Learning More Universal Representations for Transfer-Learning

Youssef Tamaazousti, Hervé Le Borgne, Céline Hudelot, Mohamed-El-Amine Seddik and Mohamed Tamaazousti

Abstract—A representation is supposed universal if it encodes any element of the visual world (*e.g.*, objects, scenes) in any configuration (*e.g.*, scale, context). While not expecting pure universal representations, the goal in the literature is to improve the universality level, starting from a representation with a certain level. To do so, the state-of-the-art consists in learning CNN-based representations on a diversified training problem (*e.g.*, ImageNet modified by adding annotated data). While it effectively increases universality, such approach still requires a large amount of efforts to satisfy the needs in annotated data. In this work, we propose two methods to improve universality, but pay special attention to limit the need of annotated data. We also propose a unified framework of the methods based on the diversifying of the training problem. Finally, to better match Atkinson's cognitive study about universal human representations, we proposed to rely on the transfer-learning scheme as well as a new metric to evaluate universality. This latter, aims us to demonstrates the interest of our methods on 10 target-problems, relating to the classification task and a variety of visual domains.

Index Terms—Deep-Learning, Universal Representations, Universality Evaluation, Transfer-Learning, Visual Recognition.

1 Introduction

Humans are able to recognize a scene and the objects that composed it with disconcerting ease in comparison to a machine. According to Atkinson's cognitive study [1], such capabilities result from the development of a powerful internal representation in their infancy, which they re-use later in life to solve multiple problems. In analogy, machines based on neural-networks also perceives the data with *representations* that they use to solve *tasks*. However, while vision in a human works well for multiple problems, machines are only able to solve one at a time. This latter motivated some recent works, that wanted to "mimic" such abilities of humans by learning *universal models* [2], [3] or *universal representations* [4], [5], [6]. By the former, we mean models that are able to solve every possible task (recognition, detection, segmentation, etc.), and by the latter, we mean representations that are able to encode every possible element (materials, objects, scenes, etc.), in every possible configuration (scale, occlusion, context, etc.). Note that, without a good encoding of the data, a task can not be solved efficiently, thus a first step toward universal models is to get universal representations. We are far from expecting a representation that could encode any information, thus the actual purpose of our work is to *improve* the universality of a given representation, at a minimal cost.

Given a reference universal representation, a straightforward way to increase its universality, is undoubtedly the use of more training data (tasks and domains). For instance, [4], [5], [6] proposed to simultaneously learn a wide set of different visual *domains*, by better considering the additional data, through the use of scaling parameters that learns the statistics of each domain. In the same vein, Subramanian *et al.* [7] considered multiple datasets with different *tasks* and proposed to find which of the available multi-task learning algorithms and set of tasks provides the best universal level. Note that, this approach of adding more data is limited by availability of data. Hence, here we are interested by increasing universality *from a fixed set of data* (tasks and domains). Conneau *et al.* [8], [9], [10] have also the same goal. However, while their approach consists to find the best algorithm, we argue that we could structure the training-data in a way that provides different tasks to learn more or better features.

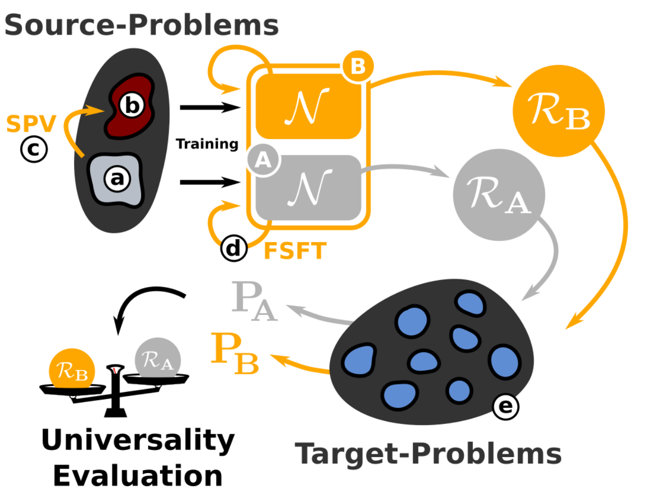


Fig. 1. A *universal* representation is a set of features learned on a source-problem (SP), that exhibit good performances when it is transfered on many target-problems (TPs). A universal representation $\mathcal{R}_A$ is obtained by a reference method (A) that consists in training a network $\mathcal{N}$ on a SP (a). When it is transfered on a large set of TPs (e), it leads to a *high* aggregated performance P_A on it. In this paper, we propose to build a *more universal* representation $\mathcal{R}_B$ by: (i) normalizing and combining the features of (A) with those of a network (yellow $\mathcal{N}$ block) learned after a *source-problem variation* (SPV) (c) – that, transforms an initial SP (a) into a new one (b) – and (ii) re-training both networks on their initial SP through *focused self fine-tuning* denoted FSFT (d). Since our method (B) is more universal than (A), it has a higher universality score according to the proposed evaluation-metric.

- *Y. Tamaazousti, is at the CSAIL of MIT, USA. E-mail: ytamaaz@mit.edu*
- *H. Le Borgne, M.E.A. Seddik and M. Tamaazousti are at the CEA, LIST, France. E-mail: firstname.lastname@cea.fr*
- *C. Hudelot is at the MICS laboratory of CentraleSupélec (University of Paris-Saclay). E-mail: celine.hudelot@centralesupelec.fr*

Manuscript received September 2, 2018

The overall contribution of this article (illustrated in Fig. 1) consist in a method that increases universality from a fixed set of data. In the vein of MuCale [11], we proposed MuLDiP-Net that: (i) *variates* the initial source-problem (SP); (ii) uses its resulting SPs to train new features; and (iii) combines all the features to form a more universal representation. By construction, MulDiP-Net learns new discriminative features without any additional images or annotations. Regarding the variation, we propose a formalism that consist to start from a set of *specific* categories (*e.g.*, *rottweiler*, *pit-bull*) and group them in more *generic* ones (*e.g.*, *dog*), according to the upper-levels of a given hierarchy (*e.g.*, WordNet, hierarchy obtained by clustering the specific categories, or categorical-levels relating to Human-categorization [12], [13]). The proposed formalism is quite general and makes MuCale a particular case of our method, that groups categories according to categorical-levels only. We also introduce a new method named *focused self fine-tuning* (FSFT), based on self-training [14] of CNNs on the *same* SP. This approach increases the universality without any additional data, and more interestingly, at the same network-capacity. FSFT can also be used to reduce the dimension of representations. Combined to MulDiP-Net, it thus improves the performances while reducing the network-capacity.

We also address the problem of universality evaluation. Rebuffi *et al.* [5] proposed an interesting *Visual Decathlon Challenge* (VDC) to measure universality on multiple domains. Furthermore, they proposed a VDC metric that coherently aggregates the scores on the multiple domains. However, they restricted their evaluation in a classical end-to-end scheme: learn the representation on the *train*-set and evaluate on the *test*-set. Such scheme do *not* completely match with the claim of Atkinson[1], that is, learning on one environment and use the learned representation on *different* problems. Hence, we propose to go further by evaluating universality in a transfer-learning (TL) scenario [14], [15]. TL, naturally fits with [1], with the *source*-problem that corresponds to the "infancy learning environment" and the *target*-problems to those "solved later in life". Also, while the VDC metric respects some desirable criteria, we argue that more are required for universality, and thus proposed a new metric that respects more of them.

The present manuscript differs in many ways from our previous work on this topic [11]. In particular, it includes the following new contributions: (1) a new universalizing method based on self-training of neural-networks [14] (Sec. 3); (2) a more general formalism of the MulDiP-Net approach based on the principle of source-problem variation by grouping (Sec. 4), allowing the use of multiple variations and thus making [11] a special case of our approach; (3) the formalization of FSFT as a dimensionality reduction method and its combination with MulDiP-Net (Sec. 4.5), making this latter more performing while significantly reducing its network-capacity; (4) the evaluation of universality in a TL scheme, that better fits the claim of [1] and the highlight of four desirable criteria for universality evaluation as well as the proposition of a new metric (Sec. 5); and (5) more extensive and detailed experimental results, including a comparison to the state-of-the-art based on ten available target-problems (Sec. 6), as well as an in-depth analysis of our approach including an ablation study and the impact of some important component of our approach (Sec. 6.3 and supplementary material).

2 Related Work

2.1 Learning Universal Representations

With UberNet, Kokkinos [2] considered a unified architecture that is trained end-to-end to tackle several vision tasks (universal model) and addressed resulting technical challenges. In the same vein, Subramanian *et al.* [7] used multiple datasets with different NLP *tasks* and proposed to find which of the available multi-task learning algorithms and set of tasks are the best to learn universal representations. Alternatively, Bilen and Vedaldi [4] rather considered that a universal representation should be able to address multiple visual domains. They proposed to learn a compact representation able to perceive a wide set of domains, using scaling parameters to learn the statistics of each. It has been extended in [5], [6] by respectively using sequential and parallel residual adapters. All these works use more data, and it is undoubtedly the most straightforward way towards universality, but it is limited by its availability. It is thus interesting to improve universality from a *fixed set of data*, which is our goal as well as that of Conneau *et al.* [8], [9]. However, in contrast to their approach, that consists to find the best task and algorithm for universality, ours is to structure the data in a way that automatically provides different tasks to learn more or better features.

All the above works consider *multiple* source-problems (*e.g.*, tasks [7], domains [5] or problems with same images but different categories as our work) to learn the universal representation. It is important to note that learning the representation with one network (using multi-task, multi-label or recursive [3] learning) is just a way to do it. In this paper we proposed to rely on multiple networks through independent learning, and the advantages and drawbacks of these methods will be detailed in Sec. 2.2.2.

