Interpretable Representation Learning for Visual Intelligence

by

Bolei Zhou

B.Eng., Shanghai Jiao Tong University (2010)
M.Phil., The Chinese University of Hong Kong (2012)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Author .....
Department of Electrical Engineering and Computer Science
May 17, 2018

Certified by .....
Antonio Torralba
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Interpretable Representation Learning for Visual Intelligence

by

Bolei Zhou

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Recent progress of deep neural networks in computer vision and machine learning has enabled transformative applications across robotics, healthcare, and security. However, despite the superior performance of the deep neural networks, it remains challenging to understand their inner workings and explain their output predictions. This thesis investigates several novel approaches for opening up the "black box" of neural networks used in visual recognition tasks and understanding their inner working mechanism. I first show that objects and other meaningful concepts emerge as a consequence of recognizing scenes. A network dissection approach is further introduced to automatically identify the internal units as the emergent concept detectors and quantify their interpretability. Then I describe an approach that can efficiently explain the output prediction for any given image. It sheds light on the decision-making process of the networks and why the predictions succeed or fail. Finally, I show some ongoing efforts toward learning efficient and interpretable deep representations for video event understanding and some future directions.

Thesis Supervisor: Antonio Torralba
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

The 5-year Ph.D. study at MIT was an incredible experience. There are several people that I wish to acknowledge who not only made this thesis work well accomplished, but also brought unlimited amount of joys and encouragement to this amazing scientifc journey.

I would first like to thank my advisor, Antonio Torralba, for his enthusiasm, inspiration, and guide throughout my Ph.D. study. 10 years ago when I was still a sophomore, the seminal work of Spatial Envelope done by Antonio and Aude was among the very first few vision papers I read, which has influenced me a lot. Now during this PhD study in which we collaborated on a lot of exciting projects, his sharp intuition and endless passion motivate me to dig deeper into the problems and discover something new.

I would also like to thank my thesis committee, Aude Oliva and Bill Freeman, for their kind support and advice on my research. Various key projects in this thesis have been collaborated closely with Aude. Aude's insights and enthusiasm on cognition and human visual system give me a brand new and inspiring perspective on the computer vision problems I work on. I am also amazed by the projects done by Bill, from the visual microphone to the work of turning corners into cameras. Bill's eye for the fundamentals of computer vision and the signal processing approach provides alternatives other than the dominating deep learning methods.

Xiaoou Tang and Xiaogang Wang were my master thesis advisor and collaborator at CUHK for which I am very thankful. They opened the door of computer vision for me at the very begining and encouraged me to further my research at MIT . I couldn't achieve all of this without their support. I also would like to say thanks to my previous mentors, James Hayes, Rob Fergus, Larry Zitnick, Yuandong Tian, John Fisher, who gave me insightful suggestions at different life moments.

These dear labmates at Torralba Lab and Oliva Lab made my graduate school journey more colorful and fulfilling: David Bau, Hang Zhao, Xavier Puig, Aditya Khosla, Adria Recasens, Agata Lapedriza, Wei-Chiu Ma, Yunzhu Li, Carl Vondrick, Zoya Bylinskii, Jimmy Wu, Joseph Lim, Tomasz Malisiewics, Alex Andonian, Yalda Mohsenzadeh, Kandan Ramakrishnan, Caitlin Mullin, Mathew Monfort, Santani Teng. The CSAIL Vision lab pro-

vided a friendly and inclusive environment , I am grateful for the accompany of the CSAIL Vision friends: Donglai Wei, Tianfan Xue, Jiajun Wu, Chris Dean, Philip Isola, Dilip Krishnan, George Chen, Hossein Mobahi, Jianxiong Xiao, Julian Straub, Vadim Smolyakov, Randi Cabezas, Katie Bouman, etc. I also would like to thank administrative and support staffs Fern Keniston and Garrett Wollman, who made life much easier in CSAIL.

Lastly I would like to dedicate this thesis to my parents, Yiquan and Hong. They always trust my decisions on both big and small, and fully support me for better or worse. I know that even if nobody would read this thesis they would go through it, I love you two.

This work was partially supported by the Facebook Fellowship, the MIT Greater China Fellowship, BRC & LING Fellowship.

Contents

1 Introduction **23**

1.1 Thesis Outline 25

1.2 Related Work 26

1.2.1 The Rise of the Large-Scale Image Datasets 26

1.2.2 Visualizing and Interpreting the Deep Neural Networks 27

1.2.3 Models for Activity Recognition in Videos 29

2 Places: A 10 Million Image Database for Scene Recognition **33**

2.1 Dataset Construction 34

2.1.1 Places Benchmark 35

2.2 Comparing Scene-centric Datasets 37

2.2.1 Dataset Diversity 37

2.2.2 Cross Dataset Generalization 41

2.3 CNNs for Scene Classification 42

2.3.1 Results on Places205 and Places365 42

2.3.2 Generic Visual Features from Places-CNNs and ImageNet-CNNs . 44

3 ADE20K for Scene Parsing **49**

3.1 ADE20K: Fully Annotated Image Dataset 50

3.1.1 Dataset summary 50

3.1.2 Image annotation 51

3.1.3 Annotation consistency 52

3.1.4 Dataset statistics 55

3.1.5 Comparison with other datasets . . . 55
3.2 Scene Parsing Benchmark and Networks . . . 57

4 Comparing the Deep Visual Representations for Objects and Scenes 61
4.1 ImageNet-CNN and Places-CNN . . . 62
4.2 Uncovering the CNN representation . . . 64
4.2.1 Simplifying the input images . . . 64
4.2.2 Visualizing the receptive fields of units and their activation patterns 66
4.2.3 Identifying the semantics of internal units . . . 69
4.3 Emergence of objects as the internal representation . . . 70
4.3.1 What object classes emerge . . . 70
4.3.2 Object Localization within the Inner Layers . . . 74

5 Quantifying the Interpretability of Deep Visual Representations 77
5.1 Framework of Network Dissection . . . 78
5.1.1 Broden: Broadly and Densely Labeled Dataset . . . 79
5.1.2 Scoring Unit Interpretability . . . 81
5.2 Interpreting Deep Visual Representations . . . 84
5.2.1 Human Evaluation of Interpretations . . . 85
5.2.2 Measurement of Axis-aligned Interpretability . . . 88
5.2.3 Network Architectures with Supervised Learning . . . 91
5.2.4 Representations from Self-supervised Learning . . . 94
5.2.5 Explaining the Predictions for the Deep Features . . . 97

6 Explaining the Deep Neural Networks via Class Activation Mapping 99
6.1 Class Activation Mapping . . . 100
6.2 Weakly-supervised Object Localization . . . 103
6.2.1 Setup . . . 103
6.2.2 Results . . . 104
6.3 Deep Features for Generic Localization . . . 108
6.3.1 Fine-grained Recognition . . . 109

6.3.2 Pattern Discovery 111

7 Efficient Deep Visual Representations for Activity Recognition 113

7.1 Temporal Relation Networks 115

7.1.1 Defining Temporal Relations 115

7.1.2 Multi-Scale Temporal Relations 116

7.1.3 Efficient Training and Testing 116

7.2 Experiments 117

7.2.1 Network Architectures and Training 118

7.2.2 Results on Something-Something Dataset 118

7.2.3 Results on Jester and Charades 120

7.2.4 Interpreting Visual Common Sense Knowledge inside the TRN . . 121

8 Discussion and Future Work 129

List of Figures

2-1 Image samples from four scene categories grouped by queries to illustrate the diversity of the dataset. For each query we show 9 annotated images. . . 34

2-2 Sorted image number per category in the Places Database. Places contains 10,624,928 images from 434 categories. Category names are shown for every 6 intervals. 34

2-3 Comparison of the number of images per scene category for the common 88 scene categories in Places, ImageNet, and SUN datasets. 37

2-4 Examples of pairs for the diversity experiment for a) playground and b) bedroom. Which pair shows the most similar images? The bottom pairs were chosen in these examples. c) Histogram of relative diversity per each category (88 categories) and dataset. Places88 (in blue line) contains the most diverse set of images, then ImageNet88 (in red line) and the lowest diversity is in the SUN88 database (in yellow line) as most images are prototypical of their class. 40

2-5 Cross dataset generalization of training on the 88 common scenes between Places, SUN and ImageNet then testing on the 88 common scenes from: a) SUN, b) ImageNet and c) Places database. 41

2-6 The predictions given by the Places365-VGG for the images from the validation set. The ground-truth label (GT) and the top 5 predictions are shown. The number beside each label indicates the prediction confidence. 44

3-1 Images in ADE20K dataset are densely annotated in detail with objects and parts. The first row shows the sample images, the second row shows the annotation of objects, and the third row shows the annotation of object parts. 51

3-2 Annotation interface, the list of the objects and their associated parts in the image. 52

3-3 Analysis of annotation consistency. Each column shows an image and two segmentations done by the same annotator at different times. Bottom row shows the pixel discrepancy when the two segmentations are subtracted, and the percentage of pixels with the same label. On average across all re-annotated images, 82.4% of pixels got the same label. In the example in the first column the percentage of pixels with the same label is relatively low because the annotator labeled the same region as 'snow' and 'ground' during the two rounds of annotation. In the third column, there were many objects in the scene and the annotator missed some between the two segmentations. 53

3-4 a) Object classes sorted by frequency. Only the top 270 classes with more than 100 annotated instances are shown. 68 classes have more than a 1000 segmented instances. b) Frequency of parts grouped by objects. There are more than 200 object classes with annotated parts. Only objects with 5 or more parts are shown in this plot (we show at most 7 parts for each object class). c) Objects ranked by the number of scenes they are part of. d) Object parts ranked by the number of objects they are part of. e) Examples of objects with doors. The bottom-right image is an example where the door does not behave as a part. 56

3-5 a) Mode of the object segmentations contains 'sky', 'wall', 'building' and 'floor'. b) Histogram of the number of segmented object instances and classes per image. c) Histogram of the number of segmented part instances and classes per object. d) Number of classes as a function of segmented instances (objects and parts). The squares represent the current state of the dataset. e) Probability of seeing a new object (or part) class as a function of the number of instances. 56

3-6 Ground-truths, scene parsing results given by the baseline networks. All networks can give correct predictions for the common, large object and stuff classes, the difference in performance comes mostly from small, infrequent objects and how well they handle details. 59

4-1 Top 3 images producing the largest activation of units in each layer of ImageNet-CNN (top) and Places-CNN (bottom). 63

4-2 Each pair of images shows the original image (left) and a simplified image (right) that gets classified by the Places-CNN as the same scene category as the original image. From top to bottom, the four rows show different scene categories: bedroom, auditorium, art gallery, and dining room. 65

4-3 The pipeline for estimating the RF of each unit. Each sliding-window stimuli contains a small randomized patch (example indicated by red arrow) at different spatial locations. By comparing the activation response of the sliding-window stimuli with the activation response of the original image, we obtain a discrepancy map for each image (middle top). By summing up the calibrated discrepancy maps (middle bottom) for the top ranked images, we obtain the actual RF of that unit (right). 66

4-4 The RFs of 3 units of pool1, pool2, conv4, and pool5 layers respectively for ImageNet- and Places-CNNs, along with the image patches corresponding to the top activation regions inside the RFs. 67

4-5 Segmentation based on RFs. Each row shows the 4 most confident images for some unit. 68

4-6 AMT interface for unit concept annotation. There are three tasks in each annotation. 68

4-7 Examples of unit annotations provided by AMT workers for 6 units from pool5 in Places-CNN. For each unit the figure shows the label provided by the worker, the type of label, the images selected as corresponding to the concept (green box) and the images marked as incorrect (red box). The precision is the percentage of correct images. The top three units have high performance while the bottom three have low performance ($< 75\%$). 71

4-8 (a) Average precision of all the units in each layer for both networks as reported by AMT workers. (b) and (c) show the number of units providing different levels of semantics for ImageNet-CNN and Places-CNN respectively. 72

4-9 Distribution of semantic types found for all the units in both networks. From left to right, each plot corresponds to the distribution of units in each layer assigned to simple elements or colors, textures or materials, regions or surfaces, object parts, objects, and scenes. The vertical axis is the percentage of units with each layer assigned to each type of concept. 72

4-10 Object counts of CNN units discovering each object class for (a) Places-CNN and (b) ImageNet-CNN. 73

4-11 Segmentations using pool5 units from Places-CNN. Many classes are encoded by several units covering different object appearances. Each row shows the 5 most confident images for each unit. The number represents the unit number in pool5. 74

4-12 (a) Object frequency in SUN (only top 50 objects shown), (b) Counts of objects discovered by pool5 in Places-CNN. (c) Frequency of most informative objects for scene classification. 75

4-13 Interpretation of a picture by different layers of the Places-CNN using the tags provided by AMT workers. The first shows the final layer output of Places-CNN. The other three show detection results along with the confidence based on the units' activation and the semantic tags. 75

5-1 Samples from the Broden Dataset. The ground truth for each concept is a pixel-wise dense annotation. 80

5-2 Scoring unit interpretability by evaluating the unit for semantic segmentation. 81

5-3 Illustration of network dissection for measuring semantic alignment of units in a given CNN. Here one unit of the last convolutional layer of a given CNN is probed by evaluating its performance on various segmentation tasks. Our method can probe any convolutional layer. 83

