(\mathbf{x}_I) = [\phi_{c_1}^{raw}(\mathbf{x}_I), \ldots, \phi_{c_C}^{raw}(\mathbf{x}_I)]$ be the raw semantic feature. With a fixed level of sparsity (Equation (2.1)), our hypotheses are not taken into account. To do so, we propose to adapt sparsity to image content using the following thresholding scheme:

$$\phi_c^{CBS}(\mathbf{x}) = \begin{cases} \phi_{c_i}^{raw}(\mathbf{x}) & \text{if } \phi_{c_i}^{raw}(\mathbf{x}_I) \geq \Gamma(\mathbf{x}_I) \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

where $\Gamma(.)$ is a threshold that depends on the semantic feature profile of each image.

Each dimension of $\phi_{c_i}^{raw}(\mathbf{x}_I)$ is a piece of information related to the content of the image. The raw semantic signature can thus be considered as a source of information on the image semantic content. The Shannon entropy being the average amount of information generated by a source, it reflects the quantity of information conveyed by the raw semantic feature. We normalize ζ^{raw} to consider it as a random variable, then for an image I from which a mid-level feature $\mathbf{x}_I$ was extracted:

$$H(\zeta^{raw}(\mathbf{x}_I)) = \sum_{i=1}^{C} \frac{\phi_{c_i}^{raw}(\mathbf{x}_I)}{\sum_{j=1}^{C} \phi_{c_j}^{raw}(\mathbf{x}_I)} \log_2 \left(\frac{\sum_{j=1}^{C} \phi_{c_j}^{raw}(\mathbf{x}_I)}{\phi_{c_i}^{raw}(\mathbf{x}_I)} \right). \tag{3.2}$$

In that context, $\frac{\phi_{c_i}^{raw}(\mathbf{x}_I)}{\sum_{j=1}^{C}\phi_{c_j}^{raw}(\mathbf{x}_I)}$ is the probability that concept c_i is identified in I. Hence, the normalized semantic feature is the probability mass function of a discrete random variable whose values reflect the presence of object/concepts into the image. As a consequence, Equation (3.2) expresses the Shannon entropy of this source of information. The entropy is low for distributed profiles and high when the number of dominant concepts is small. Thus, the threshold $\Gamma(.)$ should decrease with respect to $H(\mathbf{x}_I)$.

To take into account the detection confidence, we consider the value of the largest semantic dimension of the profile, noted $\max_\zeta = \arg\max_{\phi_{c_i}}(\zeta^{raw}(\mathbf{x}_I))$, as the proportion of concepts to retain among the C available. Let us note that since the values are the outputs of binary classifiers normalized by a sigmoid, they are normalized in $[0, 1]$. For instance, if a profile is almost flat with the maximum confidence value around 0.5, we keep half of them and force the others to zero. With an almost flat profile and a confidence around 0.75 we retain the three quarters of them. In accordance to our hypothesis, the number of dimensions retained increases with the confidence on the detection. Hence, the threshold $\Gamma(.)$ should increase with respect to $\max_\zeta \times C$.

Finally, the threshold can thus be estimated by:

$$\Gamma(\zeta^{raw}(\mathbf{x}_I)) = \alpha \times \frac{\max_{\zeta^{raw}(\mathbf{x}_I)} \times C}{H(\zeta^{raw}(\mathbf{x}_I))}, \tag{3.3}$$

where $H(\zeta^{raw}(\mathbf{x}_I))$ is computed according to Equation (3.2), $\alpha \in [0, 1]$ is a normalizing parameter that can be determined by cross-validation, C is the total number of visual concepts that are considered in the semantic representation and the "confidence parameter" $\max_{\zeta^{raw}(\mathbf{x}_I)}$ is the largest of them. To better understand the variation of the threshold of Equation (3.3) regarding the entropy and the confidence, an illustration is depicted in Figure 3.3 for the four prototypical semantic profiles (presented in Figure 3.2).

3.1.2.2 Constrained Local Semantic Features

Recognizing objects that are present in small locations of an image (which is mostly the case in real-world images) is very hard when using mid-level features extracted from a CNN [34, 209]. This phenomenon is more accentuated with semantic-features. However, it is well established that extracting information from local regions and adding them to those of the global image, improves drastically the final performances on classification benchmarks [34, 228, 209].

Thus, in this section, we exploit local regions in the context of semantic features, and propose to combine this process with the CBS method (Section 3.1.2) in order to *better reflect* the semantics at a local scale. Consider a set of N regions $\{\mathcal{R}_i\}_{i=1...N}^{I}$ that have been identified into an image I. From each region, we extract a mid-level feature $\mathbf{x}_{R_i}$ then compute the corresponding semantic representation $\zeta^{d_{\mathcal{R}_i}}(\mathbf{x}_{R_i})$ with a level of sparsity $d_{\mathcal{R}_i}$. Then, all these local semantic representations are pooled to get the final representation of the image:

$$\zeta(\{\mathbf{x}_{\mathcal{R}_i}\}_{i=1...N}) = \underset{i=1...N}{\mathcal{P}}(\zeta^{d_{\mathcal{R}_i}}(\mathbf{x}_{\mathcal{R}_i})), \tag{3.4}$$

where $\mathcal{P}$ can be any pooling operator (max or sum) that operates on individual components (dimensions) of semantic representations of image regions. This local scheme can be applied to a semantic

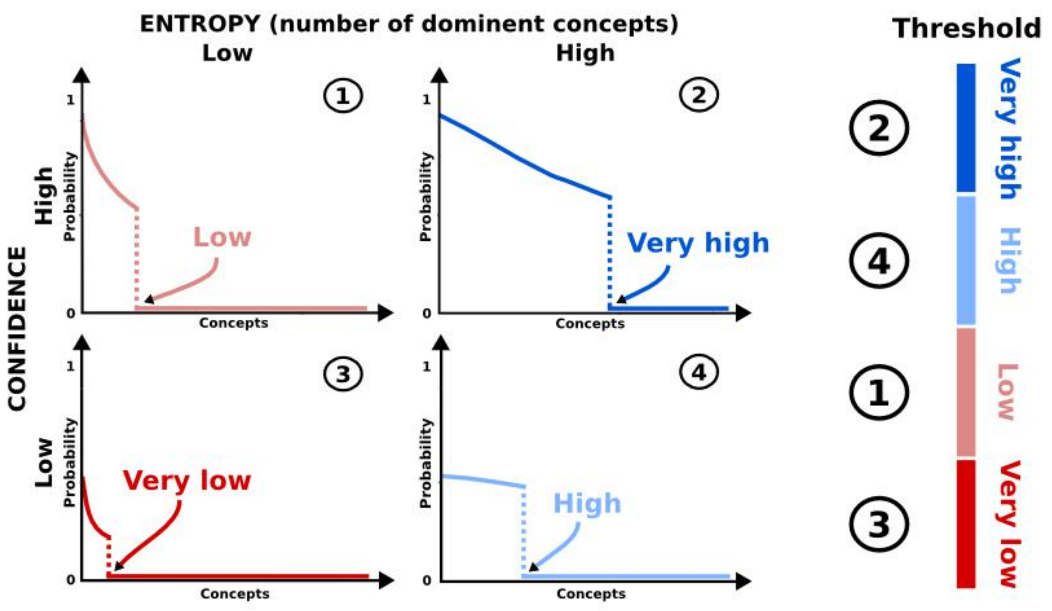

Figure 3.3: Illustration of the threshold computed by Equation (3.3) on the four different profiles of Figure 3.2. Each profile (on the left) corresponding to a particular combination of entropy and confidence, is associated to a number, and is ranged in the threshold axes (on the right). Best view in color.

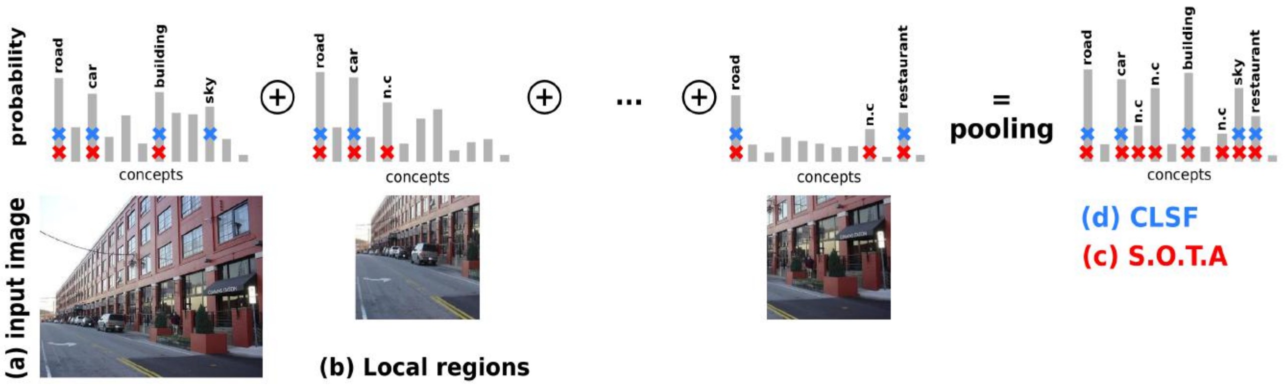

Figure 3.4: Illustration of our Constrained Local Semantic Features (CLSF) compared to common one (S.O.T.A). One input image (a) and its local regions (b) are presented here. On top of each image (input and local), we illustrate its associated semantic representation with gray bars representing the activation of the concepts in that image. Concepts selected in each image by a fixed-level scheme are marked with a red cross and those selected by CBS are marked with a blue cross. After the pooling operation, we obtain two final features (c) and (d) that have selected different concepts. We clearly observe that the representations of the fixed-level sparsity on local regions selected a lot of noisy concepts (n.c). It remains to the consideration of all the regions (informative or not) with the same intensity, which is sub-optimal. In contrast, our CLSF has selected concepts regarding the content of each region (through CBS) and thus selected only relevant concepts per region resulting in a more relevant output representation (d). Best view in color.

feature with a *fixed* sparsification-level [81] for *all* the regions ($d_{\mathcal{R}_i}$ fixed $\forall i \in [1, N]$), but it results into the assignment of the *same* number of visual concepts to *all* regions. Such an approach is sub-optimal since different regions in complex real-world images would contain very heterogeneous information (*e.g*, some regions contain a lot of information while other may contain much less). To determine which ones should be put into relief, we would like to assign a weight to *each* region. Actually, it is exactly what CBS does when it computes the level of sparsity $d_{\mathcal{R}_i}$ used in Equation (3.4) on each regions before the pooling. It automatically introduces a *constraint* at a local scale, that serves as a weight reflecting the level of information of the region. It results into our main contribution namely "Constrained Local Semantic Features" (CLSF), formulated by:

$$\zeta(\{\mathbf{x}_{\mathcal{R}_i}\}_{i=1\ldots N}) = \underset{i=1\ldots N}{\mathcal{P}}(\zeta^{CBS}(\mathbf{x}_{\mathcal{R}_i})). \tag{3.5}$$

The sparsification of *each* region is thus computed by the proposed content-based sparsity defined by Equations (3.1) and (3.3), resulting in the assignment of a sparsification-level $d_{\mathcal{R}_i}$ for *each* region $\mathcal{R}_i$. An illustration of our constrained local scheme compared to the common one is depicted in Figure 3.4.

The performance of the proposed constrained local scheme is quite correlated to the used region-detector – that determines the regions $\mathcal{R}_i$. In [245], we proposed a simple strategy, inspired by SPP [96], consisting in extracting the full image as region $\mathcal{R}_0$ and choosing following $\mathcal{R}_{i>0}$ according to regular grid at a smaller scale. The rectangular regions overlap such that, even if an object is cut, a more significant part will be present in at least one region (than in the case of SPP with contiguous regions). As an extension to our previous work [245], here, we also evaluate the robustness of our method to different region-detectors of the literature.

3.2 Diverse Concept-Level Semantic Features

3.2.1 Introduction

Most semantic representations in the literature [22, 81, 112, 115, 252] consider visual concepts independently from each other whereas they are often linked together by some (semantic) relationships (*i.e.*, hyponymy, hypernymy, exclusion, etc.). An exception is the work of Bergamo and Torresani [21] that introduces "meta-classes" to address this aspect. Those meta-classes are "abstract" categories (*i.e.*, do not really exist in the real-world) that capture common properties among many object classes. They are built using spectral clustering on low-level features of images among a set of categories. The restrictive assumption of this method is the dependence of the meta-class learning to the visual low-level features. For instance, it leads to irrelevant meta-classes when low-level features fail to capture the dissimilarity between different categories, making this method a "bottom-up" scheme.

The classical formulation of semantic features exploits all classifier outputs [21, 22, 115, 252], but as described in the previous secttion (3.1.2), forcing the semantic representation to be sparse (by setting the lowest values to zero) can be beneficial both in terms of scalability and performance [81, 112]. Nevertheless, semantic representations with a large set of concept detectors often contain a high number of visually similar concepts to describe the same object. For instance, the right image of Figure 3.5 would be predicted by using a semantic feature as a *palm cockatoo*, but also a *cockatoo*, a *parrot*, a *bird*, a *vertebrate*, and so on, inducing redundant information in the final representation. As far as a human is concerned, he would categorized this image as *a bird*, an *animal* and maybe a *palm cockatoo*, if the human is a bird-expert. In fact, cognitive studies such as those of Rosch [214] and Kosslyn [118] showed that a human tends to categorize an object through three categorical-levels (i) basic-level, (ii) superordinate, and (iii) subordinate. As depicted in Sec. 2.2.5, they are the most important concept types used to categorize objects.

In this contribution, we take into account the relations between concepts using human existing knowledge, such as semantic hierarchies (*e.g.*, WordNet [168]), which makes our approach a "top-down" scheme. More precisely, our main contribution consists in identifying three types of concepts into an existing hierarchy, according to their categorical level, then processes them differently to design the semantic representation. It is nevertheless not easy to determine to which categorical-level a concept belongs to. Hence, we propose a method to identify the three groups in practice, for a given supervised classification problem. The proposed semantic representation is named Diverse Concept-Level features and is denoted **D-CL**.

Compared to bottom-up approaches, an advantage of the proposed top-down scheme appears when the concept detector fails at the subordinate level (*e.g.* the concepts *cockatoo* and *parakeet* are highly activated), which is often the case since the category is finer thus harder to identify. In that case, our descriptor at least captures basic-level and superordinate concepts (*e.g.* bird and animal), making the full representation more robust for classification problems. Moreover, the proposed representation contains only useful concepts (from the three categorical levels), which avoids redundant information that disturbs the image classification. The whole method is described in details in Sec. 3.2.2.

We validate the proposed D-CL representation (in Sec. 3.2.3), in a multi-object classification task through Pascal VOC 2007, Pascal VOC 2012 and Nus-Wide Object. The experiments show that the proposed approach obtains better results than seven state-of-the-art semantic representations.

3.2.2 Proposed Method

3.2.2.1 Diverse Concept-Level Features

We recall that a semantic representation is a C-dimensional vector $\zeta(\mathbf{x}_I) = [\zeta_1(\mathbf{x}_I), \dots, \zeta_C(\mathbf{x}_I)]$ extracted from an image I, itself described by a mid-level feature $\mathbf{x}_I$. The feature $\mathbf{x}_I$ could be any image descriptor such as Bag-of-Word or Fisher Kernel [197] features, but also mid-level features such as that obtained from a fully-connected layer of a convolutional neural network. Each dimension $\zeta_i(\mathbf{x}_I)$ of the semantic representation is the output of a classifier for the concept c_i evaluated on $\mathbf{x}_I$.

While the concepts c_i are potentially linked together by some semantic relationships, most of works consider them independently [22, 81, 112, 115, 252]. A notable exception is the work of Bergamo and Torresani [21] that takes into account relations between categories through a "bottom-up" scheme. However, their method can lead to irrelevant identification of relations when the low/mid-level features used fail to capture the dissimilarity between different categories. To cope with such a limitation, we propose to rely on existing human knowledge regarding the relations between concepts. Such a knowledge is, for instance, reflected into existing hierarchies such as WordNet [168] that organizes a large set of concepts according to "is-a" relationships, that is to say by defining hyponyms and hypernyms. An advantage of our approach is to remove the dependences to the basic visual descriptor and to introduce human-based information within the process of image representation design.

All the concepts considered in semantic features are named according to existing *categories*. Once again, the *name* of a category is given according to a human judgment, and the exact choice of the word used is far from being neutral, as a large literature has shown it, both in Psychology [118, 214] and Computer Vision [54, 163, 189]. More precisely, they showed the importance to differentiate several levels of categories:

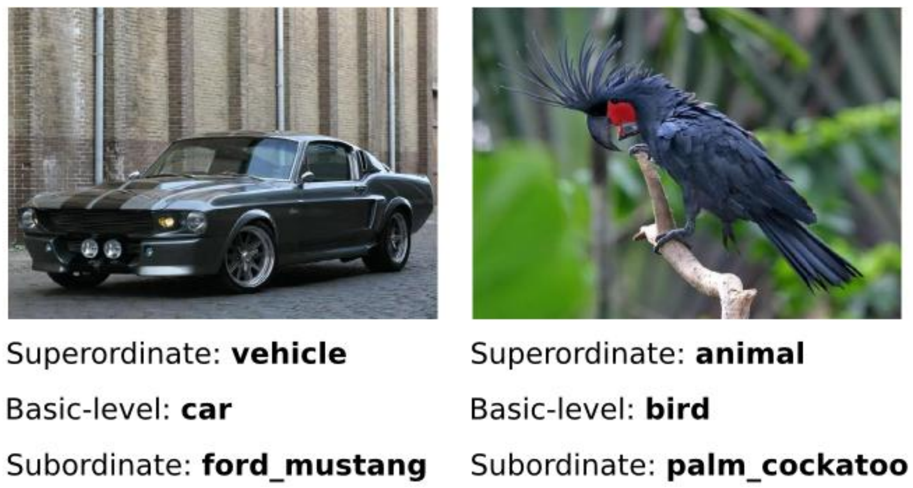

Figure 3.5: Illustration of concepts that our D-CL representation would predict, for two different images. It select concepts from different categorical levels of a semantic hierarchy, *i.e.*, superordinate, basic-level and subordinate concepts.

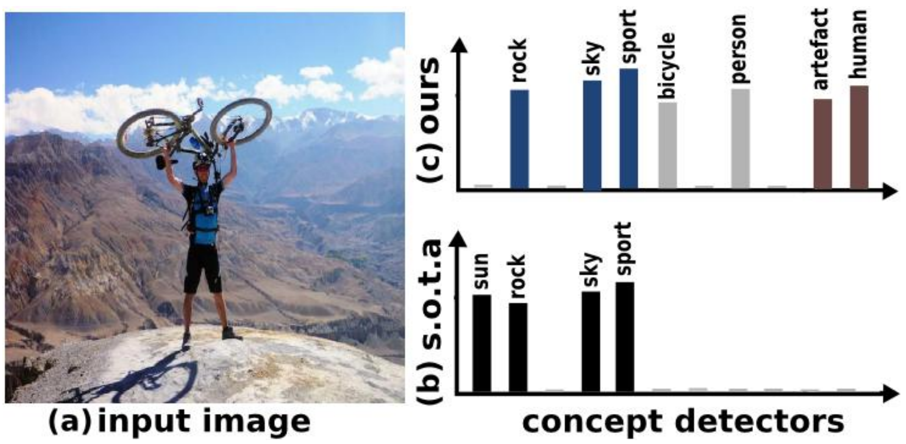

Figure 3.6: We propose a semantic representation that computes the concepts presence differently according to their categorical level. For an input image (a) with multiple objects, state-of-the-art semantic features (b) would output the concepts illustrated in black and miss useful concepts, such as person and bicycle. In contrast, the proposed scheme (c) captures properties of the image that are useful for categorization, *e.g.* *superordinate* (brown), *basic-level* (gray) and *subordinate* (blue) concepts, making the representation more relevant. Best viewed in color.

- **Basic-level concepts** are the terms at which most people tend naturally to categorize objects, usually neither the most specific nor the most general available category but the one with the most distinctive attributes of the concept.
- **Superordinate concepts** are categories placed at the top of a semantic hierarchy and thus displays a high degree of class inclusion and a high degree of generality. They include basic-level and subordinate concepts.
- **Subordinate concepts** are found at the bottom of a semantic hierarchy and display a low degree of class inclusion and generality. As hyponyms of basic-level concepts, subordinate categories are highly specific.

At the core of our proposal to design a feature representation, concepts are processed differently according to their categorical level. This asymmetrical process is based on a cognitive study proposed by [118] where they conclude that, concepts are processed differently by humans, *i.e.*, it is purely perceptual for the basic-level and subordinate concepts, while it is inferred using stored semantic information, for superordinate concepts. In our scheme, basic-level and subordinate concepts are computed through a visual process, while superordinate concepts are processed semantically using the hyponym relations between concepts into hierarchies. Figure 3.5 illustrates for two input images, the three types of concepts that would be retained by our scheme.

More precisely, for an input image I, the probability of a basic-level or a subordinate concept is the output of a binary classifier ($\phi_i^V(\mathbf{x}_I)$) for the concept c_i evaluated on the mid-level feature $\mathbf{x}_I$, further normalized by a sigmoid function such that $0 < \phi_{c_i}^V(\cdot) < 1$. The binary classifiers, that we name *visual classifiers* in the following, have been learned using images of the concept c_i as positive samples and images of a diversified class as negative samples. Each concept classification model $\phi_{c_i}^V(\cdot)$ is obtained with L_2-regularized linear SVM, but other linear models could be used. Regarding the process of basic-level and subordinate concepts, even if it is similar, a particular difference is that, all basic-level concepts are selected in the final representation, while for subordinate concepts, we select only the most salients. This particular process for subordinate concepts avoids redundancy of information, due to the fact pointed in [214] that there is more concepts at a subordinate level than at the basic-level.

Concepts (c_i) at the highest categorical level (superordinate) are computed, for an input image, through a *semantic classifier*. It is an inference of concepts that have at least one hyponym relation with the superordinate concept (c_i). We thus define the subsumption function that aims to output the set of concepts having hyponym relations with an input concept. We further, define the semantic classifiers that are used to compute superordinate concepts.

Definition 1. A subsumption function $\varsigma(\cdot)$ takes as input a concept c_i and a semantic hierarchy $\mathcal{H}$ with hyponymy relations and outputs a set $\mathcal{C}_i$ of concepts that are subsumed by the concept c_i, *i.e.*, the concepts that have an hyponymy relation with the concept c_i in the semantic hierarchy $\mathcal{H}$.

Definition 2. Considering $\mathbf{x}_I \in \mathbb{R}^N$ a N-dimensional mid-level representation extracted from an image I. A semantic classifier is an operator that predicts the probability of presence of a concept c_i in the image through a semantic inference of purely visual output classifiers: $\phi_{c_i}^S(\mathbf{x}_I, \mathcal{C}_i) = max(\phi_{\mathcal{C}_1}^V(\mathbf{x}_I), \cdots, \phi_{\mathcal{C}_M}^V(\mathbf{x}_I))$, where $\mathcal{C}_i$ is the set of concepts subsumed by concept c_i, $M = card(\mathcal{C}_i)$ and $\phi^V(\cdot)$ is the output values given by visual classifiers.

Finally, the proposed "Diverse Concept-Level" (D-CL) representation computes superordinate concepts through a semantic classifier and all other concepts, *i.e.* basic-levels and subordinates, using visual classifier. It also selects all basic-level and superordinate concepts and retains only the most salient subordinate concepts. Formally, let us denote $\mathcal{N}$ the set of all concepts associated to a semantic hierarchy, $\mathcal{BL}$ the set of all basic-level concepts, $\mathcal{P}$ the set of superordinate concepts, $\mathcal{B}$ the set of subordinate concepts and $\mathcal{B}^K$ the set of the K most salient subordinate concepts for each input image. Note that, $\mathcal{N} = \mathcal{P} \cup \mathcal{BL} \cup \mathcal{B}$. Each dimension $\zeta_i^{\text{DCL}}(\mathbf{x}_I)$ of the D-CL representation $\zeta^{\text{DCL}}(\mathbf{x}_I)$ is a concept detector computed through:

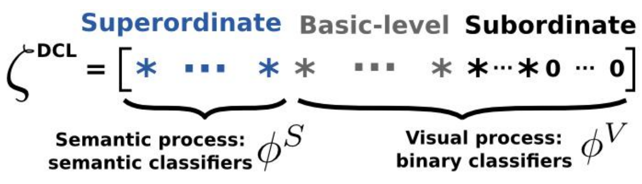
Superordinate
Basic-level
Subordinate
$\zeta^{DCL} = [* \cdots * * \cdots * * \cdots * 0 \cdots 0]$
Semantic process: semantic classifiers ϕ^S
Visual process: binary classifiers ϕ^V

Figure 3.7: Illustration of the asymmetric process in our representation. Superordinate concepts are processed semantically through semantic classifiers, while basic-level and subordinate concepts are visually processed through binary classifiers. Stars and zeros represent output values $\zeta_i(\cdot)$ of each concept of the D-CL representation $\zeta(\cdot)$. Note that, concepts are grouped by categorical levels, but any order could be obtained in a real scheme.

$$\zeta_i^{\text{DCL}}(\mathbf{x}_I) = \begin{cases} \phi_{c_i}^S(\mathbf{x}_I, \varsigma(c_i)), & \text{if } c_i \in \mathcal{P} \\ \phi_{c_i}^V(\mathbf{x}_I), & \text{if } c_i \in \mathcal{BL} \cup \mathcal{B}^K \\ 0 & \text{if } c_i \in \mathcal{B} \setminus \mathcal{B}^K \end{cases} \tag{3.6}$$

where $\varsigma(\cdot)$ is the subsumption function, $\phi_{c_i}^V(\cdot)$ the visual classifier, $\phi_{c_i}^S(\cdot)$ the semantic classifier and K is a parameter corresponding to the number of subordinate concepts retained in the representation, that can be set by cross-validation. An illustration of the asymmetric process according to the type of concepts is presented in Figure 3.7.

3.2.2.2 Identifying Concept Groups in Practice

In this section, we detail how to identify the three groups of concepts (*i.e.*, basic-level, superordinate and subordinate), in practice, for a given supervised classification problem.

As depicted in Equation (3.6), the D-CL representation is computed by activating all the basic-level concepts, all the superordinate concepts, the K most salient subordinate concepts and by deactivating all others. Let $\zeta^{\text{DCL}}(\mathbf{x}_I)$ be the D-CL representation of a mid-level representation $\mathbf{x}_I$ extracted for an image I contained in a targeted dataset. Let $\mathcal{D}^d$ be the set of d categories of the targeted dataset.

While basic-level concepts are not available at a large scale, we propose to identify, in an off-line phase, the set of basic-level concepts ($\mathcal{BL}$) selected in our D-CL representation by matching it with the set of targeted dataset categories $\mathcal{D}^d$. This latter, is based on the assumption that broader-datasets mostly contain categories at the basic-level. Specifically, all targeted dataset categories d_i are matched with a concept c_i from $\mathcal{N}$ to generate a set of basic-level concepts adapted to the dataset $\mathcal{BL}^d$. In fact, this matching has the advantage to make our D-CL representation adaptable to the application context. Regarding the sets of superordinate $\mathcal{P}$ and most salient subordinate $\mathcal{B}^K$ concepts, they are therefore automatically selected through the subsumption function $\varsigma(\cdot)$ that takes as input concepts from $\mathcal{BL}^d$ and a semantic hierarchy $\mathcal{H}$ with "is-a" relations. Formally, the Equation (3.6) becomes:

$$\zeta_i^{\text{DCL}}(\mathbf{x}_I) = \begin{cases} \phi_{c_i}^S(\mathbf{x}_I, \varsigma(c_i)), & \text{if } c_i \in \mathcal{P}^d \\ \phi_{c_i}^V(\mathbf{x}_I), & \text{if } c_i \in \mathcal{BL}^d \cup \mathcal{B}^K \\ 0 & \text{if } c_i \in \mathcal{B} \setminus \mathcal{B}^K \end{cases} \tag{3.7}$$

where $\mathcal{BL}^d$ and $\mathcal{P}^d$ are, respectively, the set of basic-level and superordinate concepts adapted to the

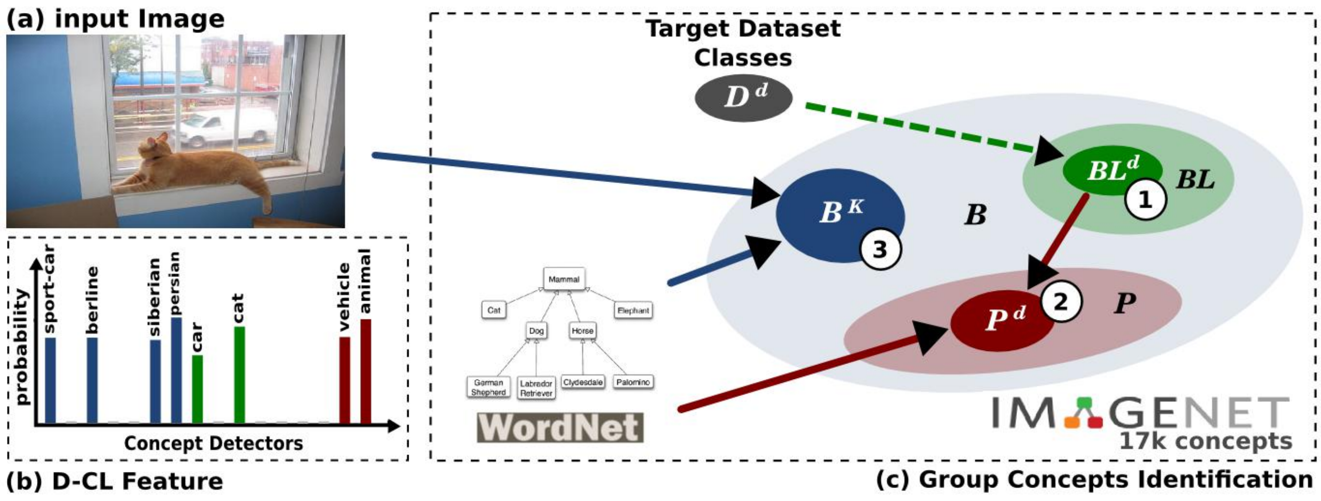

Figure 3.8: Illustration of the concept groups identification (c) in a practical case, for an input image (a) contained in a dataset collection. The proposed concept groups identification selects (1) in an off-line phase (dashed arrow), the concepts of the target dataset categories ($\mathcal{D}^d$) as a portion ($\mathcal{BL}^d$) of all basic-level concepts ($\mathcal{BL}$), (2) the part ($\mathcal{P}^d$) of its superordinate concepts ($\mathcal{P}$) and in a final step (3) the most salient ($\mathcal{B}^K$) subordinate concepts ($\mathcal{B}$). For steps (2) and (3), a semantic hierarchy (WordNet) is used to compute the hyponymy relations. This latter results in the final D-CL representation (b), to an activation of diverse concept levels (*i.e.*, superordinate, basic-level and subordinate) and a deactivation of all other concepts.

targeted dataset $\mathcal{D}^d$. Selecting a portion of the whole concepts, and setting others to zero is closely related to the sparsification process that sets to zero the lowest output values and keeps activated only the other concepts. The key novelty of our work is the adaptability of the concept selection to the input images. Contrary to former work, the sparsity is adapted to each image, according to its actual content, and relative to the problem of interest.

Our D-CL feature is illustrated in Figure 3.8. It is able to capture from an image containing multiple objects, all the basic-level concepts (colored in dark green) adapted to the target dataset, all its super-ordinate concepts (colored in dark red) and the most salient subordinate ones (colored in dark blue). It results in a final representation capturing the most informative concepts for a target collection of images.

3.2.3 Experimental Results

In this section we compare our two contributions to each other and we also compare them to the state-of-the-art methods. More precisely, we first describe (in Sec. 3.2.4) the baseline methods used for comparison and we precise the datasets on which we perform the comparisons, then, we give the results of all the methods (ours, baselines and state-of-the-art methods) in image classification (in Sec. 3.2.4.1) and image retrieval (in Sec. 3.2.4.2). Finally, we discuss (in Sec. 3.3.1) the impact of each contribution and the differences between them. We also mention their possible complementaries and provide the most natural way to combine them.

3.2.4 Settings

Since both contributions deal with semantic representations, we will mainly compare them to semantic-based methods of the literature. However, since they are build on top of CNN features, we will also conduct a comparison to CNN-based features. The following items describe the different baseline methods used for comparison:

- **VGG** [228] corresponds to the penultimate fully-connected layer ($fc7$) extracted from a CNN pre-trained on ILSVRC [216] (1.2 million images labeled among $1,000$ categories). It results in a $4,096$ dimensional vector.
- **VGG [fc8]** corresponds to the extraction of the last layer ($fc8$) of the pre-trained VGG model. Since each dimension is associated to a semantic connotation, the representation corresponds to a semantic representation built on top of the fc7 layer with $1,000$ concept-detectors. The dimensionality of the vector is $1,000$.
- **Semfeat** [81] is a semantic-feature trained on top of CNN features (fc7) extracted from the Caffe network [114]. For fair comparisons, we rebuilt it using the fc7 layer of VGG as basic features. As in the original work, we use $17,462$ concept-detectors corresponding to classifiers trained on categories of ImageNet that contain at least 100 images. As in their work, a fixed sparsification is modeled here.
- **Classemes+** is our own implementation of Classemes [252]. While in the original work they use many hand-crafted features, here we use the fc7 VGG features. This latter, insures a fair comparison with the different methods. To correctly replicate their methodology, we do not apply any sparsification.
- **Meta-Class** [21] is one of the prior semantic representation. It corresponds to the output probabilities of $15,232$ concept-detectors. The representation of images consists of five low-level features concatenated after the appliance of a spatial pyramid histogram with 13 pyramid levels. Since the number of concepts is closely similar to other methods and the code is available[1], we use it as it is released.

To extend the comparison, we also report released scores by other semantic-based approaches in the literature (ObjectBank [115], Picodes [22] and Classemes [252]).

Since our CLSF method models locality and all the baselines does not, we added two baseline methods that model local information. More precisely, we added VGG+FT+Local and Semfeat+Local that are both described in details in the two following items:

- **VGG+FT+Local** [34, 228] corresponds exactly to the work of [34] with the VGG architecture [228]. Hence, it is a VGG network fine-tuned on each target-dataset with data-augmentation (10 crops per image). On the train and test phases of the target-datasets, each image is augmented 10 times (center crop, four corners and the flip of each of them) and represented by the fine-tuned $fc7$ features. A pooling operation is applied to merge the local features.

[1]http://vlg.cs.dartmouth.edu/projects/metaclass/metaclass/Home.html

- **Semfeat+Local** corresponds to the best semantic features (Semfeat) for which we added the modelization of local information. Since, in the literature, there is no efficient use of local information in the context of semantic features, we modeled it with the proposed local scheme (presented in Sec. 3.1.2.2).

Note that, while all comparison CNN-based features (*i.e.*, VGG, VGG[fc8], VGG+FT+Local, [34, 186, 273, 228]) are learned on training images labeled among $1,000$ categories, the semantic-based ones (including our proposal based on them) considered in these experiments are learned on $17,000$ categories. Thus CNN and semantic-based features are not directly comparable. However, it should be noted that, the CNNs could hardly benefit from large available annotated data since it is often quite difficult to get the convergence with so many categories. Above all, adding a new concept in a CNN is more difficult since it requires to retrain all the model. On the contrary with semantic features, we are not limited by the number of categories and, more importantly, it can be learned in an incremental way (without re-training all other detectors), which is highly desirable to tend toward universality.

Regarding the evaluation of the different methods, it is carried in a transfer-learning scheme with four publicly available datasets used as target-tasks. More precisely, we used Pascal VOC 2007 [71], Pascal VOC 2012 [70], Nus-Wide Object (N-W Ob.) [38] and MIT Indoor 67 (MIT-67) [205]. We perform image classification and retrieval and we precise in their associated sections which datasets and evaluation-metrics we used. In all cases, for reproducibility and comparability reasons, our evaluation is conducted with standard experimental protocols that are described for all the datasets in Table 4.2.

3.2.4.1 Image Classification Results

In this section we evaluate our both contributions on some of the datasets presented above in an image classification task. As stated in the introduction of the Thesis, the goal here is to evaluate the increase of universality compared to a reference method, and more precisely here the comparison is carried with different image representations (including a reference one, namely Semfeat [81]) in a transfer-learning scheme. This latter consists to train a representation on a large auxiliary dataset and re-use it as image representation on many relatively small target-datasets. Hence, images of the target-datasets are represented by pre-learned image representations of the different methods. Each category of the target-dataset is learned with a one-versus-all SVM. A cross-validation process is used to estimate the SVM cost parameter and the α parameter of Equation (3.3). Some reasons of these choices are mentioned in the introduction of the Thesis.

Since our CLSF method models image locality, it is unfair to compare it with CBS and D-CL. Hence, we perform two comparison experiments: (i) comparison of all semantic representations that do not model locality and (ii) comparison of methods that model the locality aspect. For the first experiment, we compared our CBS and D-CL methods to all semantic representations presented above. For the second experiment, in addition to the baseline methods, we also compare our method (CLSF) with the best state-of-the-art ones.

The results for the global-based semantic-feature methods (first experiment) are presented in Table 3.2. Our both representations (CBS and D-CL) significantly outperform all the other semantic representations on the three datasets. The two methods are also significantly better than the reference Semfeat method (that models a top-K sparsity) which clearly highlights the interest of our structured sparsity. Moreover, compared to all the baselines, the improvements of the proposed CBS and D-CL

Method	**Nus-Wide Object (20%)**	**Pascal VOC 2007 (45%)**	**Pascal VOC 2012 (30%)**
ObjectBank [115]	n.a	45.2*	n.a
Classemes [252]	n.a	43.8*	n.a
Classemes+ [252]	70.3	82.4	81.7
Picodes [22]	n.a	43.7*	n.a
Meta-Class [21]	36.5	48.4 (53.2*)	49.3
VGG-16 (fc8) [228]	67.3	77.4	77.2
Semfeat [81]	74.7	82.8	81.7
CBS (ours)	n/a	**85.1**	**83.0**
D-CL (ours)	**76.0**	**85.1**	**83.0**

Table 3.2: Comparison of overall performances (mean Average Precision in %) of seven semantic representation baseline methods with our both methods (CBS and D-CL) that are highlighted in bold. The evaluation is carried on three datasets: Nus-Wide Object, Pascal VOC 2007 and Pascal VOC 2012. We mention, for each dataset (between brackets), the rate of images in the dataset labeled with multiple categories. Results marked with * are those reported in the original papers.

methods, are much better on Pascal VOC 2007 than on Pascal VOC 2012 which are themselves better than on Nus-Wide Object. This result is aligned with the expectation since Pascal VOC 2007 contains a larger part (45%) of images labeled by multiple categories, compared to Pascal VOC 2012 and Nus-Wide Object, that contain only 30% and 20%, respectively. Regarding this, the performances of our methods increase with the level of co-occurrence objects in the dataset. In summary, this experiment demonstrates that the association of our structured-sparsity with large semantic representation (*i.e.*, that contains a huge amount of object-detectors; 17,462 here), aims to take better advantage of the *largeness* of the representation than with a top-k sparsity [81].

The results for the second experiment (local-based methods) are presented in Table 3.3. Roughly, our method achieves better results compared to baseline methods and competitive results compared to the best state-of-the-art methods. More precisely, CLSF performs better than the VGG baseline [228] and all semantic representations that do not model local information. Moreover, it also significantly outperforms the methods that comparably model the locality aspect (VGG+FT+Local and Semfeat+Local). Note that, the fact that CLSF is better than Semfeat+Local clearly highlights the impact of the CBS method when applied on local regions. Compared to the best state-of-the-art methods, in the context of object recognition, we observe that the methods of Simonyan *et al.* [228] and Duran *et al.* [63] achieve very competitive results compared to ours. However, it is important to consider that in their methods, a large amount (*i.e.*, 64 for [63] and 250 for [228]) of local regions is extracted per image, while in our scheme, we only extract 6 regions per image. In the next paragraph, we discuss in more details the cost of the top-performing methods of the literature. The success of our method is due to the efficient combination of the three proposed contributions (CBS + Local Semantic + FTDG). Notice that our FTDG contribution roughly corresponds to a CNN-features better than VGG features and is presented in more details in the next chapter. An evaluation of the impact of each component of our proposal is conducted in Sec. 3.2.5.

It is important to note that, our results are not the best ones of the literature, since many works reported slightly better results. However, they use costly computational methods. In the following, we roughly compare the computational cost of their methods with ours. Indeed, in [228], they use a multi-scale pyramid pooling with five scales, that consists to extract 50 crops per scale, which

	Method	VOC 2007 mAP (in%)	VOC 2012 mAP (in%)	N-W Object mAP (in%)	MIT 67 Acc. (in%)
CNN	Oquab *et al.* [209]	$77.7^{\dagger}$	n/a	n/a	n/a
	Chatfield *et al.* [34]	$82.4^{\dagger}$	$83.2^{\dagger}$	n/a	n/a
	Wei *et al.* [273]	$81.5^{\dagger}$	$81.7^{\dagger}$	n/a	n/a
	Zhou *et al.* [297]	n/a	n/a	n/a	$70.8^{*\ddagger}$
	VGG [228]	86.1	84.5	71.3	65.8
	VGG+FT+Local [34, 228]	86.8	86.0	75.9	69.9
	Simonyan *et al.* [228]	$89.7^{\dagger *}$	$\mathbf{89.3}^{\dagger *}$	n/a	n/a
	Duran *et al.* [63]	90.2	88.5	n/a	n/a
Semantic	**VGG [fc8]** [228]	77.4	77.2	67.3	48.7
	Meta-Class [21]	48.4 ($53.2^{\dagger}$)	49.3	36.5	35.7 ($44.6^{\dagger}$)
	Classemes+ [252]	82.4	81.7	70.3	58.9
	Semfeat [81]	82.8	81.7	74.7	61.5
	Semfeat+Local	85.3	85.4	65.4	68.7
	CLSF (ours)	**90.4**	**88.6**	**76.6**	**74.1**

Table 3.3: Comparison of our method (denoted CLSF) with baselines (highlighted in bold) and state-of-the-art methods on four datasets (including three object recognition datasets and one scene classification dataset). We separate CNN-based methods (top) and semantic-based ones (bottom). Scores marked with * were achieved using a large amount of crops per image (250 for [228] and 64 for [63]). Those marked with † are scores released in the original papers and those marked with ‡ were achieved using a CNN trained on 3 million scenes.

results in the extraction of 250 crops per image, which is very costly. Moreover, they represent each crop by pooling the fc7 features extracted from two CNNs, one of 16 layers and one of 19. In contrast, we obtained 88.6 on VOC 2012 (versus 89.3 for [228]) using only six regions per image and an architecture of 16 layers, while we get better results on VOC 2007 (90.4 versus 89.7). Very recently, Yang *et al.* [281] achieves 92.0 and 90.6 in VOC-07 and VOC-12, respectively. However, they use the selective search algorithm [257] to extract $1,500$ regions-per-image. More restrictively, they use bounding box annotations during the learning phase, which makes it clearly incomparable to our approach. Less restrictively, Duran *et al.* [63] proposed a new deep-learning method that incorporates top instance and negative evidence insights using image labels only. They obtained top performances on many datasets (including 90.2 on VOC-07 and 88.5 on VOC-12). However, they represent images with the features obtained after the fine-tuning (on each target-dataset) of a CNN pre-trained on ILSVRC. They also model local information for their positive/negative instances selection by using a multi-scale scheme (8 scales) and 8 local regions per scale (4 positives and 4 negatives), resulting in 64 extractions per image. On our side, our proposal obtains slightly better results with much less cost computations on target-datasets, *i.e.*, transferring the pre-trained CNN features *without* fine-tuning and by modeling the local information with *only* five regions-per-image.

Regarding the results in scene recognition (last column of Table 3.3), our method has a similar behavior than in object recognition, that is to say, it outperforms all the baselines and the best state-of-the-art methods. The most competitive result is that of Zhou *et al.* [297] (70.8) with an Hybrid-CNN, corresponding to the combination of two CNNs, one learned for object recognition on the 1.2 million images of ILSVRC, and another learned for scene recognition on 3.5 million images labeled among $1,183$ scene categories. In contrast, our approach obtains significantly better results (74.1) while we use only one CNN model that has been trained on objects. Note that, while Duran *et al.* [63] also report results on MIT 67, here we do not compare our results to theirs since the MIT 67 dataset do not provide an official train/test split and they do not use the same as us. In fact, with the on-line code of VGG-16, they achieve an accuracy of 69.9 on their split while we obtain (with the same on-line code) 65.8 on our split, proving that they use an easier split than the one we use. In summary, the proposed method obtains very competitive results on Pascal VOC 2012 and achieves state-of-the-art performances on three benchmarks (Pascal VOC 2007, Nus-Wide Object and MIT Indoor 67) of image classification, pushing it to one of the best performing approaches of the literature.

3.2.4.2 Image Retrieval Results

In this section we compare the CLSF contribution with the best baseline methods in a context of image retrieval. Note that, D-CL is not considered in this comparison because the actual method to find the concept-groups uses the categories of the target-dataset and this is not applicable on the task of image retrieval (since it is not supposed to contain categories). So, the evaluation in the context of image retrieval is carried on three datasets and we adopt Average Precision at top K (AP@K) retrieved results. The final evaluation metric is the mAP@K that corresponds to the average of the AP@K over all the queries. $K \in \{1, 2, \cdots, 100\}$ for VOC-07 and N-W Ob. and $K \in \{1, 2, \cdots, 80\}$ for MIT 67, since only 80 positive-images are available per query. We use cosine similarity to compare the image representations.

Figures 3.9 gives mAP@K curves for our method, the best semantic representations of the literature (Classemes+ and Semfeat) and the best CNN-based features (VGG) on VOC-07, MIT-67 and N-W Ob., respectively. In contrast to image classification, it is in image retrieval context that we can observe the interest of semantic features, both, on the computational complexity (much lower than with CNN features) and on the performances that are slightly better than with CNN features. In addition to have the low computational complexity, the proposed semantic representation significantly outperforms all the methods (CNN and semantic based).

Semfeat forces a fixed level of sparsification that, consequently, deletes useful concepts and Classemes+ keeps all the dimensions of the representation, that remains to consider noisy concepts in the final feature. In contrast to these methods, ours applies an effective cooperation of local representations and a CBS that deletes non-informative concepts on the final features, which explains the above phenomena of performance.

3.2.5 In-Depth Analysis

Analysis of Constrained Local Semantic Features

In this section, we describe the implementation details of our method, then we conduct three experiments that highlight some insights of our method. More specifically, we evaluate the impact of each component of our method, then we systematically evaluate the correlation between the semantic representations and the mid-level ones used to build them and finally, we conduct an experiment that shows the robustness of our approach to the chosen region-detector.

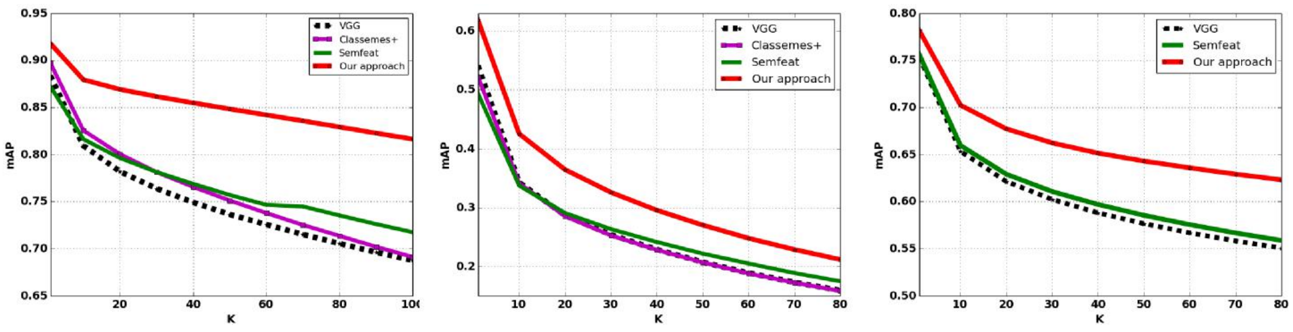

Figure 3.9: Results (mAP@K) of the proposed CLSF method on image retrieval through Pascal VOC 2007 (left), MIT Indoor 67 (middle) and Nus-Wide Object (right) datasets. Our method is compared to the two best semantic-based methods and the best CNN-feature of the literature, *e.g.*, Classemes+ [252] and Semfeat [81] for semantic-based methods and VGG [228] (in dashed line) for CNN-feature. Best view in color.

3.2.5.1 Implementation details

Semantic representation learning: To build our semantic representations, we followed the implementation of [81]. More precisely, the semantic representation is built using ImageNet [53] concepts that have at least 100 associated images, resulting in $C = 17,462$ concept-detectors. When learning the classifiers $\phi^{raw}_{c_i}(\mathbf{x}_I)$, we use images representing the concept as positive samples and images from a diversified class as negative ones. This negative set contains images of ImageNet-concepts not considered to build the semantic representation (*i.e.* the concepts are not in $\mathcal{C}$). We used a ratio of $1/100$ between positive and negative samples to learn the detectors, this ratio having been determined empirically as a good compromise between the resulting performance and the computational complexity. To learn each concept-detector $\phi^{raw}_{c_i}$, we use L_2-regularized linear SVM.

CNN-Features: The proposed method can be applied to a semantic signature built on top of any mid-level representations. However, the quality of the semantic representation directly depends on that of the mid-level ones used. We thus created semantic representations on top of a competitive mid-level CNN representation released in the literature, namely VGG-Net [228]. More precisely, the CNN representation used corresponds to the extraction of the penultimate fully-connected layer ($fc7$ layer) from a CNN learned on ILSVRC 2012 dataset [216] (containing 1.2 million images labeled among $1,000$ output categories), which results in $4,096$ dimensional vectors.

Region-Detectors: Here, we proposed a simple strategy inspired by SPP [96], consisting to extract the full image as region $\mathcal{R}_0$ and choose following $\mathcal{R}_{i>0}$ according to a regular grid (four corners and the center) at a smaller scale ($2/3$ of the size of the original image). The rectangular regions overlap, such that even if an object is cut, a more significant part will be present in at least one region (than in the case of SPP with contiguous regions).

3.2.5.2 Impact of Each Component of Our Method

In this section, we evaluate the impact of each component of the proposed approach and compare it to the best semantic baseline (*i.e*, Semfeat [81]). To do so, we first isolate each component of our approach and evaluate the performances of each of them and their different combinations. More

Method	CBS	SPP	FTDG	VOC-07	MIT-67
Semfeat [81]	✗	✗	✗	82.8	61.1
CSF	✓			85.1	63.6
LSF		✓		85.3	68.7
SF-FTDG			✓	88.1	71.7
CLSF	✓	✓		88.2	71.6
CLSF-FTDG (CLSF+)	✓	✓	✓	90.4	74.1

Table 3.4: Evaluation of the contribution of each component of the proposed approach through Pascal VOC 2007 (VOC-07) and MIT Indoor (MIT-67) datasets. The first row corresponds to the baseline (*i.e.*, best semantic representation of the literature [81] that models the sparsification in a fixed manner and does not consider local information). The three next rows show the results of each individual contribution and the two last rows correspond to the different fusion schemes of the contributions. Note that, since FTDG is a method to improve CNN-features, it is presented in the next chapter.

specifically, our proposal contains three elements that correspond to the content-based sparsity (CBS), the local scheme for semantic features (SPP) and the diversified mid-level representation used for the building of semantic features (FTDG). Since this latter is a contribution in the context of CNN-features, it is presented in Section C.1.1 of the Appendix, but it roughly corresponds to a better mid-level representation than those used to build semantic representations.