2.2 Approaches for Universalizing Methods

Many works can be considered as universalizing methods, in the sense that they diversify and increase feature detectors, either by modifying the source-problem (SP) on which a network is learned, or relying on ensemble models that learn several networks on different problems. We propose a unified view of these approaches, using the notion of Source-Problem Variation (SPV, Sec. 4.1).

2.2.1 Learning one Network on a Modified SP

To diversify an initial problem, a first approach consists in adding new categories and corresponding annotated images. Three kinds of categories are added: *specific* [4], [5], [15], [16] (*e.g.* rottweiler), *generic* [17], [18], [19] (*e.g.* dog) and *noisy* [20], [21]. In some cases, the categories added contain data from multiple domains [4], [5]. These methods slightly increase 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 additional data and corresponding annotations. Another limitation lies in the learning methodology. Indeed, the network is often trained jointly on the generic and specific data, resulting into a mix of generic and specific features in the intermediate layers. Since a softmax loss is often used, it considers generic and specific categories as mutually exclusive, that is not the case. At the opposite, we propose to group the categories according to their level and separately learn the features on each to respect the real-world semantics. Alternatives to this proposal are the works of [22], [23] that respectively group hierarchically or by clustering. However (as shown in the experiments), for the sake of universality, it is preferable to group at categorical-levels as we propose.

[1] In all the document, the "claim of Atkinson" states for: "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 multiple kind of different problems" [1].

2.2.2 Learning a Set of Networks on a Set of SPs

The methods of the second approach [24], [25], [26], [27], [28], [29] give an answer to the problem that in joint training, different kinds of features are undesirably mixed. Indeed, 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 smaller 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 types 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 proposed MulDiP-Net method, provides 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 [24], [25], [27], [29] from this approach re-label specific categories into *non-semantic* generic ones, that is to say categories that do not exist in the real world) [23] and capture the common properties among many object classes independently of an actual common semantics. 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. For the sake of universality, it is preferable to rely on a grouping process that uses explicit human categorization expertise in order to reflect semantically-real relations between categories.

2.3 Universality Evaluation

Inspired by studies on the visual brain (claim of [1]), authors of [4], [5], [6] evaluated universality of representations as their ability to simultaneously cover a large range of visual domains. Their evaluation consists in learning and testing on the *same problem* and only the *visual domain* differs. Moreover, since many problems are considered, they proposed a metric to aggregate the scores in each task, that respects their proposed criterion of significance. While such criterion is undoubtedly useful for universality evaluation, it should be noted that their classical evaluation framework does *not* completely match with [1]. In contrast to our transfer-learning scheme, their scheme do not consider the terms "re-used later in life" which means that *multiple* and *different* problems should be solved by a universal representation. Moreover, in addition to their criterion of significance and the coherent aggregation considered in [7], we proposed a total of six additional criteria important for universality evaluation, as well as two metrics that respect almost all the criteria. Nevertheless, while such TL scheme has been already used in the NLP community [7], [8], [9], [30] for universal representations, to the best of our knowledge, we are the first to propose it in the vision community and more importantly, to link it to the claim of [1].

2.4 Cognitive Studies in Computer Vision

A last line of work deals with the inspiration from cognitive studies in computer vision [31], [32], [33], [34]. Generally, their goal is to output basic-level concepts of an image from a set of predicted finer ones. An exception is the work of [34], that is closest to ours since they consider categorical-levels in their representation. As in our work, their system reflects the psychological hint stating that, even if humans tend to categorize objects at the subordinate-level, they are still aware of the other categorical-levels [13]. However, the key difference is how we integrate that hint as well as the purpose of its consideration. Indeed, our goal is to diversify the features learned in CNNs, while they aim at solving the problem of generic categories that output low scores because of their high intra-class variance, in order to force their beneficial consideration. Moreover, we opt for an integration at three levels (data, learning and representation), while they do it only after the computation of their semantic representation [35], [36].

3 Focused Self Fine-Tunning

We propose a new method that takes advantage of the principle of the *re-training* of neural networks *on the same problems*, and thus does not need more data [15], [18], [19], nor increasing the network capacity [11], [24], [37], [38]. Our approach relates to the work of [14] who proposes an extensive study of the effect of different *self-training* methods (*i.e.*, re-training a neural network on the same problem it was trained originally). They explored two learning-strategies: (i) frozen-based re-training (that we call Frozen Self-Training and denote **FrST**) and (ii) fine-tuning based re-training (that we call Self Fine-Tuning and denote **SFT**). More precisely, both methods re-train an initial network (that we denoted **initNet**), with $\Theta^a{=}(\theta_1^a, \theta_2^a)$ being its set of *trained* weights. Both methods consist in three steps: (i) the weights of FrST and SFT are splitted into two sets (θ_1^b and θ_2^b), with θ_1^b containing the weights of the first L layers and θ_2^b, the weights of the last ones (ii) the two sets of weights are initialized differently – in FrST and SFT, the first layers are initialized with the weights of the pre-trained initNet and the last layers are initialized randomly – ; and finally (iii) the re-training of the weights – FrST retrains only last layers and "frozes" the first ones, while SFT retrains them all with the *same* learning-rate. Their extensive study leads to some interesting conclusions that motivate our approach. In particular, they showed that: (i) FrST hurts the performances of initNet because"initNet contained fragile co-adapted features on successive layers" (*i.e.*, features that interacted with each other in a fragile way during initNet learning such that this co-adaptation could not be relearned by the randomly initialized upper layers *alone*) and (ii) SFT slightly increases the performance because "it aims to recover co-adapted features that were trained by the initNet". Simply said, it is important to preserve some knowledge acquired during the learning of the original network, but it is a sub-optimal to *completely* focus the training on the last layers. As a consequence, we propose a new method called Focused Self Fine-Tuning (FSFT) that can be seen as an hybrid view of the two previous ones. As in [14], the re-training principle consists in dividing the weights learned on the initNet into two sets, initializing them differently (first with the pre-trained initNet, others randomly), jointly minimizing them but with *different* learning-rates. An illustration is given in Fig. 2.

More formally, let us consider a source training database $\mathcal{D}^{\mathcal{S}}$ containing N images x_i with their associated labels y_i. Let us

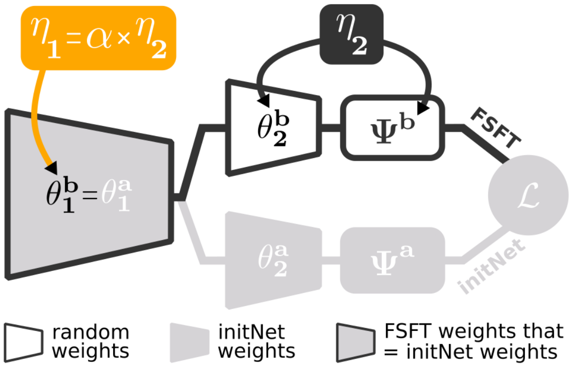


Fig. 2. Focused Self Fine-Tunning (FSFT). An initial network (gray branch) denoted initNet (first layer's weights denoted θ_1^a, last ones θ_2^a and the classifier Ψ^a) is trained on a source-database, by minimizing the loss-function $\mathcal{L}$. Once initNet trained, another network (black branch) denoted FSFT (first layer's weights denoted θ_1^b, last ones θ_2^b and the classifier Ψ^b) is trained on the *same* source-database and solves the same problem (minimize same loss $\mathcal{L}$). The weights θ_1^b (black block filled in gray) of the first layers of FSFT are initialized with the firsts of initNet (θ_1^a) and *weakly* updated through training. The weights of the last layers θ_2^b (black blocks filled in white) are randomly initialized and *fully* trained.

also consider a network $\mathcal{N}$ that was trained on $\mathcal{D}^{\mathcal{S}}$ by minimizing a loss function $\mathcal{L}$ with a certain optimization algorithm. Let note $\Theta^a{=}(\theta_1^a, \theta_2^a)$ being its set of *trained* weights splitted in two sets, namely θ_1^a that contains the weights of the first L layers and θ_2^a that contains those of the remaining layers. FSFT consists to re-train the network $\mathcal{N}$ by minimizing the same loss-function $\mathcal{L}$ as initNet, on the same training database $\mathcal{D}^{\mathcal{S}}$, which is expressed by:

$$\arg\min_{\Theta^b} \mathcal{L}((\Psi^b(\mathbf{x}, \Theta^b = (\theta_1^b, \theta_2^b)); \mathbf{y}), \tag{1}$$

where $\Psi^b(\mathbf{x}, \Theta^b)$ is the predicted probability vector for images $\mathbf{x}$ using learnable weights $\Theta^b{=}(\theta_1^b, \theta_2^b)$. The set of weights θ_1^b is initialized with those of the first layers of initNet (*i.e.*, $\theta_1^b{=}\theta_1^a$) and θ_2^b is initialized randomly. The individual weights w_{ij}^b, that forms the full sets θ_j^b (with $j \in \{1, 2\}$), are updated through:

$$w_{ij}^b \leftarrow w_{ij}^b - \eta_j \frac{\partial \mathcal{L}}{\partial w_{ij}^b}, \forall w_{ij}^b \in \theta_j^b, \tag{2}$$

where, η_j respectively corresponds to the learning-rate of the first layers (when $j{=}1$) and last ones (when $j{=}2$). More specifically, $\eta_1{=}\alpha{\times}\eta_2$, with $\alpha \in [0, 1]$ is a parameter that can be set by cross-validation. To summarize, our FSFT method *preserves* some knowledge acquired during the learning of the initNet (by initializing the weights of its first layers, with the firsts of initNet). It also *focuses* the training on the last layers (by completely training the last layers, while allowing, through factor α, some slight change of the first ones).