5-4 The annotation interface used by human raters on Amazon Mechanical Turk. Raters are shown descriptive text in quotes together with fifteen images, each with highlighted patches, and must evaluate whether the quoted text is a good description for the highlighted patches. 86

5-5 Comparison of the interpretability of all five convolutional layers of AlexNet, as trained on classification tasks for Places (top) and ImageNet (bottom).Four examples of units in each layer are shown with identified semantics. The segmentation generated by each unit is shown on the three Broden images with highest activation. Top-scoring labels are shown above to the left, and human-annotated labels are shown above to the right. Some disagreement can be seen for the dominant judgment of meaning. For example, human annotators mark the first `conv4` unit on Places as a 'windows' detector, while the algorithm matches the 'chequered' texture. 87

5-6 Interpretability over changes in basis of the representation of AlexNet `conv5` trained on Places. The vertical axis shows the number of unique interpretable concepts that match a unit in the representation. The horizontal axis shows α, which quantifies the degree of rotation. 89

5-7 Visualizations of the best single-unit concept detectors of five concepts taken from individual units of AlexNet conv5 trained on Places (left), compared with the best linear-combination detectors of the same concepts taken from the same representation under a random rotation (right). For most concepts, both the IoU and the visualization of the top activating image patches confirm that individual units match concepts better than linear combinations. In other cases, (e.g. head detectors) visualization of a linear combination appears highly consistent, but the IoU reveals lower consistency when evaluated over the whole dataset. 90

5-8 Complete rotation ($\alpha = 1$) repeated on AlexNet trained on Places365 and ImageNet respectively. Rotation reduces the interpretability significantly for both of the networks. 91

5-9 Interpretability across different architectures trained on ImageNet and Places. 92

5-10 Histogram of the object detectors from the ResNet and DenseNet trained on ImageNet and Places respectively. 92

5-11 Comparison of several visual concept detectors identified by network dissection in DenseNet, ResNet, GoogLeNet, VGG, and AlexNet. Each network is trained on Places365. The two highest-IoU matches among convolutional units of each network is shown. The segmentation generated by each unit is shown on the four maximally activating Broden images. Some units activate on concept generalizations, e.g., GoogLeNet 4e's unit 225 on horses and dogs, and 759 on white ellipsoids and jets. 93

5-12 Comparison of interpretability of the layers for AlexNet, VGG16, GoogLeNet, and ResNet152 trained on Places365. All five conv layers of AlexNet and the selected layers of VGG, GoogLeNet, and ResNet are included. 94

5-13 Semantic detectors emerge across different supervision of the primary training task. All these models use the AlexNet architecture and are tested at conv5. 95

5-14 The top ranked concepts in the three top categories in four self-supervised networks. Some object and part detectors emerge in `audio`. Detectors for person heads also appear in `puzzle` and `colorization`. A variety of texture concepts dominate models with self-supervised training. 95

5-15 Segmenting images using top activated units weighted by the class label from ResNet152-Places365 deep feature. a) the correctly predicted samples. b) the incorrectly predicted samples. 96

5-16 Class-specific units from ResNet152-ImageNet and ResNet152-Places365 on one class from action40 and sun397. For each class, we show three sample images, followed by the top 3 units from ResNet152-ImageNet and ResNet152-Places365 ranked by the class weight of linear SVM to predict that class. SVM weight, detected concept name and the IoU score are shown above each unit. 98

6-1 A simple modification of the global average pooling layer combined with our class activation mapping (CAM) technique allows the classification-trained CNN to both classify the image and localize class-specific image regions in a single forward-pass e.g., the toothbrush for *brushing teeth* and the chainsaw for *cutting trees*. It produces a visual explanation for the decision made by the CNN. 100

6-2 Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions. 101

6-3 (a) The CAMs of four classes from ImageNet. The maps highlight the discriminative image regions used for image classification e.g., the head of the animal for *briard* and *hen*, the plates in *barbell*, and the bell in *bell cote*. (b) Examples of the CAMs generated from the top 5 predicted categories for the given image with ground-truth as dome. The predicted class and its score are shown above each class activation map. We observe that the highlighted regions vary across predicted classes e.g., *dome* activates the upper round part while *palace* activates the lower flat part of the compound. 103

6-4 Class activation maps from CNN-GAPs and the class-specific saliency map from the backpropagation methods. 106

6-5 a) Examples of localization from GoogleNet-GAP. b) Comparison of the localization from GooleNet-GAP (upper two) and the backpropagation using AlexNet (lower two). The ground-truth boxes are in green and the predicted bounding boxes from the class activation map are in red. 107

6-6 CAMs and the inferred bounding boxes (in red) for selected images from four bird categories in CUB200. In Sec. 6.3.1 we quantitatively evaluate the quality of the bounding boxes (41.0% accuracy for 0.5 IoU). We find that extracting GoogLeNet-GAP features in these CAM bounding boxes and re-training the SVM improves bird classification accuracy by about 5% (Tbl. 6.4). 110

6-7 Generic discriminative localization using our GoogLeNet-GAP deep features (which have been trained to recognize objects). We show 2 images each from 3 classes for 4 datasets, and their class activation maps below them. We observe that the discriminative regions of the images are often highlighted e.g., in Stanford Action40, the mop is localized for *cleaning the floor*, while for *cooking* the pan and bowl are localized and similar observations can be made in other datasets. This demonstrates the generic localization ability of our deep features. 111

6-8 Informative objects for two scene categories. For the dining room and bathroom categories, we show examples of original images (top), and list of the 6 most frequent objects in that scene category with the corresponding frequency of appearance. At the bottom: the CAMs and a list of the 6 objects that most frequently overlap with the high activation regions. 112

6-9 Examples of highlighted image regions for the predicted answer class in the visual question answering. 112

7-1 What takes place between two observations? (see answer below the first page). Humans can easily infer the temporal relations and transformations between these observations, but this task remains difficult for neural networks. 114

7-2 The illustration of Temporal Relation Networks. Representative frames of a video (shown above) are sampled and fed into different frame relation modules. Only a subset of the 2-frame, 3-frame, and 4-frame relations are shown, as there are higher frame relations included. 115

7-3 Prediction examples on a) Something-Something, b) Jester, and c) Charades. For each example drawn from Something-Something and Jester, the top two predictions with green text indicating a correct prediction and red indicating an incorrect one. Top 2 predictions are shown above Charades frames. 122

7-4 The top representative frames determined by single frame baseline network, the 2-frame TRN, 3-frame TRN, and 4-frame TRN. TRNs learn to capture the essence of an activity only given a limited number of frames. Videos are from the validation set of the Something-Something dataset . . . 124

7-5 Temporal alignment of videos from the (a) Something-Something and (b) Jester datasets using the most representative frames as temporal anchor points. For each action, 4 different videos are aligned using 5 temporal anchor points. 125

7-6 (a) Accuracy obtained using ordered frames and shuffled frames, on Something-Something and UCF101 dataset respectively. On Something-Something, the temporal order is critical for recognizing the activity. But recognizing activities in UCF101 does not necessarily require temporal relational reasoning. (b) The top 5 action categories that exhibited the largest gain and the least gain (negative) respectively between ordered and shuffled frames as inputs. Actions with directional motion appear to suffer most from shuffled inputs. 126

7-7 t-SNE plot of the video samples of the 15 classes using the deep features from the single-frame baseline, 2-frame TRN, and 5-frame TRN. Higher frame TRN can better differentiate activities in Something-Something dataset. 126

7-8 Early recognition of activity when only given the first 25% frames. The first 25% of each video, represented by the first frame shown in the left column, is used to generate the top 3 anticipated forecasts and corresponding probabilities listed in the middle column. The ground truth label is highlighted by a blue arrow which points to the last frame of the video on the right. 127

List of Tables

2.1 Classification accuracy on the test set of Places205 and the test set of SUN205. We use the class score averaged over 10-crops of each test image to classify the image. $*$ shows the top 2 ranked results from the Places205 leaderboard. 43

2.2 Classification accuracy on the validation set and test set of Places365. We use the class score averaged over 10-crops of each testing image to classify the image. 44

2.3 Classification accuracy/precision on scene-centric databases (the first four datasets) and object-centric databases (the last four datasets) for the deep features of various Places-CNNs and ImageNet-CNNs. All the accuracy/precision is the top-1 accuracy/precision. 47

3.1 Comparison of semantic segmentation datasets. 57

3.2 Baseline performance on the validation set of SceneParse150. 58

4.1 The parameters of the network architecture used for ImageNet-CNN and Places-CNN. 63

4.2 Comparison of the theoretical and empirical sizes of the RFs for Places-CNN and ImageNet-CNN at different layers. Note that the RFs are assumed to be square shaped, and the sizes reported below are the length of each side of this square, in pixels. 68

5.1 Statistics of each label type included in the dataset. 80

5.2 Tested CNN Models 83

5.3 Human evaluation of our Network Dissection approach. 87

6.1 Classification error on the ILSVRC validation set. 106
6.2 Localization error on the ILSVRC validation set. *Backprop* refers to using [88] for localization instead of CAM. 107
6.3 Localization error on the ILSVRC test set for various weakly- and fully-supervised methods. 107
6.4 Fine-grained classification performance on CUB200 dataset. GoogLeNet-GAP can successfully localize important image crops, boosting classification performance. 109
6.5 Classification accuracy on representative scene and object datasets for different deep features. 110

7.1 Statistics of the datasets used in evaluating the TRNs. 117
7.2 (a) Results on the validation set of Something-Something Dataset. All the models use the center cropping of the equidistant frames in the video. Multi-scale TRN with 10-crop augmentation achieves the best performance. TRN also outperforms TSN in a large margin, showing the importance of temporal order. (b) Results on the test set of the Something-Something Dataset. Comparison methods are from the official public leaderboard. . . . 120
7.3 Jester Dataset Results on (a) the validation set and (b) the test set. 120
7.4 Results on Charades Activity Classification. 121
7.5 Early activity recognition using the MultiScale TRN on Something-Something and Jester dataset. Only the first 25% and 50% of frames are given to the TRN to predict activities. Baseline is the model trained on single frames. . 127

Chapter 1

Introduction

Recent progress of deep neural networks in computer vision and machine learning has enabled transformative applications across robotics, healthcare, and security. However, despite the superior performance of the deep neural networks, it remains challenging to understand their inner workings and explain their output predictions.

Deep neural network models such as Convolutional Neural Networks (CNNs) are often criticized as being black boxes that lack interpretability, because of their millions of unexplained model parameters. The interpretability of AI models is closely relevant to various important issues, for example the safety of the AI models in critical applications such as autonomous driving and medical image diagnosis, and the fairness and bias of AI models which potentially result to social and moral impacts. Thus lacking interpretability greatly limits the usage of the complex models for wider Artificial Intelligence (AI) applications.

This Ph.D. thesis work advances the interpretable representation learning, with the goal of better understanding the interpretability of the deep learning models used in computer vision. To open up the black boxes of the deep neural networks, this work develops methods for visualizing and interpreting their internal representations, then proposes a general framework for quantifying the interpretability of any given deep visual representations. To better investigate the decision making of the networks, this work further designs a simple yet effective approach for generating visual explanations for the prediction made by the deep model, by highlighting the most informative image regions. Furthermore, in order to go from image recognition to event understanding in videos, this work constructs a new

efficient and interpretable deep models for video recognition.

Comprehending the visual world in a single glance is one of the most magnificent feats of human intelligence. By sampling the scene dozens of times per second, we are exposed to millions of natural images in the course of a year. This rich and diverse visual experience contained in the scene context guides our interpretation of the world and shapes our knowledge of the reality. Inspired by this, recent deep learning models build their internal representation in a data-driven approach of learning from millions of training samples from large-scale scene datasets. Generalized from millions of training samples, many of the visual recognition models powered by deep neural networks have achieved human-level performance.

An AI model achieves human-level visual recognition, thus it is likely enbale to learn to identify and disentangle the underlying explanatory factors in the observed input data. So the central question about learning interpretable representations is *what are the underlying explanatory factors when the deep neural networks are trained to recognize the objects or scenes*, and furthermore, *are these factors are meaningful and interpretable to human or not*.

In this thesis, we show that there are various internal units emerged as detectors to detect visual concepts at multiple levels: the units at lower layers are detecting edges or texture while the units at higher layers are detecting more semantically meaningful concepts such as bicycle wheel or dog head. We develop methods that interpret the meaning of each unit through human annotation or automatically by measuring the degree of alignment between activation and a list of concepts. We further find that the meaningful and interpretable units emerge in the deep visual representations with various network architectures trained from a wide range of supervision tasks. Finally we show that the interpreted units can be used to provide explicit explanations for a prediction made by a CNN. Our results highlight that interpretability is an important property of deep neural networks that provides new insights into their hierarchical structure.

1.1 Thesis Outline

This thesis starts from building large-scale scene centric datasets Places and ADE20K, which is the foundation for training and benchmarking the deep neural networks. Then this thesis introduces various approaches developed to interpret the deep models trained for recognizing objects, scenes, and various other supervision tasks. Specifically this thesis is organized in following chapters:

Chapter 2 describes the *Places Database* [121, 122], the first ever massive-scale image dataset containing tens of millions of scene images. This dataset has established the most comprehensive set of benchmarks for scene recognition in the vision community. It also provides a training set other than the ImageNet where CNNs are able to be trained from scratch. It lays the foundation for the comparison of deep visual representations trained for classifying scenes and objects in the Chapter 4 and Chapter 5.