The Semfeat baseline corresponds to the case where none of the three components is used (line 1 of Tab. 3.4). Selecting each of them independently results in: (i) a constrained semantic representation (CSF) when we select CBS, (l. 2 of Tab. 3.4) (ii) a local semantic representation (LSF) when we select SPP (l. 3 of Tab. 3.4) and (iii) a semantic representation based on a diversified mid-level feature (SF-FTDG) when we select FTDG (l. 4 of Tab. 3.4). Fusing the different components, results in: a constrained local semantic feature (CLSF) when we fuse CBS and SPP (line 5 of Tab. 3.4) and a constrained local semantic features based on a diversified mid-level features (CLSF+) when we combine CBS, SPP and FTDG (last line of Tab. 3.4).

The results are presented in Table 3.4. The first line corresponds to the baseline method, the three next lines represent the methods that consist of the selection of each component individually and finally the last two lines represent the different fusion schemes. We can clearly see that each individual component is better than Semfeat. More precisely, the content-based sparsity (CBS) obtains 85.1 and 63.6 while the fixed sparsity modeled in Semfeat obtains 82.8 and 61.1, making it much better. The proposed local scheme (LSF) as well as the diversified mid-level representation used to build semantic representations (SF-FTDG) considerably improve the performances compared to Semfeat. Regarding the different fusion schemes, the proposed CLSF obtains 88.2 and 71.6 on VOC07 and MIT67 and when we compare it to the LSF method (that obtains 85.1 and 63.6, resp.) we observe that the proposed local scheme associated to CBS is clearly better than the one associated with the fixed sparsity [81]. We also observe that the improvement given by the proposed CBS in a local scheme is slightly better than in a global scheme – *i.e*, CLSF is 3 points better than LSF while CSF is only 2.4 points better than Semfeat – making the CBS much interesting in a local scheme. Adding the FTDG contribution to CLSF (CLSF-FTDG) still improves the performances (compared to CLSF), indicating that using a better mid-level representation combines well with our proposal.

This ablation study shows that each of the proposed contributions brings an improvement over Sem-

Method	Pascal VOC 2007 mAP (in %)			
	Overfeat [224]	Caffe [114]	VGG [228]	FTDG (ours)
CNN [fc7]	72.0	76.3	86.1	88.9
CNN [fc8]	71.2	75.0	77.4	88.2
Classemes+ [252]	72.2	72.4	82.4	88.2
Semfeat [81]	73.6	76.0	82.8	88.1
CLSF (ours)	**78.4**	**80.5**	**88.2**	**90.4**

Table 3.5: Evaluation of the impact of CNN features (Overfeat, Caffe, VGG and FTDG) used to build semantic representations (Classemes, Semfeat and CLSF) on a transfer-learning scheme over the VOC 07 dataset.

feat. By combining all the contributions, we obtain a jump in performance. These results show the complementarity natures of the different contributions.

3.2.5.3 CNN Features Sensitivity

To our knowledge, it has never been directly showed that mid-level features used to learn the base classifiers impact the performances of semantic features. In this section we perform this empirical demonstration. More precisely, we compare the performances (on Pascal VOC 2007) of the semantic representations based on the different CNN architectures (Overfeat [224], Caffe [114], VGG [228] and VGG-FTG). Overfeat and Caffe have a similar architecture since both of them are a derivative of the reference AlexNet network [132]. The VGG group [228] proposed a more performing network that differs from the former ones by its depth that contains 16 layers, instead of 8. FTDG is described in Section C.1.1, but it roughly corresponds to the penultimate layer extracted from a VGG network learned on a large and diverse set of categories. For all features, we use the same implementation details as depicted in section 3.2.5.1.

Results reported in Table 3.5 show that semantic representation performances are directly related to the quality of CNN features. For all the semantic methods, best performance is obtained with FTDG followed by VGG, Caffe and Overfeat. In addition, the comparison of the semantic features based on FTDG and those based on VGG means that the transferability of the CNN features directly impacts those of the semantic features. Equally important, the proposed semantic representations get more performance than all others, regardless the CNN representations used to build them. Moreover, since other CNN features are easy to plug into the proposed pipeline, it will be easy to make it evolve in order to take advantage of progress in deep learning architecture design. This latter analysis greatly validates the robustness of the proposed semantic representation to the different CNN features used to build it.

3.2.5.4 Region Detector Robustness

An important part of our method is the strategy to choose which local regions (which location, which size, which form, etc.) to extract from each image. In fact, for the CLSF method (described in Sec. 3.1.2.2), we employed a fixed grid inspired by SPP all over the paper in order to demonstrate the constrained locality property of the proposed CBS method in a local scheme. Here, we evaluate the

Region Detector	**#Regions per Image** (average)	**Pascal VOC 2007** mAP (in %)
EdgeBox-100 [55]	100	87.7
EdgeBox-10 [55]	10	88.3
USOD [293]	1.29	88.3
USOD* (custom)	**1.29**	90.0
SimpleGrid	6	90.4
SimpleGrid + USOD*	7.29	**90.6**

Table 3.6: Evaluation of the effects of the region detector used in the CLSF+ method on Pascal VOC 2007 dataset. Second column indicates the average (over the whole dataset) number of regions extracted per image, while the last column presents the obtained results.

sensitivity of our method (through the VOC07 dataset) to the following region-detectors:
- **SimpleGrid** is a scheme inspired by SPP [96] where the full image is taken and 5 local regions are extracted according to regular grid at smaller scale ($2/3$ of the full image). Unlike SPM, we use overlapping rectangular regions to reduce the risk of cutting the objects represented in the image, resulting in six extractions per image.
- **EdgeBox** [55] is an algorithm for generating object bounding box proposals based on edges. We used it as it is released and set the parameters α to 0.65 and β to 0.55. The number of regions per image is also a parameter, thus we tested for 10 and 100 regions.
- **USOD** [293] is a very recent method of unconstrained object detection using CNNs. This method has the desirable property to detect very small objects and outputs a content-based number of salient regions. We used the online code with default parameters and it remains to an average of 1.29 regions per image.
- **USOD*** is our customization of USOD. Regions that have one side smaller than 224 pixels, are extended until we get 224 pixels (similar to add context) on the smallest side. The highest side is set using the height/width ratio of the initial extracted region.
- **SimpleGrid+USOD*** is the combination of regions extracted from the SimpleGrid and the customized USOD schemes. In practice, it remains to an average of 7.29 regions extracted per image. This combination has the desirable property to extract big objects through the SimpleGrid scheme and small objects with the USOD* algorithm.

Results of our approach using each region detector are presented in Table 3.6. First, EdgeBox is quite bad compared to other detectors, especially when we take 100 boxes. This is due to the fact that 100 regions per image is high and thus, it certainly contains a lot of noisy information. But our method does not highly distort the results (-0.6 point compared to EdgeBox-10). This is mainly due to the CBS part of our method that desirably neglects regions without relevant information. On the contrary, with USOD or EdgeBox with a small number of regions, CLSF+ obtains 88.3 which is lower than the results of USOD* or SimpleGrid (respectively 90.0 and 90.4).

First, USOD* gives better results than USOD because of the pixelization limit of this latter. In fact, USOD tends to extract very small regions but when they are passed through the CNN for mid-level feature extraction, the regions are re-sized to 224×224 (CNN constraint), which highly pixelizes the images. The proposed customization USOD* extracts bigger regions through the consideration of the context when USOD outputs small ones. Second, SimpleGrid gives high results compared to other region detectors. This is mainly due to the small and weakly-overlapping regions taken from the whole image which cover all the content of the image. But it still has a limitation which is the possibility to extract regions without information and thus disturbing the final representation. However, thanks to the CBS module in our method, we neglect these noisy regions and do not consider them in the final representation.

Finally, this experiment globally shows that our method is quite robust to the bad region-detectors, since it does not highly distort the performances and reacts in a good way with good region-detector algorithms.

Analysis of Diverse Concept Level Semantic Features

We describe the implementation details of our method, then we conduct two experiments that highlight some insights of it. More specifically, we first conduct an experiment that shows the interest of the proposed semantic classifiers (for the superordinate concepts) compared to the traditional visual classifiers, then we systematically evaluate the correlation between the concept-groups selected and the performance of the final semantic representation.

3.2.5.5 Implementation details

D-CL learning: To build our base semantic feature, we follow the implementation of Ginsca *et al.* [81]. Indeed, we used ImageNet [53] to learn the different semantic classifiers. We especially use a subset of ImageNet that contains $17,462$ concepts. It has been obtained from the set of $22,000$ concepts of ImageNet after the selection of only the categories that contain more than 100 images. Hence, each individual concept detector is then learned using images representing the concept c_i as positive samples, and images from a diversified class as negative samples. Note that, the concepts can be placed at any categorical-level of a semantic hierarchy, making our method applicable on top of any semantic representation.

CNN feature: Semantic representations (including the proposed D-CL), are built on top of any low-level or mid-level features (CNN). However, the quality of the D-CL representation will directly depends on the low/mid-level feature used. We thus created semantic features on top of a competitive mid-level representation released in the literature, namely VGG-Net [228]. It is extracted from the last fully-connected layer (layer 16) of a Convolutional Neural Networks (CNN) learned on ILSVRC 2012 dataset [216] (containing 1.2 million images labeled among $1,000$ output categories), resulting in $4,096$ dimensional vectors. Note that, for a fair comparison, the same mid-level representation is used to build Classemes+ [252] and Semfeat [81]. For our study, fine tuning of the CNN may result into an improvement of the results at the cost of significant computational cost and the possible use of additive data. Such a specific optimization of the CNN has not been considered in our experiment, to insure their reproducibility with the available CNN models.

Concept Groups Identification: As depicted in Sec. 3.2.2.2, the set of basic-level concepts ($\mathcal{BL}^d$ in Equation (3.7)) is matched with the set of targeted dataset categories, for each dataset. Since all the concepts of ImageNet are organized in accordance to the WordNet [168] hierarchy, we use it as input to the subsumption function $\phi(\cdot)$ to select the corresponding superordinate concepts ($\mathcal{P}^d$ in Equation (3.7)). Specifically, only the first and the fourth level of the WordNet hierarchy are used. This avoids redundancy of semantically close superordinate concepts. In fact, those levels contain the most popular superordinate concepts employed in cognitive experiments [118, 214, 248]. For the set of the K most salient subordinate concepts ($\mathcal{B}^K$), the parameter K of Equation (3.7), is cross-validated on each training dataset using the usual train/val split.

Superordinate concepts	**Basic-level concepts**	**Visual** mAP (in %)	**Semantic** ($\uparrow$) mAP (in %)
Animal	bird - cow - dog - horse - sheep	92.9	**97.7** (+4.8)
Elec. equip.	tv monitor	52.1	**72.6** (+20.5)
Furniture	chair - sofa - table	70.0	**74.9** (+4.9)
Person	person	77.2	**85.7** (+8.5)
Plant	potted plant	26.5	**40.5** (+14.0)
Vehicle	airplane - bike - boat - bus - car - mbike - train	93.4	**96.9** (+3.5)
Vessel	bottle	18.7	**31.4** (+12.7)
mAP		61.5	**71.4** (+9.9)

Table 3.7: Evaluating purely visual binary classifiers (denoted as "Visual") and our proposed semantic classifiers (denoted as "Semantic") for superordinate concepts (first column) of Pascal VOC 07 dataset classes (second column). The improvements of semantic classifiers over visual classifiers are shown in parentheses. Note that, the class *person* of Pascal VOC 2007 is already at the highest level in the WordNet hierarchy.

3.2.5.6 Accuracy of Semantic Classifiers

In this section, we assess the effectiveness of the proposed semantic classifier ($\phi^S(\cdot)$ of Equation 3.6), and compare it with purely visual classifiers, *i.e.*, binary classifiers ($\phi^V(\cdot)$ of Equation 3.6) on generic concepts (*i.e.* concepts that have at least one hypernym relation with another concept).

This analytic study is an analogy to the experiment conducted by the cognitive works of Stephan Kosslyn [118]. More precisely, they wanted to provide a converging evidence that superordinate concepts are semantically processed by humans, rather than by a visual perception processing. Thus, to respect the analogy with [118], we evaluate the proposed semantic classifier and the visual classifiers on superordinate concepts only.

Regarding our experiment, the selection of superordinate concepts imposes, in the Equation 3.7 of the proposed D-CL representation, to set to zero all the basic-level and subordinate concepts ($\phi_i^V(\mathbf{x}) = 0$, $\forall c_i \in \mathcal{BL}^d \cup \mathcal{B}^K$). Thereby, the experiment has been conducted on the context of multi-class object classification through the Pascal VOC 07 dataset. All the images of the dataset have been re-labeled at superordinate level, *e.g.* all images labeled as *bird*, *dog*, *cow*, *horse* or *sheep* are now labeled as *animal*, all images labeled as *chair*, *sofa* or *table* are now labeled as *furniture*, etc (see first and second column of Table 3.7 for the re-labeling of other classes). Hence, we learn each superordinate class of the dataset by a one-vs-all SVM classifier. The cost parameter of the SVM classifier is optimized through cross-validation on the training dataset, using the usual train/val split. Performance results of both classifiers are reported in Table 3.7 using average precision (AP in %) for each class and mean Average Precision (mAP in %) over all classes in the last row. The second column of the table corresponds to the basic-level categories of the dataset and the first column corresponds to the list of selected superordinate concepts in our D-CL feature. The average precision of each superordinate concept computed through binary classifiers (denoted as **Visual**) and the proposed semantic classifier (denoted as **Semantic**), are presented in the last two columns, respectively. Remarkably, the

Concept Groups Selection	$\mathcal{P}$	$\mathcal{BL}$	$\mathcal{B}$	$\mathcal{B}^K$	**mAP**
Superordinate	✓				44.4%
Basic-level		✓			76.1%
Subordinate			✓		82.1%
K-Subordinate				✓	78.9%
Fusion 1	✓	✓	✓		82.7%
Fusion 2 (D-CL)	✓	✓		✓	85.1%

Table 3.8: Evaluation of the contribution of different concept groups selection (check-mark = selected group) in the proposed semantic feature on Pascal VOC 2007 dataset.

proposed semantic classifier clearly outperforms binary classifiers (purely visual) for all the superordinate concepts. From this study, we conclude that the superordinate concepts are better recognized by D-CL, due to its ability to compensate low within-category resemblance of generic concepts. The most surprising aspect of this experiment is that it shows, as concluded by the analogical cognitive experiment of Stephan Kosslyn [118], that semantic process is most adapted than purely visual process for superordinate concepts.

3.2.5.7 Concept Groups Selection Sensitivity

We now evaluate the contribution of the concepts from different groups (*i.e.* categorical levels) on a multi-object classification task (Pascal VOC 2007). To this end, we need to isolate each group of concepts in the D-CL representation by selecting them individually and setting other groups to zero. It results in four special cases of the D-CL representation (Equation 3.7), (i) selecting only superordinate concepts ($\forall c_i \in \mathcal{P}^d \cup \mathcal{B}, \phi(c_i) = 0$) denoted as "Superordinate", (ii) selecting only basic-level concepts ($\forall c_i \in \mathcal{P}^d \cup \mathcal{B}, \phi(c_i) = 0$) denoted as "Basic-level" (iii) selecting only subordinate concepts ($\forall c_i \in \mathcal{P}^d \cup \mathcal{BL}^d, \phi(c_i) = 0$), denoted as "Subordinate" and (iv) selecting only the K most salient subordinate concepts ($\forall c_i \in \overline{\mathcal{B}^K}$), denoted as "$K$-Subordinate". We also evaluate the contribution when selecting all the concept groups in the representation ($\forall c_i \in \mathcal{P}^d \cup \mathcal{BL}^d \cup \mathcal{B}, \phi(c_i) \neq 0$ in Eq. 3.7), *e.g.*, superordinate, basic-level and subordinate concepts, denoted as "Fusion 1". Finally, we report the results obtained by the proposed D-CL concept groups selection (as depicted in Sec. 3.2.2.2), corresponding to the selection of, all the superordinate and basic-level concepts and the K most salient subordinate concepts. It is also a fusion of other groups of concepts that we denote as D-CL. Results are reported in Table 3.8. For each concept group selection, a check-mark represents the concept groups that had been selected in the final representation. The last column gives the mAP obtained for the different concept selections. Note that, the K parameter of Equation 3.7 has been cross-validated for the "K-Subordinate" and "D-CL" concept group selections.

Obviously, selecting only superordinate concepts ($\mathcal{P}$) leads to very bad results, compared to basic-level concepts only ($\mathcal{BL}$), which are their-self lower than subordinate concepts only ($\mathcal{B}$). Selecting only the K most salient subordinate concepts ($\mathcal{B}^K$) obtains lower performances than selecting them all. Surprisingly, for the fusion, it is better with the selection of the K most salient concepts (the proposed D-CL) than with the selection of all subordinate concepts (Fusion 2). This experiment shows that the proposed D-CL selection gives a most effective semantic representation.

3.3 Conclusions

In this chapter, we introduced two novel methods of sparsification for semantic representations. Both methods aim at decreasing the noise generated by the large set of detectors in semantic representations. In contrast to existing works, which perform sparsification for novel images regardless the image content, both methods perform a *structured* sparsification based on the image content. More precisely, the Content-Based Sparsity (CBS) consists of an adaptive sparsity according the image and/or the local region content that is modeled through the entropy and the confidence of the output classifiers. The second contribution named Diverse Concept-Level (D-CL) consists of the exploitation of human knowledge (such as categorical-level categories modeled in semantic hierarchies), to identify groups of concepts, to consider (when not considered) some particular concept in the final representation. This latter, processes the groups of visual concepts differently from each other, according to their categorical-level belonging. Simply said, our schemes tend to output only the informative concepts (through the structured sparsity) in the final semantic representation.

The evaluation of the two methods is carried out for classification and retrieval tasks and is compared to the state-of-the-art semantic representation methods. Both proposed methods outperform existing semantic representations.

This work direction aims at better handling the increase of the amount of detectors and thus contributes to increase the universality of semantic-features.

3.3.1 Complementaries of the Two Contributions

CBS and D-CL methods contribute to increase the universality of semantic-features, since they both perform a structured sparsity that aims at considerably decreasing the noise (*i.e.*, noisy object activations that the semantic features output when computed on images) introduced by the large amount of detectors. Hence, this latter aims to take full advantage of the *largeness* of the amount of detectors, which thus, pushes our two methods to quite important methods for the approach that consist to enlarge the amount of detectors for increasing the universality of semantic-features built on top of CNN-features with a bank of independent SVM classifiers.

The two methods have similar performances on the target-datasets used for evaluation. However, they are quite different and have different advantages and disadvantages. The advantage of the CBS method lies in its ability to automatically compute, for a given image, the right amount of object activations to keep in the representation. However, by construction, CBS always outputs the top detections of the semantic features and ignores all other activations. For example, it is important to consider that generic detectors (detectors of *dog*, *car*, *bird*, etc.) output much lower values than specific ones (detectors of *rottweiler*, *malinois*, *german shepherd*, etc.) because of the intra-class variance property of their categories. Indeed, the *dog* category contains dogs from several breeds and thus contains *diverse* images, that thus results in a much higher intra-class variance than the *breed of dog* category, which contains only images of that breed of dog. As said in Sec. 2.2.5, different intra-class variances cause different behaviors on the learning-algorithm and thus different classifiers. In that case, the resulting behavior is that specific detectors output much higher values than generic ones. Thus, when we apply a top-k sparsity (fixed or adaptive), we empirically observed that it always selects the specific ones. This is an important drawback in the context of multi-object classification, since it generally selects the concepts of the largest objects in the image and misses all those of the smaller objects around it. The D-CL method fixes this drawback since by construction it always considers the generic concepts. Moreover, the generic concepts are also *boosted*, which fixes the non-desirable behavior of the classifiers that output low values for generic concept-detectors. However, D-CL adopts a *fixed* sparsity scheme, which thus has all the drawbacks of the fixed top-k sparsity method. Hence, in order to take full advantages of both methods, in the future, we plan to combine them through the most natural way that consists to apply an adaptive sparsity to the well structured sparsity of D-CL. Moreover, an other drawback of D-CL is its dependence to the identification technique of the concept-groups. Thus, we need to investigate how to automatize this selection of concept-groups, as we automatized the selection of the "good" concepts in the CBS scheme. In any case, we have two adaptive sparsification methods, according to different criteria and not necessarily compatible, so it is not obvious to combine them naively.

3.3.2 Discussion About Semantic-Features

CBS and D-CL methods are two universalizing methods in the sense that they take full advantage of the *largeness* of the amount of detectors both from its quantitative and qualitative part. Indeed, considering a large denoised set of detectors gives the ability to the resulting representation to cover efficiently a wide variety of targeted domains. Moreover the two approaches also improve the universality of the representation by adding more relevant semantics to the obtained representation but on different aspect of semantics (implicit versus explicit or extensional versus intentional). The CBS method, by taking into account the content of images, leads to representations that are more tailored to tackle the perceptual specificity of the targeted domains, *i.e.*, its implicit semantics or extensional definition of concepts. On the other side, the D-CL method, by adding explicit semantics on human categorization, i.e. the different kinds of categorization levels, improves the ability of the obtained representations to tackle the terminological specificity of the targeted domains, *i.e.*, its explicit semantics or intensional definition of concepts. Thus, here, we propose some directions that should be investigated in the context of semantic representations to increase their universality. We will also explain why, in this Thesis, we did not focused on the improvement of the universality of semantic-features but rather that of CNN-features.

The actual semantic representations already contain a huge amount of detectors ($17,462$ in our implementation and almost $80,000$ in [81]) and to design them, we already consider a large number of categories among those usually available in the benchmarks of the literature. This latter, aims to cover a very wide range of different data. As mentioned before, the most obvious way to learn more universal semantic representations is to increase their amount of detectors. However, to improve them in this direction, collecting *more* annotated data (*i.e.*, images and categories) is needed and can be very costly. A less costly way to increase the amount of data is to get subcategories from each category, through clustering. However, in an unsupervised scheme, to stay in a semantic context, we should associate a semantic connotation to each cluster that could be difficult. Nevertheless, annotating only the closest image to the centroid of each cluster is still much less costly than getting sub-categories through human annotations. More generally, the latter direction consists in better (and semi-automatically) structuring the available annotated data. This idea is, by the way, one of the core idea of our next contribution (Chapter 4). The other obvious direction to increase universality of semantic features is to improve the discriminatory power of each individual detector. To do so, we see two ways, (i) better classifiers – this would suggest to build better learning estimators – and (ii) better lower-level representations – learn the classifiers on top of more universal representations. This latter point actually corresponds to the increase of universality of CNN-based representations and will be discussed in details in the next chapter.

Above all we identified some limitations in using semantic features in order to reach our goal of building universal representations. Semantic representations built with the implementation of Ginsca *et al.* [81] – *i.e.*, trained with a bank of *independent* classifiers on top of a *fixed* representation –, give worse transfer performance than those obtained from the last layer of a pre-trained CNN – *i.e.*, trained with a bank of *dependent* classifiers, on top of many *trainable* representations[2]. From this, one may infer that either a set of dependent classifiers is better than a set of independent ones, or a set of classifiers trained on top of learned representations is better than a set of classifiers learned on top of fixed representations. This reveals that finding solutions to teach CNNs to solve very large classification tasks (*i.e.*, with a very large bank of classifiers at the last layer) may be better than trying to improve the implementation of semantic representations chosen in this Thesis. However, a more problematic fact is that, semantic representations can only be learned in a supervised way (*i.e.*, with classifiers trained to predict a given category). Indeed, while this was not directly observed empirically, it seems that features learned with explicit supervision are not better in terms of transfer performance than those resulting from implicit supervision (*e.g.*, internal layers of CNNs are more transferable than the last one [285]). To conclude, features learned with implicit supervision seems to provide better transferability power than those learned with explicit supervision, thus we will now focus on the investigation of building representations through the internal layers of CNNs rather than through semantic features (*i.e.,* last layer of a CNN or an auxiliary bank of classifiers learned on top of the penultimate layer).

[2]Note that, in the actual version, one-versus-all learning strategy is considered but a way to avoid this problem could be to use samples of other categories as negative examples, or even a perceptron rather than multiple independent classifiers.

4

Improving Universality using Discriminative-Problem Variation

Contents

In this chapter, we present the core contribution of the thesis, that is a new approach to learn more universal representations – especially more universal CNN-based representations. The approach starts from a given *reference representation* – a set of features learned with a standard learning strategy and a network architecture on an *initial discriminative problem* – and consists of three modules: (i) the application of the proposed principle of "Discriminative Problem Variation" (denoted **DPV**) to obtain new problems at near-zero cost; (ii) the training of new features on the obtained discriminative problems; and (iii) the combination of the learned features that results into a representation that is more universal than the original one. Simply said, our approach consists in training multiple networks on different problems obtained through the principle of discriminative problem variation. We thus name it "Multi Discriminative-Problem Network" (**MulDiP-Net**). In this Thesis, we provide a proof of concept of the approach, with justified choices for each step. We proposed an original contribution for the first and third steps of the approach, as well as a complete experimental protocol and evaluation-metric to evaluate and thus compare universalizing methods. In Section 4.1, we present in details our approach (*i.e.*, MulDiP-Net), then in Section 4.2, we describe the *general* evaluation protocol for universalizing methods and finally in Section 4.3, we report the experimental results of our approach in a transfer-learning scheme.

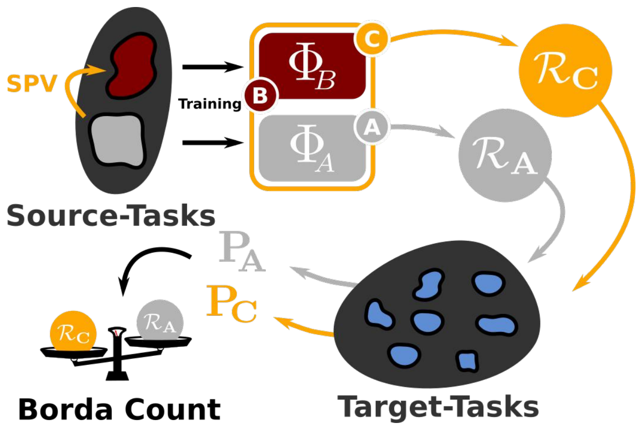

Figure 4.1: A *universal representation* is a set of features learned on a **source-task**, that leads to high performances when it is used on a large set of **target-tasks** (transfer-learning). Let us suppose a universal representation $\mathcal{R}_A$ (light gray circle) obtained by a reference method (A) that consists in training a network (set of features Φ_A on a source-problem (bottom gray blob in set of source-tasks) according to a classical learning methodology. When it is transfered on a large set of target-tasks, it leads to a *high* average performance P_A on it. Our proposed approach consists in building a *more universal* representation $\mathcal{R}_C$ (orange circle) by: normalizing and combining the features Φ_A with features Φ_B obtained from a network trained on the output of a discriminative-problem variation (SPV) – *i.e.*, starting from an initial source-problem (gray blob), variates it to get a new one (red blob). Since our method (C) is more universal than (A), it has a higher ranking score according to the *Borda Count* metric, that we propose for evaluating the increase of universality (described in sec. 4.2).

4.1 Multi Discriminative-Problem Networks

4.1.1 Introduction

While it is known that learning neural networks with more data (categories and/or images per category) leads to representations that have better performances on a set of target-datasets (in a transfer-learning scenario) – see Table C.1 in Appendix –, it is clearly limited by its cost in terms of adding annotated data. It would thus be advantageous to get more universal representations with the same amount of data (annotated images). To reach this goal, two aspects could be considered: improving the learning approach or improving the data on which it is applied (referred in the previous chapter as problem of better structuring the available annotated data). The first point has been extensively explored. In particular, many works proposed better optimization algorithms [126, 99], network architectures (deeper, wider) [289, 237, 97], activation functions [277], regularization [232], normalization [110], and much more. Most of these works use the ILSVRC (or the Place305) datasets to learn the CNNs. On the other side, improvements of the structure of the data have been less explored. Indeed, a couple of works proposed to increase the set of input data [166, 11] or solve a scene recognition task instead of objects [297]. More frequently, data-augmentation is used but such an approach is limited in terms of possible realistic variations with regards to those meet in real conditions. In this Thesis, we think that working on data could be as interesting as a better learning approach, and has the advantage to be an orthogonal contribution to this last. If one considers two discriminative problems (DP-A and DP-B) such that DP-A discriminates *green-lemon*, *yellow-lemon*, *yellow-apple* and *green-apple* and DB-P discriminates *lemon* from *green*, there is a high chance that the resulting learned features differ significantly. If one combines the both sets of filters, it will thus obtain a much richer set, able to discriminate more classes. The key remark is that for this particular example, if DP-A is fully available with its annotated images, obtaining DP-B does not require a costly process. It is possible to use the *same* images and simply re-annotate each category, that is much cheaper than annotating all the images for DP-B. The approach we propose is based on this principle and explores such a re-annotating process to obtain more features at a near zero-cost, leading to a more universal representation able to adequately represents various concepts. An illustration of our approach is given in Figure 4.1.

4.1.2 Proposed Method

We formalize the source problem variation (SPV) in Section 4.1.2.1 with a particular emphasis on the SPV by grouping of specific categories. Once a CNN has been learned on several discriminative problems, the resulting representations have to be combined. Concatenation is a simple approach that already gives interesting results, but at the cost of using a larger vector to represent the images. Hence, we propose a method that we named Focused Self-Fine Tuning (FSFT), which consists in retraining a set of parameters with a smaller set by focusing the learning on them, rather than on all the parameters of the network (Section 4.1.2.5). When all our contributions are combined, it results in MulDiP-Net that provides more universal representations of comparable or smaller size than the original one (Section 4.1.2.6). An illustration of the MulDiP-Net method is given in Figure 4.5.

Above the proposal itself and the demonstration of its performance in transfer learning (Section 4.3), we also put into relief some hints to explain the performance improvements (Section 4.3.1). As explained above, the richer set of features allows a better representation for a larger set of discriminative problems. With MulDiP-Net, the representation contains features that capture common properties among generic categories making it more robust for problems relative to this category.

4.1.2.1 SPV: Source-Problem Variation

A source problem, $\mathcal{D}_k^S = \{(x_i^k, y_i^k)\}_{i=[\![1,N_k]\!]}$, consists in a set of N_k couples (x_i^k, y_i^k) where x_i^k is a training image and $y_i^k \in \mathcal{Y}_k = \{c_1^k, \ldots, c_{C_k}^k\}$ its associated label among C_k categories. By solving this source problem (denoted SP in the following), the CNN learns features to discriminate the images of the categories of the SP. For instance, if we consider a network that solves a SP containing images of *lemons* and *green-apples*, the network learns different features than those obtained when it solves a SP containing images of *lemons* and *strawberries*. Hence, changing the SP to be solved by a CNN can lead to a change in the set of features learned by the network. Motivated by this observation, we propose the principle of *Source Problem Variation* (SPV) that is formalized by a variation function $\vartheta_0^k(\cdot)$ that transforms an initial source problem $\mathcal{D}_0$ into a new one $\mathcal{D}_{k,k>0}$. Such a function has the following form:

$$\begin{aligned} \vartheta_0^k : \quad \{\mathbb{R}^{S_I} \times \{0,1\}^{C_0}\}^{N_0} &\rightarrow \{\mathbb{R}^{S_I} \times \{0,1\}^{C_k}\}^{N_k} \\ \mathcal{D}_0 &\mapsto \mathcal{D}_k = \{(x_i^k, y_i^k)\}_{i=[\![1,N_k]\!]}, \end{aligned} \tag{4.1}$$

with the following constraints:

$$\begin{cases} \forall i \in [\![1, N_k]\!], \forall j \in [\![1, N_0]\!], \exists (x_i^k, y_i^k), x_i^k \neq x_j^0 \text{ or } y_i^k \neq y_j^0 \\ \forall i \in [\![1, N_k]\!], \forall j \in [\![1, N_0]\!], \exists (x_i^k, y_i^k), x_i^k = x_j^0 \text{ or } y_i^k = y_j^0 \end{cases} \tag{4.2}$$

where $S_I = W \times H \times D$ is the size of images (*i.e.*, width W, height H and depth $D = 3$ for RGB images), (x_i^k, y_i^k) is an element of the new SP $\mathcal{D}_k$, with each training image x_i^k labeled according to the label-set $\mathcal{Y}_k = \{c_1^k, \ldots, c_{C_k}^k\}$ containing C_k categories. The first constraint imposes that at least one element of $\mathcal{D}_0$ has changed in $\mathcal{D}_k$ and the second constraint warrants that at least one label or one image of $\mathcal{D}_0$ has to be in $\mathcal{D}_k$. With these constraints, taking a dataset completely different to the initial one is *not* a SPV and in contrast, changing all the data (images and labels) without keeping any element from the initial SP is also *not* a SPV.

In practice, the variations can be of any kind and on any of the elements of D_0 (*i.e.*, on the set of images $\{x_i^0\}_{i \in [\![1,N_0]\!]}$ and/or on the set of labels $\{y_i^0\}_{i \in [\![1,N_0]\!]}$). We propose to consider three kinds of variations of the SP: variations that consist in *adding* some elements (images or categories) in the new SP, variations that consist in *splitting* the original SP into different smaller problems and finally variations based on *grouping*. This principle, although not explicitly described as a SPV has been applied in the literature. For instance, variations on the image set while preserving the set of categories have been proposed in [98, 224]; variations by *adding* have been used by [23, 210] (by adding data labeled among specific categories) and [166, 246] (by adding data labeled among generic categories); variations by splitting the set of categories has been performed in [101, 2, 280, 11, 275] and finally, variations by *grouping* of categories have been explored in [33, 109, 66].

An illustration of some common SPVs, including the SPV by *grouping* that we are particularly interested in, is given in Figure 4.3. The SPV formalism can be used for different goals, including diversifying the available features with SPV by *grouping* as explained in the next section. In any case, the considered SPV only concerns the training data and is thus independent of the method used to learn the CNN as well as its architecture.

4.1.2.2 SPV Based on Grouping

The use of SPV by *adding* has been explored in the literature but has the drawback to require additive annotated data. Thus, we propose to rely on SPV by *grouping*, that learned from the same data. Source Problem Variation (SPV) based on grouping has the advantage to alleviate drawbacks of those based on adding and splitting (of a whole set of categories, to get multiple label-sets, *i.e.*, discriminative-problems). Indeed, compared to adding SPV – which needs more annotated data that is costly to obtain – grouping SPV does not need more annotated data. Moreover, compared to splitting SPV – which decreases the performances of the networks, since it decreases considerably the amount of training data –, grouping SPV maintains exactly the same amount of images. Several approaches can be imagined to group categories, including a simple random grouping or a bottom-up clustering grouping as explored by Bergamo *et al.* [21] in the context of semantic features. We propose to rely on an external a priori knowledge on human categorization in order to learn features that better reflect it. Indeed, most of the categorization problems we are interested in, aim at identifying some categories that make sense for humans (need of interpretability), thus we assume that injecting such knowledge in the learning process should be able to lead to results that better match the (human) user needs. This semantic knowledge is generally represented in the form of hierarchies. Here, we thus focus on the *categorical levels* [118, 214, 248] that consist in a hierarchy of categories mostly used by Humans to categorize objects.

Let us consider a semantic hierarchy with hyponymy relations (*i.e.*, a set of categories organized according to "is-a" relations). An example of such a hierarchy is WordNet, on which are mapped the categories of ImageNet [53]. Formally, this is a directed acyclic graph $\mathcal{H} = (\mathcal{V}, E)$ consisting of a

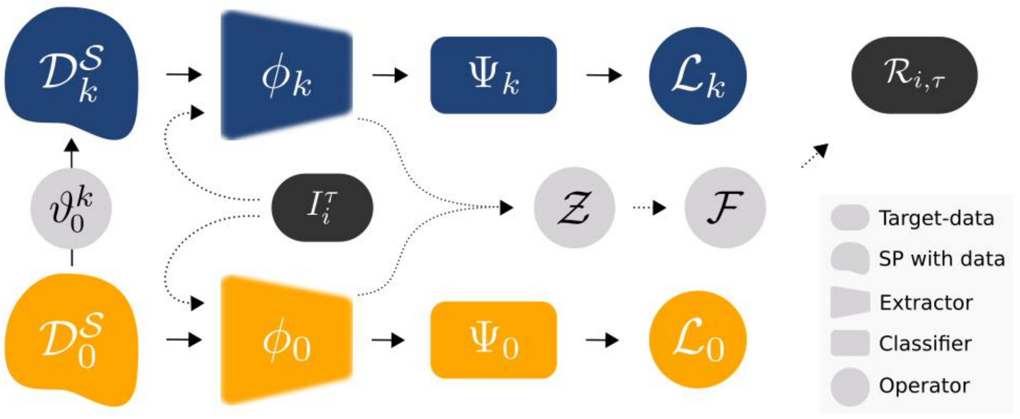

Figure 4.2: Detailed illustration of our MulDiP-Net method. Let consider an *initial* source problem (SP) $\mathcal{D}_0^{\mathcal{S}}$ consisting in a set of couples of images and their associated labels (x_i^0, y_i^0). MulDiP-Net consists in three phases: (i) variation of the initial SP into new ones ($\mathcal{D}$ and ϑ blocks); (ii) training networks on the obtained set of SPs (ϕ, Ψ and $\mathcal{L}$ blocks) ; and (iii) combination of the features extracted from each trained network in order to form a more universal representation (remaining blocks). More precisely, the first phase is a source problem variation (SPV) (ϑ_0^k) applied on the initial SP $\mathcal{D}_0^{\mathcal{S}}$, which outputs a new SP $\mathcal{D}_{k,k>0}^{\mathcal{S}}$. After applying K SPV functions, we get a set of K+1 SPs containing the new SPs and the initial one. The second phase consists in the learning of multiple networks on the set of K+1 SPs. More precisely, we train one network per SP, resulting in a set of K+1 networks $\{\mathcal{N}_k\}_{k=[\![0,K]\!]}$. Indeed, each network $\mathcal{N}_k$ – that is a composition of a features-extractor ϕ_k and a classifier Ψ_k – is trained by minimizing the loss function $\mathcal{L}_k$ computed using the output of the predictor Ψ_k and the ground-truth of the SP $\mathcal{D}_k^{\mathcal{S}}$. Then, the third phase consists in the extraction of universal representation $R_{i,\tau}$ from images $I_i^{\mathcal{T}}$ of a target-task T_τ. More precisely, it passes the images $I_i^{\mathcal{T}}$ into the trained networks and get features (through the extractors ϕ_k) that are independently normalized ($\mathcal{Z}$) and fused ($\mathcal{F}$) in order to output the final representation $R_{i,\tau}$.

set $\mathcal{V}$ of nodes and directed edges $E \subseteq \mathcal{V} \times \mathcal{V}$. Each node $v \in \mathcal{V}$ is a label and $(v_i, v_j) \in E$ is a hierarchy-edge indicating that label v_i subsumes label v_j. Let us also consider a source-problem $\mathcal{D}_0^{\mathcal{S}}$ containing N_0 images labeled among C_0 *specific* categories belonging to $\mathcal{C}_0^{\mathcal{S}} = \{c_1^0, c_2^0, \ldots, c_{C_0}^0\}$, such that $\mathcal{C}_0 \subset \mathcal{V}$.

We now consider a categorical-level defined according to human cognition. Let us note $\mathcal{B}_L^{cat}$ a set of categories that belong to a categorical-level L (*i.e*, *subordinate* level for L=0, *basic* for L=1 and *superordinate* for L=2). It is important to note that the categories of $\mathcal{B}_L^{cat}$ do *not* correspond to a given level of the hierarchy $\mathcal{H}$. Hence, we consider that all $c_i^{cat_L} \in \mathcal{B}_L^{cat}$ are mapped to some nodes of $\mathcal{H}$. To *group* the categories of $\mathcal{C}_0^{\mathcal{S}}$ into G *generic* categories. This latter is equivalent to get the partitioning of $\mathcal{C}_0^{\mathcal{S}}$ into G subsets *i.e.*, $\mathcal{C}_0^{\mathcal{S}} = \bigcup_{i=1}^{G} \mathcal{G}_i$. To do so, we define a partitioning function according to a categorical-level $\mathcal{B}_L^{cat}$ as:

$$\begin{aligned} P_{cat_L}: \quad & \mathcal{V} && \to 2^{\mathcal{C}_0^{\mathcal{S}}} \\ & c_i^{cat_L} && \mapsto \mathcal{C}_0^{\mathcal{S}} \cap D_{\mathcal{H}}\left(c_i^{cat_L}\right), \end{aligned} \tag{4.3}$$

where $D_{\mathcal{H}}(c_i^{cat_L})$ is the set of all descendant nodes of the categories $c_i^{cat_L}$ according to $\mathcal{H}$. Using this function, we obtain G generic categories, with $G \ll C_0$.

We can now define our re-labeling function relative to a given categorical-level $\mathcal{B}_L^{cat}$ by:

$$\begin{aligned} R_{\mathcal{B}_L^{cat}}: \quad 2^{\mathcal{C}_0^{\mathcal{S}}} &\to \mathcal{B}_L^{cat} \\ \mathcal{C}_i &\mapsto \mathcal{B}_L^{cat} \cap \mathcal{A}_{\mathcal{V}}\left(LCA_{\mathcal{H}}\left(\mathcal{C}_i\right)\right), \end{aligned} \tag{4.4}$$

where $LCA_{\mathcal{H}}\left(\mathcal{C}_i\right)$ is the lowest common ancestor of all the categories in the set $\mathcal{C}_i$ and $\mathcal{A}_{\mathcal{V}}(c_i)$ returns the set of all the ancestors of the category c_i. Formally, $\mathcal{A}_{\mathcal{V}}(c_i) = \{\delta_{\mathcal{H}}^j(c_i)\}_{j=1}^{\infty}$, with $\delta_{\mathcal{H}}(\cdot)$ being a *deductive function* that associates to a category v_i of $\mathcal{V}$ its direct ancestor, that is to say, the category directly above v_i according to $\mathcal{H}$ and $\delta_{\mathcal{H}}^n(\cdot)$ its corresponding iterated function (*i.e* $\delta_{\mathcal{H}}(\cdot)$ composed with itself n times and for which we assume that the image of the root node of $\mathcal{H}$ is itself).

Simply said, while Eq. (4.3) partitions the set of specific categories into "arbitrary" generic categories (those of WordNet), Eq. (4.4) re-labels them according to the categorical-level $\mathcal{B}_L^{cat}$. Also important, images are also automatically re-labeled according the same process as the initial categories they belong. The SPV by *grouping* ϑ_G is illustrated in Figure 4.4, showing the combination of the partitioning and re-labeling functions.

The same process can be applied to re-label categories belonging to hierarchical-levels $\mathcal{B}_L^h$ of $\mathcal{H}$, assuming that descendants of leaf-nodes are themselves (*i.e.*, $D_{\mathcal{H}}(c_i^0) = c_i^0$) and that if a leaf-node is at a certain hierarchical-level $\mathcal{B}_L^h$ with $L \neq 0$, its least common ancestor is itself (*i.e.*, if $c_i^0 \in \mathcal{B}_L^h, LCA(c_i^0) = c_i^0$). The SPV by *grouping* can not only be applied on semantic hierarchies but also on hierarchies constructed based on data, through clustering. In MulDIP-Net, we thus can consider three three types of grouping SPV, the first having our preference and the two other being considered as baselines: (i) Categorical grouping; (ii) Hierarchical grouping; and (iii) Clustering grouping. This latter results in a final set of SPs denoted $\mathcal{D}_\Omega$ containing the initial SP as well as the resulting SPs of the three SPV by *grouping*.

4.1.2.3 MulDiP-Net Training

Let us consider a set of source problems $\mathcal{D}_\Omega^{\mathcal{S}} = \{\mathcal{D}_0^{\mathcal{S}}, \mathcal{D}_1^{\mathcal{S}}, \cdots, \mathcal{D}_\omega^{\mathcal{S}}\}$, with $\mathcal{D}_0^{\mathcal{S}}$ being the initial SP and $\mathcal{D}_{k>0}$, being the ω SPs obtained from ω different SPVs functions applied on the initial SP. MulDiP-Net consists in training one network per source problem, including the *initial* SP. Let us note that each SP can be learn with a different CNN or with different learning settings since our method does not depend on this aspect. In the following, we will nevertheless use the same CNN for all the SPs, in order to avoid this source of variation for their study.

Specifically, we use common CNN architectures (AlexNet [132], VGG [228] and DarkNet [212]) and follow a standard learning procedure that consists in a random initialization of the weights (Gaussian distribution), a softmax loss-function after the last layer, and asynchronous SGD to optimize it. Using the softmax loss-function, the posterior probability of an image x_i^k given a category c_j^k for the source-problem $\mathcal{D}_{k\in[\![0,\omega]\!]}^{\mathcal{S}}$ is thus:

$$p_{ij}^k = \frac{\exp(\Psi_k^j(x_i^k))}{\sum_{n=1}^{C_k} \exp(\Psi_k^n(x_i^k))}, \tag{4.5}$$

where $\Psi_k^j(x_i^k)$ is the j^{th} dimension of the output of the last fully-connected layer of the network and the dimensionality of $\Psi_k(\cdot)$ is equal to the number of categories in the set of categories $\mathcal{C}_k$ (*i.e.*, C_k). Thus, assuming that the ground-truth probability for image x_i^k and class c_j^k at SP $\mathcal{D}_k^{\mathcal{S}}$ is defined as $\overline{p_{ij}^k}$,

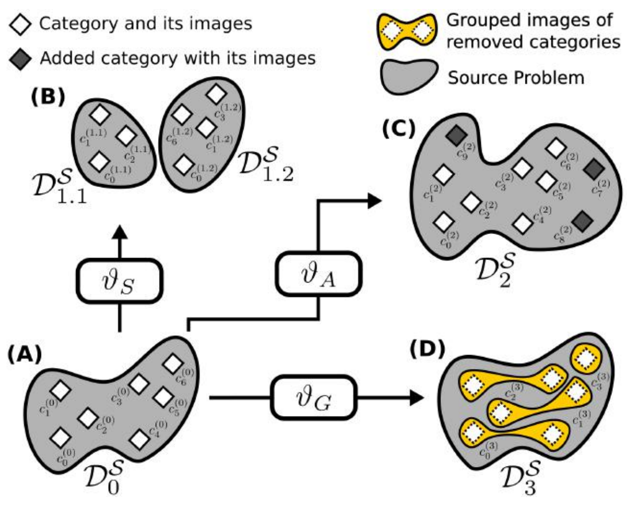

Figure 4.3: Illustration of the three kinds of source problem variations (SPVs): *Splitting*, *Adding* and *Grouping*. An initial source problem (SP) $\mathcal{D}_0^S$ is illustrated in (A). See the legend on top of the figure for a description of each graphical element. (B) represents the output of the SPV by splitting (ϑ_S) which results in two diminished sets of training-data (each of them contains less images and categories) compared to the initial SP $\mathcal{D}_0^S$ – *i.e.*, $N_i < N_0$ and $|\mathcal{C}_i| < |\mathcal{C}_0|$, with $i \in \{1.1, 1.2\}$). In contrast, as illustrated in (C), aSPV by *adding* (ϑ_A) results in an increased set of training data (more images and categories), compared to $\mathcal{D}_0^S$ (*i.e.*, $N_2 > N_0$ and $|\mathcal{C}_2| > |\mathcal{C}_0|$)). The last example is our SPV by *grouping* (ϑ_G), illustrated in (D). It results in the *same amount* of training-data (same set of images but labeled according different but less numerous categories), compared to $\mathcal{D}_0^S$ (*i.e.*, $\{x_i^3\} = \{x_i^0\},\ \forall i$ and $|\mathcal{C}_3| < |\mathcal{C}_0|$). Best view in color.

the cost function is:

$$\mathcal{L}_k(\Theta) = -\frac{1}{N_k} \sum_{i=1}^{N_k} \sum_{j=1}^{C_k} \overline{p_{ij}^k} \log(p_{ij}^k), \tag{4.6}$$

where, N_k and C_k are respectively the amount of training images and amount of categories in the SP $\mathcal{D}_k^S$ and Θ is the whole set of learnable parameters of network $\mathcal{N}_k$. Note that, each cost function $\mathcal{L}_k(\Theta)$ is minimized *independently*. Moreover, since the SPV functions are based on grouping, all the SPs contain the same images as the initial SP (*i.e.*, $\{x_i^k\} = \{x_i^0\}, \forall i \in [\![1, N_k]\!]$ and $\forall i,\ N_0 = N_k$) and thus each cost function is minimized on the same set of training images, but with different labels. At convergence, we obtain a set $\mathcal{N}_\Omega^* = \{\mathcal{N}_0^*, \ldots, \mathcal{N}_\omega^*\}$ of ω+1 trained networks.

4.1.2.4 MulDiP-Net Representation

Let us consider a MulDiP-Net (*i.e.*, a set $\mathcal{N}_\Omega^* = \{\mathcal{N}_0^*, \ldots, \mathcal{N}_\omega^*\}$ of ω+1 trained networks) and an image I_i^τ of a target-task T_τ. The process of extraction of the representation $\mathcal{R}_{i,\tau}^\Omega$ from an image I_i^τ, through the MulDiP-Net, consists in two steps: (i) extraction of the features of the image from each subnetwork $\mathcal{N}_k^*$ of MulDiP-Net and (ii) dimensionality reduction, independent normalization and combination of these features into a more relevant representation. Let $\phi_k^K(\cdot)$ be the feature vector extracted from the K^{th} layer of the k^{th} learned network. It is thus a scoring function of the image I_i^τ that produces a vector of activations (if K is a fully-connected layer) or a tensor of activations (if K is a convolutional layer). In the latter case, the tensor is flattened to get a vector. The MulDiP-Net

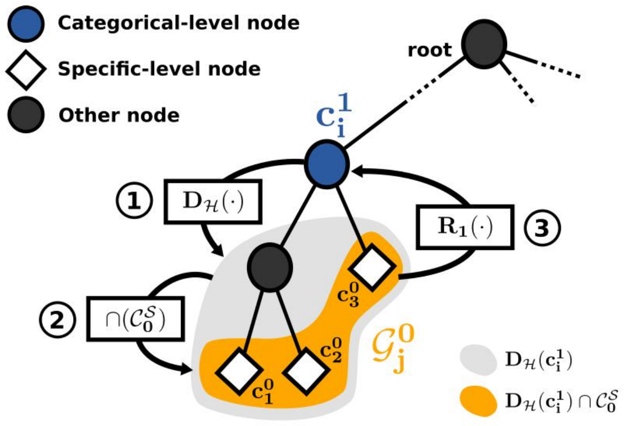
Categorical-level node
Specific-level node
Other node
root
$\mathbf{c_i^1}$
① $\mathbf{D}_{\mathcal{H}}(\cdot)$
② $\cap(\mathcal{C}_0^{\mathcal{S}})$
③ $\mathbf{R_1}(\cdot)$
$\mathbf{c_1^0}$
$\mathbf{c_2^0}$
$\mathbf{c_3^0}$
$\mathcal{G}_\mathbf{j}^\mathbf{0}$
$\mathbf{D}_{\mathcal{H}}(\mathbf{c_i^1})$
$\mathbf{D}_{\mathcal{H}}(\mathbf{c_i^1}) \cap \mathcal{C}_0^{\mathcal{S}}$

Figure 4.4: Detailed illustration of our SPV by *grouping*. Given a set of specific categories (leaf nodes of the hierarchy in white diamonds) and a set of generic categories (here c_i^1) at a certain level (here categorical denoted $\mathcal{B}_L^{cat}$), our SPV by *grouping* ϑ_G consists in three steps: (1) computation of all descendants of c_i^1 according to the hierarchy $\mathcal{H}$, (2) computation of the descendants that belong to the initial set of categories, producing a group $\mathcal{G}_j^0$, and (3) re-labeling of the categories (as well as their images) of the latter group into the categories of $\mathcal{B}_L^{cat}$. The first two points correspond to Equation (4.3), while the last one corresponds to Equation (4.4).

representation for the query image I_i^τ is thus:

$$\mathcal{R}_{i,\tau}^{\Omega} = \mathcal{F}_{k\in[\![0,\omega]\!]}\Bigg(\mathcal{Z}\Big(\mathcal{S}\big(\phi_k^K(I_i^\tau)\big)\Big)\Bigg), \tag{4.7}$$

where $\mathcal{F}$ is a fusion operator of ω+1 input vectors, $\mathcal{Z}$ is a normalization function, and $\mathcal{S}$ is an operator that reduces the representation dimension. In practice for the fusion, normalization and dimensionality reduction functions, we respectively used the concatenation ($\bigoplus_{k\in[\![0,\omega]\!]}$), the L_∞ norm and a proposed learning-based dimensionality-reduction method called FSFT (see Section 4.1.2.5).