4 Multi Discriminative-Problem Network

The previous method is efficient without needing additional data nor capacity but always solves the *same* problem. To learn *more* universal representations, one must solve *different* problems. In this section, we propose another universalizing method that combine *different* but *complementary* features learned on *different* problems. It consists in three components: (i) source problem variation (SPV) (Sec. 4.1) – that define a new source problems (SPs) – with a special emphasis on variation with *grouping* (Sec. 4.2); (ii) independent training of networks on a set of multiple SPs obtained by the SPV (Sec. 4.3); and (iii) extraction of features from the set of trained networks followed by a particular combination to form the more universal representation (Sec. 4.4). Since each SP is a discriminative model solved by a neural network, the method is named "Multi Discriminative Problem Network" (MulDiP-Net, illustrated in Fig. 3). In Sec. 4.5, we use the FSFT (Sec. 3) as a *dimensionality reduction* method in the MulDiP-Net.

4.1 SPV: Source-Problem Variation

A source problem $\mathcal{D}_k^{\mathcal{S}} = \{(x_i^k, y_i^k)\}_{i=[\![1,N_k]\!]}$, consists in a set of N_k pairs (x_i^k, y_i^k) with x_i^k being a training image and y_i^k its associated label. The images x_i^k are labeled according a label set $\mathcal{Y}_k = \{c_1^k, \dots, c_{C_k}^k\}$ of C_k categories. By solving this source problem (SP), the CNN learns features that discriminate between the images of the different categories of the given SP. For instance, if we consider a network that solves a SP containing images of *lemons* and *green-apples*, the network will learn different features than one that solves a SP containing *lemons* and *strawberries*'s images. Hence, changing the SP to be solved by a CNN can lead to a change in the set of learned features. Motivated by this assumption[2], we propose the principle of *Source Problem Variation* (SPV) that is 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:

$$\begin{aligned} \vartheta_0^k : \quad \{\mathbb{R}^{S_I} \times \{0,1\}^{C_0}\}^{N_0} \quad & \to \{\mathbb{R}^{S_I} \times \{0,1\}^{C_k}\}^{N_k} \\ \mathcal{D}_0 \quad & \mapsto \mathcal{D}_k = \{(x_i^k, y_i^k)\}_{i=[\![1,N_k]\!]}, \end{aligned} \tag{3}$$

with the following constraints:

$$\begin{cases} \forall i \in [\![1, N_k]\!], \forall j \in [\![1, N_0]\!], \exists (x_i^k, y_i^k), x_i^k \neq x_j^0 \text{ or } y_i^k \neq y_j^0 \\ \forall i \in [\![1, N_k]\!], \forall j \in [\![1, N_0]\!], \exists (x_i^k, y_i^k), x_i^k = x_j^0 \text{ or } y_i^k = y_j^0 \end{cases} \tag{4}$$

where $S_I = W \times H \times D$ corresponds to the size of images (*i.e.*, width W, height H and depth D with $D = 3$ 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 the label-set $\mathcal{Y}_k = \{c_1^k, \dots, c_{C_k}^k\}$ containing C_k categories. Regarding the constraints, the first ensures that at least one element of $\mathcal{D}_0$ has to be variated compared to the elements of $\mathcal{D}_k$ and the second one warrants that at least one element of $\mathcal{D}_0$ has to be in $\mathcal{D}_k$. With these constraints, taking a dataset completely different than 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 many nature and act both on the images $\{x_i^0\}_{i \in [\![1,N_0]\!]}$ and/or the labels $\{y_i^0\}_{i \in [\![1,N_0]\!]}$). We consider three types of variations: (i) *adding* images or categories to the initial SP; (ii) *splitting* the initial SP ; (iii) *grouping* categories (thus images) of the initial SP (Fig. 4).

This principle, although not explicitly described as a SPV, have been already applied in the literature. For instance, variations on the image set while preserving the set of categories have been proposed in [39], [40]; *adding* variations has been used by [4], [5], [6] (by adding data labeled among *specific* categories) and [18], [19] (by adding data labeled among *generic* categories); *splitting* variations on the set of categories has been performed in [15], [24], [25], [28], [29] and finally, variation of the categories by *grouping* them has been explored in [22], [23], [41]. However, note that a

[2] Assumption empirically validated by [16]. Indeed, they have shown that a CNN trained on scenes learns *different* features (*i.e.*, object detectors) than one trained on objects (which learns object-part detectors).

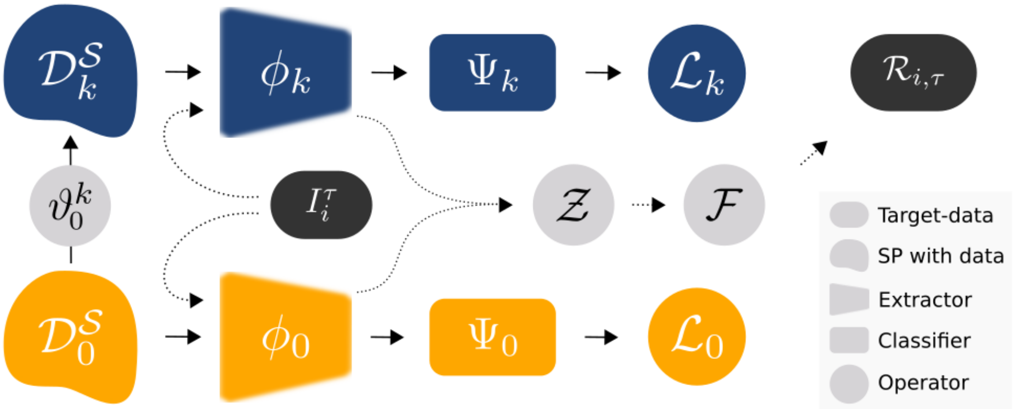


Fig. 3. Illustration of our MulDiP-Net method. Let consider an *initial* source problem (SP) $\mathcal{D}_0^{\mathcal{S}}$ consisting in a set of images and their associated labels $(x_i^0, y_i^0)_{i\in[\![1,N]\!]}$. MulDiP-Net consists in three phases: (i) variation (ϑ) of the initial SP ($\mathcal{D}_0$) into new ones ($\mathcal{D}_k$); (ii) training networks (ϕ and Ψ) on the whole set of SPs ($\{D_0, D_k\}$); and (iii) normalization ($\mathcal{Z}$) followed by combination ($\mathcal{F}$) of the features extracted from each trained network in order to form a more universal representation. More precisely, phase (i) 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 (only one variation illustrated here, thus K=1). Phase (ii) consists in the learning of one network for each of the K+1 SPs, resulting in a set of K+1 trained networks: $\{\mathcal{N}_k\}_{k=[\![0,K]\!]}$. 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 $\mathcal{D}_k^{\mathcal{S}}$. Phase (iii) consists in the extraction of the more universal representations $R_{i,\tau}$ from images I_i^τ of a target-task τ. Specifically, it passes the images I_i^τ into the trained networks and gets features (through the extractors ϕ_k) that are independently normalized ($\mathcal{Z}$) and fused ($\mathcal{F}$) in order to output the final representation $R_{i,\tau}$.

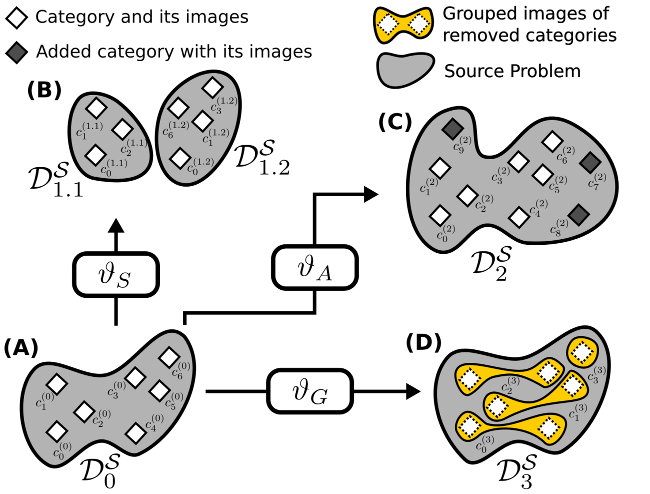

Category and its images
Added category with its images
Grouped images of removed categories
Source Problem
(B)
$\mathcal{D}_{1.1}^{\mathcal{S}}$
$\mathcal{D}_{1.2}^{\mathcal{S}}$
(C)
$\mathcal{D}_2^{\mathcal{S}}$
ϑ_S
ϑ_A
ϑ_G
(A)
$\mathcal{D}_0^{\mathcal{S}}$
(D)
$\mathcal{D}_3^{\mathcal{S}}$

Fig. 4. Illustration of three kind of SPV: *Splitting*, *Adding* and *Grouping* (see figure legend for description of each graphical elements). An initial source problem (SP) $\mathcal{D}_0^{\mathcal{S}}$ is illustrated in (A). (B) represents the output of the splitting SPV (ϑ_S) which results into two *diminished* sets of training-data (each of them contains less images and categories) compared to the initial SP $\mathcal{D}_0^{\mathcal{S}}$ – *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), an *adding* SPV (ϑ_A) results in an *increased* set of training data (more images and categories), compared to $\mathcal{D}_0^{\mathcal{S}}$ (*i.e.*, $N_2 > N_0$ and $|\mathcal{C}_2| > |\mathcal{C}_0|$)). The last example is our grouping SPV (ϑ_G), illustrated in (D). It results in the *same amount* of training-data (*same* set of images but labeled according *less* amount of categories), compared to $\mathcal{D}_0^{\mathcal{S}}$ (*i.e.*, $\{x_i^3\}$=$\{x_i^0\}$ and $|\mathcal{C}_3| < |\mathcal{C}_0|$).