Chapter 3 presents *ADE20K dataset* [124], a subset of Places Database which has 22K images with precise pixel-wise annotations of scenes, objects, parts of objects. This dataset allows training deep neural networks for pixel-wise semantic segmentation. In Chapter 5 this dataset is used as part of the testing dictionary to generate the interpretation of the individual units in the deep visual representations.

Chapter 4 compares the deep visual representations in the CNN trained to classify objects and the CNN trained to classify scenes [119]. We show that object detection emerges inside the CNN trained to recognize scenes, even more than when trained to classify objects on ImageNet. This result is interesting because objects are discovered as the explanatory and interpretable factors inside a network when it is trained to solve the scene classification. It shows that the network learns to disentangle the meaningful factors to solve the task it is being trained for.

Chapter 5 develops a method called *Network Dissection* [5] to measure the interpretability of any given deep visual representations. We define the interpretability of individual unit as the degree of alignment between the unit activation and the predefined list of visual concepts. We apply this method to evaluate and compare the interpretability of deep visual representations trained from supervised learning and self-supervised learning. We

further examine how different training regularizers and network designs affect the unit interpretability.

Chapter 6 describes a method called *Class Activation Mapping* [120] to explain the final prediction given by the classification network. The method highlights the most informative image regions relevant to the prediction, by leveraging the internal representation of the network.

Chapter 7 extends the interpretable representation learning from image recognition to video understanding. An efficient and intepretable neural representation called *Temporal Relational Network* [118] is proposed for activity recognition in videos.

Chapter 8 offers discussion and future work.

1.2 Related Work

We have a brief survey on several lines of work which are relevant to the topics covered in the thesis work.

1.2.1 The Rise of the Large-Scale Image Datasets

Image dataset is the driving force for novel models and progress made for visual recognition. What does it take to reach human-level recognition with a machine-learning algorithm? In the case of supervised learning, the problem is two-fold. First, the algorithm must be suitable for the task, such as Convolutional Neural Networks in the large scale visual recognition [48, 122] and Recursive Neural Networks for natural language processing [40, 60]. Second, it must have access to a training dataset of appropriate coverage (quasi-exhaustive representation of classes and variety of exemplars) and density (enough samples to cover the diversity of each class). The optimal space for these datasets is often task-dependent, but the rise of multi-million-item sets has enabled unprecedented performance in many domains of artificial intelligence.

Convolutional Neural Networks [48,52] have likewise achieved near human-level visual recognition, trained on 1.2 million object [15, 36, 79] and 2.5 million scene images [122]. Expansive coverage of the space of classes and samples allows getting closer to the right

ecosystem of data that a natural system, like a human, would experience. The history of image datasets for scene recognition also sees the rapid growing in the image samples as follows:

The first benchmark for scene recognition was the Scene15 database [50], extended from the initial 8 scene dataset in [69]. This dataset contains only 15 scene categories with a few hundred images per class, and current classifiers are saturated, reaching near human performance with 95%. The MIT Indoor67 database [76] with 67 indoor categories and the SUN (Scene Understanding, with 397 categories and 130,519 images) database [108] provided a larger coverage of place categories, but failed short in term of quantity of data needed to feed deep learning algorithms. To complement large object-centric datasets such as ImageNet [15], we build the Places dataset described in this thesis. Places is the other dataset where people is able to train modern deep neural networks from scratch besides ImageNet.

Meanwhile, the Pascal VOC dataset [23] is one of the earliest image dataset with diverse object annotations in scene context. The Pascal VOC challenge has greatly advanced the development of models for object detection and segmentation tasks. Nowadays, COCO dataset [56] focuses on collecting object instances both in polygon and bounding box annotations for images depicting everyday scenes of common objects. The recent Visual Genome dataset [47] aims at collecting dense annotations of objects, attributes, and their relationships. ADE20K [124] proposed in the thesis collects precise dense annotation of scenes, objects, parts of objects with a large and open vocabulary, which greatly enhance the diversity and richness of image annotations. Altogether, annotated datasets further enable artificial systems to learn visual knowledge linking parts, objects and scene context.

1.2.2 Visualizing and Interpreting the Deep Neural Networks

As deep neural networks surpass human on various tasks, people start to explore why these deep models work so well and what have been learned inside. Recently there sees a growing number of work on understanding deep neural networks. The relevant work can be categorized as following three aspects:

Visualizing deep visual representations. Though CNN models are notoriously known as black boxes, a growing number of techniques have been developed to visualize the internal representations of convolutional neural networks. The behavior of a CNN can be visualized by sampling image patches that maximize activation of hidden units [31, 112, 119], or by using variants of backpropagation to identify or generate salient image features [58, 88, 112]. Back-propagation together with a natural image prior can be used to invert a CNN layer activation [59], and an image generation network can be trained to invert the deep features by synthesizing the input images [20]. [66] further synthesizes the prototypical images for individual units by learning a feature code for the image generation network from [20]. These visualizations reveal the image patterns that have been learned in a deep visual representation and provide a qualitative guide to the interpretation and interpretability of units. In [119], a quantitative measure of interpretability was introduced: human evaluation of visualizations to determine which individual units behave as object detectors in a network trained to classify scenes. However, human evaluation is not scalable to increasingly large networks such as ResNet [37], with more than 100 layers. Therefore the aim of the thesis work is to develop methods to go from qualitative visualization to quantitative interpretation.

Analyzing the properties of deep visual representations. Various intrinsic properties of deep visual representations have been explored. Much research has focused on studying the power of CNN layer activations to be used as generic visual features for classification [3, 78]. The transferability of activations for a variety of layers has been analyzed, and it has been found that higher layer units are more specialized to the target task [110]. Susceptibility to adversarial input reveals that discriminative CNN models are fooled by particular image patterns [67, 95]. Analysis of correlation between different random initialized networks reveal that many units converge to the same set of representations after training [54]. The question of how representations generalize has been investigated by showing that a CNN can easily fit a random labeling of training data even under explicit regularization [113]. Our thesis work focuses on another less explored property of deep visual representations: interpretability.

Unsupervised learning of deep visual representations. Recent work on unsupervised

learning or self-supervised learning exploits the correspondence structure that comes for free from unlabeled images to train networks from scratch [2, 17, 43, 68, 105]. For example, CNN is trained by predicting image context [17], by colorizing gray images [116, 117], by solving image puzzle [68], and by associating the images with ambient sounds [71]. The resulting deep visual representations learned from different unsupervised learning tasks are compared by evaluating them as generic visual features on classification datasets such as Pascal VOC. Chapter 5 provides an alternative approach to compare deep visual representations in terms of their interpretability, beyond just their discriminative power.

1.2.3 Models for Activity Recognition in Videos

Convolutional Neural Networks for Activity Recognition. Activity recognition in videos is a core problem in computer vision. With the rise of deep convolutional neural networks (CNNs) which achieve state-of-the-art performance on image recognition tasks [48, 122], many works have looked into designing effective deep convolutional neural networks for activity recognition [9, 19, 45, 89, 98, 103]. For instance, various approaches of fusing RGB frames over the temporal dimension are explored on the Sport1M dataset [45]. Two stream CNNs with one stream of static images and the other stream of optical flows are proposed to fuse the information of object appearance and short-term motions [89]. 3D convolutional networks [98] use 3D convolution kernels to extract features from a sequence of dense RGB frames. Temporal Segment Networks sample frames and optical flow on different time segments to extract information for activity recognition [103]. A CNN+LSTM model, which uses a CNN to extract frame features and an LSTM to integrate features over time, is also used to recognize activities in videos [19]. Recently, I3D networks [9] use two stream CNNs with inflated 3D convolutions on both dense RGB and optical flow sequences to achieve state of the art performance on the Kinetics dataset [46]. There are several important issues with existing CNNs for action recognition: 1) The dependency on beforehand extraction of optical flow lowers the efficiency of the recognition system; 2) The 3D convolutions on sequences of dense frames are computationally expensive, given the redundancy in consecutive frames; 3) Since sequences of frames fed into the network

are usually limited to 20 to 30 frames, it is difficult for the networks to learn long-term temporal relations among frames. To address these issues, the Temporal Relation Network proposed in Chapter 7 sparsely samples individual frames and then learns their causal relations, which is much more efficient than sampling dense frames and convolving them. We show that TRN-equipped networks can efficiently capture temporal relations at multiple time scales and outperform dense frame-based networks using only sparsely sampled video frames.

Temporal Information in Activity Recognition. For activity recognition on many existing video datasets such as UCF101 [92], Sport1M [45], THUMOS [33], and Kinetics [46], the appearance of still frames and short-term motion such as optical flow are the most important information to identify the activities. Thus, activity recognition networks such as Two Stream network [89] and the I3D network [9] are tailored to capture these short-term dynamics of dense frames. Therefore, existing networks don't need to build temporal relational reasoning abilities. On the other hand, recently there have been various video datasets collected via crowd-sourcing, which focus on sequential activity recognition: Something-Something dataset [34] is collected for generic human-object interaction. It has video classes such as 'Dropping something into something', 'Pushing something with something', and even 'Pretending to open something without actually opening it'. Jester dataset [1] is another recent video dataset for gesture recognition. Videos are recorded by crowd-source workers performing 27 kinds of gestures such as 'Thumbing up', 'Swiping Left', and 'Turning hand counterclockwise'. Charades dataset is also a high-level human activity dataset that collects videos by asking crowd workers to perform a series of home activities and then record themselves [86]. For recognizing the complex activities in these three datasets, it is crucial to integrate temporal relational reasoning into the networks. Besides, many previous works model the temporal structures of videos for action recognition and detection using bag of words, motion atoms, or action grammar [27, 28, 75, 101, 102]. Instead of designing temporal structures manually, we use a more generic structure to learn the temporal relations in end-to-end training. One relevant work on modeling the cause-effect in videos is [104]. [104] uses a two-stream siamese network to learn the transformation matrix between two frames, then uses brute force search to infer the action category.

Thus the computation cost is high. Ideally we want to have a deep neural network model which are efficient to train and test on videos.

Chapter 2

Places: A 10 Million Image Database for Scene Recognition

For an agent acting into the world, there is no doubt that object and event recognition should be a primary goal of its visual system. But knowing the place or context in which the objects appear is as equally important for an intelligent system to understand what might have happened in the past and what may happen in the future. For instance, a table inside a kitchen can be used to eat or prepare a meal, while a table inside a classroom is intended to support a notebook or a laptop to take notes.

Whereas most datasets have focused on object categories (providing labels, bounding boxes or segmentations), here we describe the Places database, a quasi-exhaustive repository of 10 million scene photographs, labeled with 434 scene semantic categories, comprising about 98 percent of the type of places a human can encounter in the world. Places provides several benchmarks for AI models to learn to recognize scene context in the wild, which greatly enhance their capacity to reach human-level recognition. Furthermore, Places is also the other large-scale image dataset where researchers are able to train CNNs from scratch besides ImageNet. In later chapter we conduct a comparison study to analyze the difference between the deep visual representation learned to recognize scenes and the one learned to recognize objects.

Image samples from Places Database are shown in Fig. 2-1 while Fig. 2-2 shows the number of images per category, sorted in decreasing order. In this chapter we introduce the

Figure 2-1: Image samples from four scene categories grouped by queries to illustrate the diversity of the dataset. For each query we show 9 annotated images.

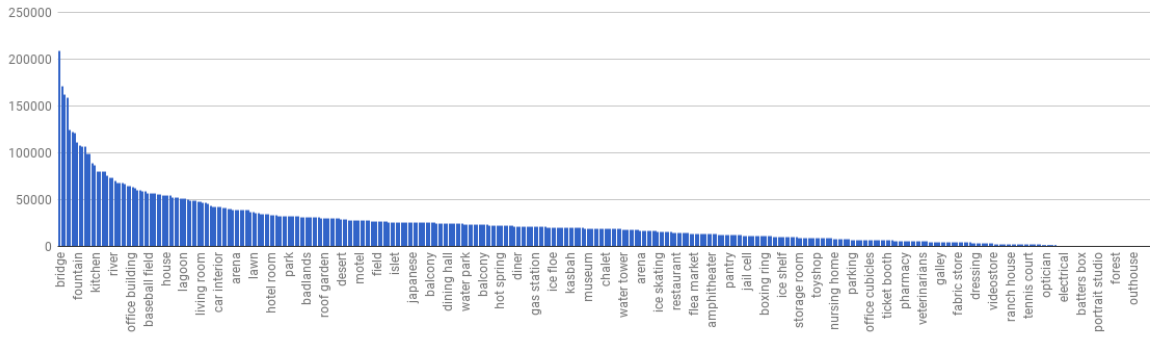


Figure 2-2: Sorted image number per category in the Places Database. Places contains 10,624,928 images from 434 categories. Category names are shown for every 6 intervals.

construction of the dataset and the scene recognition benchmarks in detail.

2.1 Dataset Construction

Since the SUN database [108] has a rich scene taxonomy, the Places database has inherited the same list of scene categories. To generate the query of image URL, 696 common adjectives (messy, spare, sunny, desolate, etc), manually selected from a list of popular adjectives in English, are combined with each scene category name and are sent to three image search engines (Google Images, Bing Images, and Flickr). Adding adjectives to the queries allows us to download a larger number of images than what is available in ImageNet and to increase the diversity of visual appearances. We then remove duplicated URLs and download the raw images with unique URLs. To date, more than 40 million images have

been downloaded. Only color images of 200×200 pixels or larger are kept. PCA-based duplicate removal is conducted within each scene category in the Places database and across the same scene category in the SUN database, which ensures that Places and the SUN do not contain the same images, allowing us to combine the two datasets.