We aim at combining features separately trained on specific and generic labels to boost the performances, assuming they are complementary. However, we noticed that features that were learned on specific data tend to produce higher values than those learned on generic data. Thus, naively concatenating these sub-representations would bias the final representation toward the dominating features (*i.e.* learned on specific data) which would result into a loss of the expected benefit. For this reason, the normalization step is crucial, because it homogenizes the scales of the sub-representations. The same problem of dominating values has been observed in [148, 272], and a normalization process was also used.

Regarding the fusion function, we could use a pooling (average or maximum) instead of the concatenation. However, by construction, it forces the output of *many* neurons values to be summarized by *one* value. Such behavior is problematic when the pooling is applied to neurons that correspond to semantically different patterns. For example, let us imagine two representations having the first dimension being a neuron highly activating on *cars* and another firing on *apples*, and let us consider an input image containing an apple on top of a car. The pooled representation will be non-discriminative for cars and apples, because the pooled neuron is activated when the input images contain *apples*, *cars* or both. In contrast, the concatenation fusion preserves all the information of the sub-representations in the final one. For the dimensionality-reduction part, we detail the proposed FSFT method in Section 4.1.2.6.

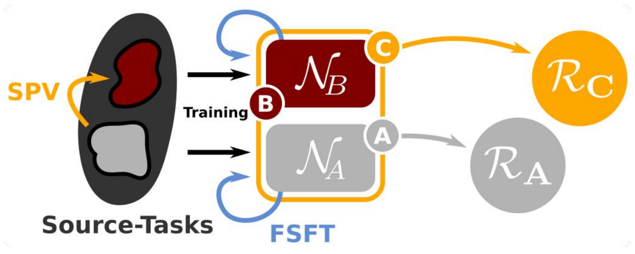

Figure 4.5: The MulDiP-Net method consists in three modules: (i) source problem variation by grouping (yellow arrow on source-tasks) that aims to get a new dataset (red blob on source-tasks) which contain the same images as the initial source-problem (SP) (gray blob on source-tasks), but with generic labels; (ii) training of multiple networks, in particular, one on the initial SP ($\mathcal{N}_A$) and one on the new SPs (here $\mathcal{N}_B$ only); and (iii) combine the features-extractors A and B to form the new extractor C (yellow block that covers A and B). Indeed, for this last point, each features-extractor is independently normalized to prevent the abusive behaviour of one from absorbing that of the others. Moreover, each features-extractor (A and B) is re-trained on its initial SPs (*i.e.*, gray SP for $\mathcal{N}_A$ and red SP for $\mathcal{N}_B$), through the proposed FSFT method (blue arrows). The final features-extractor C aims to get a more universal representation $\mathcal{R}_C$ than each of its components ($\mathcal{R}_A$ and $\mathcal{R}_B$), at the same amount of features and with the same annotated data.

In practice, when $\mathcal{N}_\Omega$ contains the models obtained from the initial SP as well as three different grouping SPVs (categorical, hierarchical and clustering-based), Equation (4.7) reduces, independently normalizes and concatenates four representations to get the MulDiP-Net representation. Figure 4.2 illustrates the training of MulDiP-Net as well as the extraction of the final representation.

4.1.2.5 Focused Self Fine-Tuning (FSFT)

We propose a new universalizing method that we named Focused Self Fine-Tuning (FSFT). It is based on the principle of *re-training* pre-trained neural networks on the *same* initial problem. Contrary to previous works, it thus does not need more data [11, 166, 246], nor a network with a larger capacity [228, 2, 272, 243]. It is closely related to the work of [285] which proposes an extensive study of the effect of different *self-training* methods. In particular, they studied two methods: (i) Frozen re-training (that we named Frozen Self-Training, noted **FrST**) and (ii) fine-tuning (that we named Self Fine-Tuning, noted **SFT**). Both methods re-train a pre-trained initial Network *initNet* on the same problem. The weights of *initNet* are split into two subsets, those of the first L layers θ_1^a and those of its last layers θ_2^a, thus $\Theta^a = (\theta_1^a, \theta_2^a)$. The methods consist in learning new weights (θ_1^b, θ_2^b) with the following process: (i) the first layers are initialized with the weights of *initNet* and the last layers are initialized randomly (ii) the weights are re-trained – FrST trains only θ_2^b and frozes θ_1^b, while SFT fine-tunes θ_1^b and fully trains θ_2^b, with the same learning-rate. Their extensive study leads to some interesting conclusions that motivates our approach. In particular, they showed that FrST hurts the performances of *initNet* due to *fragile co-adapted features on successive layers* in the initNet that suggests that the co-adaptation could not be relearned by the upper layers alone. On the contrary, SFT slightly increases the performance due to its ability to recover the co-adapted features that were trained by the initNet. Hence, it is important to preserve some knowledge acquired during the learning of the *initNet* but one should avoid to *only* focus the training on the last layers. As a consequence, Focused Self Fine-Tuning (FSFT) is an hybrid approach of these both. As in [285], the weights learned on *initNet* are split into two subsets, initialized differently and jointly minimized. The main difference is that we use *different learning rates* for both sets. An illustration is given in Figure 4.6.

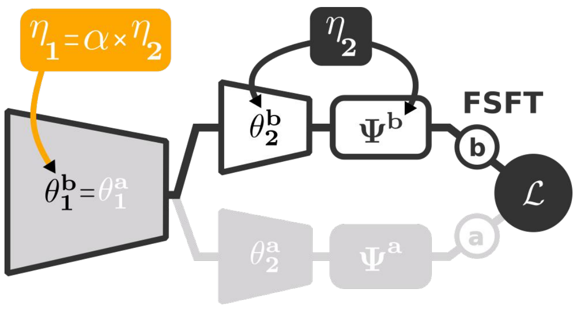

Figure 4.6: Illustration of Focused Self Fine-Tunning (FSFT). A first training phase is performed on a source-database, by minimizing a classical loss-function $\mathcal{L}$ (stream (a) colored in light gray). FSFT first discards its last set of layers (θ_2^a and Ψ^a) and replace them by new ones (θ_2^b and Ψ^b) that are randomly initialized. The first layers of stream (a) and the new last layers (that replace those discarded), form the stream (b). FSFT then consists to re-train the weights of the initial network (*i.e.*, stream (a)) on the *same* source-database (with the same loss-function $\mathcal{L}$ to minimize). More precisely, FSFT initializes the weights of its first layers θ_1^b (light gray block surrounded in black) with those of the initial network (*i.e.*, $\theta_1^b{=}\theta_1^a$ at initialization), and the remaining last layers are initialized randomly. Then, FSFT *trains* the weights of the last layers θ_2^b (white blocks surrounded in black), while *re-training* those of the first layers (θ_1^b) but α-times slower. In particular, the weights of the first layers θ_1^b are *weakly* updated during training, compared to those of the last layers – *i.e.*, θ_2^b are trained with a learning-rate η_2 that is *higher* than the learning-rate of θ_1^b (*i.e.*, $\eta_1 = \alpha \times \eta_2$, with $\alpha \in]0, 1[$). Best view in color.

More formally, let us consider a source training database $\mathcal{D}^{\mathcal{S}}$ containing N annotated data and a network $\mathcal{N}$ trained on $\mathcal{D}^{\mathcal{S}}$ to learn its weights θ_1^a (L first layers) and θ_2^a (remaining last layers). FSFT consists in re-training $\mathcal{N}$ by minimizing the same loss-function $\mathcal{L}$ as *initNet* and the same training database $\mathcal{D}^{\mathcal{S}}$:

$$\arg\min_{\Theta^b} \mathcal{L}\Big(\Psi^b\big(\phi(\mathbf{x}, \Theta^b)\big); \mathbf{y}\Big), \text{with } \Theta^b = (\theta_1^b, \theta_2^b)), \tag{4.8}$$

where, $\mathbf{x}$ and $\mathbf{y}$ respectively correspond to the learning images and their associated labels, $\Psi^b(\phi(\mathbf{x}, \Theta^b))$ is the predicted probability vector for the features $\phi(\mathbf{x}, \Theta^b)$ of images $\mathbf{x}$, and θ_1^b are initialized with the weights of the first layers of the initNet (*i.e.*, $\theta_1^b{=}\theta_1^a$) and those of the last ones θ_2^b are initialized randomly. Then, the individual weights $w_{ij}^b \in \theta_j^b$ (with $j \in \{1, 2\}$) are updated with different learning-rates:

$$w_{ij}^b \leftarrow w_{ij}^b - \eta_j \frac{\partial \mathcal{L}}{\partial w_{ij}^b}, \forall w_{ij}^b \in \theta_j^b, \tag{4.9}$$

where, η_j is the learning-rate. We set $\eta_1 = \alpha \times \eta_2$, with $\alpha \in [0, 1]$ a parameter that can be determined by cross-validation. Hence, FSFT preserves some knowledge acquired during the learning of the initNet (by initializing the weights of its first layers, with those of initNet) and *partially* focuses the training on the last layers (by fully training the last layers, while authorizing, through the factor α, some slight changes of the first ones).

4.1.2.6 FSFT to improve MulDiP-Net

The FSFT method can be used in the MulDiP-Net method for dimensionality and parameters reduction. In MulDiP-Net the more source-problems we consider, the more universal representation we get. However, a concatenation fusion results in increasing the dimension of the final representation

Property	**Avg**	**RG**	**VDC**	**BC**	**aNRG**	**mNRG**
Coherent aggregation		✓	✓	✓	✓	✓
Merit bonus			✓		✓	✓
Penalty for damage	✓			✓	✓	✓
Independent to outliers				✓		✓
Increasing & decreasing metrics				✓	✓	✓
Independent to reference method	✓			✓		
Independent to many methods	✓	✓	✓		✓	✓
Consistent with time	✓	✓	✓		✓	✓

Table 4.1: Comparison of different metrics (the Avg baseline, RG [235], VDC [210] and the three proposed in this Thesis, namely BC, aNRG and mNRG) according the different properties needed to define a good evaluation metric.

linearly with the number of SPs considered. MulDiP-Net being a kind of ensemble approach, it contains ω+1 *times* more parameters than a standard network and it contributes to obtain a more universal representation. There is nevertheless a need to find a trade-off between effectiveness and efficiency and the FSFT method can help to it. Roughly, our method consists in applying a slightly modified version FSFT to each subnetwork of MulDiP-Net, such that it results into a representation that is both more compact while leading to similar or better performance in transfer learning.

More formally, let us consider a feature extractor ϕ_k trained on a SP noted $\mathcal{D}_k^{\mathcal{S}}$, and another feature extractor $\phi_k^{'}$ obtained after the retraining of ϕ through a slightly modified version of the FSFT method. Indeed, instead of retraining ϕ with exactly the same original architecture, we reduce the dimension of the last layers on which FSFT will focus its training. It thus also decreases the amount of parameters of the final MulDiP-Net model, in particular because most of the usual CNN parameters are contained in the fully connected layers. For instance, AlexNet contains 62.3 millions parameters with 3.7 millions in the convolutional layers and 58.6 millions in the FC ones, meaning that around 94% of the parameters are contained in the FC layers. To reduce the dimension, given a set of ω+1 networks, we set each sub-representation $\phi_k^{'}$ to be of size $\lceil n/(\omega+1)\rceil$, with n the dimension of the original sub-representation ϕ_k. Regarding the performance improvement, it can be surprising because we expect to boost the performances by considerably reducing the dimensionality of the representation. Indeed, this is possible because of the effectiveness of the FSFT method compared to the standard learning procedure and thus we can suggest that a *compact* version of it will still be more performing (even by a weak gain) than the standard version. Furthermore, MulDiP-Net will follow the latter process by the combination of multiple compact representations that independently contribute weakly. At the end, the new MulDiP-Net+FSFT method contains much lower amount of parameters and outputs a much smaller representation than the classical MulDiP-Net method, while being more performing.

4.2 Evaluation of Universalizing Methods

Evaluating the ability of a method to learn universal representations or more precisely to learn more universal representations is a difficult problem that, to our knowledge, has not been much tackled in the literature. Let first recall that our problem of universality is motivated by the cognitive study of Atkinson [8] and especially its main claim, that is to say, the ability of humans to develop a universal and powerful internal representation of images in the early years of their development and re-use it (almost) "as is" later in life for solving any kind of problems. Inspired by the latter work, authors of [23, 210, 211] evaluated the universality of representations by their ability to cover, simultaneously, a large range of visual domains like objects, faces, animals, etc. More precisely, the evaluation scheme of [23, 210] does *not* completely match with the work of Atkinson since learning and testing are conducted on the *same problem* and only the *visual domain* differs. However, in [8], the terms "re-used later in life" mean that a representation is universal if it works well on many number of problems encountered latter in life, and in particular, on *different problems* than those used to develop (learn) the representation. Hence, in this Thesis, we propose to consider the case where the training problem is *different* than the testing one. A natural technical scheme for such condition is the transfer-learning scheme, where the *source-task* is used to learn the universal representation (analogically corresponds to the universal and powerful internal representation developed in the early years of the development) and the *target-tasks* are used to evaluate it (analogically corresponds to the problems solved by humans later in life). Moreover, to better match the claim of Atkinson (re-use the representation "as-is"), we propose to *not* modify the representation on the target-tasks. Simply said, *we do not fine-tune* the representation on each target-task, but rather learn a simple task-predictor *on top of* the representation. Nevertheless, since when humans develop their representations, they do not have access to the problems they will solve in the future, the target-tasks used to evaluate universality are *undetermined* when learning on the source-tasks. In summary, the substantial difference with the evaluation scheme of [23, 210] and ours lies in the fact that, in contrast to them, we consider the aspect of re-using the representation later in life *on different problems*. Simply said, our evaluation framework that lies in a Transfer-Learning (TL) scheme is closer to [8] than theirs [23, 210, 211] that lies in an End-to-End Learning. Note that, while such evaluation framework (transfer-learning on multiple target-tasks without modifying the representation) is commonly used in the NLP community [44, 42, 41, 43, 180, 235] for evaluating universality of representations, to the best of our knowledge, we are the first to propose such scheme in the vision community, and more importantly to motivate and link it with the cognitive study of Atkinson [8]. An illustration of the proposed evaluation scheme is given in Figure 4.7.

The last step to evaluate universality, consists in aggregating the scores of a method on multiple target-tasks. One could naively aggregate the scores by averaging them. Given a set of T target-tasks, the **Average (Avg)** universality score U_i^{avg} of a method M_i is: $U_i^{avg} = \frac{1}{T}\sum_{j=1}^{T} s_j^i$, with s_j^i being the performance of the method M_i on the target-task T_j. However, a first problem is the fact that metrics of all the target-tasks could be different (many will prefer to use measures such as precision or recall, or an aggregative of both with a F-measure of average precision), and such aggregation would be non-coherent. Hence a first requirement is a *coherent aggregation*. To alleviate such problem, Subramanian *et al.* [235] proposed to average the improvement over the scores of a reference universal representation across the set of multiple target-tasks (named **relative gain (RG)**): $U_i^{avg} = \frac{1}{T}\sum_{j=1}^{T} s_j^{ref} - s_j^i$, with s_j^{ref} being the performance of a reference method M_{ref} on the target-task T_j. Such aggregation brings us to a coherent aggregation, but unfortunately depends on the scores of a reference method that needs to be chosen and evaluated on all target-tasks before evaluating universality. This pushes us to define a metric that is *independent to a reference method*. Recently, Rebuffi *et al.* [210] went a bit far by identifying one additional criterion that should be also

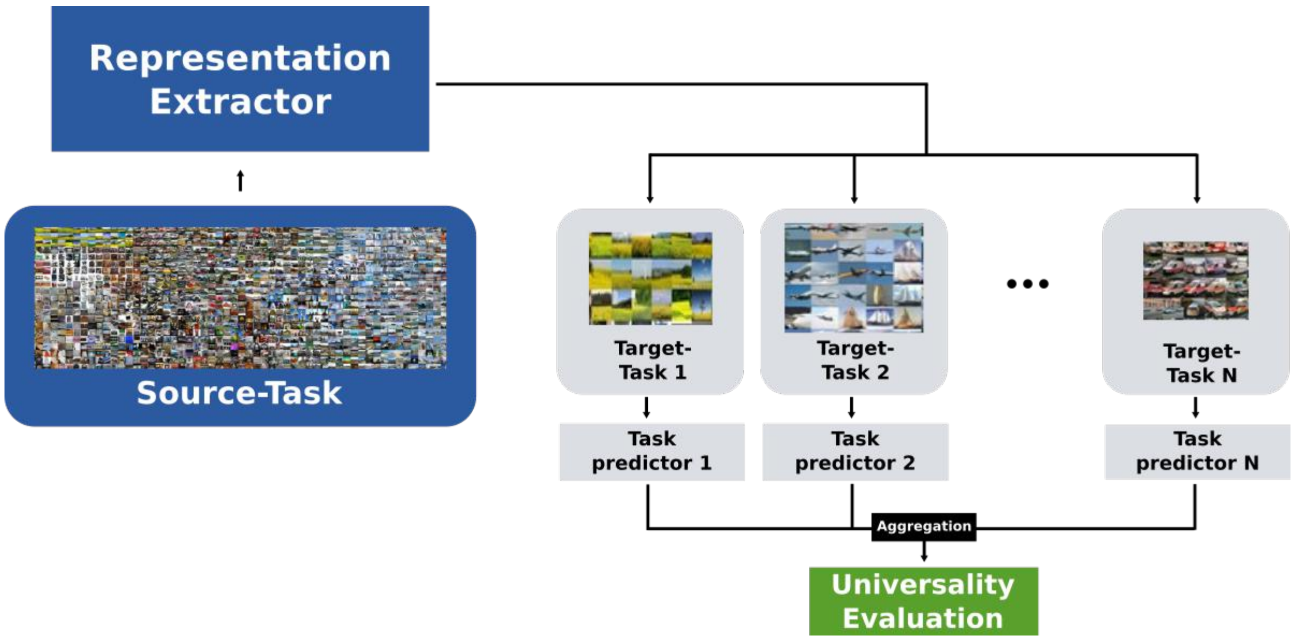

Figure 4.7: Illustration of the proposed universality evaluation scheme. A universal representation extractor (blue rectangle) is learned on one or many source-task(s) (blue blob). The extractor is used to represent the data of a set of N undetermined (unknown during the training on the source-task) target-tasks (gray blobs). For each target-task, a simple task-predictor (gray rectangles) is learned on top of the representations (without modifying the representation). A function aggregates (black rectangle) the performances of a method on the multiple target-tasks to evaluate universality (green rectangle). Best view in color.

considered, that is, the universality evaluation metric should rewards proportionally with the score of the reference method. More concretely, the metric should rewards more if the reference score is high and less if it is low. We call such criterion *merit bonus*. Hence, they proposed a metric named **Visual Decathlon Challenge (VDC)** that also consider this aspect. More precisely, the VDC universality score is computed as follows: $U_i^{VDC} = \sum_{j=1}^{T} \alpha_j \max\{0, E_j^{max} - E_j\}^{\gamma_j}$, with E_j being the error rate of the method M_i on the target-task T_j and E_j^{max} being the best reachable error rate on T_j, α_j is fixed for all T_j but his role is to normalize the methods by providing 1,000 point for a perfect ones, and finally the γ_j (with $\gamma_j \geqslant 1$) is here to model the merit bonus aspect. Undoubtedly, the principle of such a metric has an interest to the problem of universality evaluation, however, it is important to note that the γ value is heuristically set to 2 in practice, and thus do not naturally model the merit bonus aspect. More problematically, modeling the merit bonus by a power value (with necessarily $\gamma_j \geqslant 1$ since in the contrary it would do the inverse of merit bonus) forced the authors to put a max operator to avoid negative values. The latter operator clearly avoid negative values and aims at modeling the merit bonus, but a main drawback, that is to say, it avoids to perform penalty for damage scores which is problematic. Indeed, a method can improve the performance over the reference method on some target-tasks but could *decrease* it on other tasks. Thus, is it highly desirable to have a metric that performs *penalty for damage*. Hence, in this Thesis we proposed the **average normalized relative gain (aNRG)**: $U_i^{aNRG} = \frac{1}{T}\sum_{j=1}^{T}(s_j - s_j^{ref})/(s_j^{max} - s_j^{ref})$, with s_j^{max} and s_j^{ref} being respectively the best reachable and reference scores. A last important point is the fact that the metric should be *independent to outlier methods*, in the sense that a method that is particularly suited to a given benchmark will gain a lot of points that could compensate an average performance on the others, which is clearly non desirable on universality. While all the metrics (including our aNRG) do not handle this aspect, we propose to replace the *average* operator by a *median* one, which results in the **median**

Normalized Relative Gain (mNRG): $U_i^{mNRG} = \underset{j \in [\![1,T]\!]}{\text{median}} \sum_{j=1}^{T} (s_j - s_j^{ref})/(s_j^{max} - s_j^{ref})$.

Finally, while not obviously better than the proposed mNRG, we propose an alternative metric based on a voting method and on the ordinal scale, with each benchmark considered as an independent voter. The first advantage of this metric is that each voter can use its own measure to estimate the performance of the methods, depending on the need they focus on. In particular some benchmarks can use "the higher is the better" measures (mean average precision, accuracy, F1-score, etc.) while others can use "the lower is the better" (test error rate, equal error rate, median rank, etc.). The second advantage is that it becomes insensitive to some outliers such as the exceptional fit of a method to a particular benchmark. Moreover, we consider that reporting the number of times a method M_1 is better than a method M_2 is a more reliable information than the average difference of scores (as long as these score are actually comparable). It exists many ordinal voting systems to choose a best candidate among several, according to several voters, but we adopt one of the simplest ones, namely the Borda Count (BC). Formally, let us consider M methods to rank, relying on the information provided by B benchmarks. Each method is then ranked according to each benchmark, resulting into a rank r_i^j with $1 \leq i \leq M$ and $1 \leq j \leq B$. It is converted into a score $M - r_i^j$ that is itself averaged to give the final score of the method: $S_i = \sum_{j=1}^{B} M - r_i^j$.

In summary, no metric is perfect, thus in Table 4.1 we highlight the advantages and drawbacks of each of the metrics (those of the literature and those proposed in this Thesis), in which we see that the proposed mNRG seems to the best one. Moreover, in Table 4.5 of the experiments we evaluate different methods according the different metrics, in which we empirically highlight the problems of some of them, which are alleviated by our best metric, namely mNRG.

4.3 Experimental Results

4.3.1 Learned Features Analysis

In this Section, we investigate whether each component of MulDiP-Net – *i.e.*, each sub-network trained on images labeled at a particular set of categorical-level labels – introduces some relevant diversity in the final set of features. We first analyze the intermediate layers of the different pre-trained sub-networks in order to show that each of them generates different features (Section 4.3.1.1). However, since diversity does not mean relevance, we also analyze how relevant is this difference (Section 4.3.1.2).

4.3.1.1 Do sub-networks of MulDiP-Net learn different features?

Two sub-networks are trained using the AlexNet network on the same training-images (ILSVRC*) for two categorical-level labels, *i.e.*, *basic*-level (containing *generic* categories) and *subordinate*-level (containing *specific* categories), that we respectively denote by **Net-G** and **Net-S**. Following Li *et al.* [143], we statistically compare the internal convolutional-filters of the two networks. Thus, we first aggregate certain statistics of the activations within the networks. Given a pre-trained network Net-n, we denote by the scalar random variable $\mathcal{X}_{l,i}^{(n)}$ the activation values produced over a large set

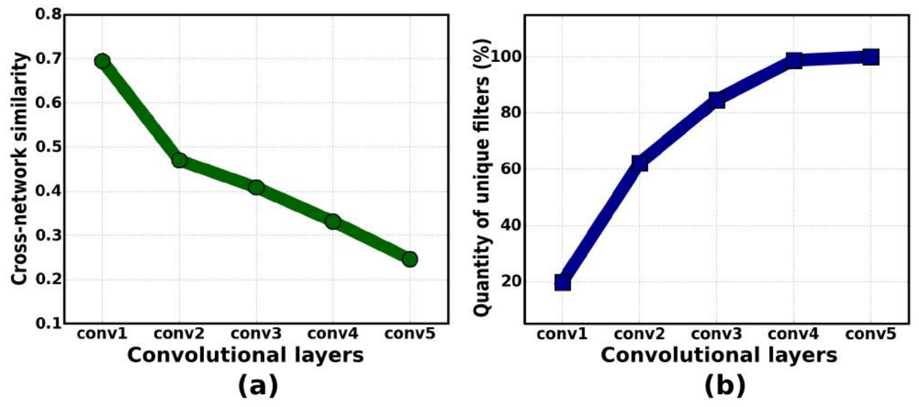

Figure 4.8: Graph (a) reports the average of maximum correlation between convolutional-filters of Net-G and Net-S (*i.e*, similarity between the two networks) according the layers. In (b), we plot the average quantity of *unique filters* (*i.e.*, that do not match with any filter of the other network) in the two networks according the layers.

of samples[1] by convolutional-filter i on layer $l \in \{conv1, \cdots, conv5\}$. From this set of samples, we collect for all filters, the average $\mu_{l,i}^{(n)}$, the standard deviation $\sigma_{l,i}^{(n)}$ and the "cross-network correlation":

$$c_{l,i,j}^{(n,m)} = \mathbb{E}[(\mathcal{X}_{l,i}^{(n)} - \mu_{l,i}^{(n)})(\mathcal{X}_{l,j}^{(m)} - \mu_{l,j}^{(m)})]/\sigma_{l,i}^{(n)}\sigma_{l,j}^{(m)}, \tag{4.10}$$

corresponding to the correlation of the random variable of a filter of a network Net-n with the one of another network Net-m at the same layer l. We thus obtain five asymmetric matrices per network, of size 96×96 for $conv1$, 256×256 for $conv2$ and $conv5$ and 384×384 for $conv3$ and $conv4$. From these matrices, we look for the filters of Net-S that match the most with those of Net-G (and inversely) to show the similarity between the networks. To show the amount of unique filters generated in each network, we look for the filters that do not have any matching filter in the other network. Regarding the cross-network similarity, we compute the average of maximum cross-network correlation (*e.g.*, the average of the maximum of each row and each column of the cross-network correlation matrices), for each layer and display the results on the graph (a) of Figure 4.8. As in [143], for the amount of unique filters, we compute the relative percentage of filters that do not match with any filter in the other network – *e.g.*, the maximum cross-correlation of that filter with all those of the other network is above a low threshold of 0.5 – for all the layers and report the results on the graph (b) of Figure 4.8.

First of all, the high similarity (0.71) observed in $conv1$ confirms previous works [143, 285, 291] showing that, whatever the training-database is, the first layers always generate very similar filters (blob and Gabor-like filters). Second, at the other extremity ($conv5$), the cross-network similarity is much lower (0.26) and the quantity of unique filters reached 100%, meaning that a very different feature space has been generated by the two networks. Hence, these two criteria show that two sub-networks of MulDiP-Net begin by generating many common filters but end by learning many unique ones, even if the same training-images have been used. Equally important, graph (4.8b) shows that we have a difference of 20% between the two sub-networks at $conv1$, clearly meaning that the representational divergence of MulDiP-Net begins at the first layer and thus confirms the importance of training the sub-networks independently without sharing any layer between them.

To enhance this study, we visualize some unique filters learned from each sub-network at all convolu-

[1]All the spatial positions (55×55 for $conv1$, 27×27 for $conv2$ and 13×13 for all others) of $2,000$ randomly selected images from the ILSVRC* validation set.

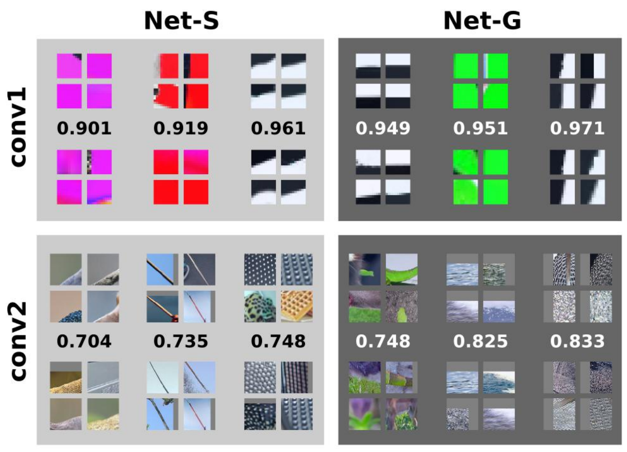

Figure 4.9: Illustration of pairs of convolutional-filters with a high similarity score. Top part contains the representation of filters (through the top-four image-patches that highly activate them) from $conv1$ and the bottom part, those from $conv2$. In each part, there are two blocks: one for the filters from Net-S (left) and one for those from Net-G (right). For each network (each block), we show its filters on top, the most similar ones from the other network on the bottom and their correlation score. We clearly see that filters that are visually-similar have a high correlation score, meaning that the "correlation" is a good similarity-metric to compare convolutional-filters.

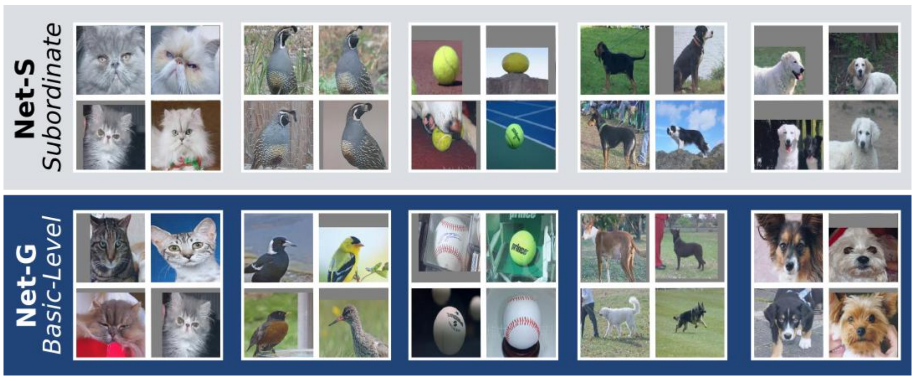

Figure 4.10: Illustration of the top-4 patches that highly activate some unique convolutional-filters of the $conv5$ layer. The top part reports unique filters learned from **Net-S** and the bottom part, unique filters from **Net-G**. Best view in color.

tional layers. To do so, we follow the literature [82, 286, 291]. In particular, for each convolutional-filter, we display, in a two by two block, the top-four image-patches (extracted from the large set of ILSVRC* images) that highly activate them. Before discussing the top-patches that activate the learned filters, it is important to assert that the results highlighted above (*i.e.*, the sub-networks learn different filters) is not due to a bad similarity-metric, thus it is crucial to show that the similarity-metric we used (cross-network correlation [143] outputs a high score (here the maximum absolute value is 1) for filters that are *visually* similar. Thus, we took the two pre-trained sub-networks (Net-S and Net-G) and show the patches that highly activate some of the *most similar* convolutional-filters between the two sub-networks (in term of correlation-metric). These similar filters are presented in Figure 4.9. We only show the filters from $conv1$ and $conv2$ since, above $conv2$-layer the filters are not very similar, thus it does not make sense to show them here. Indeed, from the results, we clearly observe that when the output value of the similarity-metric is high, the filters are highly visually-similar. In $conv2$, the range of the correlation-scores is slightly below the range of values in $conv1$, but it is still relatively high. In fact, we see that some structure-like (*water*, *dotted*, *green-point*, etc.) filters are very visually-similar as predicted by the correlation-metric. This latter clearly shows the high suitability of the correlation-metric to compare convolutional-filters. Thus in Figures 4.10 and 4.11, we respectively show a small ($conv5$-filters) and large amount (from $conv1$ to $conv5$ filters) of unique filters of each sub-network. In Figure 4.10, we clearly observe that Net-S has learned *specific* filters such as breed of objects (*e.g.*, they highly activate only *tennis-ball*, *hairy-cat*, or even *white-dog* patches) while in the Net-G, *generic* filters that are invariant to the breed of objects have been learned (*e.g.*, they highly activate *cat*, *bird*, or *ball* patches). Figure 4.11 that reports more visualizations of unique filters, highlights the five following main points:

- the more we go deeper, the more the filters represent abstract object-parts;
- the two networks (Net-S and Net-G) generate very different filters;
- the more we go deeper, the more correlation-scores of the most similar filters are low. This clearly means that the more we go deeper, the more the filters generated by one network are far from those generated by the other network;
- the filters generated by Net-S are very specific at the deeper layers ($conv4$ and $conv5$), for instance they contain very specific breeds of dogs, very specific breeds of rodent, very specific breeds of birds;
- the filters generated by Net-G are quite generic at the deeper layers, for instance they contain different breeds of dogs, different breeds of birds and different kinds of fruits.

The first point confirms what is already known [286, 291] and the others confirm what is highlighted in the above paragraph – *i.e.*, different filters are generated by different categorical-level networks.

In summary, this analysis shows that solving different discriminative problems (through different categorical-level labels) with a CNN forces the generation of different convolutional-filters. This latter, is quite surprising since the *same* training-images are used in the two discriminative problems.

4.3.1.2 Is the difference between each sub-network of MulDiP-Net relevant?

In the previous section, we have shown that each component of a MulDiP-Net generates different features. However, difference does not mean relevance, thus, it is crucial to evaluate whether this diversity in the set of learned features learned from each sub-network becomes relevant in the final set of features that forms the MulDiP-Net representation. For evaluating the relevance of a descriptor, we follow Peng *et al.* [195] and Herranz *et al.* [98] that estimate the relevance of a set of features by its discriminability on a set of categories. Formally, we define the *discriminability* of a N-dimensional representation $\Phi(\mathbf{x}) = \{\phi_1(\mathbf{x}), \cdots, \phi_N(\mathbf{x})\}$ with respect to a class b_j belonging to a set of M classes $\mathcal{B} = \{b_1, \cdots, b_M\}$ as

$$D(\Phi(\mathbf{x}), b_j) = \frac{1}{N}\sum_{i=1}^{N} I(\phi_i(\mathbf{x}), b_j), \tag{4.11}$$

where $I(\phi_i(\mathbf{x}), b_j)$ is the mutual information of filter $\phi_i(\mathbf{x})$ and class b_j. The filter $\phi_i(\mathbf{x})$ is a continuous variable, thus, as in [195], we use a density estimation method (*e.g.*, Parzen windows) to approximate $I(\phi_i(\mathbf{x}), b_j)$.

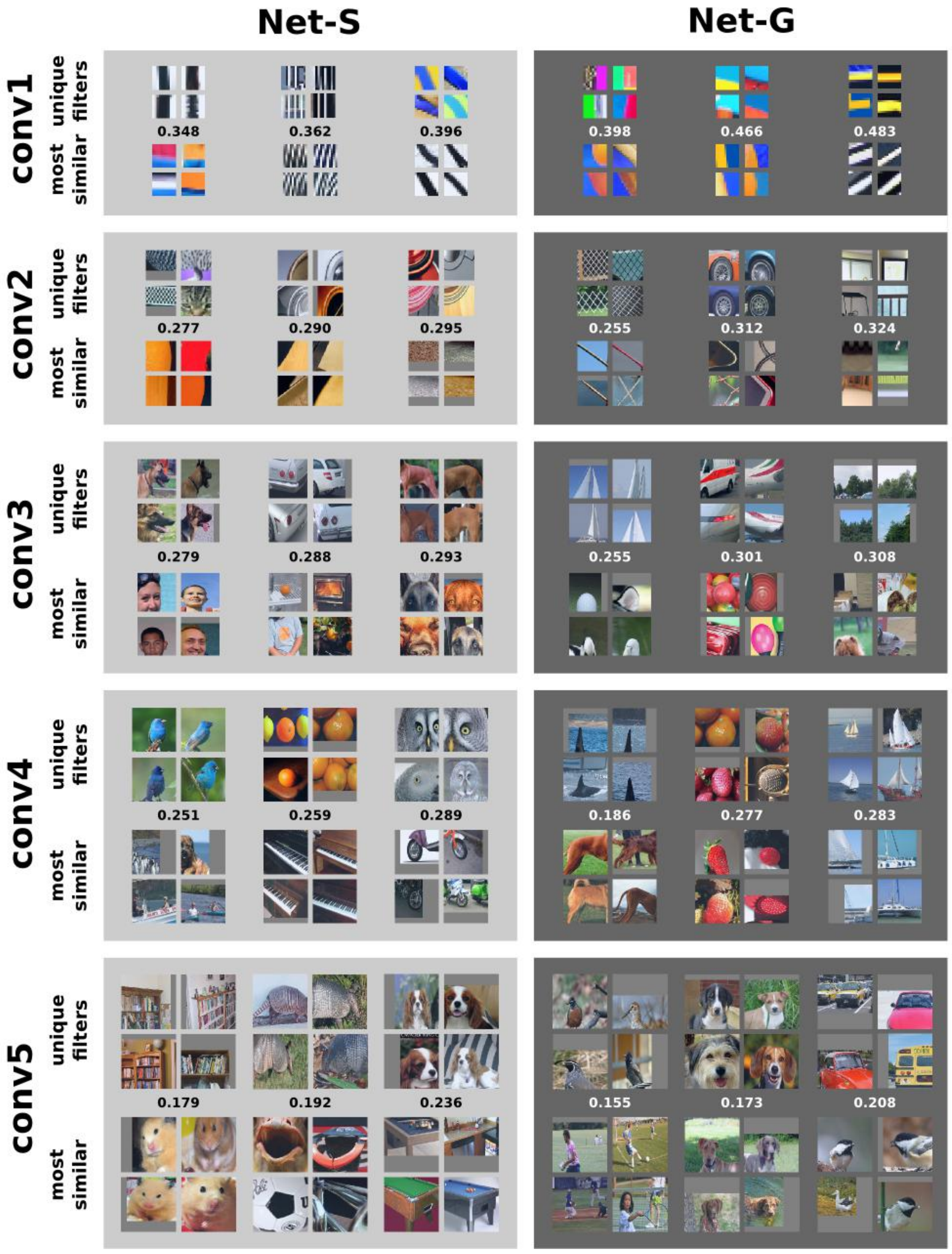

Figure 4.11: Visualization of unique filters generated by Net-S (left) and Net-G (right) for the five convolutional-layers of the AlexNet architecture. In each of the five blocks we represent on the top part, the filters (through the top-four image-patches that highly activate them) from one network (Net-S on the left and Net-G on the right) on one convolutional-layer and on the bottom part, their closest filters from the other network. Between those filters, we indicate their correlation score. Simply said, in each block (convolutional-layer), the three filters on the top-left and the three on the bottom-right belong to Net-S and the three filters on the bottom-left and the three on the top-right belong to Net-G.

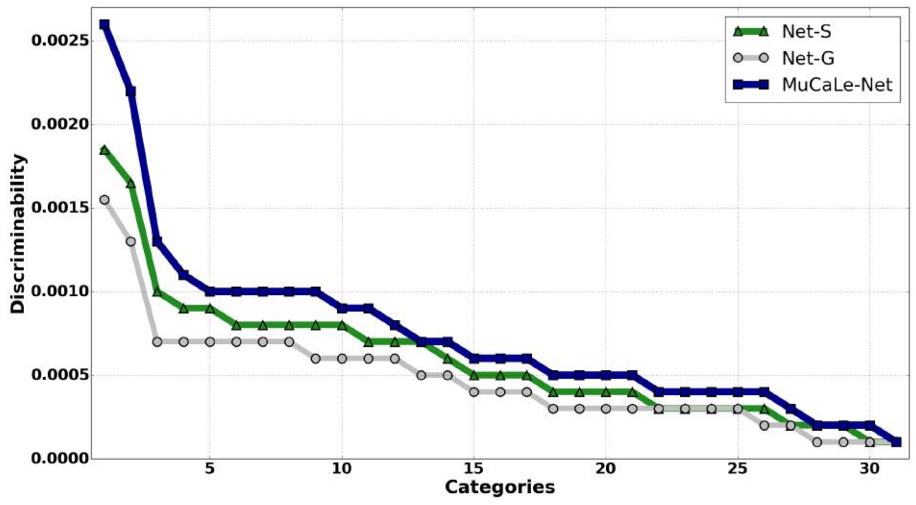

Figure 4.12: Comparison of the discriminability (ordered by decreasing values) of representations generated by Net-G, Net-S and MulDiP-Net, on the categories of the NWO dataset.

In Figure 4.12, we plot the discriminability on the 31 categories of the Nus-Wide Object dataset of the $fc7$ representations generated by three networks: (i) Net-G, (ii) Net-S, and (iii) MulDiP-Net. Note that, the MulDiP-Net representation obtained by (iii) corresponds to the combination of those obtained by the first two. From this graph, we observe that the representation generated by Net-S is roughly more discriminating and thus different than the one learned by Net-G, confirming the results of the previous section. More interestingly, the graph shows that their combination (MulDiP-Net), is more discriminating (for all the categories) than each of them independently, meaning that the difference between the features is highly relevant.

To resume, this analysis shows that CNNs are able to generate different features with the same training-images by varying their categorical-level labels and this difference is highly relevant. This means that the proposed MulDiP-Net method is going on the direction of our main goal, *i.e.*, diversify the set of features generated by CNNs in order to result in more universal representations.

4.3.2 Comparison to State-Of-The-Art Methods

In this section, we describe a transfer-learning protocol for the evaluation, the implementation details of our method, and the datasets used for evaluation. We then compare our method to those of the literature that can be used as universalizing methods.

4.3.2.1 Settings

Transfer-Learning Protocol

All methods are evaluated in a transfer-learning scheme, that contains three phases: (i) training on the source-task, (ii) transferring to the target-task and (iii) evaluating the performances on the target testing set. All the methods are trained with standard architectures (AlexNet [132] if not specified and VGG [228] or DarkNet [212]). The source-tasks (ILSVRC and ILSVRC*) used to train the networks are presented in the next section. If not specified, we use ILSVRC*, because it is smaller

Datasets	(1)	(2)	(3)	(4)	(5)	(6)
ILSVRC*	objects	483	✗	569,000	48,299	Acc.
ILSVRC	objects	1K	✗	1.2M	50,000	Acc.
VOC07	objects	20	✓	5,011	4,952	mAP
VOC12	objects	20	✓	11,540	10,991	mAP
NWO	objects	31	✓	21,709	14,546	mAP
CA101	objects	102	✗	3,060	3,022	Acc.
CA256	objects	257	✗	15,420	15,187	Acc.
MIT67	scenes	67	✗	5,360	1,340	Acc.
stACT	actions	40	✗	4,000	5,532	Acc.
CUB	birds	200	✗	5,994	5,794	Acc.
stCA	cars	196	✗	8,144	8,041	Acc.
FLO	plants	102	✗	1,020	6,149	Acc.

Table 4.2: Detailed descriptive of the different datasets used in this chapter. On top of the table, we describe the source-datasets and on bottom, the target ones. For each dataset, we detail six characteristics. Each column of the table corresponds to a certain characteristic: (1) domain of the images; (2) amount of categories; (3) whether the dataset contains multiple categories per image (✓) or no (✗); (4) amount of training examples; (5) amount of test examples; and (6) the standard evaluation metric (Accuracy and mean Average Precision, respectively denoted by **Acc.** and **mAP**).

than ILSVRC, making the training process faster. For the second transfer phase, each image is represented by one layer of the pre-trained network and each class of the target-dataset is then learned by a *one-vs-all* SVM classifier. The cost parameter of the SVM is optimized for each dataset through cross-validation on the usual train/val splits. Note that, we could use fine-tuning for the transfer part, however, as mentioned by [109], since fine-tuning modifies the representations, it leads to a bias that hides the real ability of a method to increase universality. For the evaluation on the target-tasks, we use the mean Average Precision (mAP) for multi-label datasets and Accuracy (Acc.) for mono-label ones. Finally, the performances of the methods on all target-datasets are globally compared through the *Borda Count* (**BC**) score (Section 4.2).

Source and Target-datasets For all the source and target-problems, we focus on a classification task, with datasets containing many visual domains. For the source-problems, we use two subsets of ImageNet [216]: ILSVRC and ILSVRC* that contains half of the former. Regarding the set of target-tasks, we used the ten following popular benchmarks, including: five containing coarse categories – Pascal VOC 2007 (VOC07) [71], Pascal VOC 2012 (VOC12) [70], Caltech-101 (CA101) [73], Caltech-256 (CA256) [93] and Nus-Wide Objects (NWO) [38]; three containing fine categories – Stanford Cars (CARS) [131], CUB-200 Birds (CUB) [265] and Flowers-102 (FLO) [181]; one containing scenes – MIT Indoor 67 (MIT67) [205]; and one containing actions – Stanford Actions (stAC) [283]. While the main characteristics of all the datasets used in this Thesis are described in Section A.3 of Appendix, we recapitulate, in Table 4.2, the characteristics of those used here. For all the benchmarks, we follow standard protocols (*i.e.*, common splits and evaluation-metrics). For VOC12 and stCAR, we used the official evaluation-servers.

Implementation Details

For the SPV, we started with ILSVRC (and ILSVRC*) as initial source-problem and variated at generic levels with three generic grouping methods: *categorical*, *hierarchical* and *clustering*. For

Subordinate	Basic	Superordinate
convertible - sport car	car	vehicle
helicopter - fighter plane	aircraft	vehicle
barber chair - rocking chair	chair	furniture
malinois - rottweiler	dog	animal
garfish - puffer - sturgeon	fish	animal
hammerhead - tiger shark	shark	animal

Table 4.3: Example of re-labeling of subordinate categories (left column) into basic-level (middle column) and superordinate ones (right column).

the two first methods, we used the ImageNet hierarchy in the functions of partitioning (Eq. (4.3)) and re-labeling (Eq. (4.4)). In the first method, a part of the categorical-level categories of ILSVRC is obtained from the list released in [216] and the other part is re-labeled by ourselves as depicted in Appendix (Section C.2). This latter results into 480 generic and $1,000$ specific categories for ILSVRC, 200 generic and 483 specific categories for ILVRC*. Some examples of relabeling of subordinate-level categories into basic and superordinate-level ones are given in Table 4.3. For the **hierarchical-grouping**, we follow the bottom-up approach of [109] and re-label the categories to higher levels of the ImageNet hierarchy. For the **clustering grouping**, we follow [33] and cluster the data of the categories with a Kmeans algorithm (K from 50 to 300 with step of 50). Regarding the extraction and combination of the features (Eq. 4.7), we used the best label-set per grouping SPV (*i.e.*, basic-level, 7^{th} hierarchical and the clustering with K=100), obtained by cross-validation on the evaluation set of the source-task. The latter choice results in a set of four SPs (including the initial one). For the extraction of the representations of images of target-tasks, we always use the penultimate layer from each subnetwork. To normalize before combining the representations, we used the infinite-norm (L_∞). For the FSFT method, we choose the following settings: $L = 6$, meaning that we focus the retraining on the penultimate and last layers; $\eta_2 = 10^{-2}$, as the learning-rate used to train the initial network; and $\alpha = 0.1$, meaning that we train the last layers 10 times faster than the firsts. Moreover, when used in the MulDiP-Net method, FSFT is implemented with two dimensions: $4,096$ (denoted FSFT-4K) meaning that FSFT is only used to improve performances, and $2,048$ (denoted FSFT-2K) meaning that FSFT reduces the dimension in addition.

4.3.2.2 Comparison with Universalizing Methods

The universalizing methods of the literature considered for comparison are:

- **REFERENCE** [132]: Training a CNN on the initial SP that contains *specific* categories. All other methods include this one. The universality of the methods will thus be measured with regard to this reference, in particular in the Borda Count evaluation.

- $\mathbf{SPV_A^{spe}}$ [11, 297]: A method that consists to perform a SPV by *adding* of 100 new *specific* categories (randomly obtained from the leaf nodes of ImageNet) and thus their $100,000$ images. Simply said, this method corresponds to the learning of one network on images labeled among 583 specific categories.

Method	**VOC07** mAP	**VOC12** mAP	**CA101** Acc.	**CA256** Acc.	**NWO** mAP	**MIT67** Acc.	**stACT** Acc.	**CUB** Acc.	**stCA** Acc.	**FLO** Acc.	**mNRG**
REFERENCE	66.8	67.3	71.1	53.2	52.5	36.0	44.3	36.1	14.4	50.5	n/a
SPV$_{\mathrm{A}}^{\mathrm{spe}}$ [11, 23, 297]	66.6	67.5	74.7	54.7	<u>53.2</u>	37.4	45.1	36.0	13.7	51.9	+1.5
SPV$_{\mathrm{G}}^{\mathrm{gen}}$ [132, 166, 246]	67.7	68.1	73.0	54.3	50.5	37.1	44.9	36.8	14.6	50.3	+1.4
AMECON [33]	61.1	62.1	58.7	40.6	45.8	24.3	32.7	26.1	13.1	36.4	-17.7
WhatMakes [109]	64.0	62.7	69.4	50.1	45.6	33.7	41.9	15.0	12.5	42.8	-7.5
ISM [275]	62.5	65.4	68.8	50.7	28.5	37.9	42.6	34.0	13.3	50.0	-4.3
GrowBrain-WA [272]	68.4	68.3	73.1	54.7	49.3	38.4	46.5	37.5	14.7	54.8	+3.5
GrowBrain-RWA [272]	69.1	69.0	74.8	55.9	50.4	40.0	48.4	<u>38.6</u>	14.8	56.1	+6.0
MuCaLe-Net [243]	<u>69.5</u>	<u>69.8</u>	<u>76.0</u>	<u>56.8</u>	**54.7**	<u>41.3</u>	<u>48.5</u>	35.6	15.7	54.8	<u>+7.7</u>
FSFT (Ours)	67.5	67.4	73.9	55.0	44.6	40.4	47.1	**38.7**	<u>15.8</u>	<u>56.8</u>	+4.0
MulDiP+FSFT (Ours)	**69.8**	**70.0**	**77.5**	**58.3**	47.9	**43.7**	**50.2**	37.4	**16.1**	**59.7**	**+9.8**

Table 4.4: Comparison of our methods (bottom) with **state-of-the-art universalizing-methods** (top). The comparison is carried in a transfer-learning scheme on ten target-datasets, when we report the performances of the methods with the standard evaluation-metrics for each target-dataset and as depicted in Section 4.2, methods are especially compared in terms of their individual scores on each benchmark as well as aggregated scores based on the median Relative Gain (**mNRG**) in the last column (scores in blue). This latter uses a reference method for which we colored their scores in red. For each benchmark as well as for the final mNRG score, we highlight the highest score in bold, and the second one is underlined. All the methods have been learned with the same architecture (*i.e.*, AlexNet) on the same initial SP (*i.e.*, ILSVRC*). For fair comparison reasons, we only used two label-sets in the MulDiP-Net method (*i.e.*, specific and categorical).

- **SPV$_{\mathbf{A}}^{\mathbf{gen}}$** [132, 166, 246]: Same as the previous method but with a *generic* SPV by *adding*, that consists to add 100 new *generic* categories and corresponding $100,000$ new images. This results in the training of one network on 583 specific and generic categories. The generic categories in this method were obtained from random internal-nodes of the ImageNet hierarchy.

- **WhatMakes** [109]: A SPV by *grouping* (SPV$_{\mathbf{G}}^{\mathbf{hl}}$) method of specific categories into generic ones is conducted according to the relabeling of specific categories into internal levels of the ImageNet hierarchy (*i.e.*, hierarchical-levels denoted **hl**). We performed it for all the hierarchical-levels and report the results for the best level (*i.e.*, sixth level starting from the leaf-node's level), selected by cross-validation on the validation set of ILSVRC.

- **AMECON** [33]: This method is similar to the previous one but differs by its type of SPV by *grouping*, that is performed by *clustering* (SPV$_{\mathbf{G}}^{\mathbf{clu}}$). Specifically, all the images of each specific categories are used to compute the average features (from the $fc7$ layer of the pre-trained reference network) for the categories. Then, a Kmeans algorithm is used to cluster this set of category average features. We applied this method with different amount of clusters (*i.e.*, $K \in \{50, 100, 150, 200, 250, 300\}$) and report the method with the best results on the validation set of ILSVRC (*i.e.*, K=100).