SPV can be applied for different goals and as explained in the beginning of this section, our goal is to learn *different* features than those learned on the initial SP, but *complementary* when combined together. Thus, with respect to our goal, the SPV has to be associated to a learning procedure and learn new neurons. Beyond the stated aim, our work introduce a new type of SPV based on the *grouping* of categories and their associated images.

4.2 SPV Based on Grouping

Grouping-SPV consists in the grouping of images through the grouping of their initial categories. In practice, it can be done in many ways (*e.g.*, randomly, based on clustering or semantically). Here, motivated by knowledge on human categorization, we propose a *semantic* grouping, through *categorical-levels*. The advantage of the semantic-grouping lies in its semantic aspect, which aims to get a new SP that is highly different than the initial one but relevant, which is not necessarily the case with random and clustering-based grouping. Compared to other SPVs, the grouping one has important advantages. Indeed, compared to adding SPV (which needs more annotated data, costly to obtain) grouping SPV does not need more annotated data, and 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.

Semantic-grouping SPV consists in the grouping of specific categories into generic ones, according to a semantic knowledge. This semantic knowledge is generally represented in the form of hierarchies. Here, we focus on a particular semantic knowledge, named **categorical-levels** [12], [13], [42] that consists 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). In practice here we use *categorical*-levels, but *hierarchical*-levels (*i.e.*, levels of the ImageNet [43] or WordNet) or *clustering*-levels could be used (more details in last paragraph of this section). Formally, the starting semantic hierarchy (ImageNet) is a directed acyclic graph $\mathcal{H} = (\mathcal{V}, E)$ consisting of a 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 an initial 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 ImageNet hierarchy $\mathcal{H}$. Hence, consider that all $c_i^{cat_L} \in \mathcal{B}_L^{cat}$ are mapped into certain nodes of the hierarchy $\mathcal{H}$. Our purpose is thus, 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:

$$\begin{aligned} P_{cat_L} : \quad \mathcal{V} \quad & \rightarrow 2^{\mathcal{C}_0^{\mathcal{S}}} \\ c_i^{cat_L} \quad & \mapsto \mathcal{C}_0^{\mathcal{S}} \cap D_{\mathcal{H}}\left(c_i^{cat_L}\right), \end{aligned} \tag{5}$$

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 the P_{cat_L} 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:

$$\begin{aligned} R_{\mathcal{B}_L^{cat}} : \quad 2^{\mathcal{C}_0^{\mathcal{S}}} \quad & \rightarrow \mathcal{B}_L^{cat} \\ \mathcal{C}_i \quad & \mapsto \mathcal{B}_L^{cat} \cap \mathcal{A}_{\mathcal{V}}\left(LCA_{\mathcal{H}}\left(\mathcal{C}_i\right)\right), \end{aligned} \tag{6}$$

where $\mathcal{A}_{\mathcal{V}}(\cdot)$ is a function that outputs, for all c_i categories the set of all its ancestors in $\mathcal{V}$ and is defined as $\mathcal{A}_{\mathcal{V}}(c_i) = \{\delta_{\mathcal{H}}^{j}(c_i)\}_{j=1}^{\infty}$, 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. (5) partitions the set of specific categories into "unnamed" generic categories (*i.e.*, do not have association to a humanly understandable word), Eq. (6) re-labels them into existing (and thus "named") categories of the categorical-level $\mathcal{B}_L^{cat}$. Importantly, images are also automatically re-labeled according the same process than the initial categories they belong. To recap, our ϑ_G grouping SPV (illustrated in Figure 5) is the combination of the partitioning and re-labeling functions.

While the above process is applied on the re-labeling of the *specific* categories into *generic* ones belonging to a categorical-level $\mathcal{B}_L^{cat}$, it can also be applied to re-label into generic categories belonging to **hierarchical-levels** $\mathcal{B}_L^{h}$ of $\mathcal{H}$ or **clustering-levels** (hierarchies constructed based on data, through clustering). For the former, we need the following two assumptions about $\mathcal{H}$: descendants of leaf-nodes are themselves (*i.e.*, $D_{\mathcal{H}}(c_i^0) = c_i^0$) and 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 $c_i^0 \in \mathcal{B}_L^{h}, LCA(c_i^0) = c_i^0$). For the latter, no assumptions are needed and practical details to construct the clustering-hierarchy are given in Sec. 6.1.3.

In summary, in MulDiP-Net, we consider the initial specific SP as well as those obtained by three grouping SPVs, namely, *categorical*, *hierarchical* and *clustering*-based ones. This latter, results in a set of SPs denoted $\mathcal{D}_{\Omega}$.

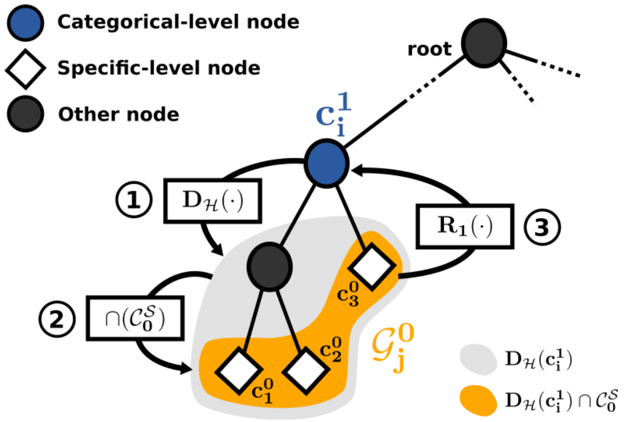

Categorical-level node
Specific-level node
Other node
root
$\mathbf{c_i^1}$
1
$\mathbf{D}_{\mathcal{H}}(\cdot)$
2
$\cap(\mathcal{C}_0^{\mathcal{S}})$
3
$\mathbf{R_1}(\cdot)$
$\mathbf{c_1^0}$
$\mathbf{c_2^0}$
$\mathbf{c_3^0}$
$\mathcal{G}_\mathbf{j}^\mathbf{0}$
$\mathbf{D}_{\mathcal{H}}(\mathbf{c_i^1})$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

Fig. 5. Illustration of our grouping SPV. Given a set of specific categories (leaf nodes of the hierarchy $\mathcal{H}$ 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 grouping SPV ϑ_G consists in three steps: computation of all descendants of c_i^1 according to the $\mathcal{H}$ (1); computation of the descendants that belong to the initial set of categories (2), producing a group $\mathcal{G}_j^0$; and re-labeling of the categories (as well as their images) of the latter group (3) into the categories of $\mathcal{B}_L^{cat}$. The first two points correspond to Eq. (5), while the last one corresponds to Eq. (6).

4.3 MulDiP-Net Training

To describe the training of our Multi Discriminative Problem Network (MulDiP-Net), let us first consider a set of SPs $\mathcal{D}_{\Omega}^{\mathcal{S}} = \{\mathcal{D}_0^{\mathcal{S}}, \mathcal{D}_1^{\mathcal{S}}, \cdots, \mathcal{D}_{\omega}^{\mathcal{S}}\}$, 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 (as depicted in the previous section). MulDiP-Net consists in training one network per SP $\mathcal{D}_{k,k\in[\![0,\omega]\!]}^{\mathcal{S}}$, with a network architecture $\mathcal{N}_k$ and a learning procedure $\mathcal{P}_k$, $\forall k \in [\![0,\omega]\!]$ (*i.e.*, including the *initial* SP). Indeed, each SP $\mathcal{D}_k^{\mathcal{S}} = \{(x_i^k, y_i^k)\}_{i\in[\![0,N_k]\!]}$ consists in a set of N_k images x_i^k labeled among C_k categories c_j^k. This latter forms a set of training-data which is used to train each network. It is worth noting that, we can have as many architectures and learning procedures (one per network) as the number of SPs in $\mathcal{D}_{\Omega}^{\mathcal{S}}$, but here we focus on the special case where they are all the same ($\mathcal{N}_k$=$\mathcal{N}_0$[3], $\mathcal{P}_k$=$\mathcal{P}_0$ for k>0). Note also that, our method does not depend on a particular architecture or learning procedure and can thus directly benefit from the advances in this domain.

Specifically, here we used common CNN architectures (AlexNet [17], VGG [37] and DarkNet [44]) and follow the classical learning procedure for $\mathcal{P}_0$, that is to say, a random initialization of the weights (with a Gaussian distribution), optimizing a softmax loss-function with stochastic gradient descent (SGD). We have as many cost functions to minimize as the number of SPs in $\mathcal{D}_{\Omega}^{\mathcal{S}}$ and each cost function $\mathcal{L}_k(\Theta)$ is minimized *independently*. Since the SPV functions are based on grouping, all the SPs contain the same images as the initial one thus each cost function is minimized on the same set of training images, but with different labels. There is thus no cost in terms of additional data, while the cost to annotate is dramatically reduced since it is reported on each category and not each image. On ILSVRC for instance, it means $1,000$ annotations instead of 1.2 million. At convergence, we obtain a set $\mathcal{N}_{\Omega}^{*} = \{\mathcal{N}_0^{*}, \ldots, \mathcal{N}_{\omega}^{*}\}$ of ω+1 networks. Such an "ensemble-like" learning method could be limited in terms of network capacity, but it exists solutions to decrease this capacity while maintaining high performance, as detailed in Sec. 3.