The images that survive this initial selection are sent to Amazon Mechanical Turk for two rounds of individual image annotation. For a given category name, its definition as in [108], is shown at the top of a screen, with a question like *is this a living room scene?* A single image at a time is shown centered in a large window, and workers are asked to press a Yes or No key. For the first round of labeling, the default answer is set to No, requiring the worker to actively pick up the positive images. The positive images resulting from the first round annotation are further sent for a second round annotation, in which the default answer is set to Yes (to pick up the remaining negative images). In each HIT(one assignment for each worker), 750 downloaded images are included for annotation, and an additional 30 positive samples and 30 negative samples with ground truth from the SUN database are also randomly injected as control. Valid HITs kept for further analyses require an accuracy of 90% or higher on these control images. As a result of the previous round of image annotation, there were 53 million remaining downloaded images not assigned to any of the 476 scene categories. Therefore, a third annotation task was designed to reclassify then re-annotate those images, using a machine learning in the loop approach. The final round of annotation is to merge some ambiguous pairs of categories and further differentiate the images from the categories with shared content.

Finally the Places database was annotated with over 10 millions labeled exemplars (10,624,928 images) from 434 place categories.

2.1.1 Places Benchmark

Here we describe four subsets of Places database as benchmarks. While Places205 and Places88 are from [122], two new benchmarks have been added in journal version of the Places database [121]: from the 434 categories, we selected 365 categories with more than 4000 images each to create *Places365-Standard* and *Places365-Challenge*. The details of

each benchmark are the following:

- **Places365-Standard** has 1,803,460 training images with the image number per class varying from 3,068 to 5,000. The validation set has 50 images per class and the test set has 900 images per class. Note that the experiments in this chapter are reported on Places365-Standard.

- **Places365-Challenge** contains the same categories as *Places365-Standard*, but the training set is significantly larger with a total of 8 million training images. The validation set and testing set are the same as the Places365-Standard. This subset was released for the Places Challenge 2016[1] held in conjunction with the European Conference on Computer Vision (ECCV) 2016, as part of the ILSVRC Challenge.

- **Places205**. Places205, described in [122], has 2.5 million images from 205 scene categories. The image number per class varies from 5,000 to 15,000. The training set has 2,448,873 total images, with 100 images per category for the validation set and 200 images per category for the test set.

- **Places88**. Places88 contains the 88 common scene categories among the ImageNet [79], SUN [108] and Places205 databases. Places88 contains only the images obtained in round 2 of annotations, from the first version of Places used in [122]. We call the other two corresponding subsets ImageNet88 and SUN88 respectively. These subsets are used to compare performances across different scene-centric databases, as the three datasets contain different exemplars per category (i.e. none of these three datasets contain common images). Note that finding correspondences between the classes defined in ImageNet and Places brings some challenges. ImageNet follows the WordNet definitions, but some WordNet definitions are not always appropriate for describing places. For instance, the class 'elevator' in ImageNet refers to an object. In Places, 'elevator' takes different meanings depending on the location of the observer: elevator door, elevator interior, or elevator lobby. Many categories in ImageNet do not differentiate between indoor and outdoor (e.g., ice-skating rink) while

[1] http://places2.csail.mit.edu/challenge.html

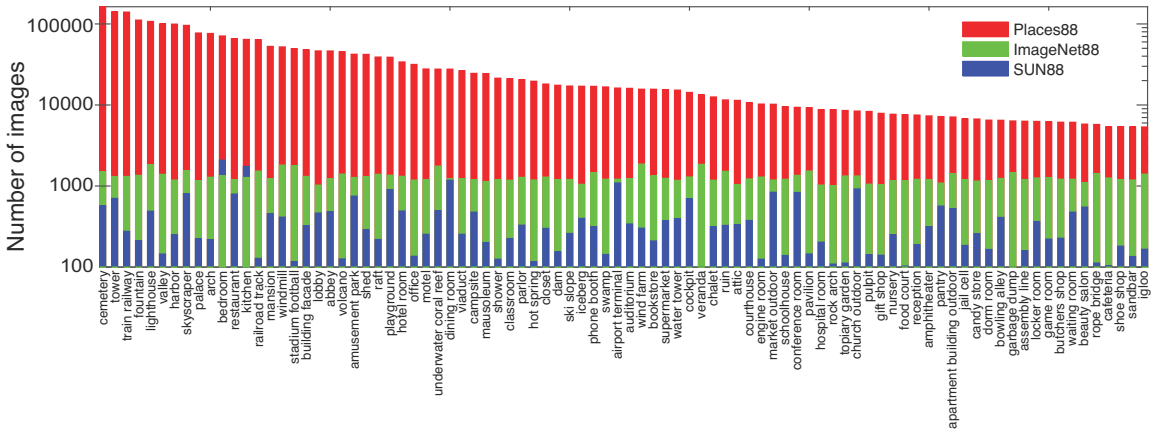


Figure 2-3: Comparison of the number of images per scene category for the common 88 scene categories in Places, ImageNet, and SUN datasets.

in Places, indoor and outdoor versions are separated as they do not necessarily afford the same function.

2.2 Comparing Scene-centric Datasets

Scene-centric datasets correspond to images labeled with a scene, or place name, as opposed to object-centric datasets, where images are labeled with object names. In this section we use the Places88 benchmark to compare Places dataset with the tow other biggest scene datasets: ImageNet88 and SUN88. Fig. 2-3 illustrates the differences among the number of images found in the different categories for Places88, ImageNet88 and SUN88. Notice that Places Database is the largest scene-centric image dataset so far. The next subsection presents a comparison of these three datasets in terms of image diversity.

2.2.1 Dataset Diversity

Given the types of images found on the internet, some categories will be more biased than others in terms of viewpoints, types of objects, or even image style [97]. However, bias can be compensated with a high diversity of images, with many appearances represented in the dataset. In this section we describe a measure of dataset diversity to compare how diverse images from three scene-centric datasets (Places88, SUN88 and ImageNet88) are.

Comparing datasets is an open problem. Even datasets covering the same visual classes have notable differences providing different generalization performances when used to train a classifier [97]. Beyond the number of images and categories, there are aspects that are important but difficult to quantify, like the variability in camera poses, in decoration styles or in the type of objects that appear in the scene.

Although the quality of a database is often task dependent, it is reasonable to assume that a good database should be **dense** (with a high degree of data concentration), and **diverse** (it should include a high variability of appearances and viewpoints). Imagine, for instance, a dataset composed of 100,000 images all taken within the same bedroom. This dataset would have a very high density but a very low diversity as all the images will look very similar. An ideal dataset, expected to generalize well, should have high *diversity* as well. While one can achieve high density by collecting a large number of images, diversity is not an obvious quantity to estimate in image sets, as it assumes some notion of similarity between images. One way to estimate similarity is to ask the question *are these two images similar?* However, similarity in the wild is a subjective and loose concept, as two images can be viewed as similar if they contain similar objects, and/or have similar spatial configurations, and/or have similar decoration styles and so on. A way to circumvent this problem is to define *relative measures* of similarity for comparing datasets.

Several measures of diversity have been proposed, particularly in biology for characterizing the richness of an ecosystem (see [39] for a review). Here, we propose to use a measure inspired by the *Simpson index of diversity* [91]. The Simpson index measures the probability that two random individuals from an ecosystem belong to the same species. It is a measure of how well distributed the individuals across different species are in an ecosystem, and it is related to the entropy of the distribution. Extending this measure for evaluating the diversity of images within a category is non-trivial if there are no annotations of sub-categories. For this reason, we propose to measure the relative diversity of image datasets A and B based on the following idea: if set A is more diverse than set B, then two random images from set B are more likely to be visually similar than two random samples

from A. Then, the diversity of A with respect to B can be defined as

$$\text{Div}_B(A) = 1 - p(d(a_1, a_2) < d(b_1, b_2)) \tag{2.1}$$

where $a_1, a_2 \in A$ and $b_1, b_2 \in B$ are randomly selected. With this definition of relative diversity we have that A is more diverse than B if, and only if, $\text{Div}_B(A) > \text{Div}_A(B)$.

For an arbitrary number of datasets, $A_1, ..., A_N$, the diversity of A_1 with respect to $A_2, ..., A_N$ can be defined as

$$\text{Div}_{A_2,...,A_N}(A_1) = 1 - p(d(a_{11}, a_{12}) < \min_{i=2:N} d(a_{i1}, a_{i2})) \tag{2.2}$$

where $a_{i1}, a_{i2} \in A_i$ are randomly selected, $i = 2 : N$.

We measured the relative diversities between SUN88, ImageNet88 and Places88 using AMT. Workers were presented with different pairs of images and they had to select the pair that contained the most similar images. The pairs were randomly sampled from each database. Each trial was composed of 4 pairs from each database, giving a total of 12 pairs to choose from. We used 4 pairs per database to increase the chances of finding a similar pair and avoiding users having to skip trials. AMT workers had to select the most similar pair on each trial. We ran 40 trials per category and two observers per trial, for the 88 categories in common between ImageNet88, SUN88 and Places88 databases. Fig. 2-4a-b shows some examples of pairs from the diversity experiments for the scene categories playground (a) and bedroom (b). In the figure only one pair from each database is shown. We observed that different annotators were consistent in deciding whether a pair of images was more similar than another pair of images.

Fig. 2-4c shows the histograms of relative diversity for all the 88 scene categories common to the three databases. If the three datasets were identical in terms of diversity, the average diversity should be 2/3 for the three datasets. Note that this measure of diversity is a relative measure between the three datasets. In the experiment, users selected pairs from the SUN database to be the closest to each other 50% of the time, while the pairs from the Places database were judged to be the most similar only on 17% of the trials. ImageNet88 pairs were selected 33% of the time.

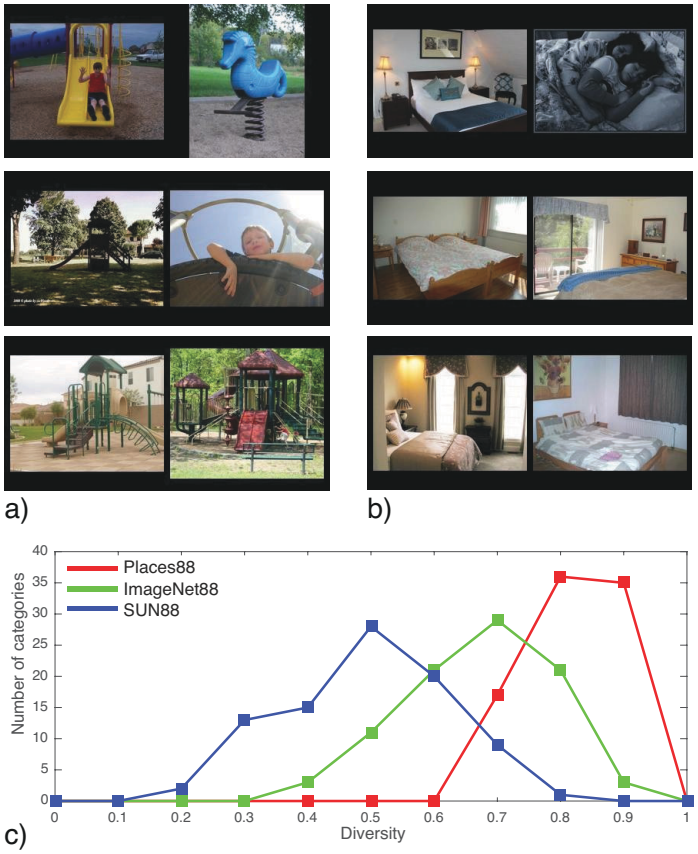


Figure 2-4: Examples of pairs for the diversity experiment for a) playground and b) bedroom. Which pair shows the most similar images? The bottom pairs were chosen in these examples. c) Histogram of relative diversity per each category (88 categories) and dataset. Places88 (in blue line) contains the most diverse set of images, then ImageNet88 (in red line) and the lowest diversity is in the SUN88 database (in yellow line) as most images are prototypical of their class.

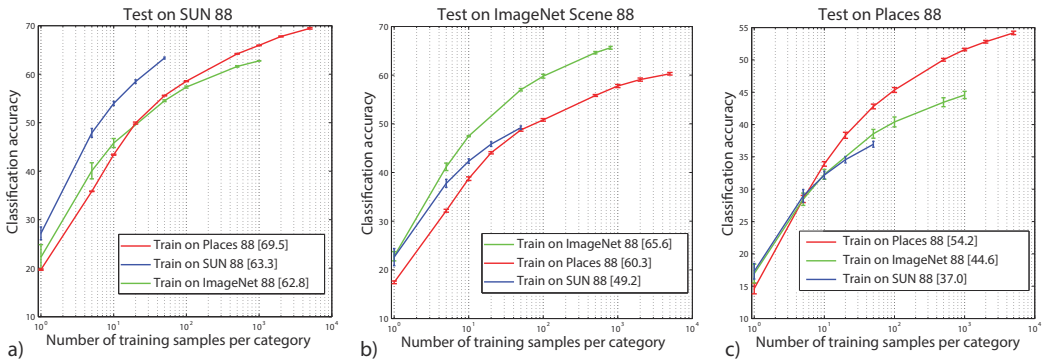


Figure 2-5: Cross dataset generalization of training on the 88 common scenes between Places, SUN and ImageNet then testing on the 88 common scenes from: a) SUN, b) ImageNet and c) Places database.