- **ISM** [275]: This method consists to train an ensemble-model with N networks, one for each problem $\mathcal{D}^{\mathcal{S}}_{n,n\in[\![1,n]\!]}$ obtained from a SPV by *splitting* (SPV$_{\mathbf{S}}^{\mathbf{half}}$). Here we applied the method with *half* splitting, that is to say, we split the initial source problem in two balanced subsets. We chose $n = 2$ because we limit the ensemble-model methods to a maximum of two networks for fair comparison. Once the networks trained, we normalize and concatenate the features extracted from the two trained CNNs.

- **GrowBrain-WA** [272]: This recent method consists to fine-tune a trained network on the same source problem it was trained initially, by growing the network capacity (wider or deeper layers). The best setting of this work is the width augmented (WA) growing that consists to add $2,048$ neurons to the penultimate layer ($fc7$). We also implemented their normalization and scaling step for the new and old layers, because they insist on its importance for performances. The final representation corresponds to the final $6,192$-dimensional $fc7$ layer.

- **GrowBrain-RWA** [272]: This method is another version proposed by [272] that consists to increase the network capacity recursively. In contrast, to the previous version, it consists to perform a recursive width augmented (RWA) growing capacity, that is to say to increase both the $fc6$ and $fc7$ layers. The best setting they report is to add $1,024$ neurons on the $fc6$ layer and $2,048$ on the $fc7$ one.

- **MuCaLe-Net** [243]: It consists to perform a normalization step followed by the concatenation of the features extracted from two CNN-models, one trained on data labeled according specific categories and another one trained on the data labeled according categorical-levels. It results in a $8,192$-dimensional representation.

The results of this comparison are presented in Table 4.4. A salient observation is that MulDiP-Net+FSFT significantly performs better than all other methods, except for two of the ten datasets. Globally, it has the highest mNRG score, meaning that it is clearly the most performing universalizing method. Another salient observation is that FSFT is quite powerful, especially because it outperforms the methods that consist to increase the data and their annotations (SPV$_A^{spe}$ and SPV$_A^{gen}$) as well as those that increase the capacity of the network (GrowBrain-WA). It is worth noting that

Method	**Avg**	**RG**	**BC**	**aNRG**	**mNRG**
REFERENCE	49.2	0.0	50	n/a	n/a
SPV$_{\mathbf{A}}^{\mathbf{spe}}$ [11, 23, 297]	50.1	0.9	62	2.3	1.5
SPV$_{\mathbf{G}}^{\mathbf{gen}}$ [132, 166, 246]	49.7	0.5	56	+1.4	+1.4
AMECON [33]	40.1	-9.1	17	-20.2	-17.7
WhatMakes [109]	43.8	-5.4	22	-10.8	-7.5
ISM [275]	45.4	-3.8	32	-8.8	-4.3
GrowBrain-WA [272]	50.6	1.4	71	+3.0	+3.5
GrowBrain-RWA [272]	51.7	2.5	87	+5.6	+6.0
MuCaLe-Net [243]	<u>52.3</u>	<u>3.0</u>	<u>92</u>	<u>+7.0</u>	<u>+7.7</u>
FSFT (Ours)	50.7	1.5	76	+3.0	+4.0
MulDiP+FSFT (Ours)	**53.1**	**3.8**	**103**	**+8.6**	**+9.8**

Table 4.5: Comparison of our methods (bottom) with state-of-the-art universalizing-methods (top). Here a special emphasis is given on the *aggregated scores* on all the benchmarks. Indeed, the scores of all the methods on all the benchmarks (those in Table 4.4) are aggregated according the different metrics mentioned in Sec. 4.2. For each metric, we highlight the highest score in bold, and the second one is underlined. Note that, for all the metrics the higher score is the better and for BC, for a set of 11 methods and 10 datasets, the best achievable score is 110, while the worse is 10.

FSFT does not need more data neither more capacity. Thus, it increases universality at zero cost of capacity and annotation. As expected, the methods that consist to increase the data and their annotations (SPV$_A^{spe}$ and SPV$_A^{gen}$) as well as those that increase the capacity of the network (GrowBrain-WA and GrowBrain-RWA) learn more universal representation compared to the reference method. Surprisingly, the ISM [275] method is not as performing as reported in their context. This is certainly due to the fact that it is designed for very large source-problems, and the half-million images used here are not sufficient. Finally, the authors of [109] highlighted that a network trained on data labeled among generic categories is almost as performing as one trained on specific categories, through the evaluation on three target-datasets distributed among two domains (general objects, actions and scenes). However, we observe that on more domains, and especially on fine-grained datasets, the difference with the reference method is much higher.

4.3.3 In-Depth Analysis

4.3.3.1 Comparison with Baseline Methods

In this section, we take further experiments to analyse the performances achieved by the proposed MulDiP-Net and FSFT, through their comparison to several baseline-methods. The comparisons for all the Section 4.3.3 are conducted on eight of the ten target-datasets presented in Section 4.3.2.1. We removed VOC12 and stCAR because the evaluation-servers needed to get the performances of a method are limited to 1 run per day. For fairness in the comparisons, we mainly used two networks in the MulDiP-Net method (*i.e*, initial specific, and basic-level) but others could be used (see Section 4.3.3.2 where we present the impact of the set of label-sets used and Section 4.3.3.3 in which three networks are used). The results are reported in Table 4.6 and an illustration as well as the description of all the baselines is given in Figure 4.13.

In particular, for the MulDiP-Net method, we first assess whether the gain of universality is due to the ensemble-model (denoted **Ensemble**) component. To do so, we compare to a baseline that consists in an ensemble-model with two networks trained on the same specific SP but with different random initializations of the weights. Compared to those of the reference method, the performances of the Ensemble are significantly better on six datasets, and less performing on the two others. In terms of aggregated performances, there is a significant gain of +2.3 in terms of mNRG. The fact that the performances of the Ensemble are only marginally above those of the reference method (1 point of improvement) could be surprising because ensemble-models generally provides more gain of performance. However, it should be noted that here the evaluation is carried in a transfer-learning scheme and not on an end-to-end scheme – for which different features (useful for the learning task) could be learned by each subnetwork –. Thus even if different useful features are learned, they are useful only for the source-task, not necessary for the target-ones. Moreover, the ensembling is not performed as usually (*i.e.*, aggregating the output of the classifiers of the networks), since here we aggregate the internal layers. In any case, this experiment shows that the performances of MulDiP-Net do not result from the ensemble-model aspect only, but specifically from the combination of features learned with data labeled among different types of categories (*i.e.*, generic and specific).

Another baseline is to compare our method without the ensemble-model, but by jointly training (**JT**) a network on the two SPs (generic and specific). We tested three variants:

- **SPV$_{\mathrm{G}}^{\mathrm{cat}}$+JT(SSL)** training the set of SPs with a sum of softmax losses (SSL) – *i.e.*, one for each SP –, which corresponds to *multi-task learning*;
- **SPV$_{\mathrm{G}}^{\mathrm{cat}}$+JT(MLS)** training the set of SPs with a multi-label loss (MLS) layer (*e.g.*, sigmoid cross entropy), where the labels for each image contain both annotations (generic and specific);
- **SPV$_{\mathrm{G}}^{\mathrm{cat}}$+JT(GtoS)** gradually training the SPs and more precisely, the network is first trained to solve the generic SP and then the training is continued on the specific SP.

Globally, the joint training gives always a much lower BC score compared to the reference method, meaning that MulDiP-Net effectively benefits from the growing capacity obtained by its ensemble-model aspect. An exception is the gradual training on the different SPs (generic then specific), that is very close to the reference method (trained once on specific) in terms of BC score. This means the **GRP(CAT)+JT(GtoS)** does not benefit from the features learned on generic data. We think that this is mainly due to the fact than when the network is trained on the second specific problem, it completely forgets the features learned on the former generic problem (this is known as catastrophic forgetting [76]). To resume, the comparison to this baselines shows the utility of the disjoint-learning of MulDiP-Net compared to the joint one (JT).

For the FSFT method, we mainly compare it to the two methods of [285] – Frozen Self Fine-Tuning (FrSFT) and Self Fine-Tuning (SFT) – that roughly consist to re-train the network on the same problem. More precisely, **FrSFT** retrains the last layers only with the previous layers being "frozen", **FST** re-trains all the layers with the same learning-rate. As highlighted in [285], we observe a performance drop of FrSFT compared to the reference. As mentioned in Section 4.1.2.5, this is due to the fragile co-adaptation neurons learned in the original network. A slight drop of performance is also observed

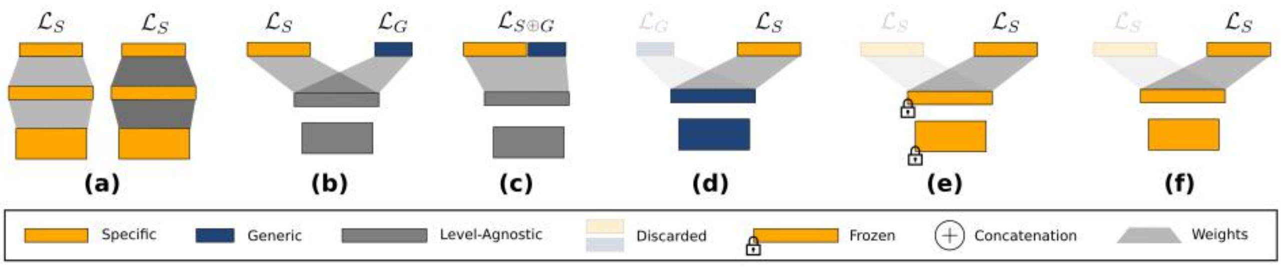

Figure 4.13: Illustration of the different baseline methods. **Yellow** blocks are features learned with specific labels, **blues** ones illustrate those learned with generic labels, and **gray** ones reflect the features learned on multiple labels (thus could be generic, specific or level-agnostic). Blocks with low opacity represent discarded layers (*i.e.*, used in a first training stage but not in the second). The padlock highlights the features that are frozen (*i.e.*, not allowed to be trained on the actual problem to solve) and the surrounded "+" indicated a concatenation of layers. The baselines are: **(a) Ensemble**: two networks trained on same specific problem $\mathcal{L}_S$, but with *different* initialization of the weights; **(b) Multi-Task**: one network trained on both, specific and generic problems, in a multi-task paradigm, *i.e.*, with a sum of their losses $\mathcal{L}_S + \mathcal{L}_G$; **(c) Multi-Label**: one network for generic and specific, but with the concatenation of the specific and generic labels and the learning of a multi-label loss $\mathcal{L}{=}\mathcal{L}_S \bigoplus \mathcal{L}_G$; **(d) Recursive**: Generic weights as initialization for specific network (GtoS), *i.e.*, training one network on the generic problem, then training of the specific problem with the learned generic weights instead of random ones; **(e) Frozen Fine-Tuning (FrFT)**: re-training on the initial problem of the penultimate and last layers of a network pre-trained on specific labels, without allowing the modification of earlier layer; **(f) Self Fine-Tuning (SFT)**: same as FrFT but with the allowing of modification of earlier layers.

for FST, meaning that fine-tuning does not always recover all the co-adapted neurons. In contrast, FSFT increases the performances, even when we compact its representation to $2,048$, highlighting its capacity to recover co-adapted neurons of the original network and even the training of others. Finally, we also assess the combination of FSFT with MulDiP-Net compared to the reference and EM+SPV$_{\mathbf{G}}^{\mathbf{cat}}$. As expected, when the FSFT is used with the same capacity (FSFT-4k), it improves the performances and when it is used with lower capacity (FSFT-2k), it both improves the performance while reducing the dimensions.

4.3.3.2 Impact of the SPV by *grouping*

We assess the MulDiP-Net performance with different configurations in the set of SPs (*i.e.*, set of generic SPs in $\mathcal{D}_{\Omega}^{S}$ of Eq. 4.7) as well as a comparison with each of its subnetworks alone. More precisely, we report the performances of each SPV by *grouping alone* – *i.e.*, **SPV**$_{\mathbf{G}}^{\mathbf{cat}}$ for categorical, **SPV**$_{\mathbf{G}}^{\mathbf{hl}}$ for hierarchical and **SPV**$_{\mathbf{G}}^{\mathbf{clu}}$ clustering levels –, as well as MulDiP-Net with each of the three grouping SPV – *i.e.*, **EM+SPV**$_{\mathbf{G}}^{\mathbf{cat}}$, **EM+SPV**$_{\mathbf{G}}^{\mathbf{hl}}$ and **EM+SPV**$_{\mathbf{G}}^{\mathbf{clu}}$ (with EM designating ensemble-model). As depicted in Section 4.1.2.4, MulDiP-Net can benefit from the combination of all the grouping SPVs, thus we also report its performances with three generic levels (initial, categorical and hierarchical) denoted **EM+SPV**$_{\mathbf{G}}^{\mathbf{cat+hl}}$ and four levels (initial, categorical, hierarchical and clustering) denoted **EM+SPV**$_{\mathbf{G}}^{\mathbf{cat+hl+clu}}$. Finally, in order to assess whether MulDiP-Net really benefits from *semantic* grouping, we also compare it to a baseline where we group the categories *randomly*. This method is denoted **EM+SPV**$_{\mathbf{G}}^{\mathbf{rand}}$. All the methods are compared with the same reference one, and the results are presented in Figure 4.14.

Method	VOC07	CA101	CA256	NWO	MIT67	stACT	CUB	FLO	mNRG
	mAP	Acc.	mAP	Acc.	Acc.	Acc.	Acc.	Acc.	
REFERENCE	66.8	71.1	53.2	<u>52.5</u>	36.0	44.3	<u>36.1</u>	50.5	n/a
(a) Ensemble	67.8	72.2	54.5	52.0	37.2	45.0	34.7	51.8	+2.3
(b) Multi-Task	61.5	61.8	45.4	49.4	30.7	36.4	25.6	38.7	-16.2
(c) Multi-Label	44.7	46.8	26.4	25.1	27.2	28.0	15.2	38.1	-45.0
(d) Recursive	65.3	68.6	50.8	52.4	33.4	50.8	29.4	45.5	-4.8
(e) FrST	62.3	64.3	47.3	50.0	30.9	38.4	29.3	41.1	-11.6
(f) SFT	67.0	71.5	52.5	49.8	36.2	44.0	36.4	50.1	-0.1
FSFT-2k	67.6	73.3	54.8	47.2	37.9	46.9	<u>38.2</u>	56.3	+3.4
FSFT-4k	67.5	73.9	55.0	44.6	40.4	47.1	**38.7**	56.8	+4.5
MulDiP-Net	<u>69.5</u>	76.0	56.8	54.7	41.3	48.5	35.6	54.8	+7.9
MulDiP+FSFT-2k	**69.8**	<u>76.6</u>	<u>57.3</u>	**49.9**	<u>41.6</u>	48.7	38.0	<u>59.2</u>	<u>+8.8</u>
MulDiP+FSFT-4k	**69.8**	**77.5**	**58.3**	<u>47.9</u>	**43.7**	<u>50.2</u>	37.4	**59.7**	**+10.7**

Table 4.6: Comparison of our universalizing-methods with **baseline-methods**. Top rows correspond to baselines. Each baseline is associated to a letter between brackets, that corresponds to the letters of their illustrations in Figure 4.13. Methods are compared in terms of their median Relative Gain (**mNRG**) scores (last column colored in blue), with a reference method that consist of a network trained on specific categories (red scores). For each benchmark as well as for the final mNRG score, we highlight the highest score in bold, and the second one is underlined. All the methods have been learned with the same architecture (*i.e.*, AlexNet) on the same initial SP (*i.e.*, ILSVRC*).

From the results, a first observation is that grouping SPs alone ($\mathrm{SPV_G}$) is always below the reference method, which is consistent with [109]. Among the three grouping methods, the categorical grouping always obtains the best results, highlighting the interest to introduce a grouping matching human categorization principles. Second, almost all the EM+$\mathrm{SPV_G}$ perform better than its subnetworks learned on specific and generic data only, showing the interest of their combination in MulDiP-Net (*i.e.*, the $\mathrm{SPV_G}$ and reference methods that respectively corresponds to the training on generic and specific data). This latter, clearly demonstrates the contribution of the combination of the specific and generic knowledge and thus that, MulDiP-Net is always beneficial, whatever the grouping SPV. In contrast, the performances of EM+$\mathrm{SPV}_\mathrm{G}^{rand}$ are always very close to the reference method, which puts forward the utility of the SPV by *semantic* grouping. Finally, the more different grouping SPs we consider in MulDiP-Net, the more the performances increase. However this performance increase seems to saturate above two SPs.

4.3.3.3 MulDiP-Net with More Training Data

To estimate the universalizing capacity of MulDiP-Net with larger SPs, we trained MulDiP-Net on the full ILSVRC containing containing around 1.2 million images labeled among $1,000$ categories. The results are respectively reported in first (reference) and third line (MulDiP-Net) of Table 4.8. MulDiP-Net performs better than the reference method on all datasets except CUB, although the results are

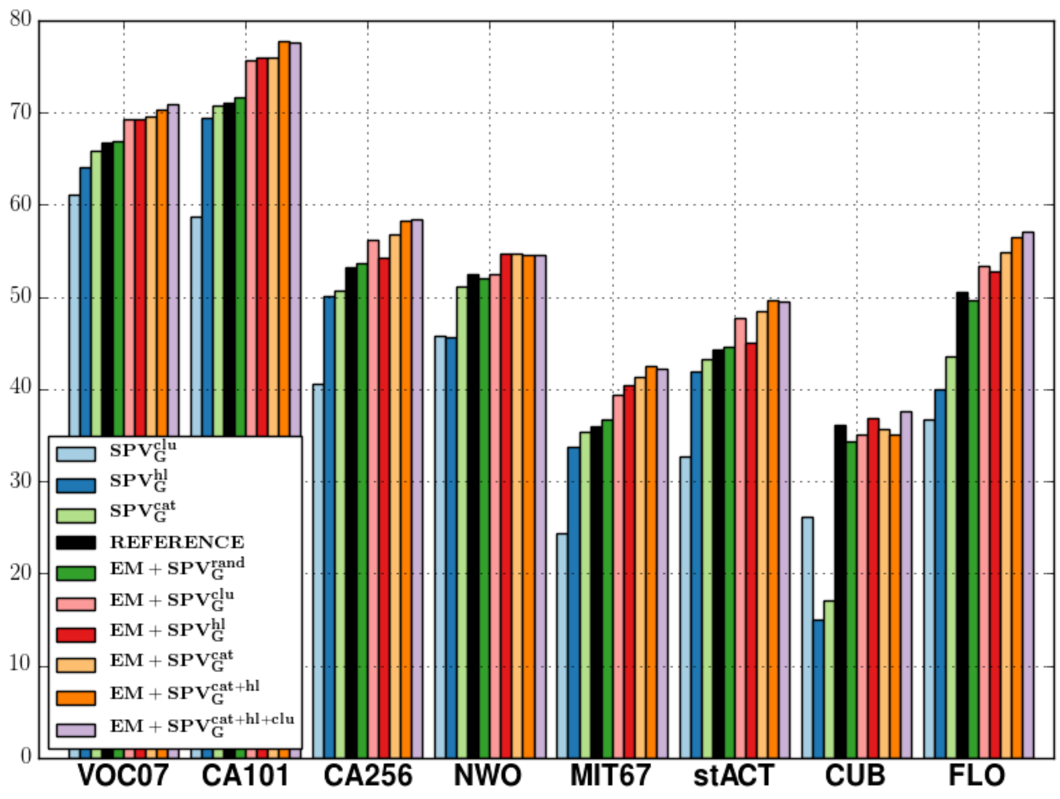

Figure 4.14: Impact of the **different SPV by *grouping*** considered in our MulDiP-Net. We plot the performance of the methods according to the benchmarks with their associated standard evaluation-metrics. Best view in color.

quite close (-0.2%), showing the previous results can be extended to a larger source dataset.

4.3.3.4 MulDiP-Net with More Networks

Up to now, we mainly reported results of MulDiP-Net with two networks learned on specific and generic categories. They correspond to the subordinate and the basic levels of the categorical levels [214]. A third level even more generic, namely superordinate level, can nevertheless be considered. We report in Table 4.7 the results obtained with this third level on two datasets.

The results obtained with the network learned on superordinate levels only are much lower than with the other networks. Indeed, the features added with this third network are very generic and thus seem to not contribute a lot to discriminate classes.

When it is added to the two other networks (concatenation), the results are slightly improved, but at the cost of a representation that is 50% larger, making the interest of the process quite limited. From these results, it seems that it better worth to include network learned on fine categories, as it is further developed in Section 6.2.1.

4.3.3.5 MulDiP-Net with Deeper Architectures

Increasing network capacity (*wider* or *deeper* layers) can help to get a better universal representation. It is thus important to determine whether the principle of MulDiP-Net has an effect with a larger network. To evaluate this point we considered AlexNet (5 convolutional layers and 3 fully-connected), the deep and wide VGG-16 (16 convolutional layers and 3 fully-connected) and DarkNet-20 (20 convolutional layers and one fully-connected layer). Results of the methods with respect to the network architectures are presented in Table 4.8. Three observations can be done from the results. First, the deeper is not the better for universalizing since the reference method with VGG-16 has a better BC score than the one with DarkNet-20. Second, for any architecture, $\mathrm{SPV_G}$ is below the reference method, except for the DarkNet one, which is surprising. Third, MulDiP-Net always performs better than its two subnetworks (*i.e.*, reference and $\mathbf{SPV_G^{cat}}$), regardless the network architecture, showing its robustness with regards to this aspect.

One can see that the network trained on specific categories (also denoted Net-S in the following) is always better than the one trained on generic categories (Net-G), except for the DarkNet architecture. This can be due to the presence of three fully connected layers in AlexNet and VGG while DarkNet is almost fully convolutional. Hence, AlexNet and VGG can finally combine more convolutional filters than DarkNet and a Net-S learns very specific filters compared to a Net-G that learns generic ones (Section 4.3.1). Such a combination of convolutional filters is able to detect more diverse patterns and being thus beneficial to universalize the representation. It is also possible to combine the filters with convolutional layers but it would require more layers. Here, it seems that the usage of fully-connected layers leads to a larger diversification of the patterns that can be detected thus improves the universality, while an even deeper fully convolutional network may also obtain good results.

Set of Label-Sets ($\mathcal{B}$)	**VOC07**	**NWO**
$\{\mathcal{L}_2\}$ (Superordinate)	58.9	37.4
$\{\mathcal{L}_1\}$ (Basic-Level)	70.0	51.0
$\{\mathcal{C}\}$ (Subordinate=specific)	70.3	51.2
$\{\mathcal{C}, \mathcal{L}_1\}$	72.5	54.1
$\{\mathcal{C}, \mathcal{L}_1, \mathcal{L}_2\}$	**73.0**	**54.9**

Table 4.7: Impact of the categorical-levels considered in MulDiP-Net, in terms of performance in transfer learning on PascalVOC 2007 (**VOC07**) and Nus wide object (**NWO**).

Method	**Network**	**VOC07** mAP	**CA101** Acc.	**CA256** mAP	**NWO** Acc.	**MIT67** Acc.	**stACT** Acc.	**CUB** Acc.	**FLOW** Acc.	**BC**
Ref.	AlexNet	71.7	79.7	62.4	58.3	46.9	51.2	**36.3**	58.4	19
$\mathbf{SPV_G^{cat}}$	AlexNet	71.5	77.4	60.4	57.8	42.8	49.3	19.5	52.4	10
MulDiP	AlexNet	**74.4**	**82.5**	**65.2**	**60.8**	**48.4**	**54.2**	36.1	**62.5**	25
Ref.	VGG	86.1	88.8	78.0	71.8	66.7	73.5	69.8	78.9	52
$\mathbf{SPV_G^{cat}}$	VGG	85.7	87.6	76.9	70.3	65.8	72.2	67.0	75.0	43
MulDiP	VGG	**87.5**	**92.0**	**80.9**	**72.6**	**68.9**	**75.0**	**71.5**	**81.9**	**68**
Ref.	DarkNet	82.7	91.0	78.4	70.5	64.8	72.2	59.5	80.0	44
$\mathbf{SPV_G^{cat}}$	DarkNet	83.2	91.5	78.1	73.2	64.4	72.6	52.5	78.9	46
MulDiP	DarkNet	**84.1**	**92.7**	**80.1**	**73.9**	**66.4**	**74.5**	**61.2**	**82.1**	<u>62</u>

Table 4.8: Comparison of MulDiP-Net with its sub-components: specific (Ref.) and generic (SPV_G^{cat}) on the **full ILSVRC** and **different network architectures** (AlexNet: 7 layers - VGG: 16 layers - DarkNet: 20 layers). The last column reports the global Borda Count score (BC). Using 9 methods, the BC score ranges in $[\![9, 90]\!]$. The highest scores are in bold and the second one is underlined.

4.4 Conclusions

In this chapter, we proposed a simple novel approach for learning more universal representations. Our approach learns a set of features on an initial source-problem (SP) then consists in three main steps: (i) applying a source problem variation to produce new source-problems; (ii) learning features on the new source problems and (iii) combining all features (learned on the initial SP and the new SPs) to form the final representation. The original contribution concerns the method to perform SPV and the combination of the features: a SPV by *grouping* according to some semantic knowledge in the form of hierarchies with "is-a" relations that is in line with our hypothesis that adding semantics is a good way to improve universality, and a combination through a dimensionality reduction (FSFT), followed by an independent normalization and a simple concatenation.

MulDiP-Net+FSFT exhibits promising results, increasing the universality of the learned representation (on a set of ten target-datasets in a transfer-learning scheme) without more annotated data. In addition, we also performed an in-depth analysis showing that our approach works better because of two properties, namely *filters diversification* and *diversification relevance*.

It is important to note that the third step can be performed differently. Such alternative approaches are discussed in details in Chapter 6.

5

Preserving Unimodal Semantics in Multimodal Representations

Contents

The question of the universality of the representation is not restricted to the visual domain. This chapter deals with building a joint representation of the visual and the textual/semantic modalities, that is an alternative to the proposal of the previous chapter to align the high-level semantic data to the low-level visual ones.

Each modality is mapped to the other through supervised learning. The originality of our approach mainly lies in using one modality to define the labels that is used as ground-truth to learn the other modality. We adopt an unsupervised learning approach to define these labels, such that they do *not* correspond to an actual concept, that would be unequivocally recognized by a human being. We argue that this ambiguity may reflect a visual concept that can hardly be defined with words as well as a semantic concept that could be illustrated by many visual aspects. In that sense, these *meta-concepts* aim to get more universal embedding space than those that are directly reflected by the unimodal features.

5.1 Non-Semantic Meta-Concepts Classification

5.1.1 Introduction

We are interested in the joint representations of the visual and the textual/semantic modalities. Such approaches have had an increasing success in the past few years in particular because the alignment of textual and visual data facilitates cross modal retrieval and can be directly applied to text illustration and automatic image captioning [123, 122]. It also allows to tackle the problems of zero-shot classification [77] and visual question answering [6]. It is also of interest on a theoretical point of view since it provides a model that tends to unify the concept independently of the modality used to represent it.

A major difficulty to understand visual and textual content lies in its ambiguity with regard to the actual user need, for instance when identifying an entity from a given textual mention [145] or matching a visual perception to a need expressed through language [6]. On the one hand, it is simply expressed by the well-known idiom "A picture is worth a thousand words" that Smeulders *et al.* [230] used to introduce their presentation of the "semantic gap". On the other hand, the number of visual representations of simple objects or persons is almost infinite, not to mention more difficult concepts such as emotions or actions that are nonetheless quite easily identified by a human.

Many approaches that aim to align visual and textual/semantic content rely on supervised learning, including in particular those that learn joint representations [127, 77, 123, 122, 271, 254, 255, 67, 222, 153, 68] (more details in Sec. 2.1.4). In such a scheme, the labels are non ambiguous thus it attempts to map a modality to the other one while skipping half of the inherent ambiguity that could exist. For example, when one wants to map a diversity of images to a given concept, it usually ignores that such images should also match its synonyms or possible "close concepts". Such a limit can of course be managed through an ad-hoc additive process that models explicitly these relations in the semantic space.

We nevertheless argue this could be directly managed during the learning of the initial mapping, by considering ambiguous concepts. In practice, we consider *meta concepts* that subsume several concepts, either representing physical entities (*concrete* concepts) or non-physical ones (*abstract* concepts). Since our approach to define these labels is strongly driven by unsupervised learning, it may result into a concept that could hardly be described by words. Hence, we qualify it as a non-semantic concept, resulting into the central notion of *non-semantic meta concept*, named NAMECON.

These NAMECONs being used as labels in a supervised learning framework, the rest of the approach is inspired by the some of the most successful recent approaches to learn joint representations for visual and textual content. Overall, the architecture consists in supervised learning of each modality that are learned jointly to optimize a bi-directional ranking loss [271, 270]. In addition, a classification task is added to provide a semantic regularization that favors high-level alignment with the common space [222, 236]. In summary our method learns the multimodal embedding space by correctly ranking aligned pairs and jointly correctly classifying NAMECONs, thus we named it **NAMRank**.

Note that, some works in the literature also proposed to rely on classes during the learning of the multimodal embedding space. Indeed, Gong *et al.* [85], Salvador *et al.* [222] and Suris *et al.* [236] proposed, to add to a standard learning strategy, an additional classification loss that learns to recognize some semantic classes derived from ground-truth annotations. Carvalho *et al.* [153] proposed to recognize these categories by learning a double-triplet scheme to express jointly the instance loss and the

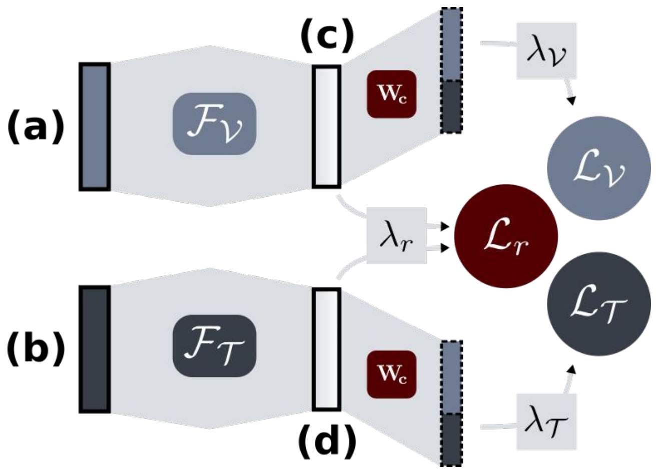

Figure 5.1: Given an input image-text pair and their respective unimodal features (a) and (b), the proposed NAMRank method maps each unimodal data into a multimodal space. In particular, each mapping function is represented by a branch (top: image to multimodal (c) through $\mathcal{F}_\mathcal{V}$; bottom: text to multimodal (d) through $\mathcal{F}_\mathcal{T}$). To learn the mapping functions, we use an objective that promotes two main constraints: (i) bring closer the positive image-text pairs and drives away the negative ones (through a bi-directional ranking loss $\mathcal{L}_r$); and (ii) ensures the multimodal representations of the image-text pairs to belong to their associated unimodal categories (visual and textual classification losses $\mathcal{L}_\mathcal{V}$ and $\mathcal{L}_\mathcal{T}$). The three objectives are calibrated with a set of parameters (λ_r, λ_V and λ_T) that results in multiple potential training strategies. The main novelty of this paper lies in the fact that the categories associated to the image-text pairs were obtained by clustering the set of unimodal features which aims to get Non-semAntic MEta-CONcepts (NAMECONs). This latter ensures the preservation of the unimodal visual and textual semantics into the learned multimodal representation.

class-based one. Such proposal is similar to the structure preserving method proposed in [271, 270], but differs by the fact that [153] uses real multimodal classes (that contains images and texts) while Wang *et al.*only used the multiple captions associated to each image, that is, unimodal categories.

Compared to all previously mentioned works, a major difference with our work is that, in our case, the non-semantic *abstract* concept deeply results from an unsupervised approach, avoiding to rely on human annotations that are costly to obtain.

5.1.2 Proposed Method

Given data from multiple modalities, our goal is to learn an embedding space where, *aligned* data from multiple modalities describing the same scene produces close representations in a multimodal space, and *non-aligned* ones produce representations that are far from each other in this space. Here, we specifically focus on visual and textual data.

Unimodal images and text representations are usually not directly comparable because they are learned on different data, with different objectives and learning-strategies. They thus do not encode the same semantics, which brings us to the famous semantic gap. Hence, to make an image and a text directly comparable, we aim at learning a common space $\mathcal{M}$ where features $\mathcal{V}_i$ of images I_i and features $\mathcal{T}_i$ of texts T_i can be projected in and being directly comparable. It thus consists in learning two mapping functions, $\mathcal{F}_\mathcal{V}$ that maps a visual feature vector $\mathcal{V}_i$ to its multimodal representation $\mathcal{M}_i^\mathcal{V}$ and $\mathcal{F}_\mathcal{T}$ that maps a textual feature vector $\mathcal{T}_i$ to its multimodal representation $\mathcal{M}_i^\mathcal{T}$. Hence, the projected visual and textual features have the same dimension ($\mathcal{M}_i^\mathcal{V} \in \mathbb{R}^M$ and $\mathcal{M}_i^\mathcal{T} \in \mathbb{R}^M$).

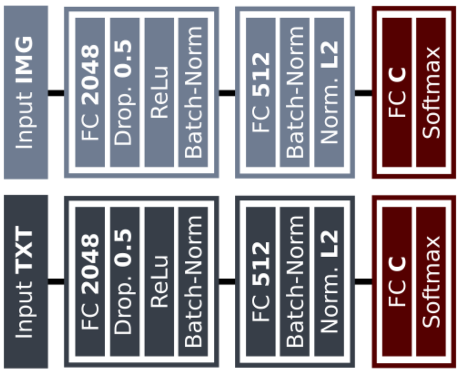

Figure 5.2: Illustration of the two-branch network and the additional namecon classification layer on top of each branch.

To learn more effective mapping functions, we propose to consider the two following principles: distributions of $\mathcal{M}_i^{\mathcal{V}}$ and $\mathcal{M}_i^{\mathcal{T}}$ need to be modality-invariant, and the projected features need to be semantically discriminative. More specifically, our contribution here is on the second objective, were we get the categories at zero-cost of annotation and especially because they directly encode the semantics of the initial modalities (visual and textual), which in a certain sense, aims to preserve semantics of the different modalities into the learned multimodal embedding space. This latter is ensured through our proposed Non-semAntic MEta-CONcepts ("NAMECON") – *i.e.*, categories that are both, *generic* and *non-semantic* – that are obtained by clustering, thus at zero-cost of annotation and obtained on the features of each modality, thus they encode the semantics of each modality. Since, we train the multimodal representation with both, the bi-directional ranking and NAMECON classification loss as objectives, we call our method "NAMRank". An overview of the method is illustrated in Figure 5.1.

In the following, we first detail the two-branch network containing the two mapping functions (Section 5.1.2.1); then we detail the bi-directional and discriminative objectives (Section 5.1.2.2). Finally, we describe our NAMECON principle (Section 5.1.2.3), detail the way we construct and infer them (Section 5.1.2.4) and, provide some important technical details to consider them with a ranking-based loss function (Section 5.1.2.5).

5.1.2.1 Two-Branches Network

To map from unimodal to multimodal representations, we need one mapping function per modality. Since we consider two modalities, the architecture we use contains two mapping functions and as in [271], it is called a *two-branch network* that is illustrated in Figure 5.2. Each branch is a L-layer perceptron, where each of its $L-1$ hidden layers is expressed as $h_l^B(\mathbf{x}) = f(W_l^B\mathbf{x} + b_l^B)$, with $\mathbf{x}$ being the input of a unimodal representation or output of a previous hidden layer ($B = \mathcal{V}$ for the visual branch and $B = \mathcal{T}$ for the textual one). W_l^B, b_l^B are the weights and bias terms of the l^{th} hidden layer of branch B and f is a non-linear activation function (ReLU). The output layer of each branch h_L^B is $h_L^B(\mathbf{x}) = (W_L^B\mathbf{x} + b_L^B)/\left\|W_L^B\mathbf{x} + b_L^B\right\|^2$, where $\mathbf{x}$ is the l2-normalized output of the previous layer h_{l-1}^B. Finally, on top of each branch B, an additional layer is present to learn an additional classification problem. This latter, does *not* belong to the mapping functions and is only here for regularizing and improving them. More precisely, the additional L+1 layer is expressed by $y_{L+1}^B(\mathbf{x}) = \sigma(W_L^B h_L^B + b_L^B)$, where $\mathbf{x}$ corresponds here to the output of the mapping function $\mathcal{F}_{\mathcal{B}}$ applied on the unimodal representation $\mathcal{B}$, and σ is the softmax function.

It is important here to mention that the multimodal representations for the visual and textual features respectively correspond to the output of the mapping functions $\mathcal{F}_\mathcal{V}$ and $\mathcal{F}_\mathcal{T}$ (*i.e.*, output of the L-layer (h_L^B)) and do not correspond to the output of $L + 1$ layer, that is only here for regularizing and improving the mapping functions (this will be described in more details in the next sections). Indeed, the L+1 layer of each branch is discarded during the inference phase. Thus, to get the multimodal representation $\mathcal{M}_B(X)$ of an input data X from modality B, during inference, we first extract its unimodal representation $\mathcal{B}$ from a pre-trained network of that modality B, then we do a simple forward pass $h_1^B \circ h_2^B \circ \cdots h_L^B$ on the branch B for which the extracted unimodal representation $\mathcal{B}$ is given as input (*i.e.*, projection of $\mathcal{B}$ through the pre-trained mapping function $\mathcal{F}_B$).

5.1.2.2 Learning Multimodal Representations with NAMECON and Ranking Objectives

The most challenging aspect of multimodal representation learning is the objective used to train the mapping functions of each branch. Hence, our method roughly consists to train the mapping functions ($\mathcal{F}_\mathcal{V}$ and $\mathcal{F}_\mathcal{T}$) by solving the commonly used bi-directional ranking task and at the same time an additional namecon classification task. In particular, the former objective enforces the multimodal representation to have close representations for aligned data and distant features for non-aligned ones; and since the ranking task only enforces alignment, the additional one is here to act as a sort of semantic regularization and ensures that the learned multimodal embedding is also discriminative with respect to the semantics of the initial modalities (visual and textual).

Formally, let us consider a training dataset $\mathcal{D} = \{X_i\}_{i=1\cdots N}$ of N text-image pairs, with images I_i^+ that are *aligned* with texts T_i^+ and images I_i^- that are *not aligned* with texts T_i^-. Using pre-trained networks, we extract the visual features $\mathcal{V}_i^+$ of dimension $d_\mathcal{V}$ and the textual features $\mathcal{T}_i^+$ of dimension $d_\mathcal{T}$ (usually, $d_\mathcal{V} \neq d_\mathcal{T}$). Each instance X_i is also assigned to a semantic label vector $Y_i = [y_{i1}, y_{i2}, \ldots, y_{iC}] \in \mathbb{R}^C$, where C is the total number of categories. If the i^{th} instance belongs to the j^{th} category, $y_{ij} = 1$, otherwise $y_{ij} = 0$. Hence, X_i can belong to one or multiple categories.

A natural framework to learn a representation that is highly similar for aligned data and weakly similar for non-aligned ones is to minimize a bi-directional ranking loss [271, 270]. More precisely, the goal is to minimize a margin-based loss function:

$$\mathcal{L}_r = \sum_{i=1}^{N} \Bigg(\sum_{i,j,k} \max\big\{0, m + d(v_i^+, t_j^+) - d(v_i^+, t_k^-)\big\} \\ + \lambda_1 \sum_{i',j',k'} \max\big\{0, m - d(t_{i'}^+, v_{j'}^+) + d(t_{i'}^+, v_{k'}^-)\big\} \Bigg), \quad (5.1)$$

where $v_i = \mathcal{F}^\mathcal{V}(\mathcal{V}_i)$ and $t_i = \mathcal{F}^\mathcal{T}(\mathcal{T}_i)$ respectively denote the output of the L-layer of the visual and textual branches, $d(a, b)$ is the Euclidean distance between a and b, m is a margin, and λ_1 balances the strength of the ranking loss in the second direction.

In order to ensure that the semantics encoded in the representations of initial modalities $\mathcal{V}_i^+$ and $\mathcal{T}_i^+$ is preserved in the multimodal embedding space, we regularize by a classification task that consists to recognize categories that were built using the unimodal representations [222, 153]. More precisely, the multimodal representations obtained from each branch $\mathcal{M}^\mathcal{V}$ and $\mathcal{M}^\mathcal{T}$ is provided as input to the last $L + 1$ layer which outputs a *predicted* C-dimensional score vector that is squashed into a probability vector ($\widehat{y_i^\mathcal{V}}$ for visual branch and $\widehat{y_i^\mathcal{T}}$ for textual branch) through a softmax function σ, and for

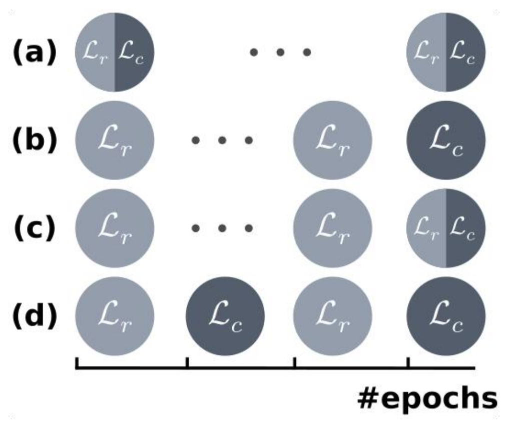

Figure 5.3: Illustration of the different training strategies. (a) Parallel, (b) Sequential (we can also start by $\mathcal{L}_c$ for a certain amount of epochs and finish by $\mathcal{L}_r$), (c) Sequential-parallel , and (d) Alternately. Note that for (b) and (c), we can also start by $\mathcal{L}_c$ for a certain amount of epochs and finish by $\mathcal{L}_r$. Best view in color.

which each dimension corresponds to a given category of the set of C categories to recognize $y_i^{\mathcal{B}}$ for branch $\mathcal{B}$. Indeed, the main novelty of this work is that, the set of categories that we named *namecons* are obtained from the unimodal visual and textual features that respectively encode some semantics of the visual and textual modalities. Namecons roughly correspond to clusters computed on unimodal representations and they thus encode the cluster structure of unimodal data, but their principle will be detailed in the next section. Once namecons obtained, we simply use them as ground-truth categories for each data. Hence, each image-text features pair $(\mathcal{V}_i^+, \mathcal{T}_i^+)$ is associated to a bank of C namecons $\mathcal{N} = \{c_1, \dots, c_C\}$, that is used as ground-truth probability for each branch. All branches are thus learned to correctly predict the namecons for a set of N samples of each mini-batch, through a *softmax cross-entropy* loss $\mathcal{L}_{\mathcal{B}}$:

$$\mathcal{L}_{\mathcal{B}} = \sum_{i=1}^{N} y_i^B \log(\widehat{y_i^B}) + (1 - y_i^B)\log(1 - \widehat{y_i^B}), \tag{5.2}$$

were, $\widehat{y_i^B} = \sigma(W_{L+1}^B \mathcal{F}^B(\mathcal{B}_i))$ is the predicted probability of presence of the C^B namecon categories with the multimodal representation $\mathcal{M}_{\mathcal{B}}$ (which has been obtained by a forward pass into $\mathcal{F}_{\mathcal{B}}$ with the unimodal feature vector $\mathcal{B}_i$ as input); and y_i^B corresponds to the banks of C^B *desired* namecons for the i^{th} element $\mathcal{B}_i$. Note that, with a softmax cross-entropy, the categories may be mutually exclusive, but their probabilities do not need to also be mutually exclusive (*i.e.*, one-hot encoding). For instance, while the *dog* and *car* classes are mutually exclusive, an image that may contain a big car near a small dog will for instance have 0.7 probability of being categorized as car 0.3 probability as dog. Simply said, all what is required is that each bank of ground-truth labels is a valid probability distribution, which allows us to also solve multi-label classification problems. This latter is important since the final namecon objective is an objective that encodes both visual and textual namecons, which results in a multi-label classification problem.

Finally, the overall loss of the NAMRank method, is minimized through Adam optimizer and modeled as a multi-task combining the bi-directional ranking loss $\mathcal{L}_r$ and the discriminative classification loss of each branch $\mathcal{L}_{\mathcal{B}}$:

$$\mathcal{L} = \lambda_r(e)\mathcal{L}_r + \lambda_{\mathcal{V}}(e)\mathcal{L}_{\mathcal{V}} + \lambda_{\mathcal{T}}(e)\mathcal{L}_{\mathcal{T}}, \tag{5.3}$$

where $\lambda(e)$ are hyper-parameters that depend on the number epochs e. More precisely, $\lambda(e)$ can be set by cross-validation or even be trainable. However, with the $\lambda(e)$ parameters, we can imagine

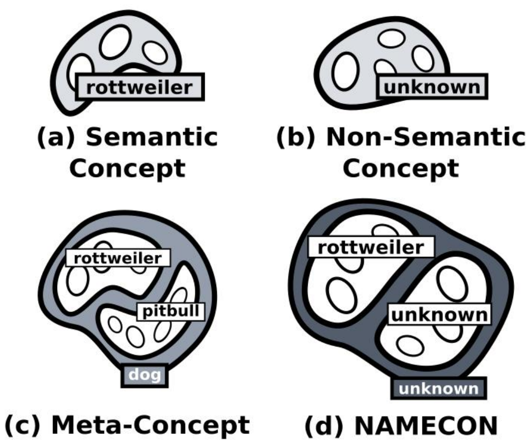

Figure 5.4: Illustration of the different kinds of concepts. A semantic concept (a) is a group (gray surrounded in black) of instances (white circles) that represents a semantic notion, and this group is associated to a semantic connotation (here rottweiler). According to definition 1, a non-semantic concept (b) is a group of instances that *do not* represent a semantic notion or is *not yet* associated to a semantic-connotation (here unknown). A meta-concept (c) is a group of elements (*i.e.*, instances or group of instances), where all the elements subsumed by the group and the group itself represents a semantic notion and is named by a human (here, dog for the group; and pitbull, rottweiler for its hyponyms). Finally, according to definition 2, the non-semantic meta-concept (d) corresponds to a group of elements, where *at least* one element subsumed by the group or the group itself *does not* represent a semantic notion or is *not yet* named by a human (here, an hyponym is unknown thus the group is indescribable thus unknown too).

different training strategies. Indeed, four strategies can be considered: (i) train both the ranking and discriminative loss at the same time from the beginning (*Parallel* strategy); or (ii) first train the network to correctly rank and then to be discriminative, or inversely (*Sequential* strategy); or (iii) first train one and *add* the other (*Sequential-parallel*); or finally (iv), alternate between one and the other (*Alternately* strategy). The different strategies that can be set by cross-validation, are illustrated in Figure 5.3.

5.1.2.3 Non-semAntic MEta-CONcepts (NAMECONs)

Let us recall that a *semantic concept* is any word or n-gram from a real-world vocabulary used by humans to describe a particular notion, that is to say, a physical entity (*e.g*, bicycle, plant, bird, etc.) or a non-physical one (*e.g*, peace, love, beauty, etc.). In the following we use the term "concrete concept" to describe a physical semantic-concept and "abstract concept" to describe a *non*-physical semantic-concept. Let also recall that a *meta-concept* is a semantic-concept (concrete or abstract) that subsumes at least one other semantic-concept (concrete or abstract). We introduce and define *non-semantic concepts* and *Non-semAntic MEta CONcepts* (NAMECONs) as:

Definition 1. A *non-semantic concept* is an instance or group of instances that can not be directly *named* by a human, or that has *not yet* been named.

The first case of the definition could be illustrated by a set of instances (images or texts) that were grouped randomly or by clustering without obvious semantic links between them. In such a case, the concept that is illustrated by the instances is clearly non-semantic and as long as a human can not associate this group entity to a known notion by attributing it a semantic connotation (*i.e.*, unique known

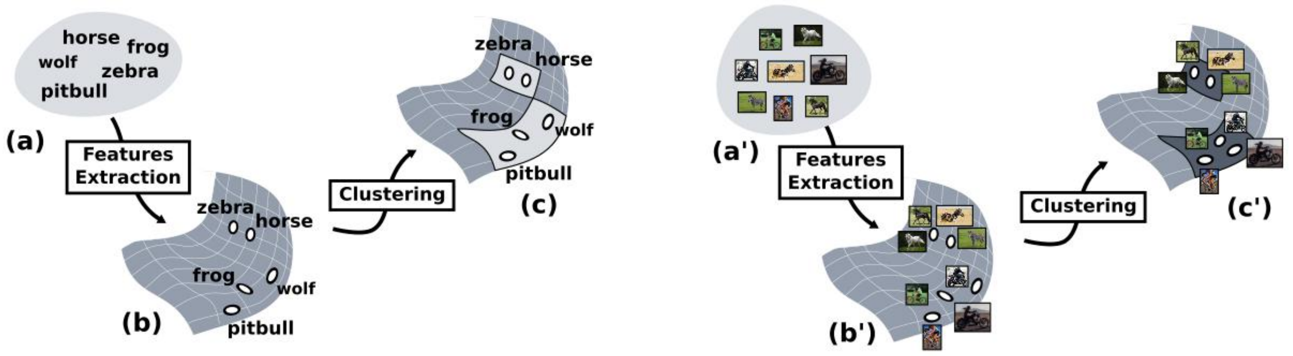

Figure 5.5: Illustration of the way we learn namecons. Given all the set of captions (or words) (a) or images (or patches) (a') in a training corpus and their projection in an unimodal embedding space ((b) for textual features and (b') for visual features), our method clusters the space ((c) and (c')) such that each cluster is a non-semantic meta-concept (namecon). A namecon is both *non-semantic* (is not yet associated to a semantic connotation by a human; or does not exist in the real-world – *i.e.*, inside a group, the instances are not semantically related, which prevents a human to attribute it a *correct* semantic concept) and *general* (*i.e.*, subsumes multiple instances). For instance, the bottom cluster in (c) is *general* since it subsumes the vectors of many words and is *non-semantic* since no semantic connotation can be attributed to it. Best view in color.

word or n-gram), the entity remains non-semantic. In the second case of the definition, an example would be a set of elements that seem to have a link between them and could be easily associated to a semantic connotation by a human. However, as long as a human did not attributed a semantic connotation, the entity remains non-semantic. Generally, when facing the same non-semantic concept (from the second case of the definition) frequently, humans assign it a new unique n-gram and add it to the real-world vocabulary used by humans in order to make it semantic. In summary, as long as a human is not able to attribute or did not yet attributed a unique n-gram (from his vocabulary) to this concept, it remains non-semantic.

Definition 2. A *non-semantic meta-concept* (NAMECON) is both, a *non-semantic* concept and a *meta*-concept. It thus subsumes at least one concept and especially, at least one non-semantic concept and is thus indescribable (or not yet described) by a human with one unique word. As in Definition 1, a namecon can be non-semantic because it has simply *not yet* described by a human.

For instance, if we group three non-semantically related semantic concepts, the probability that a human can describe this group with a unique n-gram is near-zero, making it a non-semantic meta concept (denoted *namecon* in the following). Finally, an overview illustration of the existing kinds of concepts and those introduced in this section is given in Figure 5.4.