4.4 MulDiP-Net Representation

Let us consider a MulDiP-Net (*i.e.*, a set $\mathcal{N}_{\Omega}^{*}$ of ω+1 trained networks) and an image $I_i^{\mathcal{T}}$ of a target-task $\mathcal{T}$. To extract the representation $\mathcal{R}_{i,\mathcal{T}}^{\Omega}$ from $I_i^{\mathcal{T}}$, we perform the following two steps: (i) extraction of the features of $I_i^{\mathcal{T}}$ through each subnetwork $\mathcal{N}_k^{*}$ of MulDiP-Net and (ii) normalization and combination of these features into a relevant representation (Fig. 3). For the extraction

[3] In all the document, $\mathcal{N}$ indicates the *architecture* of a network, while $\mathcal{N}^{*}$ indicates a *trained* network.

of the features, let recall that an architecture is a composition of classifiers Ψ and features-extractor Φ (*i.e.*, $\mathcal{N} = \Psi \circ \Phi$). Thus, to get the features from an image, the classifier as well as some of the last layers are discarded, and the K^{th} first layers of network $\mathcal{N}_k^*$ filter the images (*e.g*, $\{conv1, conv2\}$ when K=2 for AlexNet [17]). Hence, the features-extractor function (denoted $\phi_k^K(\cdot)$) outputs, for I_i^τ, a vector (if K points to a fully-connected layer) or a tensor of activations (if K indicates a convolutional layer). In the latter case, the tensor is flattened or pooled in order to get a vector. Formally, the MulDiP-Net representation for the query image I_i^τ is thus computed as:

$$\mathcal{R}_{i,\tau}^{\Omega} = \mathcal{F}_{k\in[\![0,\omega]\!]}\Big(\mathcal{Z}(\phi_k^K(I_i^\tau))\Big), \tag{7}$$

where $\mathcal{F}$ is the fusion operator among the ω+1 input vectors, and $\mathcal{Z}$ is a normalization function that returns normalized representations. In practice for the normalization function, we choose the L-infinite norm L-∞ and for the fusion functions, we used the concatenation $\bigoplus_{k\in[\![0,\omega]\!]}$ (reasons discussed in next paragraph).

Our goal was to combine features trained separately on specific and generic labels in order to learn more universal representations at near-zero cost of annotation, because we hypothesized that their learned features can be complementary. However, because of the different training-data, the networks have different behaviors after training. Indeed, once they are trained, they react differently on the same input images. More precisely, the features learned on specific data tend to fire with higher values than those learned on generic data. Thus, naively concatenating these sub-representations would restrict the fused representation to the activations of the dominating features (those learned with specific labels) with an additional noise from dominated features (those learned with generic labels), which will result to a degradation of performance. Hence, the independent normalization step is crucial, because it aims to homogenize the scales of the sub-representations. The same problem of dominating values has been observed in [38], [45], who solved it with a similar normalization process.

4.5 FSFT as a Dimensionality Reduction Method

The more SPs we consider in MulDiP-Net, the more universal representation we get. However, because of its concatenation fusion, the dimensionality of the fused representation linearly increases with the amount of SPs considered. Also, MulDiP-Net is an ensemble-like method since it is implemented such that it contains as many networks as the number of SP considered. This point is generally translated in terms of the amount of parameters in the final model. Indeed, while the MulDiP-Net aims at building more universal representations, it contains ω+1 *times* more parameters than a standard network. In summary, the actual form of MulDiP-Net faces two problems: (ii) the high capacity of its resulting model; and (ii) the high dimensionality of its resulting representation. To alleviate these drawbacks, we proposed to rely on the FSFT method (introduced in Sec. 3) and even more, on its use as a *dimensionality reduction* method. To do so, we roughly re-train each of the subnetworks of MulDiP-Net on the same initial SP, but with the last layers (on which FSFT focuses the re-training) containing much *less* neurons. Compared to the classical FSFT, the "FSFT as a Dimensionality Reduction method" (**FSFT-DR**) results in: (i) a final model that has much lower capacity and (ii) a representation with a lower dimensionality, at a near-zero decrease of performance, but that still significantly increases the performances compared to the initial network (experiments of Sec. 6.3). Obviously such a behavior is also observed on the MulDiP+FSFT-DR (FSFT-DR applied on each subnetwork of MulDiP-Net).

Criterion	Avg	RG	VDC	BC	mNRG
Coherent aggr.		✓	✓	✓	✓
Significance			✓		
Merit bonus			✓		✓
Penalty malus					✓
Penalty for damage	✓	✓		✓	✓
Indep. to outliers				✓	✓
Indep. to reference	✓			✓	
Time consistency	✓	✓	✓		✓

TABLE 1
Comparison of all the universality evaluation-metrics – Avg baseline, RG [7], VDC [5] and ours, namely BC and mNRG – according all the criteria mentioned (and highlighted in bold) in Sec. 5.

More formally, let us consider a features-extractor ϕ_k trained on a certain SP noted $\mathcal{D}_k^S$ with a network architecture $\mathcal{N}_k$ that has a penultimate layer of dimensionality $dim(\phi_k(\cdot))$=n. FSFT-DR consists to re-train (as depicted in Eq. 2 about the FSFT re-training) the last layers of ϕ_k (which results in a new features-extractor denoted ϕ_k') on the same source-problem $\mathcal{D}_k^S$ but with a *different* network-architecture $\mathcal{N}_k'$ that has a penultimate layer of a dimensionality $dim(\phi_k'(\cdot))$=n' *lower* than those of $\mathcal{N}_k$ (*i.e.*, n'<n). Note that in the classical formulation of FSFT n'=n. Note also that, while by construction FSFT-DR reduces the dimensionality, it also reduces the number of parameters of the final network $\mathcal{N}_k'$ (since $\mathcal{N}_k'$ contains less neurons and thus less weights than $\mathcal{N}_k$). It is all the more true since FSFT-DR is applied on the last layers, which are generally fully-connected ones and thus contain the majority of the parameters of the CNN. For instance, AlexNet contains 62.35 millions parameters with 3.75 millions in the convolutional layers and 58.6 millions in the fully-connected (FC) ones (about 94% of the total). Regarding the application of FSFT-DR on the MulDiP-Net method, let first consider a given set $\mathcal{N}_\Omega^* = \{\mathcal{N}_0^*, \ldots, \mathcal{N}_\omega^*\}$ of ω+1 trained networks. FSFT-DR is applied on each subnetwork $\mathcal{N}_{k\in[\![0,\omega]\!]}^*$ by setting the last layer of ϕ_k' to be of size $n' = \lceil n/(\omega+1)\rceil$, with n being the dimensionality of the original sub-representation ϕ_k. The rest of the pipeline remains the same than in MulDiP-Net, that is, extraction of the penultimate layer from each subnetwork and merging them after independent normalization. In summary, compared to MulDiP-Net, MulDiP+FSFT-DR (i) contains much less parameters and (ii) results into a smaller representation than the classical MulDiP-Net, while being much more performing.

5 Evaluation of Universalizing Methods

Universality is motivated by the claim of Atkinson's cognitive study [1]. Inspired by this, authors of [4], [5], [6] evaluated the universality of representations by their ability to simultaneously cover a large range of domains, like objects, faces, animals, etc. However, their evaluation scheme does *not* completely match with [1] since learning and testing are conducted on the *same problem* and only the image distribution differs (learn on train images and evaluate on test ones). However, in [1], the terms "re-used later in life" mean that universality implies to work well on *many* but *different* problems than those used to learn the representation. Hence, in this paper, we propose to consider this through the scheme of transfer-learning (TL) that naturally fits with it. Indeed, in TL, the *source-task* is used to learn the representation (analogically, the universal one developed in the early years) and the *target-tasks* are used to evaluate it (analogically, the problems solved later in life). Moreover, to better match the claim of [1] (re-use "as-is"), we propose to *not* modify the representation (do not fine-tune it) for each target-task, but rather learn a simple task-predictor *on top of* it. Nevertheless, when humans develop their representation, they do *not* have access to future problems, thus the target-tasks should not be available during the learning of the representation. What makes our evaluation framework unique is that it lies in a TL scheme, closer to [1] than those of [4], [5], [6] which lies in a classical end-to-end learning scheme.

Evaluating universality requires to aggregate the scores of a method on *multiple* target-problems. This step is not straightforward, since metrics of each problem could be *different* (*e.g.*, error-rate, mAP, F1-score) or even be increasing (*e.g.*, recall) or decreasing (*e.g.*, median rank). Thus, naively averaging the scores would result into incoherences. To get **coherent-aggregation**, Subramanian *et al.* [7] proposed to average the gain over the scores of a reference universal representation across the set of multiple problems: relative gain (**RG**) computed as: $U_i^{RG} = \frac{1}{P}\sum_{j=1}^{P} s_j^{ref} - s_j^i$, with s_j^{ref} and s_j^i being respectively the performance of a reference method M_{ref} and the method to evaluate M_i on the problem P_j. Such aggregation is coherent, but **depends on a reference method** that needs to be arbitrarily and manually chosen. Rebuffi *et al.* [5] went further by identifying one additional criterion for universality: the metric should better rewards significant gain compared to the reference scores (called **significance** criterion). To do so, they proposed the Visual Decathlon Challenge (**VDC**), which is computed as: $U_i^{VDC} = \sum_{j=1}^{P} \alpha_j \text{max}\{0, E_j^{max} - E_j\}^{\gamma_j}$, with E_j^{max} and E_j respectively being the error-rate of the actual method M_i and the best reachable one on the problem P_j, α_j is just a normalizing factor, and $\gamma_j \geqslant 1$ models the significance.

In addition to the properties already put on evidence, we introduce four more criteria. The first criteria (**merit-bonus**) indicates that the metric should reward proportionally with the score of the reference method. For instance a jump of 1% is more rewarding if it starts from a reference at 90% than one at 10%. The second criteria (**penalty for damage**) is motivated by the fact that a method could improve the performance on some tasks but *decrease* it on others. In such case, the metric should penalize them. The third proposed criteria (**penalty malus**) lies with the fact that, the aggregated improvement (over the reference) of some methods, could be lower than the aggregated decrease. Such method should also be penalized. A last important criterion is that the metric should be **independent to outlier methods**, in the sense that it should detect methods that are well suited to a given benchmark (gain a lot of points) that could compensate (when aggregated) a insignificant gain on the other benchmarks.