The results show that there is a large variation in terms of diversity among the three datasets, showing Places to be the most diverse of the three datasets. The average relative diversity on each dataset is 0.83 for Places, 0.67 for ImageNet88 and 0.50 for SUN. The categories with the largest variation in diversity across the three datasets were *playground*, *veranda* and *waiting room*.

2.2.2 Cross Dataset Generalization

As discussed in [97], training and testing across different datasets generally results in a drop of performance due to the dataset bias problem. In this case, the bias between datasets is due, among other factors, to the differences in the diversity between the three datasets. Fig. 2-5 shows the classification results obtained from the training and testing on different permutations of the 3 datasets. For these results we use the features extracted from a pre-trained ImageNet-CNN and a linear SVM. In all three cases training and testing on the same dataset provides the best performance for a fixed number of training examples. As the Places database is very large, it achieves the best performance on two of the test sets when all the training data is used.

2.3 CNNs for Scene Classification

Given the impressive performance of the deep Convolutional Neural Networks (CNNs), particularly on the ImageNet benchmark [48, 79], we choose three popular CNN architectures, AlexNet [48], GoogLeNet [93], and VGG 16 convolutional-layer CNN [90], then train them on **Places205** and **Places365-Standard** respectively to create baseline CNN models. The trained CNNs are named as Places*Subset-CNN*, i.e., Places205-AlexNet or Places365-VGG.

All the Places-CNNs presented here were trained using the Caffe package [44] on Nvidia GPUs Tesla K40 and Titan X[2]. Additionally, given the recent breakthrough performances of the Residual Network (ResNet) on ImageNet classification [38], we further fine-tuned **ResNet152** on the Places365-Standard (termed as Places365-ResNet) and compared it with the other trained-from-scratch Places-CNNs for scene classification.

2.3.1 Results on Places205 and Places365

After training the various Places-CNNs, we used the final output layer of each network to classify the test set images of Places205 and SUN205 (see [122]). The classification results for Top-1 accuracy and Top-5 accuracy are listed in Table 2.1. The Top-1 accuracy is the percentage of the testing images where the top predicted label exactly match the ground-truth label. The Top-5 accuracy is that the percentage of testing images where the ground-truth label is among the top ranked 5 predicted labels given by an algorithm. Since there are some ambiguity between some scene categories, the Top-5 accuracy is a more suitable criteria of measuring scene classification performance.

As a baseline comparison, we show the results of a linear SVM trained on ImageNet-CNN features of 5000 images per category in Places205 and 50 images per category in SUN205 respectively. We observe that Places-CNNs perform much better than the ImageNet feature+SVM baseline while, as expected, Places205-GoogLeNet and Places205-VGG outperformed Places205-AlexNet with a large margin, due to their deeper structures.

[2]All the Places-CNNs are available at `https://github.com/CSAILvision/places365`

Table 2.1: Classification accuracy on the test set of Places205 and the test set of SUN205. We use the class score averaged over 10-crops of each test image to classify the image. $*$ shows the top 2 ranked results from the Places205 leaderboard.

	Test set of Places205		Test set of SUN205	
	Top-1 acc.	Top-5 acc.	Top-1 acc.	Top-5 acc.
ImageNet-AlexNet feature+SVM	40.80%	70.20%	49.60%	80.10%
Places205-AlexNet	50.04%	81.10%	67.52%	92.61%
Places205-GoogLeNet	55.50%	85.66%	71.60%	95.01%
Places205-VGG	**58.90**%	**87.70**%	**74.60**%	**95.92**%
SamExynos$*$	64.10%	90.65%	-	-
SIAT MMLAB$*$	62.34%	89.66%	-	-

To date (Oct 2, 2016) the top ranked results on the test set of Places205 leaderboard[3] is 64.10% on Top-1 accuracy and 90.65% on Top-5 accuracy. Note that for the test set of SUN205, we did not fine-tune the Places-CNNs on the training set of SUN205, as we directly evaluated them on the test set of SUN.

We further evaluated the baseline Places365-CNNs on the validation set and test set of Places365. The results are shown in Table 2.2. We can see that Places365-VGG and Places365-ResNet have similar top performances compared with the other two CNNs[4]. Even though Places365 has 160 more categories than Places205, the Top-5 accuracy of the Places205-CNNs (trained on the previous version of Places [122]) on the test set only drops by 2.5%.

To evaluate how extra categories bring improvements, we compute the accuracy for the 182 common categories between Places205 and Places365 (we merge some categories in Places205 when building Places365 thus there are less common categories) for Places205-CNN and Places365-CNN. On the validation set of Places365, we select the images of the 182 common categories, then use the aligned 182 outputs of the Places205-AlexNet and Places365-AlexNet to predict the labels respectively. The Top1 accuracy for Places205-AlexNet is 0.572, the one for Places365-AlexNet is 0.577. Thus Places365-AlexNet not

[3] `http://places.csail.mit.edu/user/leaderboard.php`

[4] The performance of the ResNet might result from fine-tuning or under-training, as the ResNet is not trained from scratch.

Table 2.2: Classification accuracy on the validation set and test set of Places365. We use the class score averaged over 10-crops of each testing image to classify the image.

	Validation Set of Places365		Test Set of Places365	
	Top-1 acc.	Top-5 acc.	Top-1 acc.	Top-5 acc.
Places365-AlexNet	53.17%	82.89%	53.31%	82.75%
Places365-GoogLeNet	53.63%	83.88%	53.59%	84.01%
Places365-VGG	**55.24%**	84.91%	**55.19%**	85.01%
Places365-ResNet	54.74%	**85.08%**	54.65%	**85.07%**

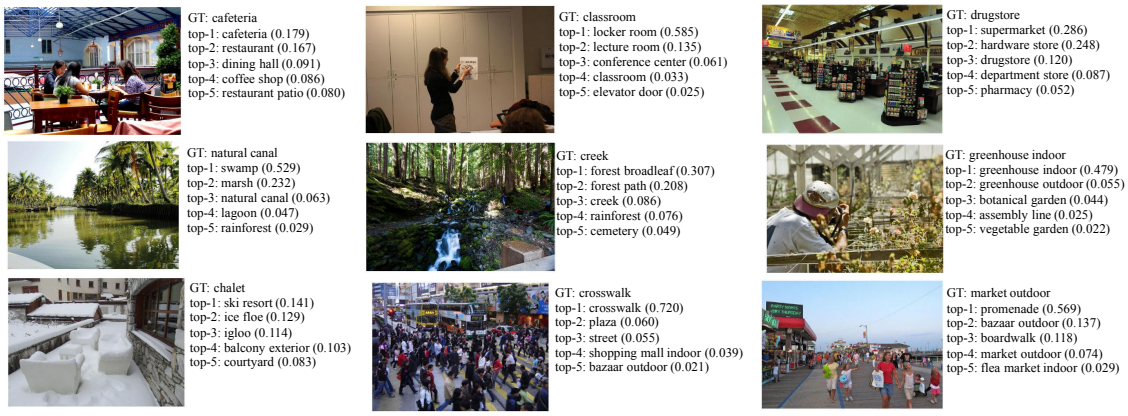


Figure 2-6: The predictions given by the Places365-VGG for the images from the validation set. The ground-truth label (GT) and the top 5 predictions are shown. The number beside each label indicates the prediction confidence.

only predicts more categories, but also has better accuracy on the previous existing categories.

Fig.2-6 shows the responses to examples correctly predicted by the Places365-VGG. Notice that most of the Top-5 responses are very relevant to the scene description.

2.3.2 Generic Visual Features from Places-CNNs and ImageNet-CNNs

We further used the activation from the trained Places-CNNs as generic features for visual recognition tasks using different image classification benchmarks. Activations from the higher-level layers of a CNN, also termed *deep features*, have proven to be effective generic features with state-of-the-art performance on various image datasets [18, 78]. But most of the deep features are from the CNNs trained on ImageNet, which is mostly an object-centric dataset.

Here we evaluated the classification performances of the deep features from scene-centric CNNs and object-centric CNNs in a systematic way. The deep features from several Places-CNNs and ImageNet-CNNs on the following scene and object benchmarks are tested: SUN397 [108], MIT Indoor67 [76], Scene15 [50], SUN Attribute [73], Caltech101 [26], Caltech256 [35], Stanford Action40 [109], and UIUC Event8 [53].

All of the presented experiments follow the standards in the mentioned papers. In the SUN397 experiment [108], the training set size is 50 images per category. Experiments were run on 5 splits of the training set and test set given in the dataset. In the MIT Indoor67 experiment [76], the training set size is 100 images per category. The experiment is run on the split of the training set and test set given in the dataset. In the Scene15 experiment [50], the training set size is 50 images per category. Experiments are run on 10 random splits of the training set and test set. In the SUN Attribute experiment [73], the training set size is 150 images per attribute. The reported result is the average precision. The splits of the training set and test set are given in the paper. In Caltech101 and Caltech256 experiment [26, 35], the training set size is 30 images per category. The experiments are run on 10 random splits of the training set and test set. In the Stanford Action40 experiment [109], the training set size is 100 images per category. Experiments are run on 10 random splits of the training set and test set. The reported result is the classification accuracy. In the UIUC Event8 experiment [53], the training set size is 70 images per category and the test set size is 60 images per category. The experiments are run on 10 random splits of the training set and test set.

Places-CNNs and ImageNet-CNNs have the same network architectures for AlexNet, GoogLeNet, and VGG, but they are trained on scene-centric data (Places) and object-centric data (ImageNet) respectively. For AlexNet and VGG, we used the 4096-dimensional feature vector from the activation of the Fully Connected Layer (`fc7`) of the CNN. For GoogLeNet, we used the 1024-dimensional feature vector from the response of the global average pooling layer before softmax producing the class predictions. The classifier in all of the experiments is a linear SVM with the default parameter for all of the features.

Table 2.3 summarizes the classification accuracy on various datasets for the deep features of Places-CNNs and the deep features of the ImageNet-CNNs. The classifier is a

linear SVM with the same default parameters for the two deep feature layers (C=1) [24]. The Places-CNN features show impressive performance on scene-related datasets, outperforming the ImageNet-CNN features. On the other hand, the ImageNet-CNN features show better performance on object-related image datasets. Importantly, our comparison shows that Places-CNN and ImageNet-CNN have complementary strengths on scene-centric tasks and object-centric tasks, as expected from the type of the datasets used to train these networks. On the other hand, the deep features from the Places365-VGG achieve the best performance (63.24%) on the most challenging scene classification dataset SUN397, while the deep features of Places205-VGG performs the best on the MIT Indoor67 dataset. As far as we know, they are the state-of-the-art scores achieved by a single feature + linear SVM on those two datasets. Furthermore, we merge the 1000 classes from the ImageNet and the 365 classes from the Places365-Standard to train a VGG (Hybrid1365-VGG). The deep feature from the Hybrid1365-VGG achieves the best score averaged over all the eight image datasets.

Table 2.3: Classification accuracy/precision on scene-centric databases (the first four datasets) and object-centric databases (the last four datasets) for the deep features of various Places-CNNs and ImageNet-CNNs. All the accuracy/precision is the top-1 accuracy/precision.

Deep Feature	SUN397	MIT Indoor67	Scene15	SUN Attribute	Caltech101	Caltech256	Action40	Event8	Average
Places365-AlexNet	**56.12**	**70.72**	89.25	92.98	66.40	46.45	46.82	90.63	69.92
Places205-AlexNet	54.32	68.24	**89.87**	**92.71**	65.34	45.30	43.26	94.17	69.15
ImageNet-AlexNet	42.61	56.79	84.05	91.27	**87.73**	**66.95**	**55.00**	**93.71**	72.26
Places365-GoogLeNet	**58.37**	73.30	**91.25**	**92.64**	61.85	44.52	47.52	91.00	70.06
Places205-GoogLeNet	57.00	**75.14**	90.92	92.09	54.41	39.27	45.17	92.75	68.34
ImageNet-GoogLeNet	43.88	59.48	84.95	90.70	**89.96**	**75.20**	**65.39**	**96.13**	75.71
Places365-VGG	**63.24**	76.53	**91.97**	**92.99**	67.63	49.20	52.90	90.96	73.18
Places205-VGG	61.99	**79.76**	91.61	92.07	67.58	49.28	53.33	93.33	73.62
ImageNet-VGG	48.29	64.87	86.28	91.78	**88.42**	**74.96**	**66.63**	**95.17**	77.05
Hybrid1365-VGG	61.77	79.49	92.15	92.93	88.22	76.04	68.11	93.13	**81.48**

Chapter 3

ADE20K for Scene Parsing

Semantic understanding of visual scenes is one of the holy grails of computer vision. The emergence of large-scale image datasets like ImageNet [15], COCO [56] and Places [122], along with the rapid development of the deep convolutional neural network (ConvNet) approaches, have brought great advancements to visual scene understanding. Nowadays, given a visual scene of a living room, a robot equipped with a trained ConvNet can accurately predict the scene category. However, to freely navigate in the scene and manipulate the objects inside, the robot has far more information to digest. It needs to recognize and localize not only the objects like sofa, table, and TV, but also their parts, e.g., a seat of a chair or a handle of a cup, to allow proper interaction, as well as to segment the stuff like floor, wall and ceiling for spatial navigation.