5.1.2.4 Learning NAMECONs that Encode Unimodal Semantics

In practice, namecons are roughly built by grouping (through clustering) all the representations of the instances of a certain modality (image or text) into a set of clusters. This latter results in a codebook that then serves as a reference space were we project each instance of the training-dataset on it, and for which we get the closest category that we encode in a vector that is used as ground-truth

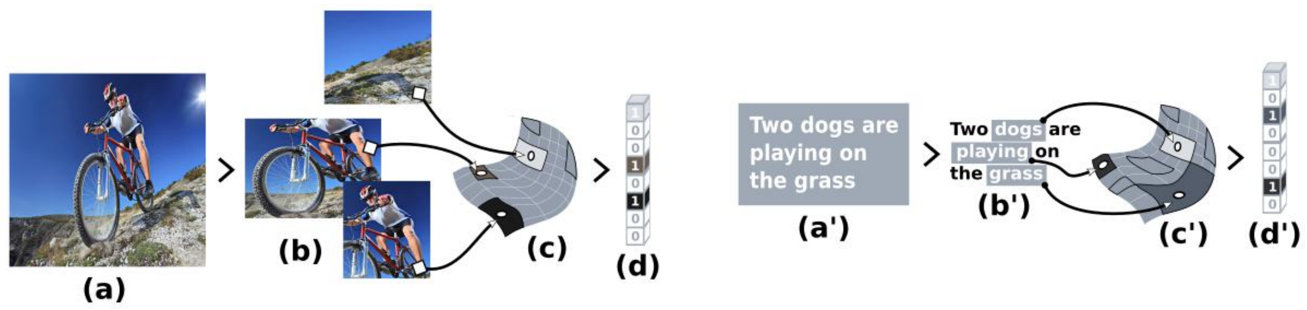
(a)
(b)
(c)
(d)
Two dogs are playing on the grass
(a')
Two dogs are playing on the grass
(b')
(c')
(d')

Figure 5.6: Illustration of the way we compute the visual and textual namecons for a given pair of image-text (a)-(a'). We first extract some global or local information (b) and (b'), then, we compute the unimodal features for each information and project them into the pre-built visual (c) and textual (c') namecon spaces. For each feature vector, we get the cluster where it falls down, activate its associated dimension and fill all other dimensions with a zero-value. For instance, in the image modality, the bottom region falls in the black cluster thus we activate its associated dimension, which is the *sixth* one. Applying this process on the global information (images or captions) results in a the final ground-truth binary *mono*-label NAMECON (d) and (d'). Note that, if the information was extracted locally (from words or patches), the binary namecons obtained from each local representation needs to be pooled (maximum). This latter would result in a bank of NAMECONs that is *multi*-label. Best view in color.

namecon during training (as depicted in Equation 5.2). More specifically, namecons are obtained by grouping visual or textual features through unsupervised clustering. The clustering (supervised or not) is applied to instances and thus aims to get groups, which makes the clusters *meta*. Since the clustering is unsupervised, we thus have a very high chance to obtain at least one cluster that is a non-semantic concept, which ensures the *non-semantic* aspect of the obtained bank of clusters. Hence, the obtained concepts verify the two characteristics of namecons: (i) it groups similar data (from the same modality) into a generic cluster that corresponds to a *meta*-concept and (ii) because of the unsupervised aspect, the resulting clusters do not have any explicit semantic connotation (*i.e* do not exist in the real-world vocabulary of humans) making them *non-semantic* concepts. An overview of the way we build the namecons is illustrated in Figure 5.5.

As stated in Section 5.1.2.2, the classification objective (here namecons) *should* encode the semantic encoded in unimodal representations. However, as mentioned above, namecons are *non*-semantic, but it is important to recall that the clustering is applied on the unimodal representational spaces, and thus capture common properties between the instances. While some of them are *indescribable* (non-semantic w.r.t *first* case of definition 1), others are *nameable* (thus they encode some semantics) but are not yet named (non-semantic w.r.t *second* case of definition 1). Simply said, some namecons are labeled as non-semantic simply because they have not yet named, but obvious links between their instances clearly exists, thus these namecons are nameable, and more especially, encode *some* semantics of unimodal features.

More formally for the description of namecons construction, let us consider a dataset $\mathcal{D}$ of n image-text pairs $\mathcal{D} = \{(\mathcal{I}_i, \mathcal{T}_i)\}_{i=1...n}$. Each element of $\mathcal{D}^T$ and $\mathcal{D}^V$ is then represented by standard representations, namely n-dimensional CNN features $\{v_i\}_{i=1...N^V}$ for the local visual elements and m-dimensional word2vec features $\{t_i\}_{i=1...N^T}$ for the textual ones.Since we want to encode both semantics (visual and textual), we build two different set of namecons (visual and textual). Indeed, given the sets of visual features $\{v_i\}_{i=1...N^V}$ and textual ones $\{t_i\}_{i=1...N^V}$, we create groups of elements resulting in two group sets, namely the visual namecons $\mathcal{N}^\mathcal{V}$ and the textual namecons $\mathcal{N}^\mathcal{V}$, containing respectively $C^\mathcal{V}$ and $C^\mathcal{T}$ clusters (C being chosen arbitrarily or obtained through cross-validation for each modality). Each group is obtained by performing clustering (*e.g* K-means, or Spectral-clustering, or MeanShift, etc.) on the whole set of representations. Hence, the whole set of clusters forms a codebook, with each cluster being a namecon that has as *prototype* the corresponding cluster-center $(\mathbf{c_i})_{i=1,\dots,C^{\mathcal{V}}} \in \mathbb{R}^{d_{\mathcal{V}}}$ for the visual representational space and $(\mathbf{c_i})_{i=1,\dots,C^{\mathcal{T}}} \in \mathbb{R}^{d_{\mathcal{T}}}$ for the textual representation space. Hence, within a namecon cluster, captions or images have similar *non*-semantic connotations (*i.e.*, similar in the sense of the textual or visual representation used, but *not* in the sense of the human semantics because the representation do not purely model it).

Once the visual and textual namecon codebooks learned, we build the ground-truth categories for each instance of the training-dataset. Indeed, the set of visual namecons $\mathcal{N}^{\mathcal{V}}$ and textual ones $\mathcal{N}^{(T)}$ that are seen as a codebook are used to encode any piece of information of the modality of interest. Generally, we adopt a coding scheme similar to local soft coding [147], originally introduced as locality-constrained linear coding [269], that is nevertheless binarized. Given the unimodal representation of an image v_i and the representation of its associated caption t_i, we encode them according to the namecon codebook of their modality (*i.e.*, $\mathcal{N}^{\mathcal{V}}$ and $\mathcal{N}^{\mathcal{T}}$), where there k^{th} dimension is equal to:

$$y_i^{\mathcal{B}}(k) = \begin{cases} 1 & \text{if } k \in NN^p(b_i) \\ 0 & \text{otherwise,} \end{cases} \tag{5.4}$$

where b_i is the representation of the i^{th} instance (image if $b{=}v$, caption if $b{=}t$), $NN^p(b_i)$ is the set of indexes of the p nearest NAMECON clusters of b_i in the representational space of modality $\mathcal{B}$. It is thus a "local hard coding" of $\mathcal{B}_i$ according to its associated modality's codebook. Note that, here the $y_i^{\mathcal{B}}$ vector is *binary* and thus corresponds to the ground-truth namecons of instance b_i (*i.e.*, $y_i^{\mathcal{V}}$ for the visual representation v_i of image $\mathcal{I}_i$ or $y_i^{\mathcal{T}}$ for the textual representation t_i of caption $\mathcal{T}_i$). Regarding the p parameter, it can be simply set by cross-validation. In Figure 5.6, we illustrated how we compute visual and textual namecons for visual and textual features, respectively.

Note that, for each instance, the two banks of namecons ($y_i^{\mathcal{V}}$ for visual instances and $y_i^{\mathcal{T}}$ for textual ones) are used as output of the non-linear $\mathcal{F}_{\mathcal{V}}$ and $\mathcal{F}_{\mathcal{T}}$ mapping functions, as expressed in Equation 5.2. The next section details how these different banks of namecons are considered as a single task in the discriminative loss.

5.1.2.5 Same NAMECON Classification Task for All Branches

While the classification terms ($\mathcal{L}_{\mathcal{V}}$ and $\mathcal{L}_{\mathcal{T}}$) of the loss function in Equation 5.3 are especially used to preserve the unimodal semantics in the multimodal space, if not done correctly, it could lead to misalignment of pair that should be aligned. Indeed, let recall that in order to benefit from the semantics encoded in both unimodal representations, we create banks of namecons from both modalities. A natural way to consider them both in our NAMRank method is to assign each bank of namecons to its associated branch (*i.e.*, visual namecons for visual branch and textual namecons for textual branch). However, an important aspect is that even if images and captions are aligned, the captions are always obtained by annotators that are asked to roughly describe the images. Simply said, captions are an approximation of the content of the images, and do *not* reflect all the details, but only what is semantically important. Because of this difference in aligned data, the learned visual and textual banks of namecons, and thus the namecons of pairs of aligned data, will be different. Hence, assigning namecons of *only one* modality to the branch of the same modality, means that the network is learned to (i) bring closer aligned data; (ii) bring the multimodal visual features $\mathcal{M}_i^{\mathcal{V}}$ to visual namecons; and (iii) bring multimodal textual features $\mathcal{M}_i^{\mathcal{T}}$ to textual namecons (that are different from visual ones). Hence, the last two objectives are adversary, thus only one of them can be reached (*i.e.*, the network will learn mapping functions that satisfy objectives (i) and (ii) or (i) and (iii), but *not* the three of them). In other words, only the semantics of one modality will be encoded in the multimodal embedding space, and not both of them. To avoid this problem, and fully take benefit from the namecons of both modalities, we propose to use the *same namecons for all branches*. In particular, we propose to fuse visual and textual namecons into one, through concatenation, and use them directly as ground-truth for each pair of aligned data. This, ensures that each branch solves the *same problem*.

Since we use the same bank of namecons for all branches, the predictions of namecons with multimodal representations obtained from all the branches, should also be the same. Indeed, we could let the network figures it out, but in order to accelerate the training we propose to enforces this aspect during learning. To do so, a solution could be to consider an additional MSE loss on the namecon predictions from each branch, in order to map them one another. However, we propose to rely on a simpler solution proposed recently by Salvador *et al.* [222], that consists to use the *same weights in all branches* (*i.e.*, the same weights are used in each branch, and only on the layer that maps from the multimodal embedding to the namecon classes). Indeed, with this method, if the visual and textual features of aligned data are close in the multimodal embedding space, the same namecon weights in all branches will certainly ensure that the namecon predictions are the same. If they are not close, the same namecon weights would promote data-alignment, and thus go in the same direction of the ranking term, which is desirable. The principles of having the same namecons and same weights for all branches, are illustrated in Figure 5.1. In particular, the red blocks W_s highlight that the same weights W_s are used in all branches, and the last blue and gray layers show that each branch solves the *same problem*, that is to say, a problem that encodes both, the visual and textual namecons.

5.2 Experimental Results

We evaluate the efficiency of our multi-modal approach with two cross-modal retrieval tasks, namely image captioning (*i.e.*, given a query image, retrieve the most descriptive captions from a large database) and text-illustration (*i.e.*, given a caption as query, retrieve the most illustrative images from a large database). While in the literature, the evaluation is generally carried on three commonly used datasets (Flickr8k, Flickr30k and MSCOCO), here we only used Flickr30k. Indeed, this latter is very similar to Flickr8k but with more training data and generally in the literature, an improvement observed on Flickr30k is also observed on MSCOCO. This work needs to be more validated experimentally, especially in all the commonly used datasets as well as an evaluation in a Transfer-Learning scheme that will state for the universality of the learned representations, which is the main topic of this Thesis. Instead here, we evaluate in a more classical end-to-end learning scheme (*i.e.*, learn the representation on the training set and evaluate on the test set of the same dataset). In Section 5.2.1, we detail the experimental settings, that is to say, the commonly used datasets and the implementations details of our method. Then, in Section 5.2.2, we compare our method with state-of-the-art ones in a classical learning scheme and discuss the obtained results. Finally, in Section 5.2.3, we conduct an in-depth analysis of our method, that contains a comparison to baseline methods, to highlight some insights and an ablation study, showing the impact of each component of our method.

5.2.1 Settings

5.2.1.1 Datasets

As said above, three datasets are commonly used in the literature to evaluate the methods. More precisely, the datasets are **Flickr-8K** [103], **Flickr-30k** [287] and **MSCOCO** [144] databases. All the datasets contain complex images associated to five precise captions and thus can be used either to perform image-captioning or text-illustration. In practice, for the three datasets, the training data are always used to train the models and the testing data to evaluate them. For the cross-modal retrieval tasks, we follow the literature and use the test data of one modality (*i.e.*, images or captions) as queries and the test data of the other modality as the collection (*i.e.*, images or captions, respectively). Regarding the total amount of data (train), Flickr-8k contains $8,000$ images, Flickr-30k contains $31,783$ images and MSCOCO, which is the largest one, contains $82,783$ images. Since in the three datasets, each image is associated to five captions, they respectively contains, $40,000$, $158,915$ and $414,113$ captions. The tree datasets need to be splitted in three sets, namely training, validation and testing. However, only the Flickr datasets contain official splits for cross-modal retrieval tasks. Indeed, for Flickr-8k the training, validation and testing sets respectively contain $6,000$, $1,000$ and $1,000$ images with their associated five captions each. For Flickr-30k, which is larger, the training, validation and testing sets respectively contain $29,783$, $1,000$ and $1,000$ images with their associated five captions each. It is worth noting that during training, since each image I_i is associated to its five captions $(\mathcal{T}_i^1, \ldots, \mathcal{T}_i^5)$, we use them as five *different* training examples that results in the following set of training text-image pairs $\{(I_i, \mathcal{T}_i^1)$, ..., $(I_i, \mathcal{T}_i^5)\}$. Also, note that, since each image is associated to five captions, many evaluation protocols emerge in the literature. In our experiments, we used the most common one [123, 122, 279] which consists to treat each caption with its associated image as an individual pair of data. For instance, in the Flickr-8k each of the $5,000$ captions has to be illustrated by one image from the whole set of $1,000$ test images in the text-illustration task. For the MSCOCO dataset, which is the largest dataset, since no official splitting is provided, we could simply follow [122] and use $1,000$ validation images and $1,000$ test images (with their associated captions), randomly obtained from the whole set of data. Regarding the evaluation metric, we adopted recall at top K retrieved results (denoted R@K in the following) for the three datasets. As in the literature, we set K $\in \{1, 5, 10\}$, meaning that we evaluate the methods in their capacity to retrieve the good documents among the first (R@1), five (R@5) or ten (R@10) retrieved document(s). In order to highlight the best methods, we simply aggregate the results of each method in terms of R@1, R@5, R@10 on the two tasks (image-annotation and text-illustration), which we denote Avg.

5.2.1.2 Implementation Details

Here, we detail the representations used for each modality, the way we build and infer namecons and how we train the networks.

Visual and Textual Representations: We follow the literature [33, 271, 270, 128, 67] to represent images and captions. More precisely for images, a common procedure consists in extracting the penultimate layer of a pre-trained network. Here, we used **RestNet-101** [96] trained on ILSVRC [216] and the modified version (**VGG-4k**) [246] of VGG [228] that has been pre-trained on a diversified set of ImageNet [53]. More precisely, for the RestNet-101, we follow [271, 128, 67] and crop the original 256×256 image in ten different ways into 224×224 images: the four corners, the center, and their vertically flipped image. The mean value is then subtracted from each color channel. Then the resulting images are feed to the network that outputs one representation for each of them, which are then averaged. This setting will be denoted **10C** in the following. As commonly done in the literature, to represent captions, we primarily use the 300-dimensional means of word2vec [167] vectors of words in each caption (referred as **w2v-mean** in the following), and tf-idf-weighted bag-of-words vectors, with a dictionary size and descriptor dimensionality of $3,404$ (referred as **tf-idf** in the following). In particular, we pre-process all the captions with WordNet's lemmatizer [24] and remove stop words.

Learning and Inferring Namecons: To build the visual and textual NAMECONs, we respectively used the visual and textual representations described above. In particular, we used the k-means algorithm to cluster the whole space of unimodal data (visual or textual), with a set of $C^{\mathcal{V}}$ clusters for the visual representations and $C^{\mathcal{T}}$ clusters for the textual representations. The sizes of the visual $C^{\mathcal{V}}$ and textual $C^{\mathcal{T}}$ NAMECON codebooks are hyper-parameters of our NAMRank method, and can be set by cross-validation. More precisely, we cross-validated them on the validation set and use a set of values that is $1,000$ to half of the size the training-dataset (*e.g.*, $15,000$ for the Flickr-30k that contains $29,783$ instances), with a step of $1,000$. The experiments suggest that an amount of $5,000$ NAMECONs is the most performing. It results, in average, to 6 instances (visual or textual) per NAMECON. Once the NAMECON codebooks constructed for the two modalities, we infer them from all the visual $\mathcal{I}_i$ and textual $\mathcal{T}_i$ instance of the training-data. To do so, we applied a hard-coding scheme with a set of p nearest neighbors, with p being set by cross-validation between the values that goes from 1 to 10. The results suggest that $p = 1$ is the more performing configuration. Then as expressed in Section 5.1.2.5, the visual and textual NAMECON labels of all aligned data $(\mathcal{I}_i, \mathcal{T}_i)$ are merged together (through concatenation) and used as ground-truth NAMECONs (*i.e.*, as the vector $\widehat{y_i^{\mathcal{B}}}$ of Equation 5.2) for the data of the given pair to train the two-branch network.

Training the Network: As depicted in Section 5.1.2.1, we used a multi-layer perceptron (MLP) to train the $\mathcal{F}_{\mathcal{V}}$ and $\mathcal{F}_{\mathcal{T}}$ mapping functions. Following [271, 270], the MLPs of each branch contains two layers, one of size $2,048$ and one of size 512 (see Figure 5.2 to see on which layer dropout and batch-normalization, and l2-normalization are performed). We used standard parameters to train the weights of the mapping functions, that is, a starting learning rate of 10^{-2}, a momentum of 0.9, a weight decay of $5 \cdot 10^{-4}$ and a batch size of 512. Note that the learning-rate is not decreased during training, since this practice seems to stop the learning. The weights and biases of the networks are initialized randomly and optimized through the Adam optimizer [126]. Better results could be obtained with different architectures, but this is out of the scope of the paper. The input of the $\mathcal{F}_{\mathcal{V}}$ and $\mathcal{F}_{\mathcal{T}}$ mapping functions are respectively the 4096 or 2048-dimensional features extracted from VGG-4k and Resnet-101; and 300 or 3404-dimensional textual features obtained from word2vec-mean and tf-idf textual features, respectively. Regarding the output of each branch, they are the same than the size of the codebook NAMECONs, and thus respectively correspond to C^T-dimensional vector (for the bank of textual NAMECONs) and C^V-dimensional vectors (for the bank of visual NAMECONs).

5.2.1.3 Comparison Methods

Here, we describe the state-of-the-art methods used for comparison. Before listing them, it is important to note that, while not done here since it is an in-progress work of the Thesis, our final goal is the increase of universality through the learning of multimodal representations. Thus, it is important that the evaluation of the different methods is carried in a Transfer-Learning setting. However, since the communities of vision, NLP, and multimodal are rapidly moving, there is a lack of works that report *fair* comparisons of the methods, that is to say, with the same visual and textual settings. Indeed, it is obvious that multimodal representations could directly benefit from better unimodal representations (*e.g.*, word2vec features learned on more data, $36,000$-dimensional Fisher-vectors derived from hybrid gaussian-laplacian mixture models [128, 67], LSTMs that encode the natural temporarily of sentences [231] or even CNNs to represent texts [10, 125], and for the visual modality, better architectures [105, 276], more universal visual representations [243, 244], visual attention [177, 149] etc.) and/or better architecture for the mapping functions (*e.g.*, RRF [149], ensemble-models [149]) or by considering local data (*e.g.*, fragments [123, 122], fragments obtained by weak supervision [68], attention [177, 149]), semantic and context [106], etc.). Thus, to the best of our knowledge, it is at this stage, hard to say which principle to learn multimodal representations is the best. Hence, in order to tackle the problem of universality, we plan to conduct an extensive set of experiments to demonstrate which multimodal-learning principle is the most promising to universality. First, we plan to compare to one of the strongest statistical models for learning joint embeddings for different feature spaces when paired data are provided, namely Canonical Correlation Analysis (**CCA**) [95]. Another approach to consider, is the asymmetric mapping one, since recently, a couple of works [31, 33, 40, 58], demonstrated some interesting results on the cross-modal retrieval. The principle roughly consists to map the representation of one modality to those of another modality, through a simple MLP trained on paired data. In practice, two ways are possible, that is, from image to text representations (**im2txt**) and from text to image representations (**txt2im**). Thus we re-implemented the works of **AMECON [33]** and **Word2VisualVec [58]** that respectively corresponds to the im2txt and txt2im possibilities. Note that, the latter work belongs to an asymmetric approach, because, their "multimodal" embedding corresponds to a unimodal one (*i.e.*, textual for im2txt and visual for txt2im). Thus, in the cross-modal retrieval tasks, the images (resp. captions) are represented by the unimodal features, and the representations of the captions (resp. images) are projected into the visual (textual resp.) space through their learned mapping function for txt2im (resp. im2txt) way. Very recently, Eisenschtat *et al.* [67] proposed a symmetric mapping approach, that is called **2-Way-Net**. The principle of this approach is to map both, from the textual to visual representation and, from the visual to textual representation, by passing though a common multimodal layer. In practice they use two networks, one for each mapping functions, and most importantly, the same weights are used in both networks (*i.e.*, the matrix weights of the L^{th} layer of one way network is learned and used as is (with a transposition) in the branch of the other way network). Finally, one of the most efficient way to learn multimodal embeddings is to use a two-branch network that maps the representation of each modality into a multimodal embedding layer, and train it with metric-learning, and more precisely *bi*-directional ranking [271, 270] (denoted **Bi-Dir Ranking**). A special case of the latter method is the work of [77] that consists to map from visual to textual representation (*i.e.,* use one branch network) and train the network with a one-directional ranking (**One-Dir Ranking**). Note that, as in the mapping approach, this method is asymmetric. Note that for all the methods, to perform cross-modal retrieval, the multimodal representations are extracted from two through the trained mapping functions, and are compared by computing their similarity through euclidean distance. Note also that, all methods are implemented with the *same settings* (*i.e.*, same visual and textual representations, same architecture and same training set, as expressed in Section 5.2.1.2), thus only the principle to learn the multimodal representation varies. In addition to these re-implemented recent works, we also report the results of some reference works [123, 122, 127].

Method	Image-Annotation			Text-Illustration			Avg
	R@1	R@5	R@10	R@1	R@5	R@10	
Bi-Dir Ranking [271, 270]	26.6	53.0	65.9	22.7	50.1	63.3	46.9
Karpathy et al. [123]*	12.6	32.9	44.0	10.3	31.4	44.5	29.3
Kiros et al. [127]*	14.8	39.2	50.9	11.8	34.0	46.3	32.8
Karpathy et al. [122]*	22.2	48.2	61.4	15.2	37.7	50.5	38.7
Word2VisualVec [58] (txt2im)	16.8	40.3	53.0	0.1	5.6	10.0	21.0
AMECON [33] (txt2im)	18.3	41.3	53.5	6.1	9.8	12.2	23.5
AMECON [33] (im2txt)	9.6	13.5	19.2	20.0	49.8	64.2	29.4
NAMRank (ours)	**30.4**	**55.3**	**69.1**	**23.8**	**52.0**	**65.5**	**49.4**

Table 5.1: Comparison of our method (bottom) with state-of-the-art methods (top) on the Flickr30k dataset. The Bi-Dir Ranking method is used as reference, thus its scores are colored in red. All the methods are evaluated on the Image-Annotation and Text-Illustration tasks, for which we report their R@1, R@5 and R@10. We also report their average recall (last column with scores colored in blue) on all the tasks. Note that, in order to have fair comparisons, all the methods (except those marked with *) are re-implemented with the same settings (same visual and textual features, same architecture and same training set), thus only the principle to learn the multimodal representation varies.

5.2.2 Comparison to State-Of-The-Art Methods

In this section we report and discuss the results obtained by comparison methods and ours (NAMRank) on the famous tasks of image-captioning and text-illustration. In particular, the evaluation is carried on the Flickr-30k dataset. The results for both image-captioning and text-illustrations tasks on the Flickr-30k dataset are reported on Table 5.1.

Roughly, we observe that our method achieves the best results on both tasks. In average, it takes 2.5 points in terms of R@K compared to the best principle of the literature to learn multimodal embeddings, namely Bi-Dir Ranking (which is used as Reference in our evaluation scheme). This clearly demonstrates the interests of the proposed principle to additionally learn to predict NAMECONs. Another salient result lies with the fact that the methods belonging to the asymmetric approach (*i.e.*, Word2VisualVec, AMECON-txt2im and AMECON-im2txt) provide good results on on direction of cross-modal retrieval but provide much lower results on the other way. For instance, AMECON-im2txt is in line with the state-of-the-art results on text-illustration but is lower than the very old methods like [123]. This surprising phenomenon suggests that different modalities should **not** be aligned by bringing only one modality to the other, but an effort to bring both of them close to the other is necessary.

5.2.3 In-Depth Analysis

In this section, the goal is to evaluate the impact of each component of our method. To do so, we conducted an ablation study where we removed one or many components from our NAMRank method. In particular, the elements removed and studied are (i) the use of ranking loss, (ii) the use of same weights in all branches, (iii) the use of same NAMECONs in all branches, and (iv) the use of one of

Method	T.S	S.W	**Image-Annotation**			**Text-Illustration**			**Avg.**
			R@1	**R@5**	**R@10**	**R@1**	**R@5**	**R@10**	
Ranking only	n/a	n/a	26.6	53.0	65.9	22.7	50.1	63.3	46.9
NAMECON	n/a	✓	18.0	39.4	51.8	15.9	39.6	53.6	36.4
NAMRank	Par	✗	24.4	51.4	64.4	21.9	48.8	62.8	45.6
NAMRank	Par	✓	27.2	56.5	68.1	21.5	48.5	60.5	47.1
NAMRank	Seq-Par	✗	26.4	53.7	65.1	21.7	49.9	62.7	46.6
NAMRank	Seq-Par	✓	30.4	55.3	69.1	23.8	52.0	65.5	**49.4**
Ranking only	n/a	n/a	23.4	50.2	63.3	20.1	44.8	59.3	43.5
NAMRank	Par	✓	24.6	52.1	66.5	20.3	47.7	62.0	**45.5**
NAMRank	Seq-Par	✓	26.0	52.4	65.9	18.4	45.4	59.7	44.6

Table 5.2: Ablation study of our method and comparison to the baseline method *Ranking only*. The scores reported here correspond to image-annotation and text-illustration results of the methods on the Flickr30k dataset. On **top** of the table, we report the results with the following features: 3404-dimensional tf-idf features for captions and 2048-dimensional Resnet-101 10C for images. At **bottom**, we report the results with: 300-dimensional w2v-mean features for texts and 4096-dimensional VGG-4k features for images. T.S and S.W respectively state for the Training Strategy and Same Weights, while Par and Seq-Par respectively refer to the Sequential and Sequential-Parallel learning strategies.

the different learning strategies[1]. To perform the evaluation, we used the Flickr-30k dataset and two different settings for the unimodal semantics, *i.e.*, VGG-4k for images and w2v-mean for captions; and Resnet-101 10C for images and tf-idf for sentences. The results of the ablation study are reported in Table 5.2.

Beside the fact that our method outperforms all the other baselines on both features settings, we observe that compared to the *ranking only*, a gain of performance is observed only when the same weights and same NAMECONs are used in all branches. This latter, highlights the importance to have the same weights and NAMECONs in all branches. With dense textual features (bottom part of the Table), the Sequential-Parallel learning strategy seems to be more performing than the Parallel one, while it is the contrary with very sparse textual features (top of the Table). While the NAMECONs only (i.e., w/o ranking loss) seems to already provide satisfying results, it is important to also consider the ranking loss in the final objective. This may due to the fact that to perform cross-modal retrieval, learning to exactly match instances is important. Finally, our method with ranking, NAMECONs, same weights and NAMECONs in all branches is better than the the *ranking only*, clearly meaning the interest of NAMECONs, and more precisely, the utility of learning to exactly match instances but *also* to group them semantically.

5.3 Conclusions

In this work, we proposed to preserve unimodal semantics in multimodal representations learned through classical bi-directional ranking loss. To do so, we proposed to rely on the introduced principle of non-semantic meta-categories (NAMECONs) that are both non-semantic (not interpretable

[1]The results here are preliminary, and a complete study of all variants of our method is required.

or not yet interpreted) and meta-categories (capture common properties between similar instances). NAMECONs are specifically used as labels of an additional classification loss that tends to regularize the alignment of corresponding visual and textual representations.

We conducted extensive experiments on text-illustration and image-captioning tasks over three commonly used datasets (*i.e.*, Flickr-8k, Flickr-30k and MSCOCO), and our method significantly outperforms all comparable state-of-the-art methods. We also conducted an in-depth analysis of our method and demonstrate the utility of each component as well as the utility of the proposed NAMECON principle compared to only using the bi-directional ranking loss. While here we did not evaluated our in terms of universality (in a transfer-learning scheme), we think it will be even more effective on different target-tasks, but this is discussed in more details in the next section.

6

Conclusions and Perspectives

Contents

In Section 6.1, we summarize the salient points that were shown through the contributions presented in Chapters 3 to 5. Section 6.2 outlines prospective directions for further research.

6.1 Summary of Conclusions & Discussions

In this Thesis, beyond the proposal of the challenge of universality and a protocol with metrics to evaluate it (Section 6.1.1), we explored three directions to universality: (i) in a context of semantic representations, we proposed to semantically reduce the noise introduced in the representation, that results from an increasing in the number of detectors through data-enlargement (Section 6.1.2); (ii) in a context of CNN-representations, we proposed to diversify the features with a simple approach that requires no additive work to collect and annotate data (Section 6.1.3); and (iii) in a context of multimodal-features, we proposed to increase the scope of action of the individual detectors through the preservation of unimodal semantics into the multimodal representation (Section 6.1.4).

6.1.1 Evaluation Protocol and Metrics for Universality

In Sections 4.2 and 4.3.2.1 of Chapter 4, we respectively proposed a set of metrics as well as a protocol for evaluating the increase of universality. More precisely, regarding the metrics we first identified a set of desirable properties for such evaluation and proposed three metrics (Borda Count based on statistical order, aRG and mRG based on the average or median of relative gain compared to the scores of a reference method) that we compared to the most natural baseline (averaging scores) as well the VDC proposed by Rebuffi *et al.* [210]. We especially showed and empirically demonstrated the limits of the last two ones, which are partially and even almost totally alleviated by our metrics. It is important to notice that the proposed metrics are not limited to evaluate visual and multimodal representations and could be easily used by the NLP community for evaluating universalizing methods. In summary, all the metrics (including our best one that verifies almost all the properties) are not perfect, meaning that the evaluation of universality remains an open problem. Thus, identification of desirable properties as well as metrics themselves are welcomed. Regarding the protocol of evaluation, we made the choice to evaluate the learned representations in a transfer-learning scheme with a classical scenario where representations are not modified during the adaptation to the target-tasks. We motivated this choice by a link with the popular cognitive study of Atkinson *et al.* [8]. While such protocol is commonly used in NLP [44, 42, 41, 124, 235, 180], to the best of our knowledge, we are the first to propose to use such protocol for the visual and multimodal representations. Finally, in practice we evaluated the universalizing methods with target *classification* problems. It is obviously desirable to evaluate universality with many different tasks (*i.e.*, boundaries, saliency, detection, segmentation, etc.) as performed by Kokkinos [130], more complex tasks (such as 3D recognition [184], Augmented Reality [173, 174] and Diminished Reality [220] in vision, or event detection [129, 59], POS tagging [165], automatic extraction of bilingual multiword expressions [223] in NLP), and even on many modalities to handle more tasks such as cross-modal retrieval, VQA [6], or Visual Dialog [51]. Moreover, while ImageNet already covers a large set of semantic domains, it could be also interesting to evaluate universality on more domains, such as the medical one (as in [23]), images of scenes [297], or even low-level objects such as structures [39], materials [19], illuminations [172], or even satellite images [268] with low-resolution [141]. More fundamentally, such goals raise the question of the characterization of the source and target datasets, as well as their tasks, ideally independently of the method used to process them, in order to be able to predict how they relate to each other on a semantic point-of-view, and consider them in the evaluation metric.

6.1.2 Semantically Reducing Noise on Large Semantic-Representations

In Chapter 3 we proposed two methods of sparsification for semantic representations. Both methods aim to reduce the noise generated by the increase of object-detectors in semantic representations. The first method (CBS) consists of an adaptive sparsity according the image and/or the local region content that is modeled through the entropy and the confidence of the output classifiers. The second one (D-CL) consists of the exploitation of human knowledge organized according to a semantic hierarchy, to identify groups of concepts according to their categorical-level. This latter, aims to process the groups of visual concepts differently from each other, according to their categorical-level. More importantly it boosts the scores of generic concepts, avoiding their consideration as noise when a sparsification process is applied. Both methods outperform existing semantic representations in a transfer-learning scheme on multiple target-datasets, showing their ability to increase the universality of semantic representations.

The CBS contribution was applied to semantic representations, but is nevertheless not limited to this context. In particular, it would be interesting to see whether the same type of noise could be encountered on internal layers of CNNs and whether our proposal could be a solution. Another aspect is on the proposed solution to the problem of the presence of noise. One could rely on learning it from data as proposed in [204]. Regarding the D-CL that is especially designed to semantic representations with high-level detectors, we proposed a simple protocol to determine the categories from the basic and superordinate-levels, but it could be interesting to rely on large scale categorical-level hierarchies as we proposed in the second part of this thesis [243] or even try to identify them automatically as proposed recently by Wang *et al.* [267]. Last but not least, four directions of improvement seem to emerge for semantic representations: (i) more annotated data; (ii) more structured data; (iii) better individual classifiers; (iv) better lower-level representations. The first and third point seem less interesting since using more annotated data requires costly human annotation and using better classifiers would remain to a simple application of another classifier, except if a focus is done on proposing a completely new classifier. Regarding point (ii), the structure used could be determined in another way, that would bring more trainable information. For example, one could consider the full captions describing the categories or other relations than the “is-a” we used here. It is also possible to get new categories through some unsupervised clustering to get non-semantic meta-concepts [33] or even sub-categories to define more precise detectors. For the direction (iv), an advantage of semantic representations is their interpretability while they are trained on top of lower-level representations that are not directly interpretable. However, these last representations can themselves be improved to include more semantics. In particular, they can be themselves more universal, as we discuss in the next section.

6.1.3 More Features on CNN-Representations Without More Annotated Data

In Chapter 4, we proposed a universalizing method, based on an ensemble-model strategy, that consists to vary the discriminative problems solved by each network of the ensemble to learn CNN features. The variation of the discriminative problem is based on category-grouping and more precisely, on categorical-levels used in the human-categorization process. We showed that it increases the image classification performances in a transfer learning context, compared to a standard learning strategy. It thus increases the universality of a given CNN-based representation, and the proposed method uses exactly the same data while the annotation process concerns the categories and not each image. It thus results into a particular cheap process in terms of required manual work. Finally, we investigated the reason why it increases the universality. On the one hand, we showed that the learned features are more different than those resulting from the usual learning strategies [143]. In addition, we also showed that this difference is always beneficial in terms of discrimination, exhibiting a kind of complementarity for this aspect that was not a priori obvious. While difficult, it certainly worths to push further the investigation to better understand the underlying learning process. In particular, it would be interesting to determine how much sparse or distributed is the coding of a particular image, or a set of images belonging to a given category, at the different layers.

Alternative approaches can be considered to implement each of the three main modules of our approach (SPV, CNN architecture and fusion strategy). For the SPV, one can modify the set of images or the set of categories. Regarding the images, the SPV includes *data augmentation* that modifies the images according to a geometric transform or a noisy process (compression, jittering). It also includes the addition of new annotated data of existing categories. Previous works usually preferred to extend the set of categories, although one could imagine a process of replacement of some of them.

We nevertheless showed that grouping categories provides better results and has the advantage to be cheap in terms of manual work. We decided to group according to categorical levels and showed this choice is better than several other baselines (grouping randomly, grouping through levels of the ImageNet hierarchy, and grouping by clustering). We argue that using categorical-levels leads to better match the human user need but there is no guaranty that it is the best approach and alternative grouping strategy may obtain better results. For our part, we nevertheless think that it worth to explore a splitting strategy, as it is detailed in Section 6.2.1.

Regarding the second module, it also exists many ways to do it. For instance, a natural way to do it is to train one net per discriminative problem (DP), but it may have a large cost in terms of amount of parameters, when considering many DPs. At that time, we wanted to demonstrate that a clever use of data with the same algorithm (same architecture and learning-strategy), is beneficial. Thus, we did it with the same network.

Regarding the fusion strategy, the concatenation originally used had the drawback to increase the dimensionality of the representation linearly with the number of DPs considered. We thus proposed to add a dimensionality reduction method to solve this problem. However, instead of concatenating the representations, one could consider pooling (*e.g.*, sum, average, max) them, but it is not better than concatenation (as mentioned in Section 4.1.2.6). Regarding the dimensional reduction, we could use a classical method (*e.g.*, PCA or even LDA since we have categories on the train-set). Important to note, if so, we should do this on the training environment, not the target-dataset. It is worth noting that FSFT not only reduces the dimensionality but also increases the performances, while a classical dimension reduction usually hurts the final performances.

6.1.4 Preserving Unimodal Semantics on Multimodal Representations

In Chapter 5 we claim that universality is not limited to vision and that representations should go towards the ability to perceive the world through multiple modalities (vision, language, sound). In that sense, learning multimodal representations on top of unimodal ones contributes to improve the universality of representations. In particular, in this work, we proposed to preserve unimodal semantics in multimodal representations learned through classical bi-directional ranking loss. It relies on the notion of non-semantic meta-categories (namecons), that is used as labels of an additional classification loss that tends to regularize the alignment of corresponding visual and textual content according to a more abstract view of the underlying concept.

We evaluated on Flickr30k and highlighted its interest compared to methods that learn through one of the best principle to learn multimodal representations to date, namely bi-directional ranking [271, 270]. An evaluation on other cross-modal datasets will be conducted in a near future to assert the validity of the approach in various contexts. Beyond cross-modal retrieval, we are also interested in evaluating it in a transfer-learning scenario, consisting in learning the network on one *large* dataset in a general domain and testing the ability of the system to provide a universal representation that exhibits good performances on *other* domains with *small* datasets. As in [3], since the source-task (*e.g.*, MSCOCO), will be in a large-scale setting, we hope that the bi-directional ranking [271, 270], which is a ranking-based algorithm will be even less performing than our method that also considers a one-vs-rest classification strategy. Indeed, the problem with the former method is that it encourages the learning of detectors that are too specific to each instance and thus less able to recognize concept in very similar instances. Simply said, we think it is less suited to universality than our method.

6.2 Directions for Further Research

In this Thesis, we have three main contributions to improve universality, but the MulDiP-Net seems the most promising one. We made particular choices for each of the three modules that composes the approach, leading to significant results in terms of universality improvement. However, alternative choices can be considered, leading to three short-term perspectives. In addition, we propose longer term perspectives that could build interesting contribution upon the proposed work.

6.2.1 SPV by Splitting: From Specific to Finer Categories

For the variation of the source problem, we preferentially adopted a grouping of specific categories according categorical levels to obtain generic ones. As underlined, there is no insurance this choice is optimal and other grouping may give better results. However, another results is of interest, namely that the performances in transfer learning with a unique CNNs are even better than when the categories learned are specific (see table below).

Set of Label-Sets ($\mathcal{B}$)	**VOC07**	**NWO**
$\{\mathcal{L}_2\}$ (Superordinate)	58.9	37.4
$\{\mathcal{L}_1\}$ (Basic-Level)	70.0	51.0
$\{\mathcal{C}\}$ (Subordinate=specific)	70.3	51.2
$\{\mathcal{C}, \mathcal{L}_1\}$	72.5	54.1
$\{\mathcal{C}, \mathcal{L}_1, \mathcal{L}_2\}$	**73.0**	**54.9**

Table 6.1: Impact of the categorical-level considered in MulDiP-Net, in terms of performance in transfer learning on Pascal VOC 2007 (**VOC07**) and Nus wide object (**NWO**).

Hence, one can naturally wonder whether a network learned on *finer categories* than specific ones could obtain better performances. However, to get the finer categories, it is not possible to rely on semantic knowledge organized in the form of hierarchies with "is-a" relations, since specific categories are the *leaf nodes* of the hierarchy. Thus, alternatives to get finer categories should be considered.

A first idea would be to use the pose of the objects, although the precise implementation is not obvious since this pose needs to be determined without annotation. A more fruitful direction would be to use unsupervised learning to determine finer categories. Indeed, when the categories are grouped hierarchically without considering the categorical levels , the results are improved over the reference and the random grouping (Fig. 4.14). It is also the case when the groups are determined by KMeans on the penultimate output of the network (clustering grouping). Since the specific categories are, by definition, quite homogeneous in terms of semantics, it thus makes sense to try to determine finer categories using such clustering. Actually, with a master internship (Julien Girard), we proposed a method on this topic and the first results are quite encouraging.

6.2.2 Exploring FSFT in Theory and Practice

In the MulDiP-Net method, we proposed a new method called *Focused Self Fine-Tuning* (FSFT) to reduce the dimensionality of the representation and the amount of parameters in the final model, while maintaining or improving the universality.

The theoretical arguments to justify this efficiency are nevertheless still limited since it mainly relies on empiric observations of [285] concerning the comparison of self-training and fine tuning with regards to the co-adaptation of features learned.

Beyond theoretical aspects, we did not explore all the potential of FSFT. In particular, we applied it to the penultimate fully-connected layers of the networks only, since we were mainly interested in merging the representations to perform transfer learning with vectors of limited dimension. It would be interesting to apply the FSFT on other internal layers and in particular on convolutional layers. This would be in any case necessary with a network without internal fully-connected layers, such as GoogleNet [237], DarkNet [212] or ResNet [96].

More interestingly, one can even imagine a recursive process, consisting in training the full network with K layers then applying FSFT to the $K-1$ last layers, then the $K-2$ last and so on. While costly, this process should lead to improve incrementally the universality of the penultimate representation. Indeed, at iteration N, the network will have to focus the training on the $K - N$ layers, with exactly the same data (that was used to train much more parameters in the previous iteration).

6.2.3 Efficient Parametrization of the Model

In the second module of the MulDiP-Net method, we proposed to rely on the training of a full network on the newly obtained source-problems (SPs). While, a significant jump of performances was observed with this method, it has the major drawback to linearly increase the amount of parameters, dimensionality of the representation and inference time, with the amount of SPs considers. As a consequence, imagining more than two or three SPs seems complicated. While the proposed FSFT was a first attempt to alleviate the dimensionality limitation, it is natural to pay special attention to the aspect of parameters.

To do so, an important question need elements of answers: *Should we pay attention to parametrization during learning or after learning?* Simply said, should we *directly* learn efficiently or learn in "brute-force" (without considering parametrization) *then* compress the model? For the first point, one could follow [272, 210, 219] which consists in relying on the background of a master-network (full network trained on the initial SP) by adding a small amount of parameters in each layer (on top of previously learned ones) and learning them on the new SPs. While such approach could be interesting, it is very biased by the knowledge encoded in the neurons learned on the previous environments, which is non-desirable to learn really *new* features. Such problem could be alleviated by learning in "brute-force" then compress the model. In the literature, many works proposed to compress neural-network models, through Pruning [160, 170, 159] or Knowledge Distillation (KD) [101, 213]. A natural idea is to consider the application of such methods on the learned ensemble (that consists of multiple SP$_i$-Net). However, while the pruning methods are a trade-off between accuracy and amount of parameters, it could significantly decrease the performances. Regarding the KD method, the problem is even more complicated, since the original formulation consists to train a small network that mimics the behavior of an ensemble of models that were trained on the *same* problems (*i.e.*, with the same *amount* of neurons at their output layers). However, in our case, this requirement is not encountered, since each SP$_i$-Net has been learned on a particular problem with a particular number of categories to recognize. An alternative idea could thus to imagine, a mapping process, that consists in training an ensemble with full networks (one on the initial SP, denoted *Master-Net* and one per new SP, denoted SP$_i$-Net), and then learning to map from the penultimate layer (and thus all the knowledge encoded in the previous layer) of the Master-Net, to the penultimate layer of any SP$_i$-Net. This will have the main advantage to only add (to the final model) the small amount of parameters in the mapping functions, without considering all those of the lower layers of the SP$_i$-Nets. Potentially, the mapping functions could provide even better performances than the initial ensemble, since they are in a certain sense, learned to collaborate, which was not the case during the independent learning of the ensemble.

6.2.4 Longer Term Perspectives

We showed that guiding the grouping with explicit knowledge on human categorization [214] leads to more universal representations. It is possible to go deeper in the universality by increasing the scope of the use of *a priori* knowledge and by considering other kinds of semantic and structural relations than hierarchical relations. In particular, one can consider to exploit spatial knowledge and spatial reasoning which is known to be of prime importance for visual recognition and understanding [15, 7, 107, 108]. The importance of learning structured representations capturing objects and their semantic relationships, as for instance scene graphs [116], label relations graphs [52] or knowledge graphs [162], has been at the core of some recent works [157, 152, 200, 278, 142, 288]. Nevertheless, to our knowledge, visual and semantic relationships between objects have never been used in the context of learning universal representations.

The principle of knowledge distillation [101] could also contribute to improve universality in a complementary direction to the use of semantic and visual relationships, since it adds knowledge at another step of the learning process. In particular, the recent teacher-student framework proposed by Hu [104] enables the distillation of diverse knowledge sources, including logic-based ones and thus, in some sense, is a first step to include expressive formal knowledge models such as ontologies. This type of approach could lead to the integration of rich and expressive spatial knowledge [14, 108, 288] in this kind of frameworks.

In the context of transfer-learning and domain adaptation, an important point that has been weakly processed, to our knowledge, is the study of the dependencies (intersection, complementarity, conflict, etc.) between the source problem and the targeted problems. From our point-of-view, knowing precisely these dependencies could be of great interest to conduct and parameter the transfer-learning process. A possible direction of research is to study the alignment of *ontologies* describing both problems as in Todorov *et al.* [251, 250] or even more complicated, the alignment of *tasks* as proposed very recently by Zamir *et al.* [290].

7

Publications

7.1 Articles in Peer-Reviewed Journals

- **Vision-Language Integration using Constrained Local Semantic Features**,
  Youssef Tamaazousti, Hervé Le Borgne, Adrian Popescu, Etienne Gadeski, Alexandru Lucian Ginsca, and Céline Hudelot,
  Computer Vision and Image Understanding (CVIU), 2017.

- **Descripteur sémantique local contraint basé sur un descripteur RNC diversifié**,
  Youssef Tamaazousti, Hervé Le Borgne, Adrian Popescu, Etienne Gadeski, Alexandru Lucian Ginsca, and Céline Hudelot,
  Journal Traitement du Signal, 2017.

7.1.1 In Preparation or Revision

- **Preserving Unimodal Semantics in Multimodal Representations Through Non-Semantic Meta-Concepts**,
  Youssef Tamaazousti, Hervé Le Borgne, Ines Chami, Céline Hudelot, and Yannick Le Cacheux,
  In preparation for ACM Transactions on Multimedia (ToM).

- **Learning More Universal Representations for Transfer-Learning**,
  Youssef Tamaazousti, Hervé Le Borgne, Céline Hudelot, Mohamed-El-Amine Seddik, and Mohamed Tamaazousti
  In revision for IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI).

7.2 International Peer-Reviewed Conferences

- **MuCaLe-Net: Multi Categorical-Level Networks to Generate More Discriminating Features**,
  Youssef Tamaazousti, Hervé Le Borgne and Céline Hudelot,
  IEEE Computer Vision and Pattern Recognition (CVPR), 2017.

- **AMECON: Abstract Meta Concept Features for Text-Illustration**,
  Inès Chami*, Youssef Tamaazousti* and Hervé Le Borgne,
  *Both authors contributed equally,
  ACM International Conference on Multimedia Retrieval (ICMR), 2017. (Oral)

- **Supervised Learning of Entity Disambiguation Models by Negative Sample Selection**,
  Hani Daher, Romaric Besançon, Olivier Ferret, Hervé Le Borgne, Anne-Laure Daquo, and Youssef Tamaazousti,
  International Conference on Computational Linguistics and Intelligent Text Processing (CICLing), 2017.

- **Diverse Concept-Level Features for Multi-Object Classification**,
  Youssef Tamaazousti, Hervé Le Borgne and Céline Hudelot,
  ACM International Conference on Multimedia Retrieval (ICMR), 2016. (Oral)

- **Constrained Local Enhancement of Semantic Features by Content-Based Sparsity**,
  Youssef Tamaazousti, Hervé Le Borgne and Adrian Popescu,
  ACM International Conference on Multimedia Retrieval (ICMR), 2016. (Oral)

7.2.1 In Preparation

- **Learning Finer-class Networks for Universal Representations**,
  Julien Girard*, Youssef Tamaazousti*, Hervé Le Borgne and Céline Hudelot,
  *Both authors contributed equally,
  In preparation for British Machine Vision Conference (BMVC), 2018.

7.3 National Peer-Reviewed Conferences

- **Désambiguïsation d'entités nommées par apprentissage de modèles d'entités à large échelle**,
  Hani Daher, Romaric Besançon, Olivier Ferret, Hervé Le Borgne, Anne-Laure Daquo, and Youssef Tamaazousti,
  COnférence en Recherche d'Information et Applications (CORIA), 2017.

- **Descripteurs à divers niveaux de concepts pour la classification d'images multi-objets**,
  Youssef Tamaazousti, Hervé Le Borgne and Céline Hudelot,
  Reconnaissance des Formes et Intelligence Artificielle (RFIA), 2016. (Oral)

- **Agrégation de descripteurs sémantiques locaux contraints par parcimonie basée sur le contenu**,
  Youssef Tamaazousti, Hervé Le Borgne and Adrian Popescu,
  Reconnaissance des Formes et Intelligence Artificielle (RFIA), 2016. (Oral)

7.4 Patents

- **Procédé d'obtention d'apprentissage d'un premier réseau de neurones convolutif vers un deuxième réseau de neurones convolutif**,
  Youssef Tamaazousti, Julien Girard, Hervé Le Borgne and Céline Hudelot,
  Patent filled on June 2018 at INPI, ref 1854785.

- **Procédé d'obtention d'un système de labellisation d'images, programme d'ordinateur et dispositif correspondant**,
  Youssef Tamaazousti, Hervé Le Borgne and Céline Hudelot,
  Patent filled on December 2016 at INPI, ref 1662013.

7.5 Other Publications and Talks

- **Learning More Universal Representations**,
  Youssef Tamaazousti, Hervé Le Borgne and Céline Hudelot,
  Both at GdR ISIS (CNAM) and Thales ("Journées de Palaiseau"), 2018 (talks)

- **Image Annotation and Two Paths to Text Illustration**,
  Hervé Le Borgne, Etienne Gadeski, Ines Chami, Thi Quynh Nhi Tran, Youssef Tamaazousti, Alexandru Lucian Ginsca, and Adrian Popescu,
  Working Notes for the ImageCLEF 2016 Workshop, (ImageCLEF), 2016.

- **Contrained Local Semantic Features**,
  Youssef Tamaazousti, Hervé Le Borgne,and Adrian Popescu,
  International Computer Vision Summer School (ICVSS) 2016 (poster)

- **Diverse Concept-Level Features for Image Classification**,
  Youssef Tamaazousti, Hervé Le Borgne and Céline Hudelot,
  Mathematical Modelling of Complex Systems (MMCS), dec 2016, Chatenay-Malabry (talk)

A

Tasks and Datasets

We describe the classical pipelines used in the literature for the tasks of classification (Section A.1) and cross-modal retrieval (Section A.2). Then, in Section A.3, we describe all the datasets used in this Thesis to evaluate methods on the classification and cross-modal retrieval tasks.

A.1 Classification Task

The general image classification scheme, illustrated in Figure A.1, follows a supervised learning approach, and consist in two main phases: an off-line *training* phase and an on-line *testing* phase. More precisely, given a training set composed of images and their associated labels, the off-line training phase usually consists in first, describing each image with an adequate **image representation**, then use these representations and their associated labels as input elements for a machine learning algorithm to learn a classification model. This model ("learned model") is learned to map input image representations to their associated labels. The learned model is then used in the on-line phase to predict the labels of the unknown images (*i.e.*, not contained in the database of the training phase). Indeed, on the test phase, each image that belongs to the test set (that are different from images of the training set but are associated to the same labels of the training-set), is described through an image representation (the same description as that used in the training phase), which is passed to the learned model in order to predict (through comparison with the learned data) the labels describing this image. Note that, on the on-line phase, the unknown images can also come from any other place than the testing set (*i.e.*, for any application that needs an automatic visual recognition system). Regarding the evaluation of the different classification methods, many evaluation-metrics have been proposed in the literature. Most of them consist in comparing the predicted labels (*i.e.*, labels predicted by the learned classification model) to the ground-truth labels.