To respect these constraints, we propose the median normalized relative gain (**mNRG**): $U_i^{mNRG} = \underset{j \in [\![1,P]\!]}{\text{median}}(s_j - s_j^{ref})/(s_j^{max} - s_j^{ref})$. The numerator is the simple RG metric, thus it respects the same criteria (coherent aggregation and penalty for damage because negative scores are allowed). The denominator acts as a normalization and naturally handles the merit-bonus and penalty malus criteria. Finally, instead of an average to aggregate, we used the median to make our metric independent to outliers.

Note that in VDC, negative values are not allowed (because of the max), thus it does not respect the penalty for damage nor the penalty malus. In addition, the significance criterion is modeled through a power γ, which not only prevents the merit-bonus, but even exhibits the inverse behavior. Indeed, VDC will reward more a method that gains 3% starting from a reference at 10%, than one that gains 2.5% starting from 90%.

Finally, none of the above metrics is independent to a reference method, thus we propose an alternative metric that we call Borda Count (**BC**), which is based on a voting method and the ordinal scale, with each benchmark considered as an independent voter. First, each voter can use its own measure to estimate the performance of the methods, which provides coherent aggregation. Second, 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 can be a more reliable information than the average difference of scores (as long as these score are actually comparable). Note that, BC lacks the **consistency with time** since its score differs with the chosen comparison methods. Formally, let us consider M methods to rank, relying on the information provided by P problems. Each method is then ranked according to each problem, resulting into a rank r_i^j with $M \in [\![1, i]\!]$ and $P \in [\![1, j]\!]$. It is converted into a score $M - r_i^j$ that is itself averaged to give the final score of the method: $S_i = \sum_{j=1}^{P} M - r_i^j$.

All the metrics are compared in Table 1 according to all the mentioned criteria. None of them address all the criteria but the proposed mNRG is the most complete of them.

6 Experimental Results

6.1 Experimental Settings

6.1.1 Transfer-Learning Protocol

As mentioned above, the evaluation of all the methods is carried in a transfer-learning scheme on multiple target-problems. The methods are trained with standard architectures: AlexNet [17] (default), VGG [37] and DarkNet [44]). By default, we train on ILSVRC* (see Sec. 6.1.2) because it is smaller than ILSVRC, making the process faster. The network on the source-problem is used to extract the image signatures on the target-problem. Here we focus on the classification task for the target-problems and learn each of their classes with a *one-vs-all* linear SVM classifier. Note that, as mentioned in Sec. 5, to match the claim of [1], there is no fine-tuning. The cost parameter of the classifiers is optimized on each dataset through cross-validation with the usual train/val splits. The performances of each target-problem are evaluated with standard splittings and metrics, namely the mean Average Precision (mAP) for multi-label datasets and Accuracy (Acc.) for mono-label ones. To evaluate the universality, we use the proposed **mNRG** evaluation-metric (Sec. 5).

6.1.2 Source and Target-datasets

For the source-problem, we used two subsets of ImageNet [46]: ILSVRC and ILSVRC* that contains half of the former. The characteristics of the datasets are presented on top of Table 2. Regarding the target-problems, we used ten image classification benchmarks from different visual domains (actions, scenes, objects, birds, plants, etc.). Specifically, five of them contain

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 2
Source-datasets (top) and target datasets (bottom) used in this paper. The columns report some characteristics: (1) images-domain; (2) number of categories; (3): multiple categories per image (✓) or not (✗); (4) number of training samples; (5): number of test samples; and (6) evaluation-metric (**Acc.** or **mAP**).

coarse categories – Pascal VOC 2007 (**VOC07**) [47], Pascal VOC 2012 (**VOC12**) [48], Caltech-101 (**CA101**) [49], Caltech-256 (**CA256**) [50] and Nus-Wide Objects (**NWO**) [51] –, three contain fine categories – Stanford Cars (**CARS**) [52], CUB-200 Birds (**CUB**) [53] and Flowers-102 (**FLO**) [54] –, one contain scenes – MIT Indoor 67 (**MIT67**) [55] – and one contain actions – Stanford Actions (**stACT**) [56]. Their characteristics are presented at the bottom of Table 2. For all the benchmarks, we follow standard protocols (*i.e.*, common splits and metrics). For VOC12 and stCAR, we used the official evaluation-servers.

6.1.3 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 the two first methods, we used the ImageNet hierarchy in the functions of partitioning (Eq. (5)) and re-labeling (Eq. (6)). Practically, in the first method, a part of the categorical-level categories of ILSVRC is obtained from the list released in [46] and the other part is re-labeled by ourselves as depicted in [11]. This latter results into 480 generic categories for the $1,000$ specific ones of ILSVRC (200 generic for the 483 specific of ILVRC*). For the hierarchical-method, we follow the bottom-up approach of [22] and re-labeled the categories to higher levels of the ImageNet hierarchy. For the clustering method, we follow [23] and clustered the data of the categories with a Kmeans algorithm (K from 50 to 300 with steps of 50). Regarding the extraction and combination of the features (Eq. 7), we used one label-set per grouping SPV (*i.e.*, basic-level, 7^{th} hierarchical and the clustering with K=100). 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. Regarding the normalization step before combining the representations, we used the infinite-norm (L_∞). In the FSFT methods, we choose the following settings: L=6, meaning that we focus the retraining on the two last layers; η_2=10^{-2}, as the learning-rate used to train the original network; and α=0.1, meaning that we train the last layers 10 times faster than the firsts. Note that, for FSFT we also use the penultimate layer as representation of the target-images.

6.2 Comparison to the State-Of-The-Art

We compare the proposed FSFT and MulDiP+FSFT to state-of-the-art methods that could be used as universalizing methods:

- **REFERENCE** [17]: A CNN trained on the initial source-problem, that contains *specific* categories (here 483). Since it is a classical method that works quite well on many problems, we use it as *reference* to evaluate universality of other methods.
- $\mathbf{SPV}_{\mathbf{A}}^{\mathbf{spe}}$ [4], [15], [16]: A method that consists in a *adding* SPV followed by the training on the obtained SP. Specifically, we added 100 *specific* categories (randomly obtained from the leaf nodes of ImageNet) with their 100K images. Simply said, the method is trains a network on 583 specific classes.
- $\mathbf{SPV}_{\mathbf{A}}^{\mathbf{gen}}$ [17], [18], [19]: Same as the previous method but with a *generic* adding SPV that adds 100 *generic* categories (with their 100K images). This results in training a network on 583 *specific and generic* classes. The generic ones were obtained from random internal-nodes of the ImageNet hierarchy.
- **WhatMakes** [22]: A *grouping* SPV followed by the training of a network on the obtained SP. Specifically, the grouping SPV corresponds to a relabeling of specific categories into internal *hierarchical-levels* of the ImageNet hierarchy. We performed it for all the levels and report the results for the best one (6^{th} level starting from the leaf-node's one).
- **AMECON** [23]: Similar to the previous method but differs by its kind of grouping SPV. Indeed, the grouping is performed by *clustering*. Specifically, all the images of each specific categories are used to compute the mean features (obtained through the $fc7$ layer of the pre-trained reference network) for the categories. Then, a Kmeans algorithm is used to cluster this set of category mean features. We applied this method with different amount of clusters (K from 50 to 300 with a step of 50) and report the best results (K=100).
- **ISM** [28]: This method trains an ensemble-model with N networks, one for each SP obtained from a *splitting* SPV. Here, we applied the method with *half* splitting (we split the initial SP in two balanced subsets). Note that, we chose two subsets to limit the method to a maximum of two networks for fair comparisons. Once the networks trained, we normalize and concatenate the features extracted from each of them.
- **GrowBrain-WA** [38]: This recent method consists to fine-tune a trained network on the same source-problem it was trained originally, by growing the network capacity (wider or deeper). The best setting is the width augmented (WA) growing that consists to add $2,048$ neurons to the $fc7$ layer. We also implemented their normalization and scaling step for the new and old layers, because they are crucial to make this method performing. The final representation corresponds to the $6,192$-dimensional $fc7$ layer. **GrowBrain-RWA** is an extended version that performs a recursive growing of the network capacity. The best setting they report is to add $1,024$ neurons on the $fc6$ layer and $2,048$ on the $fc7$ one.
- **MuCaLe-Net** [11]: 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 one according categorical-levels. It results in a $8,192$-dimensional representation.