Scene parsing, or recognizing and segmenting objects and stuff in an image, remains one of the key problems in scene understanding. Going beyond image-level recognition, scene parsing requires a much denser annotation of scenes with a large set of objects. However, the current datasets have limited number of objects (e.g., COCO [56], Pascal [22]) and in many cases those objects are not the most common objects one encounters in the world (like frisbees or baseball bats), or the datasets only cover a limited set of scenes (e.g., Cityscapes [14]). Some notable exceptions are Pascal-Context [64] and the SUN database [108]. However, Pascal-Context still contains scenes primarily focused on 20 object classes, while SUN has noisy labels at the object level.

Our goal of the work in this chapter is to collect a dataset that has densely annotated

images (every pixel has a semantic label) with a large and an unrestricted open vocabulary. The images in our dataset are manually segmented in great detail, covering a diverse set of scenes, object and object part categories. The challenges for collecting such annotations include finding reliable annotators, as well as the fact that labeling is difficult if the class list is not defined in advance. On the other hand, open vocabulary naming also suffers from naming inconsistencies across different annotators. In contrast, our dataset was annotated by a single expert annotator, providing extremely detailed and exhaustive image annotations. On average, our annotator labeled 29 annotation segments per image, compared to the 16 segments per image labeled by external annotators (like workers from Amazon Mechanical Turk). Furthermore, the data consistency and quality are much higher than that of external annotators.

The chapter is organized as follows. Firstly we describe the construction and statistics of the ADE20K dataset [124]. Then several semantic segmentation networks are evaluated on the scene parsing benchmark of ADE20K as baselines. We further apply the scene parsing networks to the tasks of automatic scene content removal and scene synthesis.

3.1 ADE20K: Fully Annotated Image Dataset

In this section, we describe our ADE20K dataset and analyze it through a variety of informative statistics.

3.1.1 Dataset summary

There are 20,210 images in the training set, 2,000 images in the validation set, and 3,000 images in the testing set. All the images are exhaustively annotated with objects. Many objects are also annotated with their parts. For each object there is additional information about whether it is occluded or cropped, and other attributes. The images in the validation set are exhaustively annotated with parts, while the part annotations are not exhaustive over the images in the training set. The annotations in the dataset are still growing. Sample images and annotations from the ADE20K dataset are shown in Fig. 3-1.

Figure 3-1: Images in ADE20K dataset are densely annotated in detail with objects and parts. The first row shows the sample images, the second row shows the annotation of objects, and the third row shows the annotation of object parts.

3.1.2 Image annotation

For our dataset, we are interested in having a diverse set of scenes with dense annotations of all the objects present. Images come from the LabelMe [80], SUN datasets [108], and Places [122] and were selected to cover the 900 scene categories defined in the SUN database. Images were annotated by a single expert worker using the LabelMe interface [80]. Fig. 3-2 shows a snapshot of the annotation interface and one fully segmented image. The worker provided three types of annotations: object segments with names, object parts, and attributes. All object instances are segmented independently so that the dataset could be used to train and evaluate detection or segmentation algorithms. Datasets such as COCO [56], Pascal [22] or Cityscape [14] start by defining a set of object categories of interest. However, when labeling all the objects in a scene, working with a predefined list of objects is not possible as new categories appear frequently (see fig. 3-5d). Here, the annotator created a dictionary of visual concepts where new classes were added constantly to ensure consistency in object naming.

Object parts are associated with object instances. Note that parts can have parts too, and we label these associations as well. For example, the 'rim' is a part of a 'wheel', which in turn is part of a 'car'. A 'knob' is a part of a 'door' that can be part of a 'cabinet'. The total part hierarchy has a depth of 3. The object and part hierarchy is in the supplementary materials.

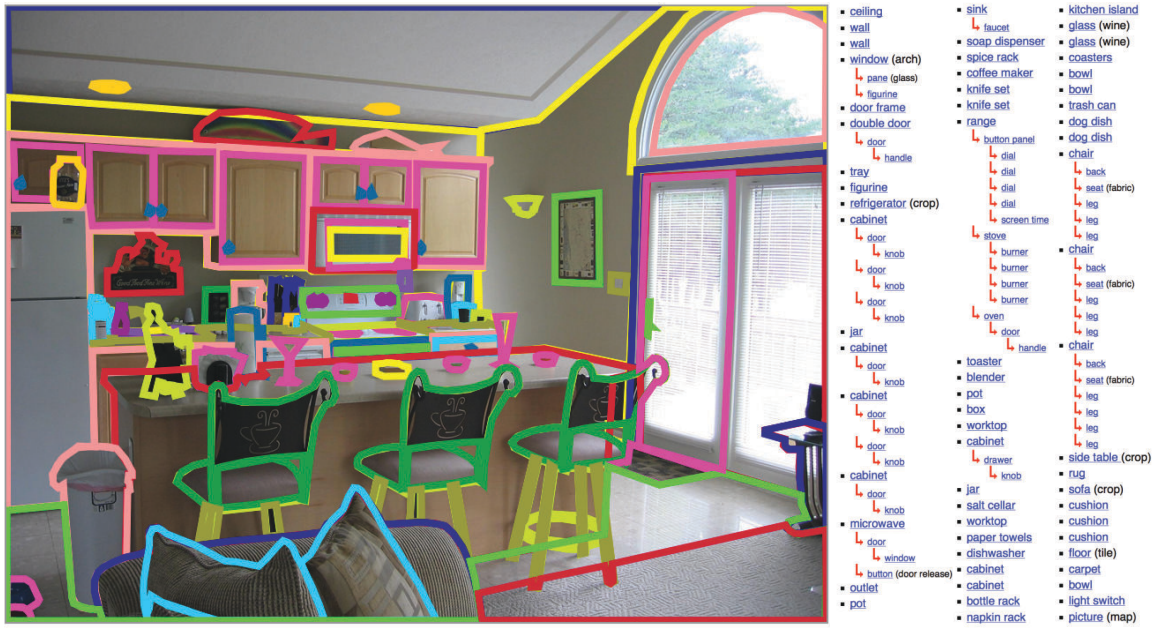


Figure 3-2: Annotation interface, the list of the objects and their associated parts in the image.

3.1.3 Annotation consistency

Defining a labeling protocol is relatively easy when the labeling task is restricted to a fixed list of object classes, however it becomes challenging when the class list is open-ended. As the goal is to label all the objects within each image, the list of classes grows unbounded. Many object classes appear only a few times across the entire collection of images. However, those rare object classes cannot be ignored as they might be important elements for the interpretation of the scene. Labeling in these conditions becomes difficult because we need to keep a growing list of all the object classes in order to have a consistent naming across the entire dataset. Despite the annotator's best effort, the process is not free of noise.

To analyze the annotation consistency we took a subset of 61 randomly chosen images from the validation set, then asked our annotator to annotate them again (there is a time difference of six months). One expects that there are some differences between the two annotations. A few examples are shown in Fig 3-3. On average, 82.4% of the pixels got the same label. The remaining 17.6% of pixels had some errors for which we grouped into three error types as follows:

- **Segmentation quality**: Variations in the quality of segmentation and outlining of

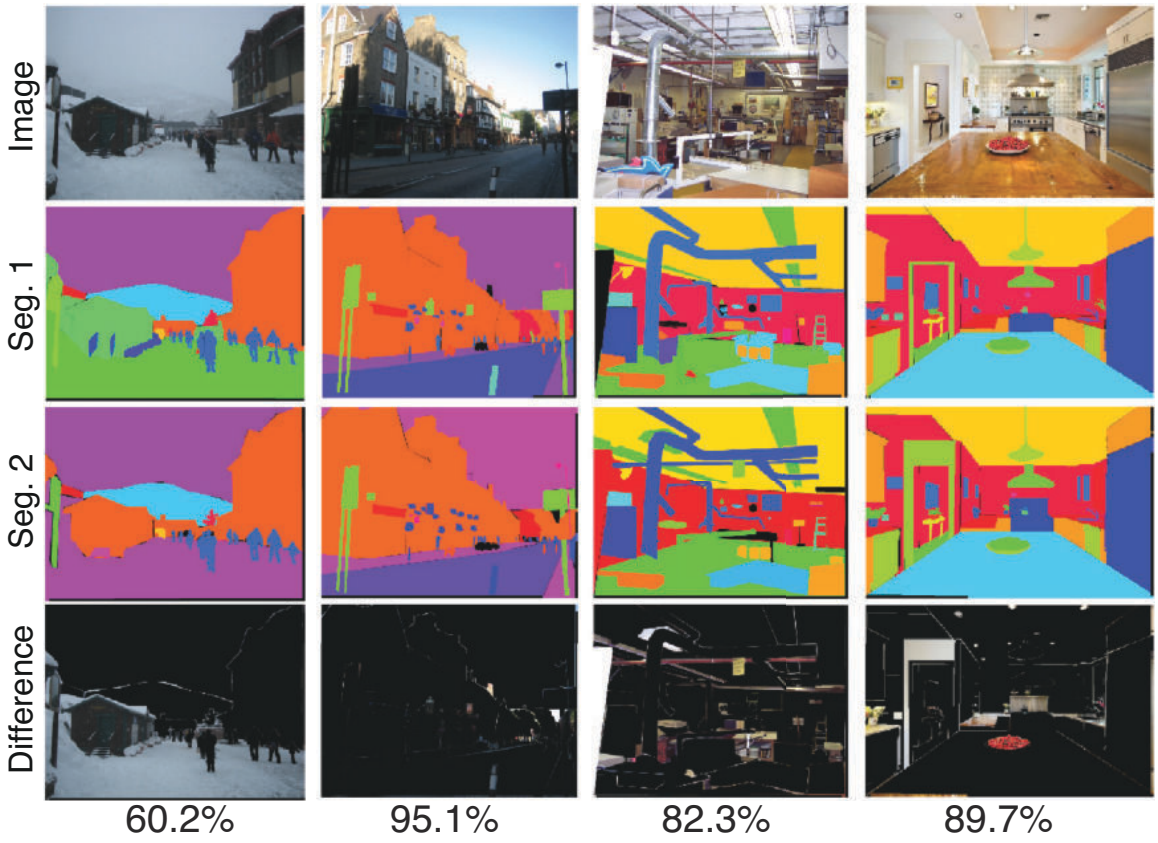


Figure 3-3: Analysis of annotation consistency. Each column shows an image and two segmentations done by the same annotator at different times. Bottom row shows the pixel discrepancy when the two segmentations are subtracted, and the percentage of pixels with the same label. On average across all re-annotated images, 82.4% of pixels got the same label. In the example in the first column the percentage of pixels with the same label is relatively low because the annotator labeled the same region as 'snow' and 'ground' during the two rounds of annotation. In the third column, there were many objects in the scene and the annotator missed some between the two segmentations.

the object boundary. One typical source of error arises when segmenting complex objects such as buildings and trees, which can be segmented with different degrees of precision. 5.7% of the pixels had this type of error.

- **Object naming**: Differences in object naming (due to ambiguity or similarity between concepts, for instance, calling a big car a 'car' in one segmentation and a 'truck' in the another one, or a 'palm tree' a 'tree'. 6.0% of the pixels had naming issues. These errors can be reduced by defining a very precise terminology, but this becomes much harder with a large growing vocabulary.

- **Segmentation quantity**: Missing objects in one of the two segmentations. There is a very large number of objects in each image and some images might be annotated more thoroughly than others. For example, in the third column of Fig 3-3 the annotator missed some small objects in different annotations. 5.9% of the pixels are due to missing labels. A similar issue existed in segmentation datasets such as the Berkeley Image segmentation dataset [61].

The median error values for the three error types are: 4.8%, 0.3% and 2.6% showing that the mean value is dominated by a few images, and that the most common type of error is segmentation quality.

To further compare the annotation done by our single expert annotator and the AMT-like annotators, 20 images from the validation set are annotated by two invited external annotators, both with prior experience in image labeling. The first external annotator had 58.5% of inconsistent pixels compared to the segmentation provided by our annotator, and the second external annotator had 75% of the inconsistent pixels. Many of these inconsistencies are due to the poor quality of the segmentations provided by external annotators (as it has been observed with AMT which requires multiple verification steps for quality control [56]). For the best external annotator (the first one), 7.9% of pixels have inconsistent segmentations (just slightly worse than our annotator), 14.9% have inconsistent object naming and 35.8% of the pixels correspond to missing objects, which is due to the much smaller number of objects annotated by the external annotator in comparison with the ones

annotated by our expert annotator. The external annotators labeled on average 16 segments per image while our annotator provided 29 segments per image.

3.1.4 Dataset statistics

Fig. 3-4a shows the distribution of ranked object frequencies. The distribution is similar to a Zipf's law and is typically found when objects are exhaustively annotated in images [108]. They differ from the ones from datasets such as COCO or ImageNet where the distribution is more uniform resulting from manual balancing.

Fig. 3-4b shows the distributions of annotated parts grouped by the objects they belong to and sorted by frequency within each object class. Most object classes also have a non-uniform distribution of part counts. Fig. 3-4c and Fig. 3-4d show how objects are shared across scenes and how parts are shared by objects. Fig. 3-4e shows the variability in the appearances of the part 'door'.