A.2 Cross-Modal Retrieval Task

Cross-modal retrieval consists in retrieving image from text (text illustration) or captions from images (automatic captioning). As in the classification scheme, the cross-modal retrieval also contain

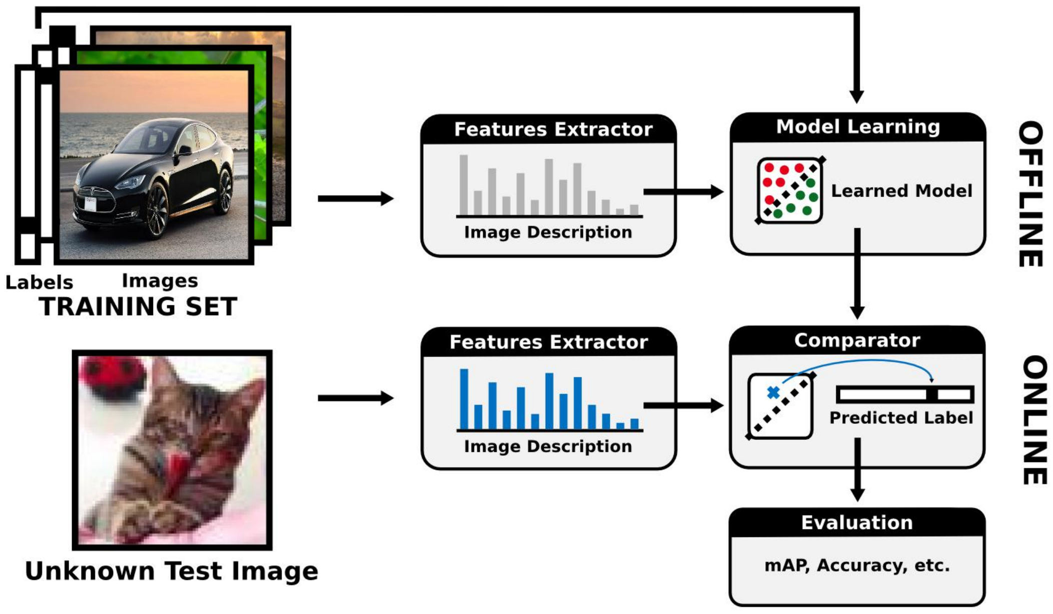

Figure A.1: Illustration of the general classification scheme. Each block here is a system and inside each block, we illustrate its output. An arrow that goes from the inside of a block to another block means that it takes the output of the first system and use it as input for the second system. In the off-line phase, images of the training set are represented using a features extractor then each representation is used as data point with their associated labels to learn a model in a supervised manner. In the on-line phase, unknown images (from the test-set or from any application) are represented (through the same features extractor as in the off-line phase) then, they are passed to the learned classifier model in order to output predicted labels. The predicted labels are then compared with the desired ones using evaluation metrics (mAP, Accuracy, etc.) in order to assess the prediction. Best view in color.

an off-line training phase and on-line testing one. For the training phase, the methods are generally learned in a supervised approach with a set of paired data from two modalities (*e.g.*, on caption describing an image). In particular, the goal is to learn a multimodal representation (on top of unimodal representations) that, after projection of the unimodal representations of paired data into the common space, the multimodal vectors are close to each other, and inversely far for unpaired data. The learned model is then used in the on-line phase to find, for a query of given modality (image or text), the closest instances of the other modality (text or image, respectively). More precisely, unimodal representations from each data (from the query and collection) are extracted and feed to the learned model that outputs the multimodal representation. Then, the representations of the query and all those of the collection are classically compared through an euclidean distance. The classical way to evaluate such cross-modal retrieval systems is to use Recall@K. It evaluates the capacity of the methods to retrieve the good documents among the first K ones.

A.3 Training and Evaluation Datasets

Most of the datasets are composed of three pre-defined splits: training, validation and testing splits. We organized them in two categories: object-centric datasets and scene-centric datasets, regarding the entities contained in their images and we will described these datasets according to the following criteria: (i) some statistical information: number of images, number of concepts, average number of concepts per image, data repartition into train/test splits; (ii) quality of the images and (iii) semantics (or level of semantics) of the classes.

Object datasets:
In the context of classification, we consider thirteen widely used datasets. The first two (based on ImageNet [53]) and the third one (Places205 [297]) are considered as large datasets since they contain millions of images and thousands of categories. All others are much smaller and are much more used since they are historically quite popular. All of them are described in the following.

- **ImageNet** [53]: It is a large database that, currently, contains 14 million images labeled among 22,000 categories. The dataset is organized according to a semantic hierarchy (WordNet [168]) with hyponymy relations (*i.e.*, "is-a" relations). It is important to note that, all categories contain unique images (*i.e.*, that are not present in other categories). It contains eleven levels of specificity (*i.e.*, maximum amount of entities subsuming a leaf-node of the hierarchy) and contains many categories from many different domains, such as *animals*, *vehicles*, *furnitures*, etc. However, it has the particularity to be very unbalanced, since it contains some categories with a huge amount of images and others with a near-zero amount. Moreover, ImageNet does not contain pre-defined train/val/test splits, since it is too large to be commonly used. Instead, the providers of ImageNet also provide a clean subset called ILSVRC described below.

- **ILSVRC** [216]: It is a subset of ImageNet and has been proposed as data for a challenge of large-scale visual recognition. Indeed, its name (ILSVRC) comes from ImageNet Large Scale Visual Recognition Challenge. It has been designed with three objectives: (i) being large – *i.e.*, with a large amount of categories and images per class; (ii) being fine-grained – *i.e.*, with each category labeled according to a set of specific words, such as *rottweiler*, *persian cat*, *office desk*, etc.; and (iii) being clean – *i.e.*, all images inside a category are very representative of the category, with the entity represented by the category positioned in the center of the image with a reasonable size compared to the size of the image. Thus, in terms of size, ILSVRC has been splitted in three pre-defined sets: the training set that contains 1.2 million images and the validation and testing sets that contain $50,000$ images each. Almost all categories contain at least $1,000$ images and most of them contain $1,300$ images. For the fine-grained categories, they took almost all the categories of ImageNet that correspond to the leaf nodes of its hierarchy.

- **Caltech-101** [73] / **Caltech-256** [93]: They are mono-object datasets where images are labeled by one of 101 categories for the former and 256 for the latter. Both of them contains also one background category that contain images from different categories. A popular split of the literature is to take $30/60$ images per-class for training (which results in $3,060/15,420$ in total) and the rest for testing (total: $3,022/15,187$) in Caltech-101/256, respectively. A key particularity of these datasets is that they mostly contain images in a white background, as e-commerce images, and each object is localized in the center of the image. Regarding their categories, they are part of basic-level categories (*i.e.*, categories most commonly used by Humans to categorize objects) [1].

- **Pascal VOC 2007** [71]: It is an object classification benchmark that contains a total of $9,963$ images, labeled among 20 basic-level categories (such as *car*, *table*, *bird*, etc.). A pre-defined split is released, consisting of $5,011$ training-images and $4,952$ testing ones. It has the particularity to contain *multiple* objects in each image. Hence, each object can be very localized in the

[1]More details are given in Section 2.2.5 regarding categorical-levels and more precisely, basic-level categories.

images, that is to say, different images of one object-category can contain the object at different spatial locations of the image.

- **Pascal VOC 2012** [70]: It is an extension of the Pascal VOC 2007 dataset. The extension carried on the enlarging of the number of images, since it contains $22,531$ images that are split into $5,717$ training-images, and $10,991$ testing-images. Note that, even the validation data is pre-defined and it contains $5,823$ images. An official evaluation server is available for this dataset, thus, the ground-truth of the test set is not publicly available.

- **Nus-Wide Object** [38]: It is a subset of the relatively large Nus-Wide dataset. The entire Nus-Wide dataset contains many images ($161,789$ training images and $107,859$ images for the test). This subset contains $36,255$ images in total labeled among 31 generic object categories. The whole data is separated in $21,709$ training-images and $15,546$ images for test. An important aspect is that it is a multi-label dataset, thus each image is labeled by one or several labels from the 31 categories.

Flickr-8K [103], **Flickr-30k [287]** and **MSCOCO [144]** are datasets used for the cross-modal retrieval task. Thus they, all provide pairs of data, that is to say, captions (generally 5) obtained by different annotators that describes the images. In the literature, for cross-modal retrieval tasks, the test data of one modality (*i.e.*, images or captions) are used as *queries* and the test data of the other modality as the *collection* (*i.e.*, images or captions, respectively). Regarding the total amount of data (train), Flickr-8k contains $8,000$ images, Flickr-30k contains $31,783$ images and MSCOCO, which is the multimodal dataset, contains $82,783$ images. Since in the three datasets, each image is associated to five captions, they respectively contains, $40,000$, $158,915$ and $414,113$ captions. The tree datasets need to be splitted in three sets, namely training, validation and testing. However, only the Flickr datasets contain official splits for cross-modal retrieval tasks. Indeed, for Flickr-8k the training, validation and testing sets respectively contain $6,000$, $1,000$ and $1,000$ images with their associated five captions each. For Flickr-30k, which is larger, the training, validation and testing sets respectively contain $29,783$, $1,000$ and $1,000$ images with their associated five captions each. MSCOCO, contain $80,783$, $1,000$ and $1,000$ train, validation and test images.

Scene datasets:
Regarding the context of scenes, two main datasets are used. The first one, is called Places and contains a large amount of images and labels, while the second one, is much smaller and thus contains much less images and labels. Each of them is described in details in the following items:

- **Places205** [297]: It is one of the largest scene dataset in the literature. It contains 205 scene categories and 2.5 millions of images. The 205 categories can be from any type, as long as it represents a scene (*e.g.*, *train-railway*, *bedroom*, *football-stadium*, etc.) and each of them contains at least 5000 images. Regarding the splits, for the training a set of 2,448,873 images from the 205 categories is provided, the validation set contains 100 images per category (which results in a total of 20,500 images) and the test set contains 200 images per category (which results in a total of 41,000 images)

- **MIT Indoor** [205]: It is a popular scene recognition dataset that consists to categorize images among 67 categories of indoor places (*e.g.*, *kitchen*, *library*, *office*, etc.). Each image is associated to one category only. A common training-test split consists to take 80 images per category for training and 20 other as testing ones, resulting in $5,360$ training-images and $1,360$ testing-images.

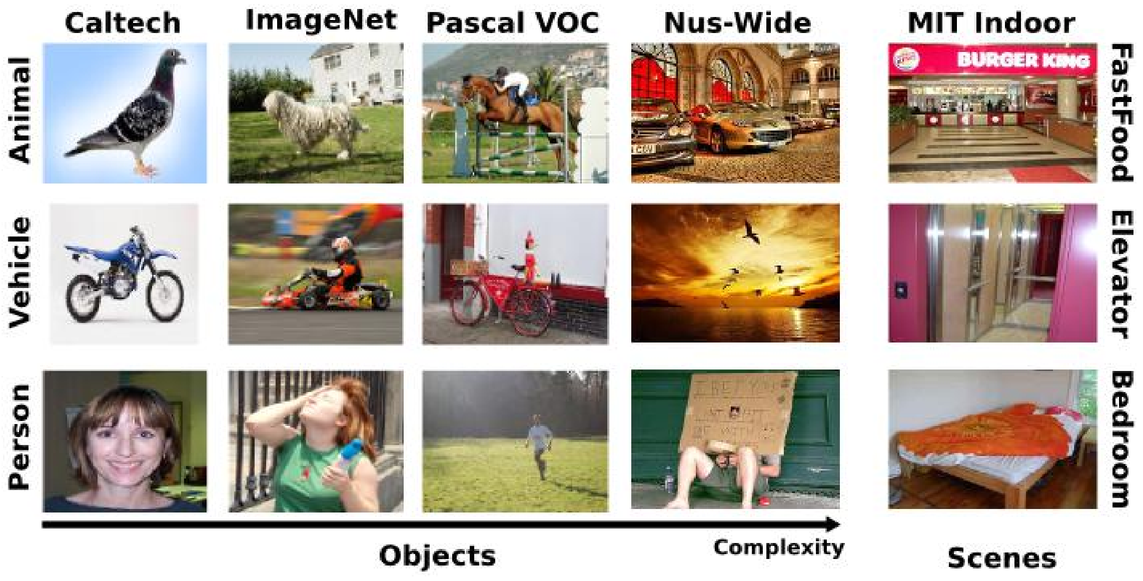

Figure A.2: Illustration of examples of images for the datasets that are commonly used in the literature. We splitted the images into two groups, the group of object datasets (on the left) and the scene dataset (right). Each row contains the images of the different datasets for a particular category (vertical text) that is given on the left for object datasets and on the right for the scene dataset. We illustrated a complexity arrow under the group of object datasets, that highlights the increase of complexity (*i.e.*, the complexity of the image can be defined by the intra-class variance of the categories of the dataset) between the datasets. For instance Caltech contains objects centered in the image in a white background, while Nus-Wide contains very localized objects in the images with potentially high occlusions of the objects. Images from ILSVRC and Places205 are not illustrated here because those of ILSVRC are the same than those of ImageNet and those of Places205 are the same than those of MIT Indoor except that they also belong to categories of outdoor places. Best view in color.

Stanford Cars (CARS) [131], **CUB-200 Birds (CUB)** [265] and **Flowers-102 (FLO)** [181] are fine-grained categories, containing respectively images in the domain of *cars*, *birds* and *flowers*. **Stanford Actions (stAC)** [283], is a dataset of images that are labeled according the actions performed by humans in the images.

In Figure A.2 we illustrated some examples of images for each of the datasets. We also resumed the description of all the datasets in Table A.1. In the following, we will see that ILSVRC and Places are commonly used as source-datasets for training CNNs in a classical transfer-learning scheme and all other datasets are generally used as target-datasets for evaluating learned representations, since they are not sufficiently large to efficiently train a neural-network.

Datasets	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
ILSVRC* [216]	objects	specific	483	1,2K	✗	569,000	48,299	Acc.
ILSVRC [216]	objects	specific	1K	1,2K	✗	1.2M	50,000	Acc.
Places205 [297]	scenes	specific	205	1.2K	✗	2.5M	41,000	Acc.
VOC-07 [71]	objects	generic	20	250	✓	5,011	4,952	mAP
VOC-12 [70]	objects	generic	20	577	✓	11,540	10,991	mAP
NW Ob. [38]	objects	generic	31	700	✓	21,709	14,546	mAP
CA-101 [73]	objects	generic	102	30	✗	3,060	3,022	Acc.
CA-256 [93]	objects	generic	257	60	✗	15,420	15,187	Acc.
MIT-67 [205]	scenes	specific	67	80	✗	5,360	1,340	Acc.
ACTIONs [283]	actions	specific	40	100	✗	4,000	5,532	Acc.
CUB [265]	birds	specific	200	30	✗	5,994	5,794	Acc.
CARs [131]	cars	specific	196	41	✗	8,144	8,041	Acc.
FLOWERs [181]	plants	specific	102	10	✗	1,020	6,149	Acc.
Flickr8k [103]	general	captions	$40,000$	1	✗	6,000	1K	R@K
Flickr30k [287]	general	captions	158K	1	✗	29,783	1K	R@K
MSCOCO [144]	general	captions	414K	1	✗	80,783	1K	R@K

Table A.1: Detailed descriptive of the different datasets used in this Thesis. On top of the table, we describe datasets used as *source-task* and at bottom, those used as *target-task*. For each dataset, we detail eight characteristics. Each column of the table corresponds to a certain characteristic: (1) domain of the images; (2) annotation-level of the categories (generic or specific); (3) amount of categories; (4) average amount of training-images per category; (5) whether the dataset contains multiple categories per image (✓) or no (✗); (6) amount of training examples; (7) amount of test examples; and (8) the standard evaluation metric (Accuracy and mean Average Precision, respectively denoted by **Acc.** and **mAP**).

B

Deep-Learning Background

In this section we review the main deep learning techniques relevant for learning and using neural networks. First, we describe (in Sec. B.1) the main component of a neural network, namely the “neuron”. Then, we detail (in Sec. B.2) the way they are assembled together in order to give an “Artificial Neural Network” (ANN). We follow this latter, by a in-depth description (in Sec. B.3) of the way we learn their parameters (weights and biases). We then describe (in Sec. B.4) the “Convolutional Neural Network” (CNN) and detail the novelties compared to artificial neural networks. Since the performances of CNNs critically depend on the design of their architectures, we describe (in Sec. B.5) the main architectures proposed in the literature.

B.1 Modeling One Neuron

The area of Artificial Neural Networks has originally been primarily inspired by the goal of modeling biological neural systems. In this section, we first briefly discuss the biological system that has mainly inspired this area. Then, we describe the formal modeling of one neuron and finally we describe the most commonly used activation functions in the formal neuron.

B.1.0.1 Biological Motivation

The basic computational unit of the brain is a neuron. We can found, approximately, 86 billion neurons in the human nervous system. These neurons are connected with an amount of approximately 10^{14} synapses. In Figure B.1, we present a commonly used cartoon-illustration of biological neurons (*i.e.*, cell body) with their main components, namely “dendrites”, “nucleus”, “axon”, “axon’s branches”, “axon terminals” and “synapses”. Each neuron receives input signals from its dendrites and produces output signals (through electrical signals) along its axon. Note that, each neuron contains only *one* axon, but this axon can eventually branch out and connect via synapses to *many* dendrites of one or many *other* neurons. Simply said, the output signal of *one* neuron can activate *multiple* other neurons.

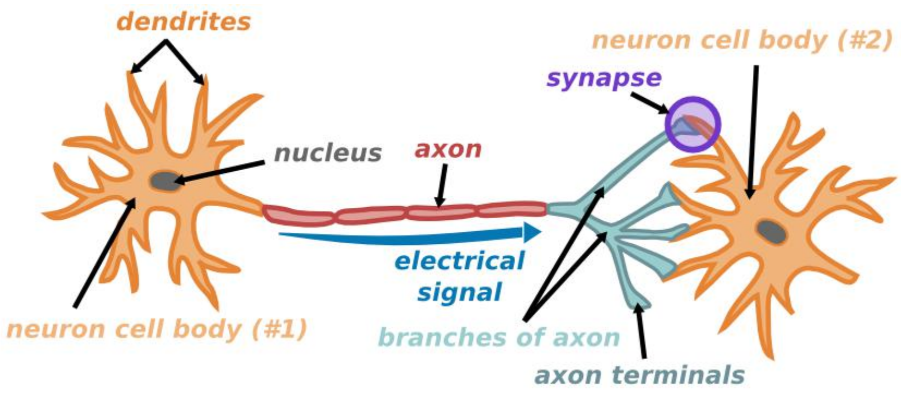

Figure B.1: Illustration of two biological neurons (cell body #1 and #2) and their main components, namely dendrites, nucleus, axon, axon's branches, axon terminals and synapses. Best viewed in color.

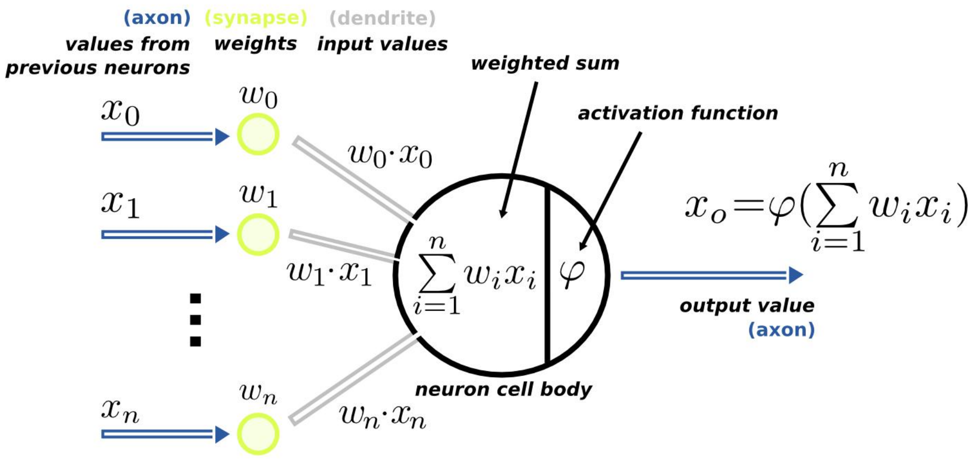

Figure B.2: Illustration of a common formal model of a neuron. We colored, in blue all the values that output from a neuron, in yellow all the weights that connect two neurons and in light gray, all the input-values of a neuron. Best viewed in color.

B.1.0.2 Formal Neuron

To simulate the functioning of a biological neuron, the "formal neuron" has been introduced by McCulloch and Pits [164] in 1943. A formal neuron is stimulated by its previous neurons, that we will call "input neurons", and regarding this stimulation, the neuron will output a value between 0 and 1. This latter will be called "output value" in the following.

Formally, given a set of n output values $\{x_0, x_1, \cdots, x_n\}$ from preceding neurons that stimulate the formal neuron denoted $x_i^{(l)}$, each of these values will be considered as input values for $x_i^{(l)}$ and will influence its output values. More precisely, let consider a set of n values $\{w_0, w_1, \cdots, w_n\}$ corresponding to the synapses respectively associated to each input neuron. These values are formally called "weights" and are pre-determined. This being said, in the computational model of the formal neuron $x_i^{(l)}$, the signals that travel along the axons (*i.e.*, input neurons x_i) interact multiplicatively (*i.e.* $w_i \cdot x_i$) with the dendrites (input values) of the formal neuron through the synaptic strength (the weights w_i) at that synapse (*i.e.* w_i). The idea is that the synaptic strengths (that are learnable) control the strength of influence of previous neurons to the formal neuron. The influence of a previous neuron to the formal neuron can be either positive (which will cause the excitation of the formal neuron) or negative (which will inhibit the excitation of the formal neuron). In the basic model, the dendrites carry the signal to the cell body, which explains why preceding neurons correspond to *input values* of the formal neuron. All input values are then get summed through the "weighted sum" $s_i^{(l)}(\cdot)$ that is formally expressed by: $s_i^{(l)}(x_i^{(l)}) = \sum_{i=1}^{n} w_i \cdot x_i$. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon, which corresponds to the output value denoted x_o. The firing rate of the neuron is obtained through an "activation function" denoted φ that takes as input the computed sum as a real value and outputs a value between 0 and 1. The output of that function directly corresponds to the output value (axon) x_o of the formal neuron which is formally expressed as:

$$x_o = \varphi(\sum_{i=1}^{n} w_i \cdot x_i). \tag{B.1}$$

Originally, the activation function used is the Heaviside function that simply consists in a threshold. An illustration of a common formal model of a neuron is given in Figure B.2.

It is important to note that, the current model of the biological neuron in the Neuroscience area is much more advanced than the one presented here. In fact, there exist many different types of neurons (each with different properties), the dendrites in biological neurons perform complex nonlinear computations, the synapses are not just a single weight but rather they are a complex non-linear dynamical systems, etc. Thus, due to all these and many other simplifications the previous description of the biological neuron is very brief and corresponds only to a coarse model.

B.1.1 Commonly used Activation Functions

In the formal neuron we have seen that the neuron sums the input values and applies on it an activation function (denoted φ). Every activation function (also called "non-linearity") takes a single number and performs a certain fixed mathematical operation on it. Historically, the "sigmoid" function (denoted σ) has been frequently used as the non-linearity in the formal neurons. This latter, is due to the fact that the sigmoid function has a nice interpretation as the firing rate of a neuron: from not firing at all (0 value) to fully-saturated firing at an assumed maximum frequency (1 value). Today, there are three main popular activation functions that we describe briefly in the following items:

- **Sigmoid**: The sigmoid function (illustrated on the left graph of Figure B.3) is mathematically formalized by:

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{B.2}$$

  It takes a real-valued input (the signal strength after the sum in the neuron) and restricts it to range between 0 and 1. In particular, large negative numbers become 0 and large positive numbers become 1. Importantly, when the neuron's activation saturates at either 0 or 1, the gradient at this regions is near to zero. Thus, the backpropagation algorithm (see Sec. B.3) fails at modifying the weights of the formal neuron and obviously, those of the preceding neurons. For these reasons, the sigmoid has recently fallen out of favor and is rarely ever used.

- **Hyperbolic Tangent (Tanh)**: The Tanh *squashes* (*i.e.*, restricts) a real-valued number to the range $[-1, 1]$ as illustrated on the middle graph of Figure B.3. Unlike the sigmoid function its output is zero-centered. Therefore, in practice the Tanh function is often preferred to the

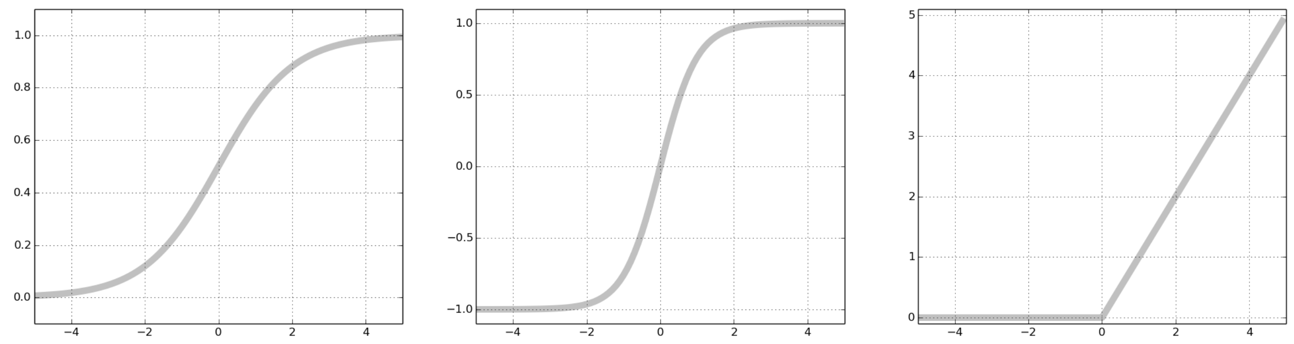

Figure B.3: Illustration of the three main activation functions, namely Sigmoid (left), Tanh (middle) and ReLU (right). The Sigmoid non-linearity squashes the input values (x) in range $[0, 1]$, the Tanh one squashed them in range $[-1,1]$ and the ReLU, in range $[0, +\infty[$.

sigmoid function. Formally, the Tanh function is expressed as:

$$\varphi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1, \tag{B.3}$$

with σ corresponding to the sigmoid function. Note that, the Tanh function is simply a scaled sigmoid function, thus, it has the same drawbacks as the sigmoid.

- **Rectified Linear Unit (ReLU)**: Mathematically, the ReLU function is expressed by:

$$\varphi(x) = \max(0, x). \tag{B.4}$$

The ReLU has become very popular in the last few years, due to its great ability to accelerate the convergence of the popular stochastic gradient descent optimization algorithm compared to the previously described Sigmoid and Tanh functions. Another advantage is its cheap computational cost compared to the expensive operations (*e.g.*, divisions, exponentials, etc.) in the Sigmoid/Tanh functions. The function is illustrated on the right graph of Figure B.3.

The three non-linearities are illustrated in Figure B.3. Two other functions have been proposed to fix some limitations of the ReLU (*e.g.*, if the learning rate of the optimization algorithm is set too high, it can lead to the "death" of some neurons, etc.), namely **Leaky ReLU** [155] and **Maxout** [90]. The former is one attempt to fix the "dying ReLU" problem, however it seems that the consistency of the benefit across tasks is currently unclear. The latter (Maxout) is a generalization form of the ReLU and its leaky version (Leaky ReLU). Note that, both ReLU and Leaky ReLU are a special case of Maxout, thus, this latter enjoys all the benefits of a ReLU unit (linear regime of operation, no saturation) and does not have its drawbacks (dying ReLU). However, compared to the ReLU neurons, it doubles the number of parameters for every single neuron, leading to a high total number of parameters. Finally, the ReLU is a good trade-off between accuracy and efficiency and is by far the most frequently used activation function in the deep-learning community.

B.2 Artificial Neural Networks

In this section, we describe "Artificial Neural Networks" (denoted ANN). Historically, many ANNs have been proposed after the formal neuron [164]. In particular, the standard linear Perceptron has been proposed by Frank Rosenblatt [215] in 1957. In 1969, Minsky and Papert [169] highlighted some limitations of the perceptron and proposed an extension, namely the Multi-Layer Perceptron (denoted MLP). In 1989, it has been demonstrated in [47] that, contrary to the standard linear perceptron, the MLP can distinguish data that are not linearly separable [47]. Hence, because of their efficiency and popularity, in this section, we will focus on MLPs only and we will exaggerate the assimilation of ANNs to MLP. More specifically, we will first detail the layer-wise organization of the neurons in the ANN, then we detail the feed-forward computation across the layers of an ANN.

B.2.1 Architecture of ANNs

First, let introduce the term "layer" as a collection of neurons. Hence, an ANN is a set of layers that are stacked one after the other. Every pair of layers of an ANN is connected through neuron's connections (*i.e.*, weights), thus the outputs of some neurons of one layer can become inputs of other neurons of another layer. However, neurons within a single layer are *not* connected at all. For regular ANN, the most common type of connection is the *full*-connection (*i.e.*, for a given layer, all the neurons of the preceding layer are connected to all the neurons of the actual layer) thus the layers of an ANN are called "fully-connected". The ANN can be seen as a directed acyclic graph with each neuron as a node and each weight as an edge between two neurons. Any node of a regular ANN is not connected to itself (no cycles). ANNs with cycles are commonly called "Recurrent Neural Networks" (RNNs) and are out of the scope of this thesis.

Formally, let consider an ANN of $\mathcal{L}$ layers $\{(0), (1), \cdots, (\mathcal{L})\}$ with each layer $\mathbf{x}^{(l)}$ being a set of $n^{(l)}$ neurons $\{x_0^{(l)}, x_1^{(l)}, \cdots, x_{n^{(l)}}^{(l)}\}$. Each pair of two consecutive layers $((k), (l))$ in the ANN is fully-connected and this connection is formally represented by a matrix of weights that contains $n^{(l)}$ lines and $n^{(k)}$ columns, and that we denote $\mathbf{W^{(1)}}$. Each line $\mathbf{W_i^{(1)}}$ of that matrix contains a vector $\{w_{i1}, w_{i2}, \cdots, w_{ik}\}$ that respectively corresponds to the weights $w_{i1}, w_{i2}, \cdots, w_{ik}$ of one neuron $x_i^{(l)}$ of the layer (l). Notice that, a "bias term" is also added to the set of input neurons of the neuron $x_i^{(l)}$ of layer (l). Indeed, the bias term is a sort of neuron $b^{(l)}$ that has the particularity to be not connected to the neurons of the preceding layer but only to those of the next layer. More specifically, its weights are connected to each neuron of the next layer, and we denote them b_i for each neuron $x_i^{(l+1)}$ of the next layer. The whole set of bias terms for the neurons of a layer (l) will be denoted $\mathbf{b}^{(l)}$ and is represented as a vector equal to $\{b_1, \cdots, b_{n^{(l)}}\}$. This being said, each element of the lines of the matrix $\mathbf{W_i^{(1)}}$ corresponds to the weights between neuron $x_i^{(l)}$ of a layer (l) and neurons $x_i^{(k)}$ of the preceding layer (k) and each element of the vector $b^{(l)}$ corresponds to the weights between neuron $x_i^{(l)}$ of a layer (l) and the bias term $b_i^{(l)}$ of the preceding layers.

As said in the previous section, the value of the weight between two neurons can be interpreted as the power of influence of the preceding neuron to the actual neuron. More generally, the main goal of an ANN is to map input vectors to output vectors. The input vectors, usually correspond to raw data vectors (images reshaped to vectors, one-hot encoding word vectors, signals, etc.) and the output vectors usually correspond to the representation of the class scores (*e.g.* in classification) which are one-hot coding numbers, or some kind of real-valued target (*e.g.* in regression). Thus, there exist three types of layers in ANNs, namely "input", "hidden" and "output" layers. While the input layer has the particularity to have no preceding layer (*i.e.*, no input neurons), the *output* one has the inverse particularity of having no next layer. Finally, the first layer is the input layer, the last is the output layer and all the others are named hidden layers. An illustration of a common ANN is given in Figure B.4.

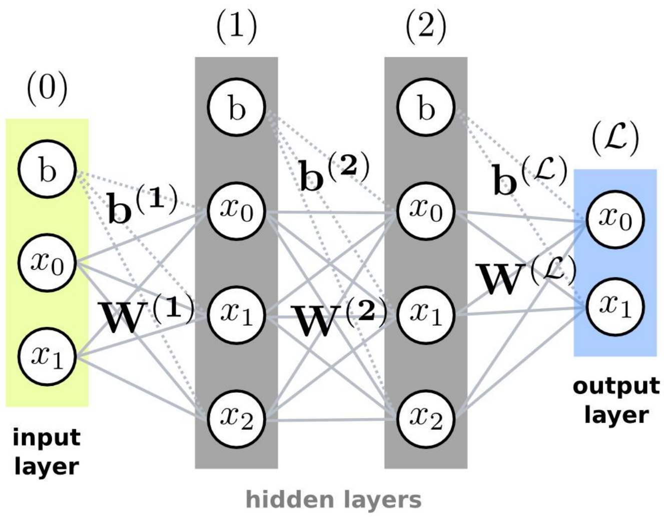

Figure B.4: Illustration of an ANN with four layers (one input layer (0) colored in yellow, two hidden layers (1) and (2) colored in dark gray and one output layer $(\mathcal{L})$ colored in blue). We colored in light gray all learnable parameters $\Theta = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}, \mathbf{W}^{(\mathcal{L})}, \mathbf{b}^{(\mathcal{L})}\}$ of the ANN and specified the bias weights in dashed lines.

The architecture of an ANN is generally defined by two (hyper)parameters, (i) the number of layers (denoted $\mathcal{L}$) which is frequently named *depth* of the ANN and (ii) the number of neurons per hidden-layer (denoted $n^{(l)}$) which is called *width* of the ANN. It has been shown in the literature [292] that, properly increasing the depth and width of an ANN increases its capacity to approximate functions and thus goes further towards its goal of mapping a set of input vectors to a set of output vectors.

B.2.2 Feed-Forward Computation in ANNs

The main goal of an ANN is the *mapping* of a set of input vectors to a set of output vectors. Hence, each input vector will be passed to the ANN through its hidden layers until it reaches the output layer. This latter is called “feed-forward computation” or “forward pass”.

Formally, a forward pass in an artificial neural network is a stack of function computations. Indeed, let consider an ANN with $\mathcal{L}$ layers (one input, $\mathcal{L}-2$ hidden and one output), its set of parameters $\Theta = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}, \cdots, \mathbf{W}^{(\mathcal{L})}, \mathbf{b}^{(\mathcal{L})}\}$ (where $\mathbf{W}^{(l)}$ is a matrix that contains the weights of layer (l) and $\mathbf{b}^{(l)}$ is a vector that contains the weights of the bias term of layer (l)) and an N-dimensional input vector $\mathbf{x}^{(0)} = \{x_0^{(0)}, \cdots, x_N^{(0)}\}$. The forward pass first consists to apply an affine transformation on the input vector ($s(\mathbf{x}^{(0)})$, corresponding to the weighted sum described in the previous section), followed by an element-wise activation function $\varphi(s(\mathbf{x}^{(0)}))$ that aims to obtain the first hidden layer $h^{(1)}(\mathbf{x}^{(0)})$. This latter is expressed by:

$$h^{(1)}(\mathbf{x}^{(0)}) = \varphi(\mathbf{W}^{(1)}\mathbf{x}^{(0)} + \mathbf{b}^{(1)}), \tag{B.5}$$

with $b^{(1)}$ the bias terms of the input layer. Then, the forward-pass still continues by applying the same kind of functions to the outputs of the preceding layers. Concretely, the output of each following hidden layer is expressed by:

$$h^{(l)}(h^{(l-1)}) = \varphi(\mathbf{W}^{(l)}h^{(l-1)} + \mathbf{b}^{(l)}), \forall l \in [1, \dots, \mathcal{L}-2], \tag{B.6}$$

where $\mathbf{W}^{(l)}$ parametrizes the affine transformation of the l^{th} hidden layer and $b^{(l)}$ is the set of bias terms. In the same vein, we compute the output layer $\mathbf{x}^{(\mathcal{L})}$ (that we will denote $\mathbf{y}^{(p)}$ in the following) by:

$$\mathbf{y}^{(p)} = h^{\mathcal{L}}(h^{(\mathcal{L}-1)}) = \varphi(\mathbf{W}^{(\mathcal{L})}h^{(\mathcal{L}-1)} + \mathbf{b}^{(\mathcal{L})}). \tag{B.7}$$

Once this previous equation is computed, the ANN has reached the output layer $\mathbf{x}^{(\mathcal{L})}$, which is the last step of the forward-pass. Note that, from now, the output vector obtained after the forward-pass is called "predicted vector" and is denoted $\mathbf{y}^{(p)}$.

B.3 Training Neural Networks

In the previous section, we defined the architecture of a regular ANN and the way it maps from an input vector to an output vector. However, we have not discussed the way the weights of each layer are obtained. Hence, in this section, we describe the way we obtain the weights (initialization and learning of their values according to some training-examples), which, generally corresponds to the *training* of neural networks. More precisely, we describe their training in four parts: (i) initialization of the weights, (ii) estimation of the difference between the *actual* and the *desired* mapping-ability, (iii) training (*i.e.*, updating) the weights in order to minimize the difference of the previous point and finally (iv) we will present some popular techniques used to improve the training of ANNs.

B.3.1 Weights and Biases Initialization

Learning an ANN consists in determining its parameters (*i.e.*, weights and biases) from some annotated training-data. Above mapping the raw inputs to the desired outputs, the challenge of generalization consists in being able to map an unviewed data to the correct output. For this, it exists many techniques, grouped under the term "regularization". This latter will be described in more details below. But before learning the parameters, it is important to fix the size of the weight-matrices (which is done when setting the network architecture) and then initialize their values.

To initialize the weights of an ANN, a reasonable-sounding idea might be to set all the initial weights to zero, however, this turns out to be a mistake. The reason of this latter needs the understanding of the optimization algorithm (backpropagation) used to train the network (*i.e.*, *update* the weights). Nevertheless, a rough explanation is that, if all the weights are initialized to zero, thus every neuron in the network will compute the same output (since the weighted sum is always equal to zero and thus the activation function will always outputs the same value), then all neurons will also compute the *same* gradients (that are needed by the optimization algorithm to update the weights) during the optimization, which will undergo the exact same weight-updates. More generally, all the neurons will be the same, if their weights are initialized to be the *same* (not only initialized to be zero), which clearly induces to the loss of utility of the ensemble of neurons (which is the most power part of ANNs). Hence, it is very important to initialize all neurons with *different* values, which is called "breaking symmetries".

Therefore, we still want the weights to be very close to zero (in order to prevent the network to be highly biased from the beginning), but as argued above, not all identically the same. A common solution is to initialize them with small numbers such that all neurons are *random* and *unique* in the beginning, in order that they compute *different* updates and integrate themselves as *distinct* parts of the full network. In practice, the weights are initialized with random floats sampled small variance (*e.g.*, $0,001$).

In contrast to the initialization of the weights, for the initialization of the biases it is common to set them with zero-values. Indeed, it is not problematic to initialize them with zero for the following reason: since all the weights are already initialized randomly and especially differently, setting all the biases to zero will never (at least, with a near-zero probability) result in a set of neurons with the exact same set of outputs (and thus, gradients), hence it will never cause the same update of the input weights and biases. When using ReLU non-linearities, it is frequent to use small constant value such as 0.01 for all biases because this ensures that all neurons fire in the beginning and therefore obtain and propagate some gradient. However, it is not clear if this provides a consistent improvement and it is more common to simply use 0 as bias initialization.

B.3.2 Loss Functions

Given some training data (*i.e.*, raw data and ground-truth labels), the ANN will be *learned to fit* the data – *i.e.*, given input vectors (raw data), the ANN will learn to map to the output vectors (ground-truth labels) – and the "loss function" is a measure of the correctness of the mapping.

The first step (*i.e.*, first forward-pass in the ANN given an input vector) of this mapping will result to a random output vector (because of the random weights-initialization), which will be very far from the wanted vector (ground-truth labels). Simply said, the prediction at the first forward-pass will be random and thus false. Hence, it is important to measure the compatibility between the predictions (*e.g.* the output vectors after a forward-pass) of the ANNs and the ground-truth label. More rigorously, the set of output vectors of the ANN is called "predicted distribution" and the set of ground-truth labels is called "desired distribution". Hence, the goal is to *minimize* the *difference* (or *maximize* the *compatibility*) between the predicted and the desired distributions. This notion of difference is interchangeably called *loss-function*, *objective-function* and *cost-function*. The output of this loss-function will indicate to the optimization algorithm the directions to update the weights of the ANN. An illustration of the loss-function principle is given in Figure B.5.

Formally, let consider a set of training-data $\mathcal{D}_C^N$ that contains N raw data $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ labeled among C-dimensional vectors corresponding to ground-truth labels $\mathbf{Y} = \{\mathbf{y}_1^{(d)}, \cdots, \mathbf{y}_C^{(d)}\}$ that we will call "desired vectors", in the following. Generally, $\mathbf{y}_i^{(d)}$ is a binary vector, that is to say, full of 0 besides 1-values in the corresponding class dimensions. Let us also abbreviate the ANN as a function f that takes as input a vector $\mathbf{x}_i$, a tensor of weights $\mathbf{\Theta}$ and outputs a predicted vector $\mathbf{y}_i^{(p)}$ through $\mathbf{y}_i^{(p)} = f(\mathbf{x}_i; \mathbf{\Theta})$. It exists three main cost-functions used to train ANNs:

- **Mean Square Error (MSE)**: It is a multi-class mono-label loss-function that has the following mathematical form:
$$L(\mathbf{y}_i^{(p)}, \mathbf{y}_i^{(d)}) = \frac{1}{C}\sum_{j=1}^{C} |y_j^{(p)} - y_j^{(d)}|. \tag{B.8}$$
The multi-class means that $C > 1$ and the mono-label means that $\mathbf{y}_i^{(d)}$ is a vector full of 0 besides *one* 1-value in the corresponding class dimension.

- **Cross-Entropy**: The cross-entropy is similar to the MSE, in the sense that it is a multi-class mono-label loss-function. However it differs from it since it takes probability distributions as

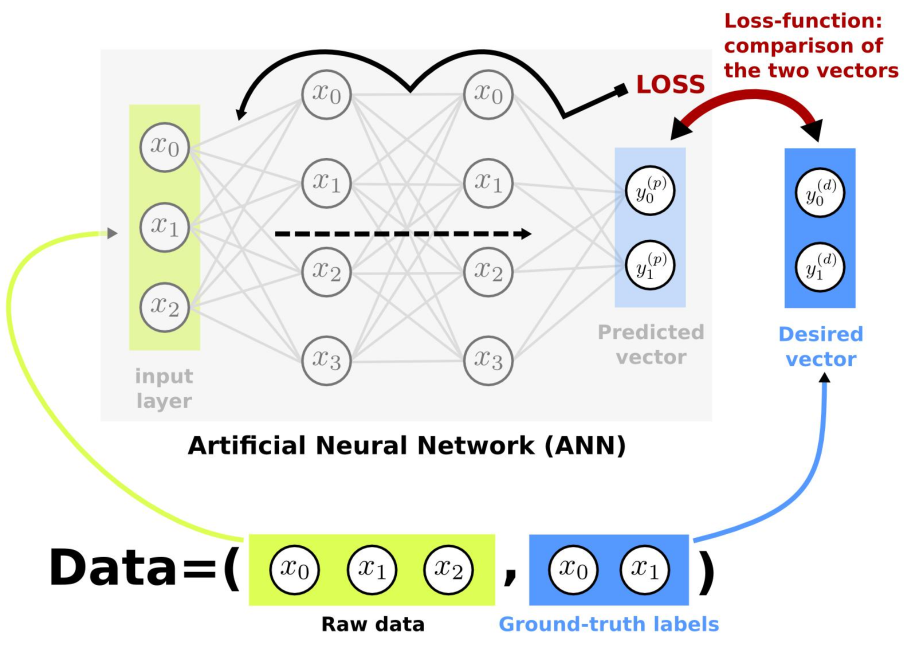

Figure B.5: Illustration of the loss-function principle. Given some training data (raw data and ground-truth labels), the ANN first computes a forward-pass (illustrated by the dashed arrow) with the actual weights (initialized randomly) until it reaches the predicted-vector, then the predicted-vector is compared to the desired-vector (ground-truth labels) through the loss-function, which outputs a loss value (called "LOSS" in this graph). The output of the loss-function (which can be interpreted as the difference between the predicted-vector and the desired one) is then back-propagated in the network (illustrated by the black arrow). Once all the error values are computed for all the neurons, they are used as input to the optimization algorithm that will update the weights. Once the weights updated, the ANN takes again the raw data and computes a forward-pass and so on. This cyclic process is done iteratively until convergence (*i.e.*, the output of the loss-function is near-zero).

input. Formally, it is expressed by:

$$L(\mathbf{y}_i^{(p)}, \mathbf{y}_i^{(d)}) = -\sum_{j=1}^{C} y_j^{(d)} \log(\varphi(y_j^{(p)})), \tag{B.9}$$

where φ is the "softmax" activation-function that takes a vector $\mathbf{x}$ of arbitrary values and passes it to a probability distribution through $\sigma(\mathbf{x}) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$. Indeed, this is necessary because the cross-entropy can only be applied on probability distributions and $y_j^{(p)}$ has arbitrary values and is not a probability distribution (the sum of its values are not equal to 1). Regarding $\mathbf{y}_i^{(d)}$, we do not apply the softmax on it because it already corresponds to a probability distribution since it is full of zeros and has *one* 1-value thus, the sum of its values is equal to 1. Also important, the softmax activation-function is not applied on *one* neuron (like activation-functions described in Sec. B.1.1), but it is applied on the whole vector (*i.e.*, the whole set of neurons).

- **Multi-Label Loss**: The multi-label loss-function is an adaptation of the cross-entropy for the problem of *multi*-label multi-class classification. Indeed, the multi-label classification problem has the property to have a desired output-vector $\mathbf{y}_i^{(d)}$ which is full of 0 besides *many* 1-values in the corresponding class dimensions, thus the sum of its values is not equal to 1, which results in an arbitrary vector (*not* a probability distribution). Formally, the multi-label loss is expressed by:

$$L(\mathbf{y}_i^{(p)}, \mathbf{y}_i^{(d)}) = -\sum_{j=1}^{C} y_j^{(d)} \log(\sigma(y_j^{(p)})) + (1 - y_j^{(d)}) \log(1 - \sigma(y_j^{(p)})), \qquad \text{(B.10)}$$

where σ, is the sigmoid activation-function. The sigmoid function is useful here because it takes any real input value (dimension of $y_j^{(p)}$) and squashes it to a value between zero and one and hence is interpretable as a probability. While the softmax takes all the dimensions of the vector and outputs a *joint* probability, the sigmoid is applied on each dimension of the vector and outputs *independent* probabilities. Note that, the Multi-Label loss is frequently called "sigmoid cross-entropy".

Notice that, the cost-function is not computed on *one* data-example, but rather on a *batch* N^b (*i.e.*, small subset from the whole set) of data. Hence, the cost-function that is minimized by the optimization algorithm is $J(\theta) = \frac{1}{N^b} \sum_{i=1}^{N^b} L(\mathbf{y}_i^{(p)}, \mathbf{y}_i^{(d)})$. In fact, in the following, the cost-function will be denoted $J(\theta)$ with θ corresponding to the whole set of weight-matrices of the ANN.

B.3.3 Learning the Weights and Biases

In the previous sections, we have described the initialization of the weights/biases and the principle of the loss-functions that consists to compute (after a forward-pass) the difference between the predicted and the desired-output. In this section, we will describe the way we learn the weights and biases of a neural-network with respect to the given random weights (at the first iteration) and the chosen loss-function. Very generally, the common method to train ANNs is to use an *optimization* algorithm in conjunction with a *backpropagation* algorithm. In fact, to understand why the problem of learning an ANN is equivalent to minimize the cost-function, we need to recall that the goal on an ANN is to find the weights/biases so that the output from the network approximates the desired output-vector for all training input-vectors. Since, the loss-function quantifies how well we are achieving this goal, then, updating the weights for minimizing the cost-function (*i.e.*, makes the loss as small as possible) clearly reaches the goal of fitting the training-examples. The backpropagation is also called "backward propagation of errors" and the main optimization algorithm used to learn ANNs is the *gradient descent* algorithm. Each of these two algorithms corresponds to a phase – *i.e.*, *backward propagation of errors* for the first phase and *weights and biases update* for the second phase – and the main principle to learn the weights and biases of an ANN is to repeat these two-phases cycle until some stopping criterion.

More precisely, the first cycle (backward propagation of errors) consists of three main steps. First, when an input vector is presented to the network, a forward-pass is computed until it reaches the output layer which gives the predicted vector (as described in Sec. B.2.2). At the first iteration, the forward-pass is computed with random weights. Second, the predicted-vector is then compared to the desired output, through the loss-function and an error value is calculated for each of the neurons in the output layer (as described in Sec. B.3.2). The error values are then propagated backwards (starting from the output layer, until the first layer) which results in an association of each neuron of the ANN to an error value which roughly represents its contribution to the original output. Third, these error-values are used to calculate the *gradient* of the loss-function with respect to the actual weights in the network.

In the second cycle phase, this gradient is fed to the optimization algorithm, which in turn uses it to update the weights/biases, in an attempt to minimize the loss-function. Very importantly, behind the process of gradient descent, as the network is trained, the neurons in the intermediate layers organize themselves in such a way that the different neurons learn to recognize different characteristics of the total input space. After training, when an arbitrary input pattern is present which contains noise or is incomplete, neurons in the hidden layer of the network will respond with an active output if the new input contains a pattern that is like a feature that the individual neurons have learned to recognize during their training.

Now that we have presented the main principle of the algorithms that aim to learn the weights and biases, we will present exactly how they work, that is to say (i) how we minimize the error of the loss-function, given the gradients and (ii) how we compute the gradients for each neuron of the ANN (even in the hidden layers), given the output loss. Hence, for the gradient descent algorithm, the goal is to find the weights $\mathbf{W}^{(l)}$ and biases $\mathbf{b}^{(l)}$ (with $l \in [1, \mathcal{L}]$) of the $\mathcal{L}$-layers ANN that minimize the loss-function L. Let first define the gradient of L to be the vector of partial derivatives $\left(\frac{\partial L}{\partial w_{ij}^{(l)}}, \frac{\partial L}{\partial b^{(l)}{}_i}\right)^T$. We denote the latter gradient vector ∇L. Thus, given each individual weights $w_{ij}^{(l)}(t)$ (weight between neuron j of the preceding layer $l-1$ and neuron i of the actual layer l) and the bias term $b_i^{(l)}$ (bias term of neuron i of the actual layer l) at iteration t, the way we update (for iteration $t+1$) each value of the weight-vector $\mathbf{W}^{(l)}$ and the value of the bias-term is respectively obtained through the following equations:

$$w_{ij}^{(l)}(t+1) = w_{ij}^{(l)}(t) - \eta \frac{\partial L}{\partial w_{ij}^{(l)}(t)}, \tag{B.11}$$

$$b_i^{(l)}(t+1) = b_i^{(l)}(t) - \eta \frac{\partial L}{\partial b_i^{(l)}(t)}, \tag{B.12}$$

with $\eta > 0$ is a small, positive parameter, known as the *learning-rate*. These equations are then used again (*i.e.*, for another iteration) to make another modification of the weights and biases. If we keep doing this, over and over, we will keep decreasing the loss-function L until the algorithm converges (*i.e*, it reaches a global or local minimum for the loss-function). Generally, the algorithm is stopped when the loss is near-zero or when it does not change after a large number of iterations.