[5], [6] explicitly tackled the universal representations problem and could be used for comparison. However, it is important

Method	**VOC07**	**VOC12**	**CA101**	**CA256**	**NWO**	**MIT67**	**stACT**	**CUB**	**stCA**	**FLO**	**mNRG**
	mAP	mAP	Acc.	Acc.	mAP	Acc.	Acc.	Acc.	Acc.	Acc.	
REFERENCE	66.8	67.3	71.1	53.2	52.5	36.0	44.3	36.1	14.4	50.5	0.0
SPV$^{\text{spe}}_{\text{A}}$ [4], [15], [16]	66.6	67.5	74.7	54.7	53.2	37.4	45.1	36.0	13.7	51.9	+1.5
SPV$^{\text{gen}}_{\text{G}}$ [17], [18], [19]	67.7	68.1	73.0	54.3	50.5	37.1	44.9	36.8	14.6	50.3	+1.4
AMECON [23]	61.1	62.1	58.7	40.6	45.8	24.3	32.7	26.1	13.1	36.4	-17.7
WhatMakes [22]	64.0	62.7	69.4	50.1	45.6	33.7	41.9	15.0	12.5	42.8	-7.5
ISM [28]	62.5	65.4	68.8	50.7	28.5	37.9	42.6	34.0	13.3	50.0	-4.3
GrowBrain-WA [38]	68.4	68.3	73.1	54.7	49.3	38.4	46.5	37.5	14.7	54.8	+3.5
GrowBrain-RWA [38]	69.1	69.0	74.8	55.9	50.4	40.0	48.4	38.6	14.8	56.1	+6.0
MuCaLe-Net [11]	69.5	69.8	76.0	56.8	**54.7**	41.3	48.5	35.6	15.7	54.8	+7.7
FSFT (Ours)	67.5	67.4	73.9	55.0	44.6	40.4	47.1	**38.7**	15.8	56.8	+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 3
Comparison of our methods (bottom) to **state-of-the-art universalizing-methods** (top). The comparison is carried in a transfer-learning scheme on ten target-datasets. Methods are compared in terms of their individual scores on each benchmark (with standard metrics) and especially their aggregated scores, with our nMRG (blue scores in last column). The universalizing methods are compared to a reference one for which we colored its scores in red. In each column, we highlight the highest score in bold, and the second one is underlined. All the methods have been learned with the same AlexNet architecture on the same initial SP (ILSVRC*).

to note that their goal was to improve universality by *adding data from multiple domains*, while our goal is to improve universality *from a fixed set of data* (one domain), making their methods not comparable to ours, since they use *more* annotated data.

As depicted in Sec. 5, the methods are evaluated in terms of the proposed mNRG score, in a transfer-learning scheme on a set of ten target-datasets from different domains. The results of the comparison are presented in Table 3. As expected, the methods that consist to add images and their annotations (SPV$_A^{spe}$ and SPV$_A^{gen}$) as well as those that increase the capacity of the network (GrowBrain) aim to learn more universal representation compared to the reference method (positive mNRG score). Surprisingly, the ISM method is not as performing as reported in their context. This might be due to the fact that it is designed for very large source-problems, and the half-million images used here are not sufficient, highlighting a clear limitation of their method. The same behavior can be observed with AMECON and WhatMakes. The former could be because the specific categories we used (leaf node of ImageNet) are not as specific as the ones (captions) used in their paper. The latter was rather a study that 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-tasks distributed among three domains (general objects, actions and scenes). However, we clearly observe here that on more domains, and especially fine-grained objects, such behavior is not observed anymore (highly negative mNRG score). Finally, we observe that our MulDiP+FSFT method significantly performs better than all other methods (highest mNRG score), meaning that it is clearly the most performing universalizing method. Note that, it also significantly outperforms MuCaLe-Net (by 2 points of mNRG), which clearly highlight the interest of combining MulDiP-Net with the proposed FSFT method.

Another salient observation is that our FSFT only is quite powerful, especially because it outperforms the methods that consist to add data and their annotations (SPV$_A^{spe}$ and SPV$_A^{gen}$) as well as one that increase the network-capacity (GrowBrain-WA). It is worth noting that FSFT does need more data neither more capacity. Thus, it increases universality at zero cost of capacity and annotation, which is quite promising.

6.3 Comparison with Baseline Methods

Here, we take further experiments to analyse the performances achieved by the proposed methods (MulDiP-Net and FSFT), through their comparison to several baseline-methods. All the baselines are described below, illustrated in Fig. 6 and the results are reported in Table 4. Note that, in this section (and supplementary), the comparisons are conducted on eight of the ten target-datasets presented in Sec. 6.1 (VOC12 and stCAR removed because evaluation-servers limited to 1 run per day needed).

We first assess whether the gain of universality obtained by our MulDiP-Net method is caused by the ensemble-model component. To do so, we compare it to a baseline that consists in an ensemble-model with two network trained on the same specific SP but with different random initializations of the weights (**Ensemble**). While Ensemble is significantly increasing universality compared to the reference, our MulDiP-Net is provides much better results. This latter, means that the performances of our method does not come from the ensemble-model aspect only, but also by the combination of features learned with data labeled among generic and specific categories. Hence, another baseline is to compare our method without the ensemble-model, but by jointly training a network on the two SPs (generic and specific). Indeed, we tested three variants: (i) training the set of SPs with a sum of softmax losses, *i.e.*, one for each SP (**Multi-Task**); (ii) training the set of SPs with a multi-label loss layer (*e.g.*, hinge loss), where the labels for each image contain both annotations, namely generic and specific (**Multi-Label**); and (iii) recursively training the SPs by training the network on the generic SP, *then* continuing the training on the specific SP (**Recursive**). Globally, the first two baselines, which belongs to the joint-training approach, strongly hurts performance compared to the reference method. The Recursive method is almost as performing as the reference one and we assume this is due to catastrophic forgetting [57], *i.e.*, when the network is trained on the second specific problem, it forgets the features learned on the former generic one. In summary, the latter baselines clearly demonstrate the utility of the independent-learning (in our method) compared to the joint one. In other words, the growing capacity of the ensemble-model aspect is crucial to make our

Method	VOC07	CA101	CA256	NWO	MIT67	stACT	CUB	FLO	mNRG
	mAP	Acc.	Acc.	mAP	Acc.	Acc.	Acc.	Acc.	
REFERENCE	66.8	71.1	53.2	52.5	36.0	44.3	36.1	50.5	0.0
Ensemble	67.8	72.2	54.5	52.0	37.2	45.0	34.7	51.8	+2.3
Multi-Task	61.5	61.8	45.4	49.4	30.7	36.4	25.6	38.7	-16.2
Multi-Label	44.7	46.8	26.4	25.1	27.2	28.0	15.2	38.1	-45.0
Recursive	65.3	68.6	50.8	52.4	33.4	50.8	29.4	45.5	-4.8
MulDiP-Net*	69.5	76.0	56.8	54.7	41.3	48.5	35.6	54.8	+7.9
FrST	62.3	64.3	47.3	50.0	30.9	38.4	29.3	41.1	-11.6
SFT	67.0	71.5	52.5	49.8	36.2	44.0	36.4	50.1	-0.1
FSFT$_{\mathbf{DR}}$*	67.6	73.3	54.8	47.2	37.9	46.9	38.2	56.3	+3.4
FSFT*	67.5	73.9	55.0	44.6	40.4	47.1	**38.7**	56.8	+4.5
MulDiP+FSFT$_{\mathbf{DR}}$*	**69.8**	76.6	57.3	**49.9**	41.6	48.7	38.0	59.2	+8.8
MulDiP+FSFT*	**69.8**	**77.5**	**58.3**	47.9	**43.7**	50.2	37.4	**59.7**	**+10.7**

TABLE 4
Comparison of our universalizing-methods with **baseline methods**. Methods are compared in terms of their mNRG score (last column in blue). All the methods have been learned with the same AlexNet architecture on the same initial SP (ILSVRC*). Methods marked with * are ours.

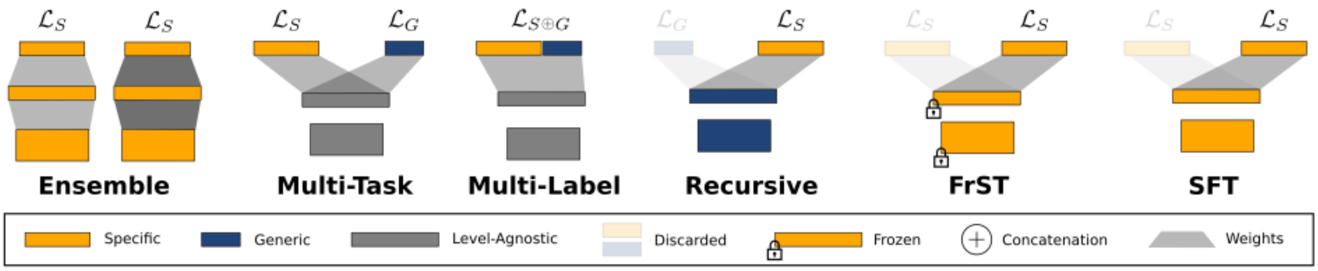


Fig. 6. Illustration of the different baseline methods.

method benefiting from multiple SPs.

For our FSFT method, we mainly compare it to the two methods (namely, Frozen Self Training (**FrST**) and Self Fine-Tuning (**SFT**)) studied by [14], that also re-train a network on the *same* problem. More precisely, **FrST** consists in retraining the last layers only with the previous layers being "frozen", while **FST** re-trains all the layers with the same learning-rate. As highlighted in [14], we observe a performance drop of FrST compared to the reference. As mentioned, in Sec. 3, this is due to the fragile co-adaptation neurons learned in the original network. A slight drop of performance is also observed for FST, meaning that fine-tuning does not always recovers all the co-adapted neurons. In contrast, our FSFT method increases performance, even when we compact its representation to $2,048$ (**FSFT$_{\mathbf{DR}}$**), clearly highlighting its capacity to recover co-adapted neurons of the original network and even the training of others.

Finally, we also assess the utility of combining MulDiP-Net with FSFT and even considering FSFT as a dimensionality reduction method (**FSFT$_{\mathbf{DR}}$**), compared to the reference and especially the MulDiP-Net method. As shown above, MulDiP+FSFT performs much better than MulDiP-Net, but more surprisingly, when MulDiP-Net is combined with FSFT$_{DR}$, it is almost as performing as the former, at a much lower capacity. Hence, by using FSFT as a dimensionality reduction method (FSFT$_{DR}$), we not only get a jump of performance (compared to MulDiP-Net), but also significantly alleviate the higher capacity produced by the ensemble-model aspect of MulDiP-Net.