The mode of the object segmentations is shown in Fig. 3-5a and contains the four objects (from top to bottom): 'sky', 'wall', 'building' and 'floor'. When using simply the mode to segment the images, it gets, on average, 20.9% of the pixels of each image right. Fig. 3-5b shows the distribution of images according to the number of distinct classes and instances. On average there are 19.5 instances and 10.5 object classes per image, larger than other existing datasets (see Table 3.1). Fig. 3-5c shows the distribution of parts.

As the list of object classes is not predefined, there are new classes appearing over time of annotation. Fig. 3-5d shows the number of object (and part) classes as the number of annotated instances increases. Fig. 3-5e shows the probability that instance $n + 1$ is a new class after labeling n instances. The more segments we have, the smaller the probability that we will see a new class. At the current state of the dataset, we get one new object class every 300 segmented instances.

3.1.5 Comparison with other datasets

We compare ADE20K with existing datasets in Tab. 3.1. Compared to the largest annotated datasets, COCO [56] and Imagenet [79], our dataset comprises of much more diverse

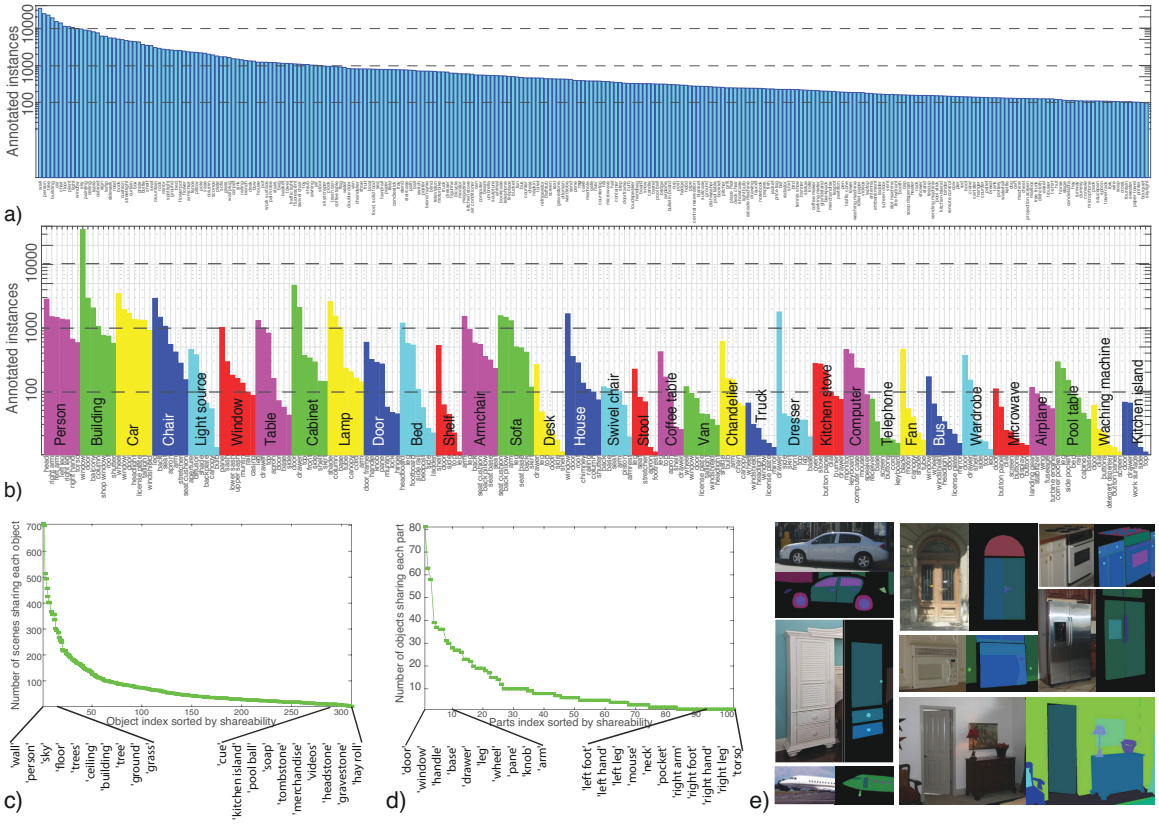


Figure 3-4: a) Object classes sorted by frequency. Only the top 270 classes with more than 100 annotated instances are shown. 68 classes have more than a 1000 segmented instances. b) Frequency of parts grouped by objects. There are more than 200 object classes with annotated parts. Only objects with 5 or more parts are shown in this plot (we show at most 7 parts for each object class). c) Objects ranked by the number of scenes they are part of. d) Object parts ranked by the number of objects they are part of. e) Examples of objects with doors. The bottom-right image is an example where the door does not behave as a part.

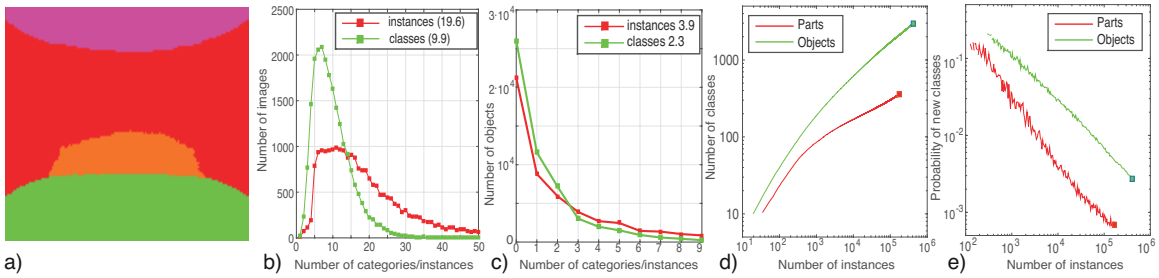


Figure 3-5: a) Mode of the object segmentations contains 'sky', 'wall', 'building' and 'floor'. b) Histogram of the number of segmented object instances and classes per image. c) Histogram of the number of segmented part instances and classes per object. d) Number of classes as a function of segmented instances (objects and parts). The squares represent the current state of the dataset. e) Probability of seeing a new object (or part) class as a function of the number of instances.

Table 3.1: Comparison of semantic segmentation datasets.

	Images	Obj. Inst.	Obj. Cls.	Part Inst.	Part Cls.	Obj. Cls. per Img.
COCO	123,287	886,284	91	0	0	3.5
ImageNet*	476,688	534,309	200	0	0	1.7
NYU Depth V2	1,449	34,064	894	0	0	14.1
Cityscapes	25,000	65,385	30	0	0	12.2
SUN	16,873	313,884	4,479	0	0	9.8
OpenSurfaces	22,214	71,460	160	0	0	N/A
PascalContext	10,103	∼104,398**	540	181,770	40	5.1
ADE20K	22,210	434,826	2,693	175,961	476	9.9

* has only bounding boxes (no pixel-level segmentation). Sparse annotations.
** PascalContext dataset does not have instance segmentation. In order to estimate the number of instances, we find connected components (having at least 150pixels) for each class label.

scenes, where the average number of object classes per image is 3 and 6 times larger, respectively. With respect to SUN [108], ADE20K is roughly 35% larger in terms of images and object instances. However, the annotations in our dataset are much richer since they also include segmentation at the part level. Such annotation is only available for the Pascal-Context/Part dataset [11,64] which contains 40 distinct part classes across 20 object classes. Note that we merged some of their part classes to be consistent with our labeling (e.g., we mark both *left leg* and *right leg* as the same semantic part *leg*). Since our dataset contains part annotations for a much wider set of object classes, the number of part classes is almost 9 times larger in our dataset.

An interesting fact is that any image in ADE20K contains at least 5 objects, and the maximum number of object instances per image reaches 273, and 419 instances, when counting parts as well. This shows the high annotation complexity of our dataset.

3.2 Scene Parsing Benchmark and Networks

Based on the data of the ADEK20K, we construct scene parsing benchark. Scene parsing is to segment the whole image densely into semantic classes, where each pixel is assigned a class label such as the region of *tree* and the region of *building*.

We select the top 150 categories ranked by their total pixel ratios[1] in the ADE20K

[1]As the original images in the ADE20K dataset have various sizes, for simplicity we rescale those large-sized images to make their minimum heights or widths as 512 in the SceneParse150 benchmark.

Table 3.2: Baseline performance on the validation set of SceneParse150.

Networks	Pixel Acc.	Mean Acc.	Mean IoU	Weighted IoU
FCN-8s	71.32%	40.32%	0.2939	0.5733
SegNet	71.00%	31.14%	0.2164	0.5384
DilatedVGG	73.55%	44.59%	0.3231	0.6014
DilatedResNet-34	76.47%	45.84%	0.3277	0.6068
DilatedResNet-50	76.40%	45.93%	0.3385	0.6100
Cascade-SegNet	71.83%	37.90%	0.2751	0.5805
Cascade-DilatedVGG	74.52%	45.38%	0.3490	0.6108

dataset and build a scene parsing benchmark of ADE20K, termed as *SceneParse150*. Among the 150 categories, there are 35 stuff classes (*i.e., wall, sky, road*) and 115 discrete object classes (*i.e., car, person, table*). The annotated pixels of the 150 classes occupy 92.75% of all the pixels of the dataset, where the stuff classes occupy 60.92%, and discrete object classes occupy 31.83%.

As for baseline networks for scene parsing on our benchmark, we train several semantic segmentation networks: SegNet [4], FCN-8s [57], DilatedVGG, DilatedResNet [10, 111], and two cascade networks proposed in [124] where the backbone models are SegNet and DilatedVGG.

The segmentation performance of the baseline networks on SceneParse150 is listed in Table 3.2. Among the baselines, the networks based on dilated convolutions achieve better results in general than FCN and SegNet. Using the cascade framework, the performance further improves. In terms of mean IoU, Cascade-SegNet and Cascade-DilatedVGG outperform SegNet and DilatedVGG by 6% and 2.5%, respectively.

Qualitative scene parsing results from the validation set are shown in Fig. 3-6. We observe that all the baseline networks can give correct predictions for the common, large object and stuff classes, the difference in performance comes mostly from small, infrequent objects and how well they handle details.

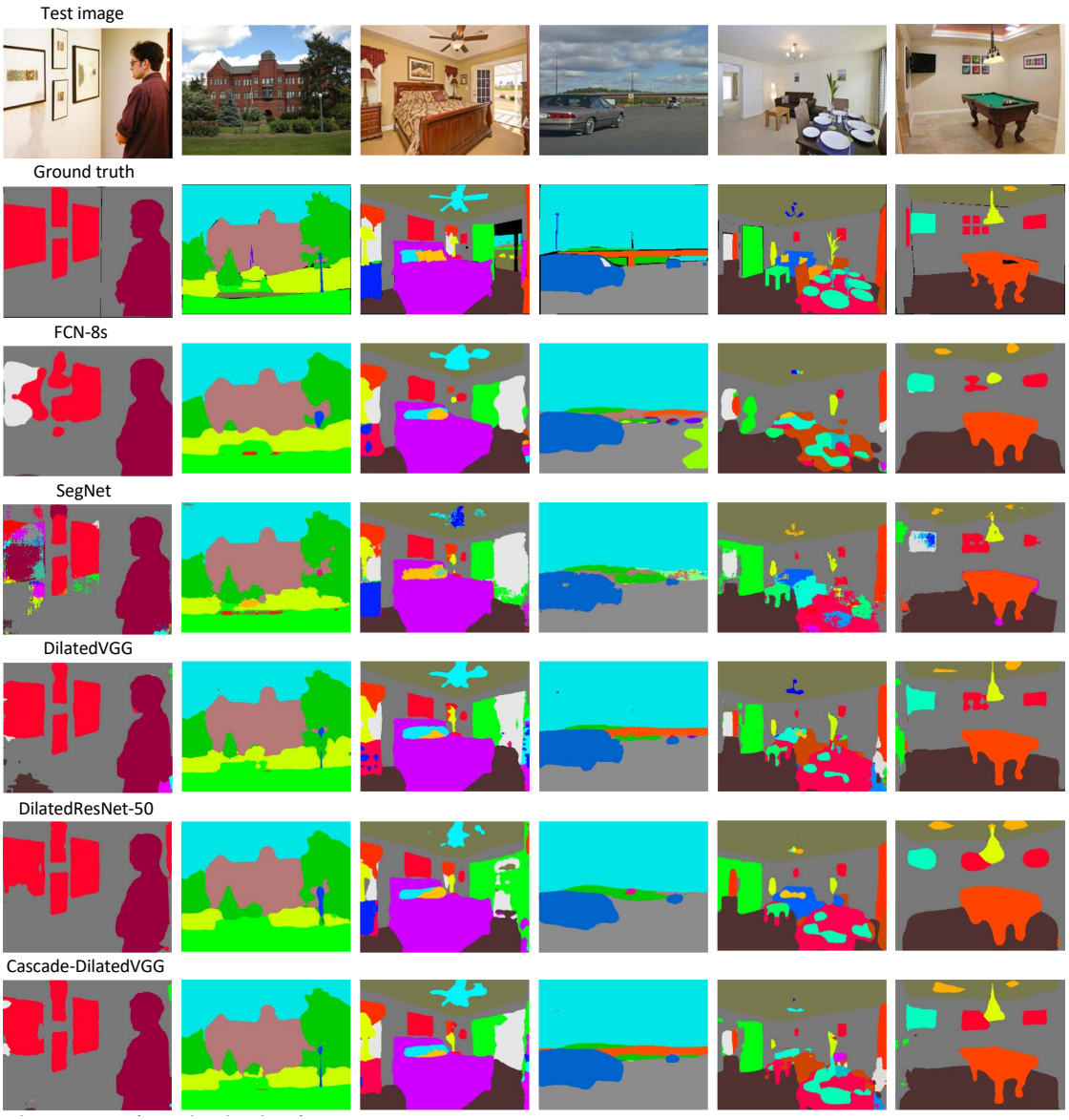


Figure 3-6: Ground-truths, scene parsing results given by the baseline networks. All networks can give correct predictions for the common, large object and stuff classes, the difference in performance comes mostly from small, infrequent objects and how well they handle details.