In the previous paragraph, we described the way we update each weight and bias at one iteration during the learning of the ANN, but we have supposed that the gradient vectors are given. Indeed, computing the gradient vectors (*i.e.*, the partial derivatives $\frac{\partial L}{\partial w_{ij}^{(l)}}$ and $\frac{\partial L}{\partial b_i^{(l)}}$) for each neuron j of layer l is crucial and it is exactly the purpose of the backpropagation algorithm. More precisely, the back-propagation is based on four fundamental equations which give us a way of computing the gradient of the loss-function. Let first introduce an intermediate quantity, $\delta_i^{(l)}$, which we call the “error” in the i^{th} neuron of layer l. The backpropagation algorithm gives us a procedure to compute the error $\delta_i^{(l)}$ for every neuron in every layers and then relating those errors to the quantities of real interest, namely the partial derivatives. Indeed the error of the i^{th} neuron in layer l is mathematically defined by:

$$\delta_i^{(l)} = \frac{\partial L}{\partial s_i^{(l)}} \tag{B.13}$$

where $s_j^{(l)}$ is the weighted sum input ($s_i^{(l)} = \sum_{j=1}^{n} x_j^{(l-1)} w_{ij}^{(l)}$) of the neuron i in the layer l. Now that we have defined the general mathematical form of the error, we will define it both for the neurons of the output layer and for the neurons of the hidden-layers. Indeed, we start by the error at the output layer since it corresponds to the initial error when we do *backward* propagation of the error. Hence, the initial error – *i.e.*, the error $\delta_i^{(\mathcal{L})}$ of the neurons of the output layer – is given by:

$$\delta_i^{(\mathcal{L})} = \frac{\partial L}{\partial x_i^{(\mathcal{L})}} \varphi'(s_i^{(l)}) \tag{B.14}$$

with $x_i^{(\mathcal{L})}$ being the output of the ith neuron in the output layer $\mathcal{L}$, $s_i^{(l)}$ the weighted sum of the ith neuron in the output layer and φ' the derivative activation function. The first term on the right of this equation $\frac{\partial L}{\partial x_i^{(\mathcal{L})}}$ measures how fast the loss is changing as a function of the output $x_i^{(\mathcal{L})}$ of the ith neuron. For instance, if we vary the output $x_i^{(\mathcal{L})}$ and the loss L varies a lot, then the error $\delta_i^{(\mathcal{L})}$ will be large. In contrast, if the loss does not vary at all (or varies very weakly) then the error will be very small. Regarding the second term $\sigma'(s_i^{(l)})$, it measures how fast the activation function φ is changing as a function of the weighted sum $s_i^{(l)}$. Note that, both terms on the right will depend on the form of the loss function L and the last term will also depend on the activation function φ.

The first step that consists to compute the initial error (error of the output layer) is then *backpropagated* (*i.e.*, propagation towards the back) in the neurons of the preceding layers. More precisely, provided the errors $\delta_i^{(l+1)}$ of all the neurons of layer $l+1$, the error $\delta_j^{(l)}$ of neuron j at layer l is expressed by:

$$\delta_i^{(l)} = \sum_{j=1}^{n^{(l+1)}} w_{ji}^{(l+1)} \delta_j^{(l+1)} \varphi'(s_i^{(l)}) \tag{B.15}$$

with $n^{(l+1)}$ being the number of neurons we have in layer $l+1$ and $w_{ji}^{(l+1)}$ the weights that goes from neuron i of the actual layer l to all the neurons j of layer $l+1$. The term $\sum_{j=1}^{n^{(l+1)}} w_{ji}^{(l+1)} \delta_j^{(l+1)}$ can be intuitively, seen as the action of moving the error *backward* through the network, which results in some kind of measure of the error at the output of the layer l. Multiplying this latter with the term $\varphi'(s_i^{(l)})$ can be seen as the move of the error backward through the activation function in layer l, which gives the error of the neurons of layer l.

By using Equations (B.14) and (B.15), we can compute the error $\delta_i^{(l)}$ of any neuron at any layer of the network. More precisely, we start by computing the error of the neurons of the output layer (through the use of Equation (B.14)), then we use these computed errors $\delta^{(\mathcal{L})}$ of layer $\mathcal{L}$ to compute (through Equation (B.15)) the errors of the previous layer $\mathcal{L}-1$, and so on, until we reach the weights of the first hidden-layer, which means that we have backpropagated the errors through all the network. Now that we have detailed the procedure to compute the errors for each neuron of each layer it only remains to relate them to the partial derivatives. More precisely, the partial derivative $\frac{\partial L}{\partial w_{ij}^{(l)}}$ of the loss with respect to any weight $w_{ij}^{(l)}$ (weight between neuron jth of layer $l-1$ and ith neuron of layer l) in the network is computed through:

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = x_i^{(l-1)} \delta_i^{(l)}, \tag{B.16}$$

with $\delta_i^{(l)}$ computed through Equation (B.14) if $l = \mathcal{L}$ or through Equation (B.15) if $l \in [2, \mathcal{L}-1]$, and $x_i^{(l-1)}$ and $x_j^{(l-1)}$ the jth neuron of layer $l-1$. In the same vein, the partial derivative $\frac{\partial L}{\partial b^{(l)}}$ of the loss with respect to any bias $b_i^{(l)}$ of a neuron i at layer l in the network is computed through:

$$\frac{\partial L}{\partial b_i^{(l)}} = \delta_i^{(l)}. \tag{B.17}$$

Note that Equation (B.16) can be rewritten more simply as

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = x_{in}\delta_{out}, \tag{B.18}$$

with x_{in} the output of the input neuron of the weight w and δ_{out}, the error of the output neuron. For the bias error, the right terms of the simple equation are the same but without the term x_{in} since the bias has no input neuron.

To summarize, the backpropagation algorithm consists in four main steps: (i) compute a forward-pass for an input vector to obtain the terms $\mathbf{s}^{(l)}$ and $\mathbf{x}^{(l)} = \varphi(\mathbf{s}^{(l)})$ for $l \in [1, \mathcal{L}]$, (ii) compute the error of the output layer $\delta^{\mathcal{L}}$ with Equation (B.14), (iii) backpropagate the error through each layer of the ANN in order to have the error δ^{l} at each layer $l \in [L-1, 2]$ via Equation (B.15) and finally (iv) compute the partial derivatives, through Equation (B.16) and (B.17). Note that, the four fundamental equations ((B.14), (B.15), (B.16) and (B.17)) to compute the partial derivatives for any neuron of any layer are only given in this section, but we do not provide the proofs of each of them.

B.3.4 Regularization Techniques

Above mapping the raw inputs to the desired outputs, the challenge of neural networks consists of being able to map an *unviewed* data to the correct output. Simply said, the main challenge is to increase *generalization* performance (*i.e.*, difference between *training* error and *test* error) of neural networks. Indeed neural network models exhibit relatively high generalization performance. However, sometimes the accuracy of neural networks with enough capacity converges towards perfection on the train-set but its performances degrade on the test-set. This phenomenon is called "overfitting". For this, it exists many techniques, grouped under the term "regularization". Hence in the following items, we quickly describe some of them:

- **L2/L1 Regularization**. The first main approach to overcome overfitting is the classical L2 regularization, which adds a term (*i.e.*, $\sum_i \|\theta_i\|^2$, with θ a vector containing all the network parameters) to the cost function to penalize the parameters in each dimension, preventing the network from *exactly* modeling the training data and therefore help generalize to new examples. L2 regularization is also known as *weight decay* as it forces the weights to decay towards zero (but not exactly zero). L1 regularization is similar to L2 regularization, but penalizes the absolute value of the weights (*i.e.*, $\sum_i \|\theta_i\|$). Unlike L2 regularization, the weights may be reduced to zero in L1 regularization.

- **Data augmentation**. It is a method of boosting the size of the training set so that the model cannot memorize all of it. This can take several forms depending on the dataset. For instance, if the objects are supposed to be invariant to rotation such as galaxies or planktons, it is well suited to apply different kind of rotations to the original images.

- **Early stopping**. It consists in stopping the training before the model begins to overfit the training-set. In practice, deep learning practitioners use it very frequently when training neural-networks.

- **Max norm constraints**. Another form of regularization is to enforce an absolute upper bound on the magnitude of the weight vector $\vec{\theta}$ for every neuron and use projected gradient descent to enforce the constraint. In practice, this corresponds to performing the parameter update as normal, and then enforcing the constraint by clamping the weight vector of every neuron to satisfy $\vec{\theta}^2 < c$. Typical values of c are on orders of 3 or 4. One of its appealing properties is that network cannot "explode" even when the learning rates are set too high because the updates are always bounded.

- **Dropout**. Finally, a recent success has been shown with a regularization technique called Dropout [232]. The idea is to randomly set a certain percentage of the activations in each layer to 0. During the training, neurons must learn better representations without co-adapting to each other being active. During the testing, all the neurons are used to compute the prediction and Dropout acts like a form of model averaging over all possible instantiations of the model.

While the weight decay and Dropout are the most popular and efficient regularization techniques, other interesting ones [83, 26] were proposed.

B.4 Convolutional Neural Networks (CNNs)

ANNs described in the previous section can be used in a wide range of applications as long as the targeted problem can be formalized as a supervised learning problem. In our case, we only consider problems that involve images as input, *i.e.*, image classification problems. Using ANNs in this case, needs to convert the images into vectors and it can be achieved very easily by reshaping them such that all the pixel's lines (or columns) of the image are concatenated in a vector. Let's take an example of an RGB image of size $256 \times 256 \times 3$ and reshape it to a vector, it will result in a vector of size $256 * 256 * 3 = 196,608$ thus, the ANN would take $196,608$-dimensional input vectors. Therefore, if we consider one neuron in the next layer (first hidden-layer), it will be connected to all input neurons (because the neurons are fully-connected), which will result to $196,608$ weights to learn for that neuron. Moreover, there is much more than one neuron per layer and also much more than one layer in an ANN, thus the number of weights will add up quickly. In summary, this full connectivity between the neurons is costly since it produces a huge number of parameters to learn, which has been shown to be very hard or impossible in practice (because the learning would quickly lead to overfitting). To fix the drawbacks of ANNs in the case of real-world images, "Convolutional Neural Networks" (denoted **CNNs**) have been introduced by LeCun *et al.* [140].

Similarly to standard ANNs, Convolutional Neural Networks are made up of neurons that have learnable weights and biases. Each neuron (the term "filter" is more appropriate in the case of CNNs) receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still maps inputs (raw images) to output-vectors and they still have a loss function on the last layer to compare the predicted outputs and the desired ones. Finally, the learning process of the weights of a CNN follows the same strategy as regular ANNs. However, CNNs make the explicit assumption that the inputs are not vectors but tensors (images), which allows us to encode certain properties into the architecture. These properties make a vast reduction of the amount of parameters in the network.

More generally, the main difference between ANNs and CNNs is in the connection between the neurons. Indeed, in ANNs, a neuron operates on the input neurons with a unique weight for each of them, and the operation between the two neurons is a dot product, while in CNNs, a neuron operates on a *group of input neurons* (called "receptive-field" in the following), through the *same group of weights* (called "filter" in the following) and the operation between the filter and the receptive-field is a *convolution*. Moreover, while in ANNs, the whole set of output neurons is organized as a vector, in CNNs, it is organized as a matrix, which is called "feature map" in the following. More generally, there are three main novelties that we will describe in this section: (i) an element called "spatial convolution" (in Sec. B.4.1), (ii) an other element called "spatial pooling" (in Sec. B.4.2), and finally (iii) the whole architecture of the CNN from which we will provide a complete overview (in Sec. B.4.3).

B.4.1 Spatial Convolution

Let consider an input image I of size $W_I \times H_I \times D_I$, with W_I, H_I and D_I respectively corresponding to the width, height and depth of the image. Let also consider a filter F of size $W_F \times H_F \times D_F$, with W_F, H_F and D_F respectively corresponding to the width, height and depth of the filter. In the case of filters and images of depth $D_I = D_F = 1$, the convolution operation is called **2D-convolution** and it consists to convolve the filter F with a receptive field at pixel (i, j) (we will call this "spatial location (i, j)", in the following) of image I which outputs a real value that is expressed by:

$$(I * F)(i,j) = \sum_{m=0}^{H_F-1} \sum_{n=0}^{W_F-1} I(i+m, j+n)F(m,n), \tag{B.19}$$

where $*$ is the convolution operator, $(i, j) \in [0, H_I - 1] \times [0, W_I - 1]$ is the pixel of image I at line i and column j. Note that, the spatial location (i, j) is a patch of the image I which is of size $i + H_F \times j + W_F$, thus it has exactly the same width and height of the filter F. By applying the convolution of filter F at all the spatial locations (i, j) (*i.e.*, all the pixels) of the image I, the result is a *feature map* M which is expressed by:

$$M(i,j) = (I * F)(i,j). \tag{B.20}$$

An illustration of a convolution of a filter applied at all spatial locations of an image with $D_I = D_F = 1$, is given in Figure B.6.

It is important to note that, the convolution of a filter is applied at a particular location (i, j) of an image and on a set of its pixels, namely spatial location (*i.e.*, all its pixels (m, n) with $m \in [i, i + H_F]$ and $n \in [i, i + W_H]$). Thus, when $i + H_F > H_I$ or $j + W_F > W_I$ the convolution can not be applied, since the size of the filter exceeds the size of the spatial location. Hence, a convolution can only be applied on pixels (i, j) with $i \in [0, H_I - H_F - 1]$ and $j \in [0, W_I - W_F - 1]$, which results in a feature map of size $H_I - H_F - 1 \times W_I - W_F - 1$ *smaller* than the size of the image. An illustration of the pixels that can not be processed with some filters of some sizes is given in Figure B.7. To fix that and output feature maps of the same size than the image, the procedure of "padding" has been introduced. It consists to increase the borders of the initial image with pixels at arbitrary values. Different kinds of padding exist in image processing (*e.g.*, zero-padding, mirror-padding, etc.) but, except aiming to output feature maps of the same size than inputs, the padding has not been shown to be very important in practice. Hence, a common procedure is to use *zero-padding* (*i.e.*, add pixels around the image with a zero value). Another important notion is the *stride*. It roughly on a parameter that represents a kind of offset on the spatial locations at which the convolution is applied. We give more details about this parameter at the end of the next section.

In the case of an image (or tensor) and filter with depths of $D_I > 1$ and $D_F > 1$, the convolution operation is called **3D-convolution**. The principle is exactly the same as with a 2D-convolution and

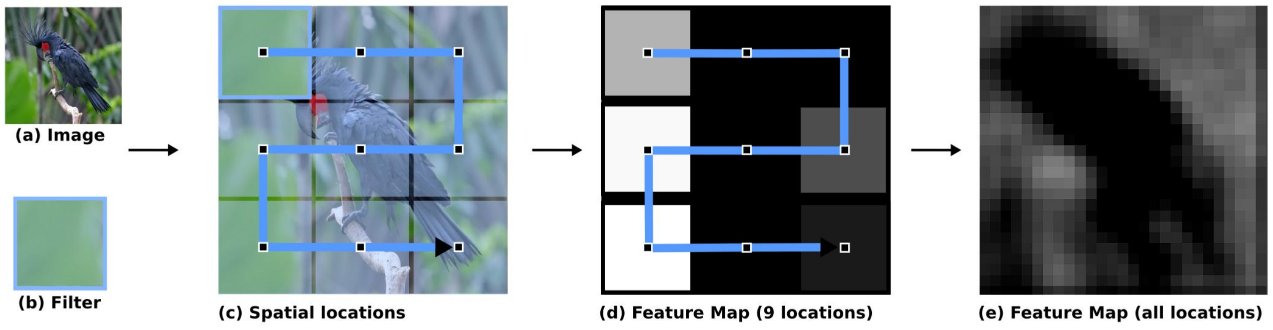
(a) Image
(b) Filter
(c) Spatial locations
(d) Feature Map (9 locations)
(e) Feature Map (all locations)

Figure B.6: Illustration of the convolution of a filter (b) – which is a green blob filter that may corresponds to a part of a tree or grass – applied at all spatial locations of an image (a) – that contains a bird with trees in the background. In (c), we observe the nine different spatial locations in which the filter is convolved with. The output of this spatial convolution is given in (d) and is called *feature map*. We see in this feature map that all the locations are colored in black, except those that have a pattern correlated with the pattern of the filter (*i.e.*, the more the correlation of the filter with the pattern at that spatial location, the more the output is white). In (e), we displayed the resulting feature maps if the filter (b) is convolved at *all* spatial locations rather than at the nine locations of (c). Best view in color.

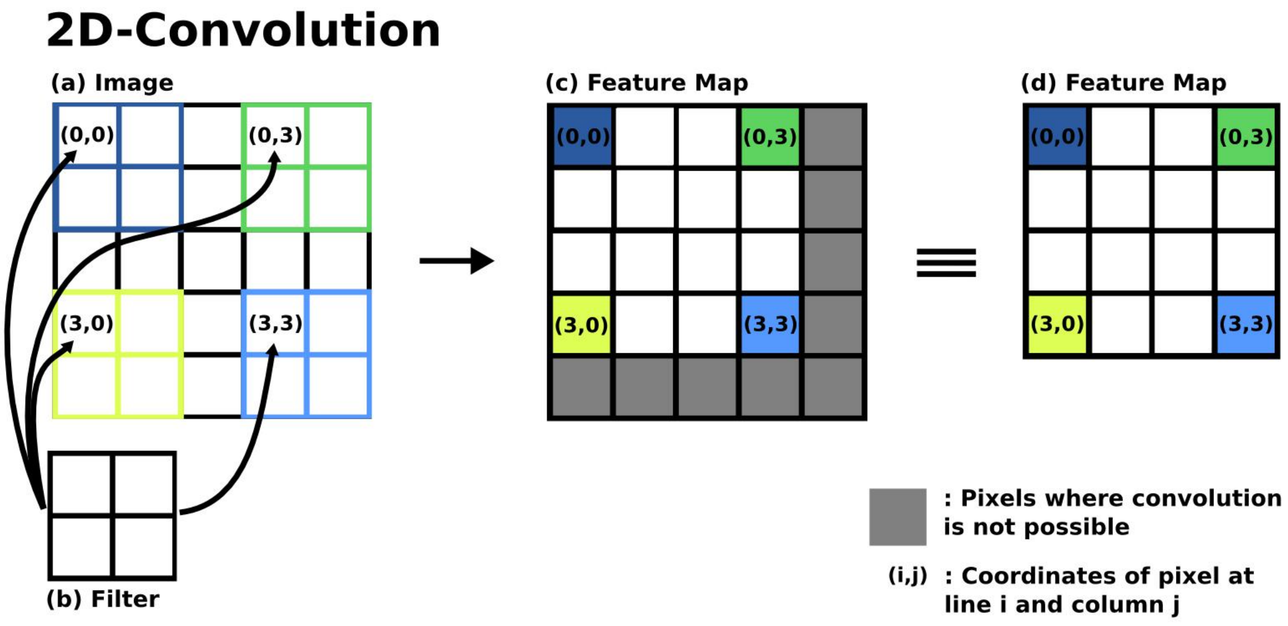

Figure B.7: Illustration of a 2D-convolution with a filter (b) of size 2×2 that convolves in an overlapping manner (*i.e.*, with stride [1,1], the process of stride will be described in the following) with an image (a) of size 5×5 at each spatial location of the image. The result of this convolution at all spatial location (feature map) is given in (c). In (c), we colored in gray the pixels that can not be convolved with this filter (of that size) since it will exceed the size of the image. In (d), we display the feature map without displaying the pixels that can not be processed with this filter. Hence, this feature map is effectively smaller than the input image. Best view in color.

it only differs by the fact that the depth of the elements to convolve should be considered. Hence, the 3D-convolution of a 3D-filter with a spatial location (i, j) of a 3D-tensor (*e.g.*, RGB image, multi-

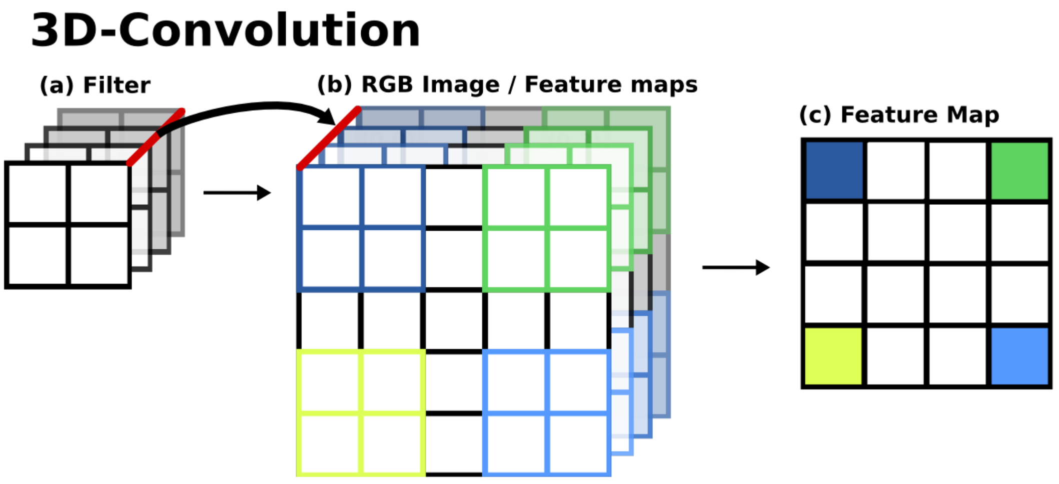

Figure B.8: Illustration of 3D-convolution of a filter of size $2 \times 2 \times 4$ with a tensor of size $5 \times 5 \times 4$. It is called 3D, since the filter and the tensor have a depth greater than 1. The red lines in (a) and (b) that are linked by a black arrow highlight the fact that the depth of the filter and the depth of the tensor should be exactly the same. The result (feature map) of this 3D-convolution applied at *all* spatial locations of the input tensor is given in (c). Note that, even if the depth of the filter and tensor is greater than 1 (here the depth equals 4), the feature map will always have two dimensions only (width, height) and thus, no depth.

spectral image, set of feature maps, etc.) is expressed by:

$$(I * F)(i, j, k) = \sum_{m=0}^{H_F-1} \sum_{n=0}^{W_F-1} \sum_{d=0}^{D_F-1} I(i+m, j+n, d)F(m, n, d), \tag{B.21}$$

where $D_F = D_I$. As for 2D-convolutions, convolving the 3D-filter at all the spatial locations of the 3D-image outputs a feature map of the same width and height than the 3D-image but *without depth* (*i.e.*, $D_M = 1$). Hence, it is important to note that whatever the depth of the 3D-filter/tensor, the output (feature map) of the 3D-convolution has *no depth*. An illustration of a 3D-convolution is given in Fig. B.8.

B.4.2 Spatial Pooling

Another important novelty in CNNs is the simple operation of "spatial pooling". This latter has two major advantages: (i) it provides invariance to slightly different input tensors (images or feature maps) and (ii) it reduces the dimension (width and height) of the input tensor. The first ability is great since it decreases the major problem in classical hand-crafted image representations, that is to say, resulting in two features that are quite *different* for images of identical content but with slight *differences* (such as illumination, point-of-view, translations, etc.). The second point is also crucial since it aims to reach the goal of CNN. Indeed, let recall that the goal of a CNN is to map input images (very large dimensions) to output vectors (small dimensions), thus, it is very important to reduce the output of the filters (feature maps) across the layers in a CNN, and this is exactly the purpose of the spatial pooling.

Formally, the *pooling* operator $\mathcal{P}_R$ applied on a small region R of a tensor is expressed by:

$$\mathcal{P}_R = \mathcal{A}(R(m, n)), \forall (m, n) \in [0, H_R] \times [0, W_R], \tag{B.22}$$

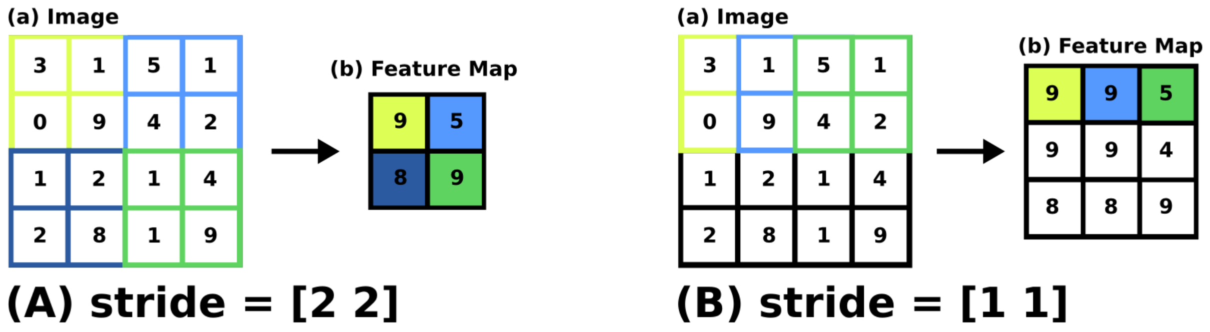

Figure B.9: Illustration of a spatial pooling at all spatial regions of an image (a) of size 4×4 with regions R of size 2×2 and two different strides, namely $[2, 2]$ in (A) and $[1, 1]$ in (B). In (A) and (B), we display the image (a) and we highlight the regions where the pooling is applied, by surrounding the pixels of each different regions with the same color. In (b), we show the obtained feature map and color each pixel with the same color than their associated region in the image (a), that is to say the pixels of the region of the image used for the pooling. Best view in color.

where R is a region of width W_R and height H_R, and $\mathcal{A}$ is an *aggregation* function (*i.e.*, a function where the values of a set of values are grouped together as input on certain criteria to form a single value of more significant meaning). Note that, the aggregation functions always output a real value, whatever the size of the region R. Regarding the region R, it is generally a small region extracted from a bigger image I, that is to say, a region located at pixel (i, j) – *i.e.*, the left-corner pixel of the region R is located at pixel (i, j) – of the image I. Hence, applying the pooling operator at all locations (i, j) of the image I is called *spatial pooling* and it gives the *pooled image* which consists of a smaller feature-map obtained through approximations of group of pixels (region R).

As for the convolution, it is important to note that, the pooling is applied at a particular location (i, j) of an image and on a set of its pixels, namely the region (*i.e.*, all its pixels (m, n) with $m \in [i, i + H_R]$ and $n \in [i, i + W_R]$). Thus, when $i + H_R > H_I$ or $j + W_R > W_I$ the pooling can not be applied, since the size of the filter exceeds the size of the spatial location. Hence, as for the convolution, a padding (zero, mirror, etc.) is generally added to the initial image, which results in outputs of the same size than inputs.

Moreover, the size of the feature map of the spatial pooling depends on the size of the input image, the size of the region R and on one parameter called "stride". Indeed the stride parameter represents a kind of offset on the spatial locations at which the pooling is applied. It is formally expressed as two-dimensional vector $[a, b]$ with a being the offset of the operation (here, pooling) in the width axis and b the offset in the height axis. For instance, if we set the stride to be $[1, 1]$, the pooling will be applied at *all* spatial locations of the image, except those where the size of the region aims to exceed the size of the image. An illustration of a spatial pooling with two different strides is given in Figure B.9.

B.4.3 Overview of a CNN

We first recall that the goal of a neural network (ANN or CNN) is to map from inputs (vectors in ANN and images in CNNs) to outputs (always vectors). Thus, a CNN takes as input an image and outputs a predicted vector after a pass on *convolutional*, *pooling* and *fully-connected* layers. A *convolutional layer* is a set of $n^{(l)}$ filters $\mathcal{F} = \{F_0, F_1, \cdots, F_{n^{(l)}}\}$ of size $W_F \times H_F \times D_F$ that convolve with an input image I of size $W_I \times H_I \times D_I$ or a set of D_M feature maps of size $W_M \times H_M$. Each image,

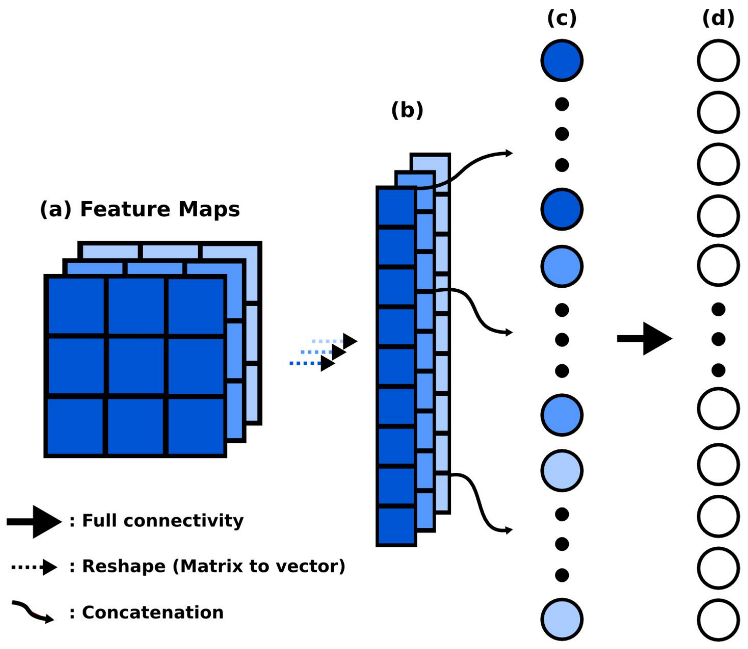

Figure B.10: Illustration of the reshaping operation that is performed after any convolutional/pooling layer before a fully-connected layer. First, each feature-map of the convolutional/pooling layer (a) is reshaped to a vector (b) – simply by concatenating all the lines or columns –, then all the vectors are concatenated to a relatively large vector (c) where each dimension is a neuron and is connected to all the neurons of the next fully-connected layer (d). Best view in color.

filter and set of feature maps can be seen as a 3D-tensor and we will respectively denote them by $\mathcal{I}$, $\mathcal{F}$ and $\mathcal{M}$. Note that the image and the feature maps are 3D-tensor since they have a width, a height and a depth, while for the set of filters, it is a 4D-tensor since it contains $N_{\mathcal{F}}$ filters that corresponds to 3d-tensors. Each of them is thus respectively of size $W_I \times H_I \times D_I$, $W_{\mathcal{F}} \times W_{\mathcal{F}} \times D_{\mathcal{F}} \times N_{\mathcal{F}}$ and $W_{\mathcal{M}} \times H_{\mathcal{M}} \times D_{\mathcal{M}}$. It is important to note that, the convolution operation forces to have filters of the same depth that the 3D-tensor used as input (image or set of feature maps) thus we always have $D_{\mathcal{F}} = D_{\mathcal{I}}$ and since each filter gives one feature map, a set of $N_{\mathcal{F}}$ filters always outputs a set of $N_{\mathcal{F}}$ feature maps, thus the depth $D_{\mathcal{M}}$ of the feature-maps tensor is always equal to the number of filters in the convolutional layer, hence $D_{\mathcal{M}} = N_{\mathcal{F}}$. Simpler than the convolutional-layer, a *pooling layer* consists of the application of the spatial pooling operation on a set of N input feature maps of size $W_{\mathcal{M}} \times H_{\mathcal{M}}$ and it thus, always outputs the same number of feature maps (*i.e.*, N) but with smaller width and height, according to the parameters of the pooling operation. We roughly recall that a fully-connected layer is a set of several neurons that are *all* connected to *all* the neurons of the previous layer.

Putting all these layers together aims to build a complete CNN. Indeed, a complete CNN consists to maps from the input images to the output vectors through the stacking of convolutional, pooling and fully-connected layers. However, while we clearly highlighted how we goes from a convolutional layer to a pooling layer and inversely, it is not clear how to goes from a convolutional or pooling layer to a fully-connected one. Indeed, after getting the feature maps of the convolutional or pooling layers, an operation of *reshaping* that consists to reshape all the feature maps to vectors is performed. Hence, this latter, aims to output a set of vectors, where each vector is the reshape of each feature maps of the previous layer. All these vectors are then concatenated to give the final output vector of this convolutional or pooling layer (*i.e.*, the one just before the fully-connected layer). This final vector has thus all the properties of a fully-connected layer since each of this dimensions is a neuron

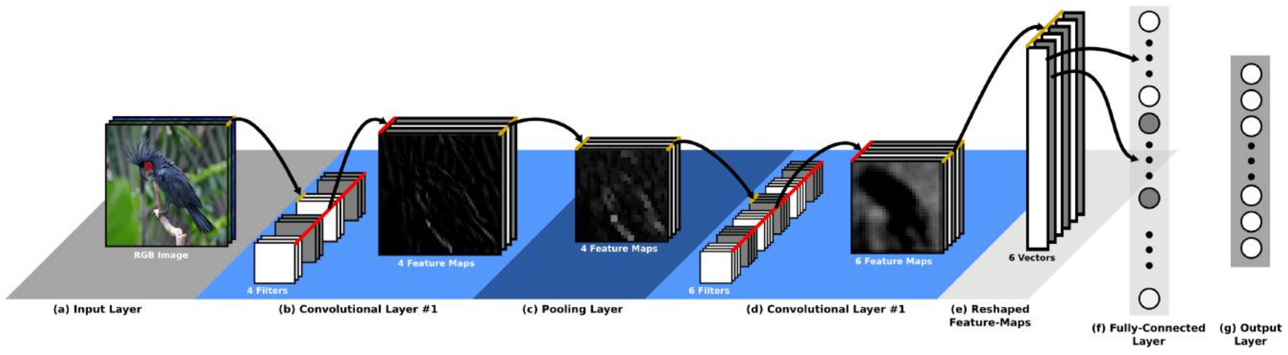

Figure B.11: Overview of a CNN. It maps from the raw input image (a) to the output vector (g) by passing through 4 stacked layers – *i.e.*, convolutional (b), pooling (c), convolutional (d), fully-connected (f). We highlighted with gold lines that the depth of filters in a convolutional layer has the same depth than the input image or set of feature-maps. In red lines, we highlighted the that fact that, we have as much feature-maps as the number of filters. Best view in color.

and is connected to all the neurons of the next layer. An illustration of this reshaping operation is given in Fig. B.10. In summary, we have illustrated an overview a complete CNN in Figure B.11,

B.5 CNN Architectures

In this section, we present the main CNN architectures proposed in the literature. We start by describing the pioneer LeNet architecture, then the most standard architecture to use in any vision problem, namely AlexNet. We then, describe two other architectures (VGG and ResNet) that have been proposed slightly later and that achieved state-of-the-art performances in a wide range of vision problems. Finally in Section B.5.5, we briefly describe the other architectures recently proposed as well as the other schemes for learning CNNs.

B.5.1 Shallow CNN (LeNet)

The pioneering CNN architecture named "LeNet" has been developed and proposed by LeCun *et al.* [140]. It has been used to recognize hand-written digits and to read zip codes. It is quite similar to the CNN that we have described in the previous section. More specifically, LeNet can be summarized by seven points: (i) the stacking of five layers: convolution - pooling - convolution - pooling - fully-connected, (ii) its inputs are normalized using mean and standard deviation to accelerate training, (iii) sparse connection matrix between layers has been used to avoid large computational cost, (iv) hyperbolic tangent or sigmoid were used as activation function, (v) trainable *average* pooling as pooling function, (vi) fully-connected layers as final classifier, and (vii) mean squared error as loss function. An illustration of the LeNet architecture is given in Figure B.12. In overall, this network was the origin of much of the recent CNN architectures, and a true inspiration for many people in the field[1]. In the next parts, we will describe the most popular of them.

[1] 7924 citations on Google Scholar.

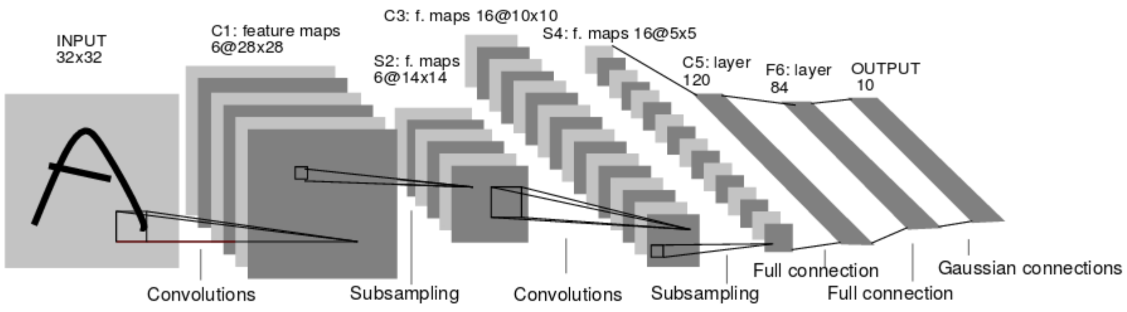

Figure B.12: Illustration (from [140]) of the pioneering LeNet architecture.

B.5.2 Deep CNN (AlexNet)

The well known "AlexNet" [132] architecture was the first work that popularized CNNs in the field of Computer-Vision. The AlexNet has been introduced by Krizhevsky *et al.* [132] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [53] in 2012 and significantly outperformed the best methods (*i.e.*, they achieved a top-5 accuracy of 84% and significantly outperformed the best previous method that was based on hand-crafted features which obtained 74% of accuracy). The network had a very similar architecture to LeNet, but was *deeper* (8 layer: 5 convolutional, 3 pooling and 3 fully-connected), *bigger* (60 millions of parameters), and featured convolutional-layers stacked on top of each other, while previously it was common to only have a single convolutional-layer always *immediately followed* by a pooling-layer. In addition to popularized the approach of CNNs, they also popularized the following three techniques: (i) the ReLU as activation function, (ii) the method of stacking convolutional-layers without being immediately followed by a pooling-layer and (iii) the method of overlapping *max*-pooling avoiding the negative averaging effects of *average*-pooling.

Notice that, the AlexNet architecture which popularized CNNs arrived only 14 years after their invention in 1998 by LeCun *et al.* [140]. This latter, is mainly due to the time to put in conjunction the three main elements that make CNNs very efficient: (i) theoretical advances [100], (ii) growing computation capability (GPGPU based development) and (iii) availability of many annotated data (ImageNet [53]). To date, there is a large investigation in the area of CNN architectures and it results in many architectures proposed every year (*e.g.*, DenseNet, Wide ResNet, DarkNet, etc.), but as said above, we will only present the most popular of them.

B.5.3 Very Deep CNN (VGG)

From the two best teams in ILSVRC 2014, there was the team of the Visual Geometry Group (VGG) that achieved great performances with a new CNN architecture called VGG-Net [228]. It highlighted the importance of the depth of the architecture and proposed to increase this latter by the use of much smaller filters (3×3 versus 11×11 in AlexNet) in each convolutional layers was necessary to increase the depth. Their network contained 19 layers (convolutional, pooling and fully-connected). VGG-Net insights the fact that that multiple 3×3 convolutions in sequence can emulate the effect of larger receptive fields (*e.g.*, 7×7). These ideas were also used in the next network architectures as Inception (Google-Net) and ResNet. In counterpart, VGG-Net is more expensive to evaluate (high number of filters per layers) and uses a lot more memory and parameters, since it contains 140 millions of parameters.

B.5.4 Residual CNN (ResNet)

In 2015, ILSVRC was won by He *et al.* [97] with an architecture of 152 layers called Residual Network (ResNet). Its main contribution was to use batch normalization and special skip connections for training deeper architectures. They also shown that with these techniques, we can train a ResNet with more than 1000 layers (even if the performances saturate after 150 layers). Importantly, the architecture is also missing the two commonly used fully-connected layers ($fc6$ and $fc7$) before the last layer of the network ($fc8$), thus it is only composed of convolutional and pooling layers. However, it has been empirically found in [260] that ResNet usually operates on blocks of lower depth (*i.e.*, 20-30 layers), which act in *parallel*, rather than *serially flow* the entire length of the network. Anyway, ResNets are currently by far the state-of-the-art CNN models and are the best choice for using CNNs if one has GPUs (one or several) and wants to reach the best performances on its task.

B.5.5 Other Architectures and Learning Schemes

Beside all these very popular architectures, many others were recently proposed. For instance, Redmond *et al.* [212] proposed an efficient fully convolutional architecture named DarkNet, that was shown to be very efficient for fast object detection (in the YOLO detection framework). DarkNet is similar to ResNet, but without batch normalization and designed for rapid inference. Another very popular is Inception (also named GoogleNet) proposed by Szegedy *et al.* [237], that is a memory efficient network with multiple loss functions (one at the output layer and other at the high intermediate layers) to enforce the mid-level layers to be discriminative. At CVPR 2017, many other architectures (DenseNet [105], PolyNet [294], ResNext [276], Xception [37]) were proposed. Note that, the most salient one is the DenseNet architecture that consists in connecting each layer of a network to every other layer in a feed-forward fashion. More precisely, for each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. The latter principle has several compelling advantages: it alleviate the vanishing-gradient problem, strengthen features propagation, encourage features reuse, and substantially reduce the number of parameters. It resulted in substantial improvement in terms of classification performance (on object recognition benchmark tasks like CIFAR-10, CIFAR-100, SVHN, and ImageNet).

Note that all these networks were trained in a supervised way, that is to say, by solving a *classification* problem through a softmax loss function. More precisely, the images were generally obtained from ImageNet [53] or Place [297] (and thus contained objects located in the center of the image or "almost empty" scenes) and associated to one label (*i.e.*, name of the object or name of the scene). However, a *supervised* learning scheme needs many annotated data and the process of labeling massive amounts of data is often prohibitively time-consuming and expensive. More importantly, too much labeling can impose human biases on the model. For these reasons, many other works were interested in schemes that relies on less supervision. For instance, many works [35, 120, 262, 263, 264] proposed to rely on *webly-supervised* learning scheme, that consist to consider images labeled by Internet tags that are noisy but can be obtained at near-zero cost. A different scheme is the *semi-supervised* learning [12, 117, 150] that aims at training models on a combination of labeled and unlabeled data. While such schemes are interesting in terms of annotations, they remains an open problem since it is still hard to highly benefit from the unlabeled data and/or noisy labeled data. Finally, recently *unsupervised* learning has gain a large interest from the machine-learning community, especially with the proposal of generative adversarial networks (GANs) [89]. Such approach seems very promising for learning features without the need of annotations, which is the most costly aspect of supervised-learning schemes. However, to the best of our knowledge such scheme were not yet learned on a large scale dataset and evaluated as a representation extractor in a transfer-learning scheme, such that we could compare its level of universality with those of representations learned in a supervised scheme. Nevertheless, some other works proposed interesting "tricks" to learn features at near-zero cost. For instance, Pathak *et al.* [193] proposed to learn by watching object moves, Noroozi *et al.* [183] proposed a way to learn features by counting objects in images, and finally Pathak *et al.* [194] proposed to rely on image completion [17, 36]. Unfortunately such scheme is still far from being really competitive with representations learned with high supervision. A reason is the difficulty to recover structures [27] and illumination [94] from small patches of pixels and more importantly infer the semantics of objects.

While all the previously mentioned works were learned on large amount of data, their principle could also be used as a way to adapt (in a transfer-learning scheme) to target-tasks with few annotated data. However, it is well known [186, 187] that fine-tuning is one of the most efficient adaptation method. Nevertheless, the recent work presented in the thesis of Thibaut Durand [60] was a big step towards high adaptation to few annotated dataset at near-zero cost of annotation. More precisely, the authors proposed a novel framework for *weakly*-supervised Learning [63, 61, 64], which consists in automatically selecting relevant image regions from weak annotations like global image labels and especially benefit from negative regions. In any case, such adaptation method starts with a CNN pre-trained on large annotated data, and more generally, with representations containing a certain level of universality.

C

Implementation details

C.1 Data-Enlargement: A Way to Universality

While it seems quite obvious, here we are interested in showing the effectiveness of adding annotated data to an initial training dataset, in order to increase universality of the learned representation. Indeed, it is well known that feeding neural networks with more data (categories and/or image per category) lead to representations that get better performances on a set of target-datasets [285], in a transfer-learning scenario. This latter, exactly means that a representation learned on a bigger set of training data is more universal than the one learned on a smaller one. Especially, here we detail a small contribution of this Thesis (named FTDG in the following) – this work results from a collaboration with Adrian Popescu, who conducted most of the experiments –, that consist to rely on fine-tuning when learning extremely large neural networks on extremely large datasets.

The principle of the FTDG (Sec. C.1.1) consists to diversify the amount of categories and learn the network with fine-tuning. We report results of experiments that highlights some insights of this method as well as the effectiveness of the data-enlargement approach for the increase of CNN-features' universality.

In this section, we conduct two experiments: (i) we systematically evaluate the performances (on a target-dataset) of a standard network trained on different sizes of training-database and (ii) we highlight the effectiveness of our FTDG method compared to a standard method.

The size of a database can be represented by three aspects: (i) the total number of images, (ii) the total number of categories and (iii) the number of images-per-class for each class. In this experiment, we only variate the two first aspects (total number of images and categories) and neglect the third one since it has been shown in [286] that the performances does not increase with roughly more than $1,200$ images-per-class, which is already the amount of images-per-class in the training-databases we use. We thus take the whole ILSVRC database containing around 1.2 million images labeled among $1,000$ categories and extract two subsets: (i) ILSVRC$^{0.5}$ containing around $500,000$ images labeled among 483 categories and (ii) ILSVRC$^{0.6}$ containing around $600,000$ images labeled among 583 categories. From each database (*i.e.*, the two subsets and the initial ILSVRC dataset), we learn one network following a standard CNN learning-strategy. The method is evaluated in a transfer-learning scheme on the Pascal VOC 2007 dataset and the results are reported on Table C.1. As expected, the

#Images	#Categories	Method	VOC07	CA101
			mAP (in %)	Acc. (in %)
5×10^5	500	Standard	70.3	79.6
6×10^5	600	Standard	71.4	82.0
1.2×10^6	1,000	Standard	76.1	87.8
1.2×10^6	1,000	Standard	86.1	n/a
2.0×10^6	2,000	Standard	88.0	n/a
3.0×10^6	3,000	Standard	88.8	n/a
4.0×10^6	4,000	Standard	89.0	n/a

Table C.1: Overall performance of a standard CNN learning-strategy with a standard architecture (AlexNet on top and VGG-16 at bottom) on two target-datasets (Pascal VOC 2007 and Caltech-101). The network is trained using different size of the source training-database. Note that, the three training-databases not only differs by their number of images but also by their discriminative-problem since they contain images labeled among different set of specific categories.

larger training-database we use, the better the methods perform on the target-datasets, meaning that the better universal representations we get.

C.1.1 Fine-Tuning for Diversified Genericness

When too much categories (*i.e.*, more than 1,000) are fed to CNNs, it is hard to make the network converge, especially when very deep CNNs (*i.e.*, GoogleNet [237], VGG [228], ResNet [97], etc.) are used. Hence, we propose to fix this drawback through a new learning strategy. More specifically, our method consists of two steps, (i) choosing an *appropriate* and large subset of categories from a large concept database and (ii) fine-tuning a pre-trained CNN model on the images of the categories retained in the previous step. Fine-tuning was initially designed, and has been shown to be very efficient [186], to *specialize* a pre-trained general CNN model on a particular domain or a small dataset. Here, we exploit it with an opposite goal, *i.e.*, the *generalization* of an already general model. To the best of our knowledge, it is the first attempt to propose the use of fine-tuning as a way of CNN models generalization, thus we name it "Fine-Tuning for Diversified Genericness" and denote it FTDG.

For the first step that consist to select an appropriate set of categories, we use three criteria:

- balance constraint: selection of categories having at least $1,000$ images;
- visual homogeneity constraint: selection of categories having performance higher than a threshold H after a K-fold cross validation. Standard VGG features [228] ($fc7$ layer) are used to learn concept models (with linear SVM) using balanced data, *e.g.*, as many negatives (from a large and diversified set of Flickr images) as there are positive samples. In the experiments, parameters are set at $K = 5$ and $H = 95\%$.
- conceptual diversity constraint: selection of diversified categories. The algorithm is initialized with a random candidate concept and is run iteratively until S concepts are selected. At each iteration, the concept that has the highest average visual distance from the already selected

	Pascal VOC 2007			
	mAP (in %)			
Number of categories	$1,000$	$2,000$	$3,000$	$4,000$
VGG$_{\text{BASE}}$	86.1	n/a	n/a	n/a
VGG$_{\text{FTDG}}$	86.9	88.0	88.8	89.0

Table C.2: Evaluation of the transferability impact for different numbers of categories retained by our diversification algorithm, through the Pascal VOC 2007 dataset.

concepts is retained. Each candidate concept is represented by its average vector ($fc7$ layer of VGG). Subsets of S unique concepts are retained after diversification.

Here, we use the full ImageNet [216] as the large database and the $1,000$ categories of ILSVRC [53] as seed CNN model for fine-tuning. Indeed, ImageNet contains a big imbalance regarding the number of images per class and it is sub-optimal to learn CNNs without correcting this imbalance [166]. These three criterion are applied on the whole set of ImageNet concepts in order to select a set of candidate concepts.

For the fine-tuning, we initialize all the convolutional layers and the first fully-connected layer ($fc6$) with the weights of the pre-trained network and initialize the weights of the last two fully-connected layers ($fc7$ and $fc8$) with respect to a Gaussian distribution. Note that, we set the learning rate of each layer to default values in the Caffe framework [114], leading to a training *from scratch* of the last two layers and an *updating* of other layers (initialized with the pre-trained network). Our methodology is generic and here we apply it to the popular deep architecture VGG provided by [228]. Regarding the extraction phase for new images, as most works in the literature [114, 209], we extract the $fc7$ layer and use it as mid-level-representation.

C.1.1.1 FTDG versus Standard Learning-Strategy

In this experiment, we set the S parameter of diversification algorithm in four values ($S = \{1000,$ $2000,$ $3000,$ $4000\}$) and learn a CNN for each of these set of categories using our Fine-Tuning for Diversified Genericness strategy. We then compare the performances of the four CNNs in a transfer-learning task. In practice, the target dataset was Pascal VOC 2007 and the $fc7$ layer of each CNN was used to produce the feature vectors of the images. The results are presented in Table C.2. They confirm the intuition that transferability is improved when concepts are added. However, the performance gain becomes marginal when moving from $3,000$ to $4,000$ categories retained. More importantly, when compared to a standard method (without FTDG), the performances of FTDG are significantly better (86.9 versus 86.1 for the standard strategy).

Here, we conducted a second experiment which aim to determine whether our proposed diversification strategy (FTDG) actually improves the learned features or only involves more semantic classifiers. Indeed, it is obvious that our method recognizes more categories since it has been designed for that objective. However, an interesting question is: *Does it improves or decreases the discriminability of the learned features ?* To answer this question, we conducted an experiment in which we compared two VGG networks: (i) the first is trained on the standard ILSVRC dataset ($1,000$ categories), denoted VGG$_{\text{BASE}}$ and (ii) the second is trained with our FTDG method on a diversified set of the ImageNet dataset ($4,000$ categories), denoted VGG$_{\text{FTDG}}$. To compare the two methods, we evaluate their discriminability on the ILSVRC 2012 validation dataset. It is important to note that, in order

Method	**ILSVRC 2012 Validation** Top-1 Accuracy (in %)
VGG_{BASE}	74.0
VGG_{FTDG}	75.5

Table C.3: Evaluation of the ability of our diversification strategy (trained on a diversified set of ImageNet, denoted VGG_{FTDG}) to improve the learned features compared to a base VGG (trained on ILSVRC 2012, denoted VGG_{BASE}) on a subset of the *validation* set of the ILSVRC 2012 dataset.

Superordinate: **vehicle**
Basic-level: **car**
Subordinate: **ford_mustang**
Superordinate: **animal**
Basic-level: **bird**
Subordinate: **palm_cockatoo**

Figure C.1: Examples of *basic*, *subordinate* and *superordinate*-level words used by Humans to categorize objects in images.

to compare these methods regarding their discriminability, we should *not* modify their learned classifiers. However, while VGG_{BASE} is able to classify the $1,000$ categories of the ILSRVC 2012 dataset (since it has been trained on it), VGG_{FTDG} is not able to do it because some of the $1,000$ categories are not contained in its set of $4,000$ categories on which it has been trained. Hence, we considered all the categories that are *common* to the original ILSVRC dataset and the set of categories obtained by our diversification algorithm, which results in 721 categories. Then, we evaluate the two methods on these 721 categories of the ILSVRC 2012 validation dataset. The evaluation is hence conducted on $36,050$ validation images.