Let note that supplementary materials are available and roughly contain: (i) a comparison of our methods to the state-of-the-art according all the universality evaluation-metrics of the literature; (ii) an evaluation of the impact of more and different grouping SPVs used in our MulDiP-Net; and (iii) the evaluation of MulDiP-Net with more training data and deeper architectures.

7 Conclusion

In this paper, we proposed four contributions: (i) a new challenge of learning more universal representations from a fixed set of data (domains and tasks); (ii) the evaluation of universality in a more suitable scheme (transfer-learning), as well as a new metric respecting most of the highlighted desirable criteria; (iii) a new method based on the re-training of networks, by focusing the training on some parameters; and finally (iv) a new method based on a general formalism of source problem variation and training of multiple networks. We demonstrated the effectiveness of our universalizing methods, in a transfer-learning scheme, through our evaluation-metric, on ten target-datasets from different domains. An in-depth analysis has also been conducted to highlight some important insights of our methods. We hope that our contributions will further support the creation of other methods to get more universal representations and open doors for many less explored aspects of transfer-learning such as, learning the source-problem, using Human knowledge or even through more realistic multi-modal and dynamic environments such as HOME [58].

References

[1] J. Atkinson, *The developing visual brain*. Oxford University Press UK, 2002.

[2] I. Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *CVPR*, 2017.

[3] J. Wang, O. Russakovsky, and D. Ramanan, “The more you look, the more you see: towards general object understanding through recursive refinement,” in *WACV*, 2018.

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

[5] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, “Learning multiple visual domains with residual adapters,” in *NIPS*, 2017.

[6] ——, “Efficient parametrization of multi-domain deep neural networks,” in *CVPR*, ser. CVPR, 2018.

[7] S. Subramanian, A. Trischler, Y. Bengio, and C. J. Pal, “Learning general purpose distributed sentence representations via large scale multi-task learning,” in *ICLR*, ser. ICLR, 2018.

[8] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, “Very deep convolutional networks for text classification,” in *Association for Computational Linguistics*, ser. ACL, 2017.

[9] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, “Supervised learning of universal sentence representations from natural language inference data,” ser. EMNLP, 2017.

[10] 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.

[11] Y. Tamaazousti, H. Le Borgne, and C. Hudelot, “Mucale-net: Multi categorical-level networks to generate more discriminating features,” in *CVPR*, 2017.

[12] P. Jolicoeur, M. A. Gluck, and S. M. Kosslyn, “Pictures and names: Making the connection,” *Cognitive Psychology*, 1984.

[13] E. Rosch, “Principles of categorization,” *Cognition and Categorization*, 1978.

[14] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *NIPS*, 2014.

[15] H. Azizpour, A. Razavian, J. Sullivan, A. Maki, and S. Carlsson, “Factors of transferability for a generic convnet representation,” *PAMI*, 2015.

[16] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, “Learning deep features for scene recognition using places database,” in *NIPS*, 2014.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.

[18] P. Mettes, D. Koelma, and C. G. M. Snoek, “The imagenet shuffle: Reorganized pre-training for video event detection,” in *ICMR*, 2016.

[19] Y. Tamaazousti, H. Le Borgne, A. Popescu, E. Gadeski, A. Ginsca, and C. Hudelot, “Vision-language integration using constrained local semantic features,” *CVIU*, 2017.

[20] P. D. Vo, A. Ginsca, H. Le Borgne, and A. Popescu, “Harnessing noisy web images for deep representation,” *CVIU*, 2017.

[21] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache, “Learning visual features from large weakly supervised data,” in *ECCV*, 2016.

[22] M. Huh, P. Agrawal, and A. A. Efros, “What makes imagenet good for transfer learning?” *arXiv:1608.08614*, 2016.

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

[24] K. Ahmed, M. H. Baig, and L. Torresani, “Network of experts for large-scale image categorization,” in *ECCV*, 2016.

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

[26] V. N. Murthy, V. Singh, T. Chen, R. Manmatha, and D. Comaniciu, “Deep decision network for multi-class image classification,” in *CVPR*, 2016.

[27] W. Ouyang, X. Wang, C. Zhang, and X. Yang, “Factors in finetuning deep model for object detection with long-tail distribution,” in *CVPR*, 2016.

[28] Y. Wu, J. Li, Y. Kong, and Y. Fu, “Deep convolutional neural network with independent softmax for large scale face recognition,” in *ACM*, 2016.

[29] 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 *ICCV*, 2015.

[30] A. Conneau and D. Kiela, “Senteval: An evaluation toolkit for universal sentence representations,” ser. LREC, 2018.

[31] 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 *CVPR*, 2012.

[32] A. Mathews, L. Xie, and X. He, “Choosing basic-level concept names using visual and language context,” in *WACV*, 2015.

[33] V. Ordonez, W. Liu, J. Deng, Y. Choi, A. C. Berg, and T. L. Berg, “Predicting entry-level categories,” *IJCV*, 2015.

[34] Y. Tamaazousti, H. Le Borgne, and C. Hudelot, “Diverse concept-level features for multi-object classification,” in *ICMR*, 2016.

[35] A. L. Ginsca, A. Popescu, H. Le Borgne, N. Ballas, P. Vo, and I. Kanellos, “Large-scale image mining with flickr groups,” in *MM*, 2015.

[36] Y. Tamaazousti, H. Le Borgne, and A. Popescu, “Constrained local enhancement of semantic features by content-based sparsity,” in *ICMR*, 2016.

[37] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *ICLR*, 2015.

[38] Y.-X. Wang, D. Ramanan, and M. Hebert, “Growing a brain: Fine-tuning by increasing model capacity,” in *CVPR*, 2017.

[39] L. Herranz, S. Jiang, and X. Li, “Scene recognition with cnns: objects, scales and dataset bias,” in *CVPR*, 2016.

[40] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” in *ICLR*, 2014.

[41] 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 *CVPR*, ser. CVPR, 2017.

[42] J. W. Tanaka and M. Taylor, “Object categories and expertise: Is the basic level in the eye of the beholder?” *Cognitive Psychology*, 1991.

[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR*, 2009.

[44] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” in *CVPR*, 2017.

[45] W. Liu, A. Rabinovich, and A. C. Berg, “Parsenet: Looking wider to see better,” in *ICLR workshop*, 2016.

[46] 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,” *IJCV*, 2015.

[47] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge,” *IJCV*, 2010.

[48] ——, “The pascal visual object classes challenge 2012,” 2012.

[49] L. Fei-Fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” *PAMI*, 2006.

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

[51] 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 *CIVR*, 2009.

[52] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, “3d object representations for fine-grained categorization,” in *ICCV Workshop*, 2013.

[53] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, “The caltech-ucsd birds-200-2011 dataset,” 2011.

[54] M.-E. Nilsback and A. Zisserman, “Automated flower classification over a large number of classes,” in *IEEE Computer Vision, Graphics & Image Processing*, 2008.

[55] A. Quattoni and A. Torralba, “Recognizing indoor scenes,” in *CVPR*, 2009.

[56] 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 *ICCV*, 2011.

[57] R. M. French, “Catastrophic forgetting in connectionist networks,” *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

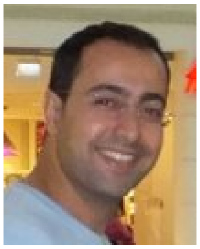
[58] S. Brodeur, E. Perez, A. Anand, F. Golemo, L. Celotti, F. Strub, J. Rouat, H. Larochelle, and A. Courville, “Home: A household multimodal environment,” *arXiv preprint arXiv:1711.11017*, 2017.

Youssef Tamaazousti is a Postdoctoral Research Associate at the Computer Science and Artificial Intelligence Lab (CSAIL) of Massachusetts Institute of Technology (MIT). He is also attached to the Qatar Computing Research Institute (QCRI). Previously, he received his PhD from CentraleSupelec and the computer-vision laboratory of CEA LIST, in 2018. His research interests span the areas of visual recognition, neural networks and representations-learning from one or multiple modalities.

Hervé Le Borgne is a researcher at the CEA LIST since 2006, carrying out research on computer vision and multimedia mining. Previously, he received his PhD from the INP Grenoble in 2004 and worked as a post-doc at Dublin City university until 2006. He published more than 40 articles in international conferences and journals and is co-inventor of seven patents. His research interest deals with information extraction from visual and textual documents, and relating this information to the human user needs.

Céline Hudelot is a Full Professor at the Mathematics and Interaction (MICS) with Computer Science Laboratory of Centrale Supelec (University of Paris-Saclay). She obtained her Ph.D in electrical and computer engineering from INRIA and the University of Nice Sophia Antipolis in 2005 and her Habilitation from Université Paris-Sud in 2014. She is in charge of the research axis on formal methods for semantic multimedia understanding in the MICS Laboratory. Her research interests include knowledge and ontological engineering for semantic image analysis, 2D and 3D image processing, information fusion, formal logics, graph-based representation and reasoning, spatial reasoning and machine learning. She was the main co-advisors of four PhD students and is advising three PhD students.

Mohamed-El-Amine Seddik received a Master of Engineering in Data Science from Institut Mines-Telecom de Lille (with the final year completed at Telecom ParisTech) and a Master Degree in Vision and Machine Learning from ENS Cachan in 2017. He is currently a PhD student in the computer vision laboratory of CEA LIST, interested in random matrix theory for machine learning and scene understanding.



Mohamed Tamaazousti received his Master's Degree in applied mathematical from the University of Orléans in 2009 and the Ph.D. degree in computer vision from the University Blaise Pascal in 2013. He is currently a permanent researcher at CEA LIST. His main research interests include structure from motion for rigid scenes, real time vision-based localization and reconstruction (SLAM) for autonomous system. He is also interested in augmented and diminished reality applications.