The overall performances of the two methods (VGG_{BASE} and VGG_{FTDG}) on the ILSVRC 2012 validation set are given in Table C.3. The obtained top-1 accuracy results are 74.0$\%$ for VGG_{BASE} and 75.5$\%$ for VGG_{FTDG}. Note that, this improvement of 1.5$\%$ is significant since VGG_{FTDG} aims to better recognize 540 images more than VGG_{BASE}. To conclude, while it is obvious that our proposed diversification strategy increases the semantic classifiers, this experiment aims us to highlight its ability to also increase the discriminability of the learned features.

C.2 More Implementation Details of MulDiP-Net

In this section, we describe more implementation details of the proposed MulDiP-Net method (Chapter 4). Especially, we provide more details about the practical categorical-level re-labeling and the partitioning protocol. Before digging into the details of the re-labeling and partitioning protocols, we give more details about the definition of categorical-level categories and compare it to those of hierarchical-level categories.

Hierarchical versus categorical-levels
Given a hierarchy $\mathcal{H}$ with "is-a" relations ($\mathcal{H} = (\mathcal{V}, E)$ consists of a set $\mathcal{V}$ of nodes and directed edges

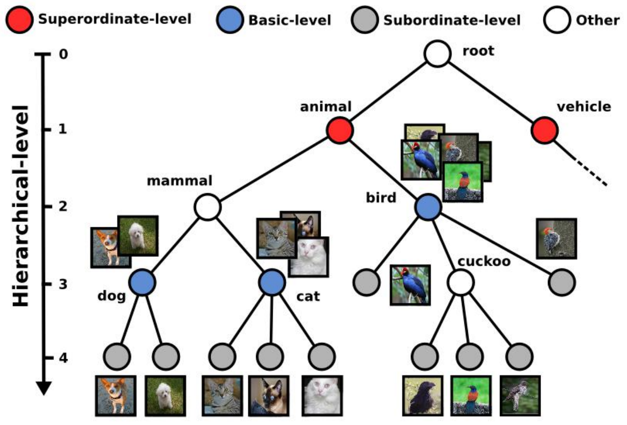

Figure C.2: Illustration of the difference between categorical and hierarchical-level categories on a hierarchy with "is-a" relations. Nodes in the same horizontal line belongs to the same hierarchical-level. Colored nodes with the same color belong to the same categorical-level. Blue nodes belong to the *basic-level*, gray nodes to the *subordinate* and red ones to the *superordinate*. By definition, a categorical-level contains the same type of categories (specific only, basic only, generic only) at a given level while a hierarchical one may contain different types at a given level. For instance, the third hierarchical-level contains generic categories (dog, cat, etc.) and specific ones (cuckoo, etc.).

$E \subseteq \mathcal{V} \times \mathcal{V}$), a hierarchical-level corresponds to the set of all nodes in the same level of the hierarchy. Formally, assuming that none of the hierarchical-level nodes has more than one direct ancestor (*e.g.*, $\forall (v_i) \in \mathcal{V}, Card(\delta_{\mathcal{H}}(v_i)) = 1)$[1] , they correspond to the nodes that have the same amount of total ancestors (*e.g.*, $\forall (v_i, v_j) \in \mathcal{V} \times \mathcal{V}, Card(\{\delta^k_{\mathcal{H}}(v_i)\}^{\infty}_{k=1}) = Card(\{\delta^l_{\mathcal{H}}(v_j)\}^{\infty}_{l=1})$. Thus, the definition is mainly based on the topology of the hierarchy (it contains inconsistent and imbalanced information if the hierarchy is imperfect, which is mostly the case in the real-world). In contrast, a categorical-level is defined by a set of categories from the same type. For instance, the *basic* categorical-level corresponds to the most common words used by Humans to categorize objects. *Subordinate/superordinate* categorical-level corresponds to the words more specific/generic than those of the basic-level. Thus, the definition of categorical-levels is mainly based on a human-cognition knowledge, namely categorization words used by Humans to classify (thus, it contains very relevant and balanced information). In Figure C.1, we show some categorical-level words used by Humans to categorize objects and in Figure C.2, we illustrate the differences between the definitions of categorical and hierarchical-levels.

Re-labeling protocol

We first describe the near-zero cost re-labeling protocol of the categorical-level categories. As mentioned in the main paper, we used *subordinate*, *basic* and *superordinate*-levels with the initial training-dataset (ILSVRC) labeled at *subordinate*-level (specific such as "rottweiler" or "malinois"). Our goal is thus to re-label the categories of the training dataset in general categorical-levels, namely *basic* (generic such as "dog" or "bird") and *superordinate*-levels (very generic such as "animal" or "vehicle"). A simple way to do that is to associate to each *subordinate* category one of its images (can be any image of the category, as long as it contains only the object and that is clearly visible, which is mostly the case on ImageNet images), show it to the annotator and ask him/her to label the image with the most common word that he/she will use to categorize it generally (*subordinate*) or very generally (*superordinate*). For instance, for the subordinate category *hammerhead*, the annotator will label it by

[1] $\delta_{\mathcal{H}}(\cdot)$ corresponds to the *deductive function* introduced in the main paper, that associates to a category v_i of $\mathcal{V}$ its direct ancestor.

the *basic*-level word *shark* and for the category *weimaraner*, it will label it by the word *dog*. For both *subordinate* categories, it will re-label it with the word *animal* for the re-labeling to *superordinate*-level. If one wants to have only words of the hierarchy in order to have homogeneity of the re-labeling between the different annotators, he can constrain the annotators by asking them to re-label the images with one of the set of words obtained from the ancestors of the *subordinate*-category.

Regarding the re-labeling of the whole ILSVRC dataset ($1,000$ specific categories) that we used in Sec. 5.3 of the main paper, we used an already available list [216] of 483 fine-grained categories labeled to 200 *basic-level* categories and re-labeled (using the above re-labeling protocol) the remaining 517 fine-grained categories. Our re-labeling of the remaining categories results into 280 new *basic-level* categories, with a total of $1,000$ subordinate categories re-labeled in 480 *basic-level* ones. In Sec. 5.2 of the main paper, we have reported some results of our method on ILSVRC$^{0.5}$ with the three categorical-level label-sets, including the *superordinate*-level. To re-label *subordinate* categories into *superordinate* ones, we considered their re-labeled 200 *basic-level* categories (obtained from [216]) as the initial categories and re-label them into *superordinate*. This trick aims us to re-label only 200 categories instead of 483 (*subordinate* categories of ILSVRC$^{0.5}$). This latter, results in 12 *superordinate* categories. The whole sets of *basic-level* categories, *superordinate* ones and their relation to the *subordinate* categories has been made available for the community at http://perso.ecp.fr/~tamaazouy/.

Partitioning protocol

Here, we detail how we automatically get the partitioning of the set $\mathcal{C}$ into G subsets (such that $\mathcal{C} = \bigcup_{i=1}^{G} \mathcal{C}_i$), as described in Sec. 3.1 of the main paper. Once the categories of the generic categorical-level (*basic* or *superordinate*) are given, it is straightforward to partition the set $\mathcal{C}$ into G subsets. In fact, let consider the following set of *subordinate* categories: $\mathcal{C} = \{$*convertible*, *landrover*, *malinois*, *rottweiler*$\}$ and the set of their re-labeling categories $\{$*car*, *car*, *dog*, *dog* $\}$, our method groups specific categories *convertible* and *landrover* to *car* and *malinois* and *rottweiler* to *dog*, which results in a set of two subsets (thus, $G = 2$) of generic categories with $\mathcal{L} = \{car, dog\}$. It is important to note that, the partitioning is directly based on the set $\mathcal{L}$ of re-labeled categories, thus $G = Card(\mathcal{L})$. From this example, we have a ratio of two specific categories per generic category but in the real-world with real-datasets, the ratio is much higher. Hence, all the images of the specific categories are re-labeled to the generic categories, resulting to *subordinate* categories that may contain much less images than *basic-level* or *superordinate* ones. For instance, as in Figure C.2 the *basic-level* category *bird* contains as much images as the amount of images in the ensemble of its subsumed *subordinate* categories.

D Résumé en Français

En raison de ses enjeux sociétaux, économiques et culturels, l'intelligence artificielle (dénotée IA) est aujourd'hui un sujet d'actualité très populaire. L'un de ses principaux objectifs est de développer des systèmes qui facilitent la vie quotidienne de l'homme, par le biais d'applications telles que les robots domestiques, les robots industriels, les véhicules autonomes et bien plus encore. La montée en popularité de l'IA est fortement due à l'émergence d'outils basés sur des réseaux de neurones profonds qui permettent d'apprendre simultanément, la représentation des données (qui était traditionnellement conçue à la main), et la tâche à résoudre (qui était traditionnellement apprise à l'aide de modèles d'apprentissage automatique). Ceci résulte de la conjonction des avancées théoriques, de la capacité de calcul croissante ainsi que de la disponibilité de nombreuses données annotées. Un objectif de longue date de l'IA est de concevoir des machines inspirées des humains, capables de percevoir le monde, d'interagir avec les humains, et tout ceci de manière évolutive (c'est à dire en améliorant constamment la capacité de perception du monde et d'intéraction avec les humains). Bien que l'IA soit un domaine beaucoup plus vaste, nous nous intéressons dans cette thèse, uniquement à l'IA basée apprentissage (qui est l'une des plus performante, à ce jour). Celle-ci consiste à l'apprentissage d'un modèle qui une fois appris résoud une certaine tâche, et est généralement composée de deux sous-modules, l'un représentant la donnée (nommé "représentation") et l'autre prenant des décisions (nommé "resolution de tâche"). Nous catégorisons, dans cette thèse, les travaux autour de l'IA, dans les deux approches d'apprentissage suivantes : (i) *Spécialisation* : apprendre des représentations à partir de quelques tâches spécifiques dans le but de pouvoir effectuer des tâches très spécifiques (spécialisées dans un certain domaine) avec un très bon niveau de performance; ii) *Universalité* : apprendre des représentations à partir de plusieurs tâches générales dans le but d'accomplir autant de tâches que possible dans différents contextes. Alors que la spécialisation a été largement explorée par la communauté de l'apprentissage profond, seules quelques tentatives implicites ont été réalisée vers la seconde catégorie, à savoir, l'universalité. Ainsi, le but de cette thèse est d'aborder explicitement le problème de l'amélioration de l'universalité des représentations avec des méthodes d'apprentissage profond, pour les données d'image et de texte. Nous avons abordé ce thème de l'universalité sous deux formes différentes : par la mise en oeuvre de méthodes pour améliorer l'universalité ("méthodes d'universalisation"); et par l'établissement d'un protocole ainsi que la définition de métriques pour quantifier et évaluer son niveau d'universalité. En ce qui concerne les méthodes d'universalisation, nous avons proposé trois contributions techniques : (i) dans un contexte de représentations sémantiques de grande dimensions, nous avons proposé une méthode pour réduire la redondance entre les détecteurs à travers un seuil adaptatif et les relations entre les concepts; (ii) dans le contexte des représentations internes de réseaux neuronaux, nous avons proposé une approche qui augmente le nombre de détecteurs sans augmenter la quantité de données annotées; (iii) dans un contexte de représentations multimodales, nous avons proposé une méthode pour préserver la sémantique des représentations unimodales (par le biais de leur structure de groupes) dans les représentations multimodales. En ce qui concerne la quantification et donc l'évaluation de l'universalité, motivé par une étude cognitive, nous avons proposé d'évaluer les méthodes d'universalisation dans le cadre d'un schéma de transfert d'apprentissage. En effet, ce schéma technique est pertinent pour évaluer la capacité universelle des représentations. Cela nous a également conduit à proposer plusieurs nouveaux critères d'évaluation quantitative pour l'universalisation des méthodes ainsi que plusieurs métriques respectant ces critères.

Bibliography

[1] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *European Conference on Computer Vision*, ECCV, 2014.

[2] K. Ahmed, M. H. Baig, and L. Torresani. Network of experts for large-scale image categorization. In *European Conference on Computer Vision*, ECCV, 2016.

[3] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Good practice in large-scale learning for image classification. *Pattern Analysis and Machine Intelligence, PAMI*, 2014.

[4] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[5] G. Andrew, R. Arora, J. Bilmes, and K. Livescu. Deep canonical correlation analysis. In *International Conference on Machine Learning*, ICML, 2013.

[6] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision*, ICCV, 2015.

[7] J. Atif, C. Hudelot, G. Fouquier, I. Bloch, and E. D. Angelini. From generic knowledge to specific reasoning for medical image interpretation using graph based representations. In *International Joint Conference on Artificial Intelligence*, IJCAI, 2007.

[8] J. Atkinson. The developing visual brain. 2002.

[9] S. Avila, N. Thome, M. Cord, E. Valle, and A. D. A. AraúJo. Pooling in image representation: The visual codeword point of view. *International Journal of Computer Vision and Image Understanding, CVIU*, 2013.

[10] Y. Aytar, C. Vondrick, and A. Torralba. See, hear, and read: Deep aligned representations. *arXiv preprint arXiv:1706.00932*, 2017.

[11] H. Azizpour, A. Razavian, J. Sullivan, A. Maki, and S. Carlsson. Factors of transferability for a generic convnet representation. *Pattern Analysis and Machine Intelligence, PAMI*, 2015.

[12] S.-Y. Bai, S. Agethen, T.-H. Chao, and W. Hsu. Semi-supervised learning for convolutional neural networks via online graph construction. *arXiv preprint arXiv:1511.06104*, 2015.

[13] H. Bannour and C. Hudelot. Towards ontologies for image interpretation and annotation. In *Content-Based Multimedia Indexing*, CBMI, 2011.

[14] H. Bannour and C. Hudelot. Building and using fuzzy multimedia ontologies for semantic image annotation. *Transactions on Multimedia Tools and Applications*, 2014.

[15] M. Bar and S. Ullman. Spatial context in recognition. *Perception*, 1996.

[16] E. Baralis, S. Chiusano, and P. Garza. A lazy approach to associative classification. *IEEE Transactions on Knowledge and Data Engineering*, 2008.

[17] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*, 2009.

[18] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[19] S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database (supplemental material). In *Computer Vision and Pattern Recognition*, CVPR, 2015.

[20] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, PAMI*, 2013.

[21] A. Bergamo and L. Torresani. Meta-class features for large-scale object categorization on a budget. In *Computer Vision and Pattern Recognition*, CVPR, 2012.

[22] A. Bergamo, L. Torresani, and A. W. Fitzgibbon. Picodes: Learning a compact code for novel-category recognition. In *Advances in Neural Information Processing Systems*, NIPS, 2011.

[23] H. Bilen and A. Vedaldi. Universal representations: The missing link between faces, text, planktons, and cat breeds. *arXiv:1701.07275*, 2017.

[24] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python.* " O'Reilly Media, Inc.", 2009.

[25] D. M. Blei and M. I. Jordan. Modeling annotated data. In *International Conference on Research and Development in Information Retrieval*, SIGIR, 2003.

[26] M. Blot, T. Robert, N. Thome, and M. Cord. Shade: Information-based regularization for deep learning. In *International Conference on Image Processing*, ICIP, 2018.

[27] F. Bousefsaf, M. Tamaazousti, S. H. Said, and R. Michel. Image completion using multispectral imaging. *IET Image Processing*, 2018.

[28] M. Brady. Artificial intelligence and robotics. *Artificial intelligence*, 1985.

[29] S. Brodeur, E. Perez, A. Anand, F. Golemo, L. Celotti, F. Strub, J. Rouat, H. Larochelle, and A. Courville. Home: A household multimodal environment. In *International Conference on Learning Representations*, ICLR-W, 2018.

[30] M. Bucher, S. Herbin, and F. Jurie. Improving semantic embedding consistency by metric learning for zero-shot classiffication. In *European Conference on Computer Vision*, ECCV, 2016.

[31] F. Carrara, A. Esuli, T. Fagni, F. Falchi, and A. M. Fernández. Picture it in your mind: Generating high level visual representations from textual descriptions. *Information Retrieval Journal*, 2017.

[32] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[33] I. Chami, Y. Tamaazousti, and H. Le Borgne. Amecon: Abstract meta concept features for text-illustration. In *International Conference on Multimedia Retrieval*, ICMR, 2017.

[34] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, BMVC, 2014.

[35] X. Chen and A. Gupta. Webly supervised learning of convolutional networks. In *Computer Vision and Pattern Recognition*, CVPR, 2015.

[36] G. Chican and M. Tamaazousti. Constrained patchmatch for image completion. In *Advances in Visual Computing*, 2014.

[37] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[38] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *ACM Conference on Image and Video Retrieval*, CIVR, 2009.

[39] M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi. Deep filter banks for texture recognition, description, and segmentation. *International Journal of Computer Vision, IJCV*, 2016.

[40] G. Collell, T. Zhang, and M.-F. Moens. Imagined visual representations as multimodal embeddings. In *Association for the Advancement of Artificial Intelligence*, AAAI, 2017.

[41] A. Conneau and D. Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

[42] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[43] A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*, 2018.

[44] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun. Very deep convolutional networks for text classification. In *Association for Computational Linguistics*, ACL, 2017.

[45] J. Costa Pereira, E. Coviello, G. Doyle, N. Rasiwasia, G. Lanckriet, R. Levy, and N. Vasconcelos. On the role of correlation and abstraction in cross-modal multimedia retrieval. *Pattern Analysis and Machine Intelligence, PAMI*, 2014.

[46] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *European Conference on Computer Vision*, ECCV-Workshop.

[47] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 1989.

[48] H. Daher, R. Besançon, O. Ferret, H. Le Borgne, A.-L. Daquo, and Y. Tamaazousti. Désambiguïsation d'entités nommées par apprentissage de modèles d'entités à large échelle. In *COnférence en Recherche d'Information et Applications*, 2017.

[49] H. Daher, R. Besançon, O. Ferret, H. Le Borgne, A.-L. Daquo, and Y. Tamaazousti. Supervised learning of entity disambiguation models by negative sample selection. In *International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing, 2017.

[50] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Computer Vision and Pattern Recognition*, CVPR, 2005.

[51] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, and D. Batra. Visual dialog. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[52] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In *European Conference on Computer Vision*, ECCV, 2014.

[53] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition*, CVPR, 2009.

[54] J. Deng, J. Krause, A. C. Berg, and L. Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *Computer Vision and Pattern Recognition*, CVPR, 2012.

[55] P. Dollar and C. L. Zitnick. Fast edge detection using structured forests. *Pattern Analysis and Machine Intelligence, PAMI*, 2015.

[56] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Computer Vision and Pattern Recognition*, CVPR, 2015.

[57] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, ICML, 2014.

[58] J. Dong, X. Li, and C. G. M. Snoek. Predicting visual features from text for image and video caption retrieval. *IEEE Transactions on Multimedia*, 2018.

[59] O. F. Dorian Kodelja, Romaric Besançon. Intégration de contexte global par amorçage pour la détection d'événements. In *conférence sur le Traitement Automatique des Langues Naturelles*, TALN, 2018.

[60] T. Durand. *Weakly supervised learning for visual recognition*. PhD thesis, Université Pierre et Marie Curie, 2017.

[61] T. Durand, T. Mordan, N. Thome, and M. Cord. Wildcat: Weakly supervised learning of deep convnets for image classification, pointwise localization and segmentation. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[62] T. Durand, D. Picard, N. Thome, and M. Cord. Semantic pooling for image categorization using multiple kernel learning. In *International Conference on Image Processing*, ICIP, 2014.

[63] T. Durand, N. Thome, and M. Cord. Weldon: Weakly supervised learning of deep convolutional neural networks. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[64] T. Durand, N. Thome, and M. Cord. Exploiting negative evidence for deep latent structured models. *Pattern Analysis and Machine Intelligence, PAMI*, 2018.

[65] T. Durand, N. Thome, M. Cord, and S. Avila. Image classification using object detectors. In *International Conference on Image Processing*, ICIP, 2013.

[66] A. Dutt, D. Pellerin, and G. Quenot. Improving image classification using coarse and fine labels. In *International Conference on Multimedia Retrieval*, ICMR, 2017.

[67] A. Eisenschtat and L. Wolf. Linking image and text with 2-way nets. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[68] M. Engilberge, L. Chevallier, P. Pérez, and M. Cord. Finding beans in burgers: Deep semantic-visual embedding with localization. In *Computer Vision and Pattern Recognition*, CVPR, 2018.

[69] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 2009.

[70] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2012.

[71] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge. *International Journal of Computer Vision, IJCV*, 2010.

[72] F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. *arXiv preprint arXiv:1707.05612*, 2017.

[73] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *Pattern Analysis and Machine Intelligence, PAMI*, 2006.

[74] Y. Feng and M. Lapata. Topic models for image annotation and text illustration. In *ACL Human Language Technologies*, HLT, 2010.

[75] France IA. Rapport de synthèse. link, 2017.

[76] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[77] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, NIPS, 2013.

[78] G. Gay-Bellile, S. Bourgeois, M. Tamaazousti, S. Naudet-Collette, and S. Knodel. A mobile markerless augmented reality system for the automotive field. In *International Symposium on Mixed and Augmented Reality Workshop*, ISMAR-W, 2012.

[79] V. Gay-Bellile, M. Tamaazousti, R. Dupont, and S. Naudet-Collette. A vision-based hybrid system for real-time accurate localization in an indoor environment. In *International Conference on Computer Vision Theory and Applications*, VISAP, 2010.

[80] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Computer Vision and Pattern Recognition*, CVPR, 2009.

[81] A. L. Ginsca, A. Popescu, H. Le Borgne, N. Ballas, P. Vo, and I. Kanellos. Large-scale image mining with flickr groups. In *International Conference on Multimedia Modelling*, MM, 2015.

[82] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, CVPR, 2014.

[83] H. Goh, N. Thome, M. Cord, and J.-H. Lim. Top-down regularization of deep belief networks. *NIPS*, 2013.

[84] C. Gomez, H. Le Borgne, P. Allemand, C. Delacourt, and P. Ledru. N-findr method versus independent component analysis for lithological identification in hyperspectral imagery. *International Journal of Remote Sensing*, 2007.

[85] Y. Gong, Q. Ke, M. Isard, and S. Lazebnik. A multi-view embedding space for modeling internet images, tags, and their semantics. *International Journal of Computer Vision, IJCV*, 2014.

[86] Y. Gong, L. Wang, M. Hodosh, J. Hockenmaier, and S. Lazebnik. Improving image-sentence embeddings using large weakly annotated photo collections. In *European Conference on Computer Vision*, ECCV, 2014.

[87] A. Gonzalez-Garcia, D. Modolo, and V. Ferrari. Do semantic parts emerge in convolutional neural networks? *arXiv preprint arXiv:1607.03738*, 2016.

[88] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[89] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, NIPS, 2014.

[90] I. J. Goodfellow, D. Warde-farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *International Conference on Machine Learning*, ICML, 2013.

[91] A. Gordoa, J. A. Rodríguez-Serrano, F. Perronnin, and E. Valveny. Leveraging category-level labels for instance-level image retrieval. In *Computer Vision and Pattern Recognition*, CVPR, 2012.

[92] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *International Conference on Computer Vision*, ICCV, 2005.

[93] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[94] S. Hadj Said, M. Tamaazousti, and A. Bartoli. Image-based models for specularity propagation in diminished reality. *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[95] D. R. Hardoon, S. R. Szedmak, and J. R. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computing*, 2004.

[96] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, ECCV, 2014.

[97] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[98] L. Herranz, S. Jiang, and X. Li. Scene recognition with cnns: objects, scales and dataset bias. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[99] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

[100] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 2006.

[101] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

[102] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[103] M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 2013.

[104] Z. Hu, X. Ma, Z. Liu, E. H. Hovy, and E. P. Xing. Harnessing deep neural networks with logic rules. In *Association for Computational Linguistics*, ACL, 2016.

[105] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[106] Y. Huang, Q. Wu, and L. Wang. Learning semantic concepts and order for image and sentence matching. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[107] C. Hudelot, J. Atif, and I. Bloch. Fuzzy spatial relation ontology for image interpretation. *Fuzzy Sets and Systems*, 2008.

[108] C. Hudelot, J. Atif, and I. Bloch. Alc(f): A new description logic for spatial reasoning in images. In *European Conference on Computer Vision*, ECCV Workshop, 2014.

[109] M. Huh, P. Agrawal, and A. A. Efros. What makes imagenet good for transfer learning? In *Advances in Neural Information Processing Systems*, NIPS-Workshop, 2016.

[110] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[111] A. Jaimes and S.-F. Chang. Conceptual framework for indexing visual information at multiple levels. In *Internet Imaging*, volume 3964, pages 2–16. International Society for Optics and Photonics, 1999.

[112] M. Jain, J. C. van Gemert, T. Mensink, and C. G. M. Snoek. Objects2action: Classifying and localizing actions without any video example. In *International Conference on Computer Vision*, ICCV, 2015.

[113] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition*, CVPR, 2010.

[114] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *International Conference on Multimedia*, ACM, 2014.

[115] L. jia Li, H. Su, L. Fei-fei, and E. P. Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Advances in Neural Information Processing Systems*, NIPS, 2010.

[116] J. Johnson, R. Krishna, M. Stark, L. Li, D. A. Shamma, M. S. Bernstein, and F. Li. Image retrieval using scene graphs. In *Computer Vision and Pattern Recognition*, CVPR, 2015.

[117] R. Johnson and T. Zhang. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in Neural Information Processing Systems*, NIPS, 2015.

[118] P. Jolicoeur, M. A. Gluck, and S. M. Kosslyn. Pictures and names: Making the connection. *Cognitive Psychology*, 1984.

[119] C. Jörgensen. Attributes of images in describing tasks. *Information Processing & Management*, 34(2-3):161–174, 1998.

[120] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *European Conference on Computer Vision*, ECCV, 2016.

[121] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.

[122] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Computer Vision and Pattern Recognition*, CVPR, 2015.

[123] A. Karpathy, A. Joulin, and F. F. Li. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in Neural Information Processing Systems*, NIPS, 2014.

[124] D. Kiela, A. Conneau, A. Jabri, and M. Nickel. Learning visually grounded sentence representations. *arXiv preprint arXiv:1707.06320*, 2017.

[125] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[126] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[127] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

[128] B. Klein, G. Lev, G. Sadeh, and L. Wolf. Fisher vectors derived from hybrid gaussian-laplacian mixture models for image annotation. *arXiv preprint arXiv:1411.7399*, 2014.

[129] D. Kodelja, R. Besancon, and O. Ferret. Représentations et modèles en extraction d'événements supervisée. In *Rencontres des Jeunes Chercheurs en Intelligence Artificielle*, RJCIA, 2017.

[130] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[131] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *International Conference on Computer Vision*, ICCV Workshop, 2013.

[132] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, NIPS, 2012.

[133] L. I. Kuncheva and J. J. Rodriguez. Classifier ensembles with a random linear oracle. *IEEE Transactions on Knowledge and Data Engineering*, 2007.

[134] S. S. Layne. Some issues in the indexing of images. *Journal of the American Society for Information Science*, 1994.

[135] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition*, CVPR, 2006.

[136] H. Le Borgne, E. Gadeski, I. Chami, T. Q. N. Tran, Y. Tamaazousti, A. L. Ginsca, and A. Popescu. Image annotation and two paths to text illustration. In *CLEF 2016 Evaluation Labs and Workshop, Online Working Notes*, 2016.

[137] H. Le Borgne and A. Guérin-Dugué. Sparse-dispersed coding and images discrimination with independent component analysis. In *International Conference on ICA and BSS*, 2001.

[138] H. Le Borgne, A. Guérin-Dugué, and A. Antoniadis. Representation of images for classification with independent features. *Pattern Recognition Letters*, 2004.

[139] H. Le Borgne, A. Guérin-Dugué, and N. E. O'Connor. Learning midlevel image features for natural scene and texture classification. *IEEE Transaction on Circuits and Systems for Video Technologies*, 2007.

[140] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

[141] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint*, 2017.

[142] Y. Li, W. Ouyang, B. Zhou, K. Wang, and X. Wang. Scene graph generation from objects, phrases and region captions. In *International Conference on Computer Vision*, ICCV, 2017.

[143] Y. Li, J. Yosinski, J. Clune, H. Lipson, and J. Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *International Conference on Learning Representations*, ICLR, 2016.

[144] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, ECCV, 2014.

[145] X. Ling, S. Singh, and D. Weld. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 2015.

[146] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 2017.

[147] L. Liu, L. Wang, and X. Liu. In defense of soft-assignment coding. In *International Conference on Computer Vision*, ICCV, 2011.

[148] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. In *International Conference on Learning Representations*, ICLR Workshop, 2016.

[149] Y. Liu, Y. Guo, E. M. Bakker, and M. S. Lew. Learning a recurrent residual fusion network for multimodal matching. In *Computer Vision and Pattern Recognition*, CVRP, 2017.

[150] K. Longi, T. Pulkkinen, and A. Klami. Semi-supervised convolutional neural networks for identifying wi-fi interference sources. In *Asian Conference on Machine Learning*, ACML, 2017.

[151] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, IJCV*, 2004.

[152] C. Lu, R. Krishna, M. S. Bernstein, and F. Li. Visual relationship detection with language priors. In *European Conference on Computer Vision*, ECCV, 2016.

[153] D. P. L. S. N. T. M. C. M. Carvalho, R. Cadene. Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. In *ACM SIGIR*, 2018.

[154] Z. Ma, Y. Lu, and D. Foster. Finding linear structure in large datasets with scalable canonical correlation analysis. In *International Conference on Machine Learning*, ICML, 2015.

[155] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, ICML, 2013.

[156] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Computer Vision and Pattern Recognition*, CVPR, 2015.

[157] T. Malisiewicz and A. A. Efros. Beyond categories: The visual memex model for reasoning about object relationships. In *Advances in Neural Information Processing Systems*, NIPS, 2009.

[158] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[159] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. *arXiv preprint arXiv:1711.05769*, 2017.

[160] F. Manessi, A. Rozza, S. Bianco, P. Napoletano, and R. Schettini. Automated pruning for deep neural network compression. *arXiv preprint arXiv:1712.01721*, 2017.

[161] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014.

[162] K. Marino, R. Salakhutdinov, and A. Gupta. The more you know: Using knowledge graphs for image classification. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[163] A. Mathews, L. Xie, and X. He. Choosing basic-level concept names using visual and language context. In *Winter Conference on Applications of Computer Vision*, WACV, 2015.

[164] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[165] S. Meftah, N. Semmar, and F. Sadat. A neural network model for part-of-speech tagging of social media texts. In *Conference on Language Resources and Evaluation*, LREC, 2018.

[166] P. Mettes, D. Koelma, and C. G. M. Snoek. The imagenet shuffle: Reorganized pre-training for video event detection. In *International Conference on Multimedia Retrieval*, ICMR, 2016.

[167] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, NIPS, 2013.

[168] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[169] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT press, Cambridge, MA, USA, 1969.

[170] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, ICLR, 2016.

[171] F. Monay and D. Gatica-Perez. Modeling semantic aspects for cross-media image indexing. *Pattern Analysis and Machine Intelligence, PAMI*, 2007.

[172] A. Morgand and M. Tamaazousti. Generic and real-time detection of specular reflections in images. In *International Conference on Computer Vision Theory and Applications*, VISAP, 2014.

[173] A. Morgand, M. Tamaazousti, and A. Bartoli. An empirical model for specularity prediction with application to dynamic retexturing. In *International Symposium on Mixed and Augmented Reality*, ISMAR, 2016.

[174] A. Morgand, M. Tamaazousti, and A. Bartoli. A multiple-view geometric model of specularities on non-planar shapes with application to dynamic retexturing. *Transactions on Visualization and Computer Graphics*, 2017.

[175] H. Murase and S. K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision, IJCV*, 1995.

[176] V. N. Murthy, V. Singh, T. Chen, R. Manmatha, and D. Comaniciu. Deep decision network for multi-class image classification. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[177] H. Nam, J.-W. Ha, and J. Kim. Dual attention networks for multimodal reasoning and matching. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[178] A. P. Natsev, M. R. Naphade, and J. R. Smith. Semantic representation: search and mining of multimedia content. In *International Conference on Knowledge Discovery and Data Mining*, KDD, 2004.

[179] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *International Conference on Machine Learning*, ICML, 2011.

[180] A. Nie, E. D. Bennett, and N. D. Goodman. Dissent: Sentence representation learning from explicit discourse relations. *arXiv preprint arXiv:1710.04334*, 2017.

[181] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *IEEE Computer Vision, Graphics & Image Processing*, 2008.

[182] Y. Nishino, K.and Sato and I. K. Eigen-texture method: Appearance compression based on 3-d model. In *Computer Vision and Pattern Recognition*, CVPR, 1999.

[183] M. Noroozi, H. Pirsiavash, and P. Favaro. Representation learning by learning to count. In *International Conference on Computer Vision*, ICCV, 2017.

[184] D. Novotny, D. Larlus, and A. Vedaldi. Learning 3d object categories by looking around them. In *International Conference on Computer Vision*, ICCV, 2017.

[185] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 2001.

[186] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Computer Vision and Pattern Recognition*, CVPR, 2014.

[187] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Computer Vision and Pattern Recognition*, CVPR, 2015.

[188] V. Ordonez, J. Deng, Y. Choi, A. C. Berg, and T. Berg. From large scale image categorization to entry-level categories. In *International Conference on Computer Vision*, ICCV, 2013.

[189] V. Ordonez, W. Liu, J. Deng, Y. Choi, A. C. Berg, and T. L. Berg. Predicting entry-level categories. *International Journal of Computer Vision, IJCV*, 2015.

[190] W. Ouyang, X. Wang, C. Zhang, and X. Yang. Factors in finetuning deep model for object detection with long-tail distribution. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[191] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *Advances in Neural Information Processing Systems*, NIPS, 2009.

[192] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 2010.

[193] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[194] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[195] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, PAMI*, 2005.

[196] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition*, CVPR, 2007.

[197] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision*, ECCV, 2010.

[198] D. Picard and P.-H. Gosselin. Improving image similarity with vectors of locally aggregated tensors. In *International Conference on Image Processing*, ICIP, 2011.

[199] D. Picard and P.-H. Gosselin. Efficient image signatures and similarities using tensor products of local descriptors. *Computer Vision and Image Understanding*, 2013.

[200] F. Plesse, A. Ginsca, B. Delezoide, and F. Prêteux. Visual relationship detection based on guided proposals and semantic knowledge distillation. *arXiv preprint arXiv:1805.10802*, 2018.

[201] D. L. Poole and A. K. Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[202] A. Popescu, G. Etienne, and H. Le Borgne. Scalable domain adaptation of convolutional neural networks. *preprint arXiv:1512.02013*, 2015.

[203] D. Putthividhy, H. T. Attias, and S. S. Nagarajan. Topic regression multi-modal latent dirichlet allocation for image annotation. In *Computer Vision and Pattern Recognition*, CVPR, 2010.

[204] F. Quanfu and C. Richard. Sparse deep feature representation for object detection from wearable cameras. In *British Machine Vision Conference*, BMVC, 2017.

[205] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Computer Vision and Pattern Recognition*, CVPR, 2009.

[206] A. Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars. Encoder based lifelong learning. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[207] N. Rasiwasia, J. Costa Pereira, E. Coviello, G. Doyle, G. R. Lanckriet, R. Levy, and N. Vasconcelos. A new approach to cross-modal multimedia retrieval. In *International Conference on Multimedia*, 2010.

[208] N. Rasiwasia, P. J. Moreno, and N. Vasconcelos. Bridging the gap: Query by semantic example. *IEEE Transactions on Multimedia*, 2007.

[209] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *Computer Vision and Pattern Recognition*, CVPR, 2014.

[210] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, NIPS, 2017.

[211] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Computer Vision and Pattern Recognition*, CVPR, 2018.

[212] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[213] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[214] E. Rosch. Principles of categorization. *Cognition and Categorization*, 1978.

[215] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[216] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision, IJCV*, 2015.

[217] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision, IJCV*, 2015.

[218] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,, 2016.

[219] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[220] S. H. Said, M. Tamaazousti, and A. Bartoli. Image-based models for specularity propagation in diminished reality. *Transactions on Visualization and Computer Graphics*, 2017.

[221] G. Salton and M. J. McGill. Introduction to modern information retrieval. 1986.

[222] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Ofli, I. Weber, and A. Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[223] N. Semmar. A hybrid approach for automatic extraction of bilingual multiword expressions from parallel corpora. In *Conference on Language Resources and Evaluation*, LREC, 2018.

[224] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations*, ICLR, 2014.

[225] A. Shabou and H. Le Borgne. Locality-constrained and spatially regularized coding for scene categorization. In *Computer Vision and Pattern Recognition*, CVPR, 2012.

[226] S. Shatford. Analyzing the subject of a picture: a theoretical approach. *Cataloging & classification quarterly*, 6(3):39–62, 1986.

[227] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[228] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, ICLR, 2015.

[229] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, ICCV, 2003.

[230] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *Pattern Analysis and Machine Intelligence, PAMI*, 2000.

[231] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association of Computational Linguistics*, 2014.

[232] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[233] N. Srivastava and R. R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in Neural Information Processing Systems*, NIPS, 2012.

[234] P. Stone, R. Brooks, E. Brynjolfsson, O. E. Ryan Calo, G. Hager, J. Hirschberg, S. Kalyanakrishnan, E. Kamar, S. Kraus, K. Leyton-Brown, A. S. David Parkes, William Press, J. Shah, M. Tambe, and A. Teller. ”artificial intelligence and life in 2030.” one hundred year study on artificial intelligence: Report of the 2015-2016 study panel. Stanford University, Stanford, CA, `http://ai100.stanford.edu/2016-report`. Accessed: September 6, 2016, September 2016.

[235] S. Subramanian, A. Trischler, Y. Bengio, and C. J. Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. In *International Conference on Learning Representations*, ICLR, 2018.

[236] D. Surís, A. Duarte, A. Salvador, J. Torres, and X. Giró-i Nieto. Cross-modal embeddings for video and audio retrieval. *arXiv preprint arXiv:1801.02200*, 2018.

[237] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*, CVPR, 2015.

[238] M. Tamaazousti. *L’ajustement de faisceaux contraint comme cadre d’unification des méthodes de localisation: application à la réalité augmentée sur des objets 3D*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, 2013.

[239] M. Tamaazousti, S. Naudet-Collette, V. Gay-Bellile, S. Bourgeois, B. Besbes, and M. Dhome. The constrained slam framework for non-instrumented augmented reality. *Multimedia Tools and Applications*, 2016.

[240] Y. Tamaazousti, H. Le Borgne, and C. Hudelot. Agrégation de descripteurs sémantiques locaux contraints par parcimonie basée sur le contenu. In *Reconnaissance des Formes et Intelligence Artificielle*, RFIA, 2016.

[241] Y. Tamaazousti, H. Le Borgne, and C. Hudelot. Descripteurs à divers niveaux de concepts pour la classification d'images multi-objets. In *Reconnaissance des Formes et Intelligence Artificielle*, RFIA, 2016.

[242] Y. Tamaazousti, H. Le Borgne, and C. Hudelot. Diverse concept-level features for multi-object classification. In *International Conference on Multimedia Retrieval*, ICMR, 2016.

[243] Y. Tamaazousti, H. Le Borgne, and C. Hudelot. Mucale-net: Multi categorical-level networks to generate more discriminating features. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[244] Y. Tamaazousti, H. Le Borgne, C. Hudelot, M. E. A. Seddik, and M. Tamaazousti. Learning more universal representations for transfer-learning. *arXiv:1712.09708*, 2017.

[245] Y. Tamaazousti, H. Le Borgne, and A. Popescu. Constrained local enhancement of semantic features by content-based sparsity. In *International Conference on Multimedia Retrieval*, ICMR, 2016.

[246] Y. Tamaazousti, H. Le Borgne, A. Popescu, E. Gadeski, A. Ginsca, and C. Hudelot. Vision-language integration using constrained local semantic features. *International Journal of Computer Vision and Image Understanding, CVIU*, 2017.

[247] Y. Tamaazousti, H. Le Borgne, A. Popescu, E. Gadeski, A. L. Ginsca, and C. Hudelot. Déscripteur sémantique local contraint basé sur un rnc diversifié. *Traitement du Signal (TS)*, 2017.

[248] J. W. Tanaka and M. Taylor. Object categories and expertise: Is the basic level in the eye of the beholder? *Cognitive Psychology*, 1991.

[249] S. J. Thorpe and M. Fabre-Thorpe. Seeking categories in the brain. *Science*, 291(5502):260–263, 2001.

[250] K. Todorov, C. Hudelot, A. Popescu, and P. Geibel. Fuzzy ontology alignment using background knowledge. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2014.

[251] K. Todorov, N. James, and C. Hudelot. Multimedia ontology matching by using visual and textual modalities. *Transactions on Multimedia Tools and Applications*, 2013.

[252] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *European Conference on Computer Vision*, ECCV, 2010.

[253] T. Q. N. Tran, H. Le Borgne, and M. Crucianu. Combining generic and specific information for cross-modal retrieval. In *International Conference on Multimedia Retrieval*, ICMR, 2015.

[254] T. Q. N. Tran, H. Le Borgne, and M. Crucianu. Aggregating image and text quantized correlated components. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[255] T. Q. N. Tran, H. Le Borgne, and M. Crucianu. Cross-modal classification by completing unimodal representations. In *International Conference on Multimedia - Workshop*, ACM, 2016.

[256] M. Turk and A. Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 1991.

[257] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International Journal of Computer Vision, IJCV*, 2013.

[258] D. C. Van Essen, J. L. Gallant, et al. Neural mechanisms of form and motion processing in the primate visual system. *Neuron*, 13(1):1–10, 1994.

[259] J. C. Van Gemert, C. J. Veenman, A. W. Smeulders, and J.-M. Geusebroek. Visual word ambiguity. *Pattern Analysis and Machine Intelligence, PAMI*, 2010.

[260] A. Veit, M. J. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, NIPS, 2016.

[261] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.

[262] P. Vo, A. L. Ginsca, H. Le Borgne, and A. Popescu. Effective training of convolutional networks using noisy web images. In *International Workshop on Content-Based Multimedia Indexing*, CBMI, 2015.

[263] P. D. Vo, A. Ginsca, H. Le Borgne, and A. Popescu. On deep representation learning from noisy web images. *arXiv:1512.04785*, 2015.

[264] P. D. Vo, A. Ginsca, H. Le Borgne, and A. Popescu. Harnessing noisy web images for deep representation. *International Journal of Computer Vision and Image Understanding, CVIU*, 2017.

[265] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset, 2011.

[266] G. Wang, D. Hoiem, and D. Forsyth. Learning image similarity from flickr groups using stochastic intersection kernel machines. In *International Conference on Computer Vision*, ICCV, 2009.

[267] H. Wang, H. Wang, and K. Xu. Categorizing concepts with basic level for vision-to-language. In *Computer Vision and Pattern Recognition*, CVPR, 2018.

[268] J. Wang, Q. Qin, Z. Li, X. Ye, J. Wang, X. Yang, and X. Qin. Deep hierarchical representation and segmentation of high resolution remote sensing images. In *International Geoscience and Remote Sensing Symposium*, 2015.

[269] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition*, CVPR, 2010.

[270] L. Wang, Y. Li, J. Huang, and S. Lazebnik. Learning two-branch neural networks for image-text matching tasks. *Pattern Analysis and Machine Intelligence, PAMI*, 2018.

[271] L. Wang, Y. Li, and S. Lazebnik. Learning deep structure-preserving image-text embeddings. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[272] Y.-X. Wang, D. Ramanan, and M. Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[273] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. Cnn: Single-label to multi-label. In *arXiv:1406.5726*, 2014.

[274] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, IJCAI, 2011.

[275] Y. Wu, J. Li, Y. Kong, and Y. Fu. Deep convolutional neural network with independent softmax for large scale face recognition. In *International Conference on Multimedia*, ACM, 2016.

[276] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[277] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[278] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[279] F. Yan and K. Mikolajczyk. Deep correlation for matching images and text. In *Computer Vision and Pattern Recognition*, CVPR, 2015.

[280] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu. Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition. In *International Conference on Computer Vision*, ICCV, 2015.

[281] H. Yang, J. Tianyi Zhou, Y. Zhang, B.-B. Gao, J. Wu, and J. Cai. Exploit bounding box annotations for multi-label object recognition. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[282] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition*, CVPR, 2009.

[283] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, and L. Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *International Conference on Computer Vision*, ICCV, 2011.

[284] X. Yin, J. Han, J. Yang, and P. S. Yu. Efficient classification across multiple database relations: A crossmine approach. *IEEE Transactions on Knowledge and Data Engineering*, 2006.

[285] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, NIPS, 2014.

[286] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *International Conference on Machine Learning*, ICML, 2015.

[287] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2014.

[288] R. Yu, A. Li, V. I. Morariu, and L. S. Davis. Visual relationship detection with internal and external linguistic knowledge distillation. In *International Conference on Computer Vision*, ICCV, 2017.

[289] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[290] A. R. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *Computer Vision and Pattern Recognition*, CVPR, 2018.

[291] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, ECCV, 2014.

[292] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, ICLR, 2017.

[293] J. Zhang, S. Sclaroff, Z. Lin, X. Shen, B. Price, and R. Měch. Unconstrained salient object detection via proposal subset optimization. In *Computer Vision and Pattern Recognition*, CVPR, 2016.

[294] X. Zhang, Z. Li, C. C. Loy, and D. Lin. Polynet: A pursuit of structural diversity in very deep networks. In *Computer Vision and Pattern Recognition*, CVPR, 2017.

[295] B. Zhou, D. Bau, A. Oliva, and A. Torralba. Interpreting deep visual representations via network dissection. *arXiv preprint arXiv:1711.05611*, 2017.

[296] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *International Conference on Learning Representations*, ICLR, 2015.

[297] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, NIPS, 2014.

[298] A. Znaidia, A. Shabou, H. Le Borgne, C. Hudelot, and N. Paragios. Bag-of-multimedia-words for image classification. ICPR, 2012.

Titre: Vers l'universalité des représentations visuelle et multimodales

Mots clés: Intelligence artificielle, Apprentissage profond, Transfert d'apprentissage, Apprentissage de représentations, Universalité.

Résumé: En raison de ses enjeux sociétaux, économiques et culturels, l'intelligence artificielle (dénotée IA) est aujourd'hui un sujet d'actualité très populaire. L'un de ses principaux objectifs est de développer des systèmes qui facilitent la vie quotidienne de l'homme, par le biais d'applications telles que les robots domestiques, les robots industriels, les véhicules autonomes et bien plus encore. La montée en popularité de l'IA est fortement due à l'émergence d'outils basés sur des réseaux de neurones profonds qui permettent d'apprendre simultanément, la représentation des données (qui était traditionnellement conçue à la main), et la tâche à résoudre (qui était traditionnellement apprise à l'aide de modèles d'apprentissage automatique). Ceci résulte de la conjonction des avancées théoriques, de la capacité de calcul croissante ainsi que de la disponibilité de nombreuses données annotées. Un objectif de longue date de l'IA est de concevoir des machines inspirées des humains, capables de percevoir le monde, d'interagir avec les humains, et tout ceci de manière évolutive (c'est à dire en améliorant constamment la capacité de perception du monde et d'intéraction avec les humains). Bien que l'IA soit un domaine beaucoup plus vaste, nous nous intéressons dans cette thèse, uniquement à l'IA basée apprentissage (qui est l'une des plus performante, à ce jour). Celle-ci consiste à l'apprentissage d'un modèle qui une fois appris résoud une certaine tâche, et est généralement composée de deux sous-modules, l'un représentant la donnée (nommé "représentation") et l'autre prenant des décisions (nommé "resolution de tâche"). Nous catégorisons, dans cette thèse, les travaux autour de l'IA, dans les deux approches d'apprentissage suivantes : (i) *Spécialisation* : apprendre des représentations à partir de quelques tâches spécifiques dans le but de pouvoir effectuer des tâches très spécifiques (spécialisées dans un certain domaine) avec un très bon niveau de performance; ii) *Universalité* : apprendre des représentations à partir de plusieurs tâches générales dans le but d'accomplir autant de tâches que possible dans différents contextes. Alors que la spécialisation a été largement explorée par la communauté de l'apprentissage profond, seules quelques tentatives implicites ont été réalisée vers la seconde catégorie, à savoir, l'universalité. Ainsi, le but de cette thèse est d'aborder explicitement le problème de l'amélioration de l'universalité des représentations avec des méthodes d'apprentissage profond, pour les données d'image et de texte. Nous avons abordé ce thème de l'universalité sous deux formes différentes : par la mise en oeuvre de méthodes pour améliorer l'universalité ("méthodes d'universalisation"); et par l'établissement d'un protocole ainsi que la définition de métriques pour quantifier et évaluer son niveau d'universalité. En ce qui concerne les méthodes d'universalisation, nous avons proposé trois contributions techniques : (i) dans un contexte de représentations sémantiques de grande dimensions, nous avons proposé une méthode pour réduire la redondance entre les détecteurs à travers un seuil adaptatif et les relations entre les concepts; (ii) dans le contexte des représentations internes de réseaux neuronaux, nous avons proposé une approche qui augmente le nombre de détecteurs sans augmenter la quantité de données annotées; (iii) dans un contexte de représentations multimodales, nous avons proposé une méthode pour préserver la sémantique des représentations unimodales (par le biais de leur structure de groupes) dans les représentations multimodales. En ce qui concerne la quantification et donc l'évaluation de l'universalité, motivé par une étude cognitive, nous avons proposé d'évaluer les méthodes d'universalisation dans le cadre d'un schéma de transfert d'apprentissage. En effet, ce schéma technique est pertinent pour évaluer la capacité universelle des représentations. Cela nous a également conduit à proposer plusieurs nouveaux critères d'évaluation quantitative pour l'universalisation des méthodes ainsi que plusieurs métriques respectant ces critères.

Title: On The Universality of Visual and Multimodal Representations

Keywords: Artificial Intelligence, Deep-learning, Transfer-learning, Representation-learning, Universality.

Abstract: Because of its key societal, economic and cultural stakes, Artificial Intelligence (AI) is a hot topic. One of its main goal, is to develop systems that facilitates the daily life of humans, with applications such as household robots, industrial robots, autonomous vehicle and much more. The rise of AI is highly due to the emergence of tools based on deep neural-networks which make it possible to simultaneously learn, the representation of the data (which were traditionally hand-crafted), and the task to solve (traditionally learned with statistical models). This resulted from the conjunction of theoretical advances, the growing computational capacity as well as the availability of many annotated data. A long standing goal of AI is to design machines inspired humans, capable of perceiving the world, interacting with humans, in an evolutionary way. We categorize, in this Thesis, the works around AI, in the two following learning-approaches: (i) *Specialization*: learn representations from few specific tasks with the goal to be able to carry out very specific tasks (specialized in a certain field) with a very good level of performance; (ii) *Universality*: learn representations from several general tasks with the goal to perform as many tasks as possible in different contexts. While specialization was extensively explored by the deep-learning community, only a few implicit attempts were made towards universality. Thus, the goal of this Thesis is to explicitly address the problem of improving universality with deep-learning methods, for image and text data. We have addressed this topic of universality in two different forms: through the implementation of methods to improve universality ("universalizing methods"); and through the establishment of a protocol to quantify its universality. Concerning universalizing methods, we proposed three technical contributions: (i) in a context of large semantic representations, we proposed a method to reduce redundancy between the detectors through, an adaptive thresholding and the relations between concepts; (ii) in the context of neural-network representations, we proposed an approach that increases the number of detectors without increasing the amount of annotated data; (iii) in a context of multimodal representations, we proposed a method to preserve the semantics of unimodal representations in multimodal ones. Regarding the quantification of universality, we proposed to evaluate universalizing methods in a Transfer-learning scheme. Indeed, this technical scheme is relevant to assess the universal ability of representations. This also led us to propose a new framework as well as new quantitative evaluation criteria for universalizing methods.