Chapter 4

Comparing the Deep Visual Representations for Objects and Scenes

When training a CNN to distinguish different object classes, it is unclear what the underlying representation should be. Objects have often been described using part-based representations where parts can be shared across objects, forming a distributed code. However, what those parts should be is unclear. For instance, one would think that the meaningful parts of a face are the mouth, the two eyes, and the nose. However, those are simply functional parts, with words associated with them; the object parts that are important for visual recognition might be different from these semantic parts, making it difficult to evaluate how efficient a representation is. In fact, the strong internal configuration of objects makes the definition of what is a useful part poorly constrained: an algorithm can find different and arbitrary part configurations, all giving similar recognition performance.

Learning to classify scenes (i.e., classifying an image as being an office, a restaurant, a street, etc) using the Places dataset [122] gives the opportunity to study the internal representation learned by a CNN on a task other than object recognition. In the case of scenes, the representation is clearer. Scene categories are defined by the objects they contain and, to some extent, by the spatial configuration of those objects. For instance, the important parts of a bedroom are the bed, a side table, a lamp, a cabinet, as well as the walls, floor and ceiling. Objects represent therefore a distributed code for scenes (i.e., object classes are shared across different scene categories). Importantly, in scenes, the spatial configu-

ration of objects, although compact, has a much larger degree of freedom. It is this loose spatial dependency that, we believe, makes scene representation different from most object classes (most object classes do not have a loose interaction between parts). While a CNN has enough flexibility to learn any of those representations, if meaningful objects emerge without supervision inside the inner layers of the CNN, there will be little ambiguity as to which type of representation these networks are learning.

The main contribution of this chapter is to show that object detection emerges inside a CNN trained to recognize scenes, even more than when trained with ImageNet. This is surprising because our results demonstrate that reliable object detectors are found even though, unlike ImageNet, no supervision is provided for objects. Although object discovery with deep neural networks has been shown before in an unsupervised setting [51], here we find that many more objects can be naturally discovered, in a supervised setting tuned to scene classification rather than object classification. Importantly, the emergence of object detectors inside the CNN suggests that a single network can support recognition at several levels of abstraction (e.g., edges, texture, objects, and scenes) without needing multiple outputs or a collection of networks.

4.1 ImageNet-CNN and Places-CNN

Convolutional neural networks have recently obtained astonishing performance on object classification [48] and scene classification [122]. The ImageNet-CNN from [48] is trained on 1.3 million images from 1000 object categories of ImageNet (ILSVRC 2012) and achieves a top-1 accuracy of 57.4%. With the same network architecture, Places-CNN is trained on 2.4 million images from 205 scene categories of Places Database [122], and achieves a top-1 accuracy of 50.0%. The network architecture used for both CNNs, as proposed in [48], is summarized in Table 4.1[1]. Both networks are trained from scratch using only the specified dataset.

The deep features from Places-CNN tend to perform better on scene-related recognition tasks compared to the features from ImageNet-CNN. For example, as compared to the

[1]We use *unit* to refer to neurons in the various layers and *features* to refer to their activations.

Table 4.1: The parameters of the network architecture used for ImageNet-CNN and Places-CNN.

Layer	conv1	pool1	conv2	pool2	conv3	conv4	conv5	pool5	fc6	fc7
Units	96	96	256	256	384	384	256	256	4096	4096
Feature	55×55	27×27	27×27	13×13	13×13	13×13	13×13	6×6	1	1

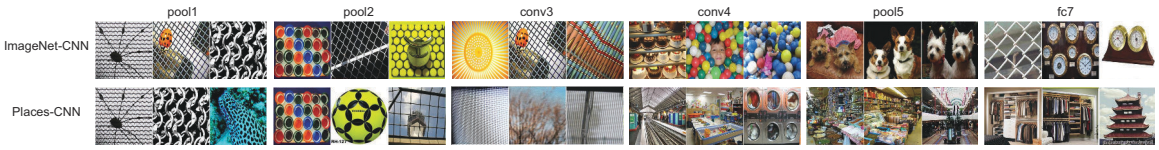


Figure 4-1: Top 3 images producing the largest activation of units in each layer of ImageNet-CNN (top) and Places-CNN (bottom).

Places-CNN that achieves 50.0% on scene classification, the ImageNet-CNN combined with a linear SVM only achieves 40.8% on the same test set[2] illustrating the importance of having scene-centric data.

To further highlight the difference in representations, we conduct a simple experiment to identify the differences in the type of images preferred at the different layers of each network: we create a set of 200k images with an approximately equal distribution of scene-centric and object-centric images[3], and run them through both networks, recording the activations at each layer. For each layer, we obtain the top 100 images that have the largest average activation (sum over all spatial locations for a given layer). Fig. 4-1 shows the top 3 images for each layer. We observe that the earlier layers such as pool1 and pool2 prefer similar images for both networks while the later layers tend to be more specialized to the specific task of scene or object categorization. For layer pool2, 55% and 47% of the top-100 images belong to the ImageNet dataset for ImageNet-CNN and Places-CNN. Starting from layer conv4, we observe a significant difference in the number of top-100 belonging to each dataset corresponding to each network. For fc7, we observe that 78% and 24% of the top-100 images belong to the ImageNet dataset for the ImageNet-CNN and Places-CNN respectively, illustrating a clear bias in each network.

In the following sections, we further investigate the differences between these networks,

[2]Scene recognition demo of Places-CNN is available at `http://places.csail.mit.edu/demo.html`. The demo has 77.3% top-5 recognition rate in the wild estimated from 968 anonymous user responses.

[3]100k object-centric images from the test set of ImageNet LSVRC2012 and 108k scene-centric images from the SUN dataset [108].

and focus on better understanding the nature of the representation learned by Places-CNN when doing scene classification in order to clarify some part of the secret to their great performance.

4.2 Uncovering the CNN representation

The performance of scene recognition using Places-CNN is quite impressive given the difficulty of the task. In this section, our goal is to understand the nature of the representation that the network is learning.

4.2.1 Simplifying the input images

Simplifying images is a well known strategy to test human recognition. For example, one can remove information from the image to test if it is diagnostic or not of a particular object or scene (for a review see [8]). A similar procedure was also used by [96] to understand the receptive fields of complex cells in the inferior temporal cortex (IT).

Inspired by these approaches, our idea is the following: given an image that is correctly classified by the network, we want to simplify this image such that it keeps as little visual information as possible while still having a high classification score for the same category. This simplified image (named minimal image representation) will allow us to highlight the elements that lead to the high classification score. In order to do this, we manipulate images in the gradient space as typically done in computer graphics [74]. We investigate two different approaches described below.

In the first approach, given an image, we create a segmentation of edges and regions and remove segments from the image iteratively. At each iteration we remove the segment that produces the smallest decrease of the correct classification score and we do this until the image is incorrectly classified. At the end, we get a representation of the original image that contains, approximately, the minimal amount of information needed by the network to correctly recognize the scene category. In Fig. 4-2 we show some examples of these minimal image representations. Notice that objects seem to contribute important information for the network to recognize the scene. For instance, in the case of bedrooms these minimal

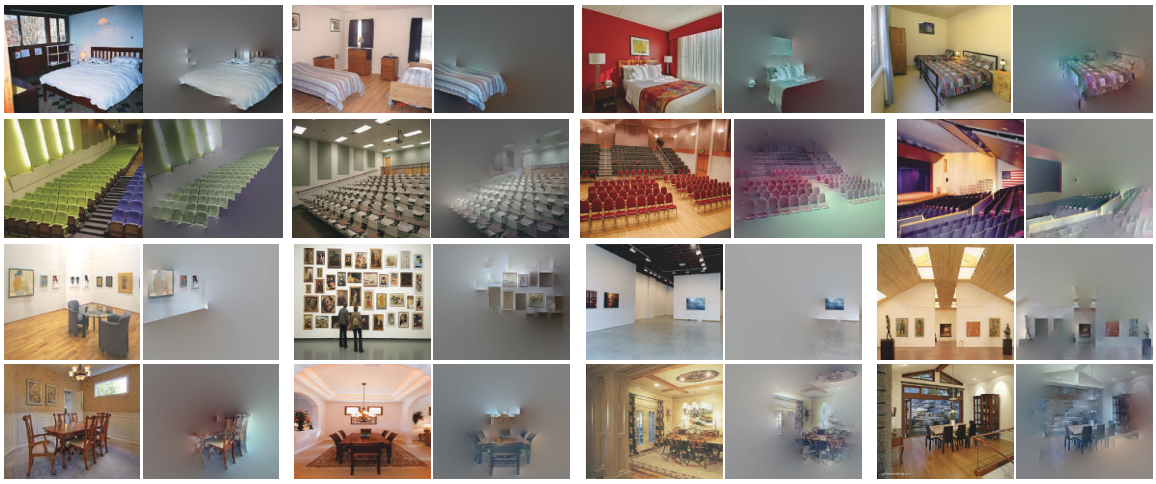


Figure 4-2: Each pair of images shows the original image (left) and a simplified image (right) that gets classified by the Places-CNN as the same scene category as the original image. From top to bottom, the four rows show different scene categories: bedroom, auditorium, art gallery, and dining room.

image representations usually contain the region of the bed, or in the art gallery category, the regions of the paintings on the walls.

Based on the previous results, we hypothesized that for the Places-CNN, some objects were crucial for recognizing scenes. This inspired our second approach: we generate the minimal image representations using the fully annotated image set of SUN Database [108] (see section 4.3.1 for details on this dataset) instead of performing automatic segmentation. We follow the same procedure as the first approach using the ground-truth object segments provided in the database.

This led to some interesting observations: for bedrooms, the minimal representations retained the bed in 87% of the cases. Other objects kept in bedrooms were wall (28%) and window (21%). For art gallery the minimal image representations contained paintings (81%) and pictures (58%); in amusement parks, carousel (75%), ride (64%), and roller coaster (50%); in bookstore, bookcase (96%), books (68%), and shelves (67%). These results suggest that object detection is an important part of the representation built by the network to obtain discriminative information for scene classification.

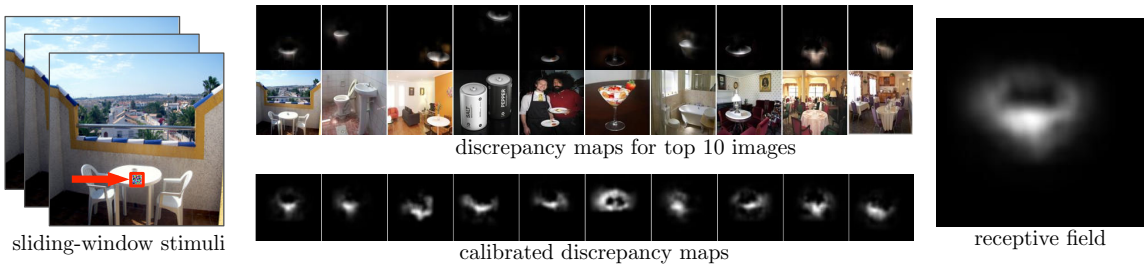


Figure 4-3: The pipeline for estimating the RF of each unit. Each sliding-window stimuli contains a small randomized patch (example indicated by red arrow) at different spatial locations. By comparing the activation response of the sliding-window stimuli with the activation response of the original image, we obtain a discrepancy map for each image (middle top). By summing up the calibrated discrepancy maps (middle bottom) for the top ranked images, we obtain the actual RF of that unit (right).

4.2.2 Visualizing the receptive fields of units and their activation patterns

In this section, we investigate the shape and size of the receptive fields (RFs) of the various units in the CNNs. While theoretical RF sizes can be computed given the network architecture [57], we are interested in the actual, or *empirical* size of the RFs. We expect the empirical RFs to be better localized and more representative of the information they capture than the theoretical ones, allowing us to better understand what is learned by each unit of the CNN.

Thus, we propose a data-driven approach to estimate the learned RF of each unit in each layer. It is simpler than the deconvolutional network visualization method [112] and can be easily extended to visualize any learned CNNs[4].

The procedure for estimating a given unit's RF, as illustrated in Fig. 4-3, is as follows. As input, we use an image set of 200k images with a roughly equal distribution of scenes and objects (similar to Sec. 4.1). Then, we select the top K images with the highest activations for the given unit.

For each of the K images, we now want to identify exactly which regions of the image lead to the high unit activations. To do this, we replicate each image many times with small random occluders (image patches of size 11×11) at different locations in the image.

[4]More visualizations are available at `http://places.csail.mit.edu/visualization`

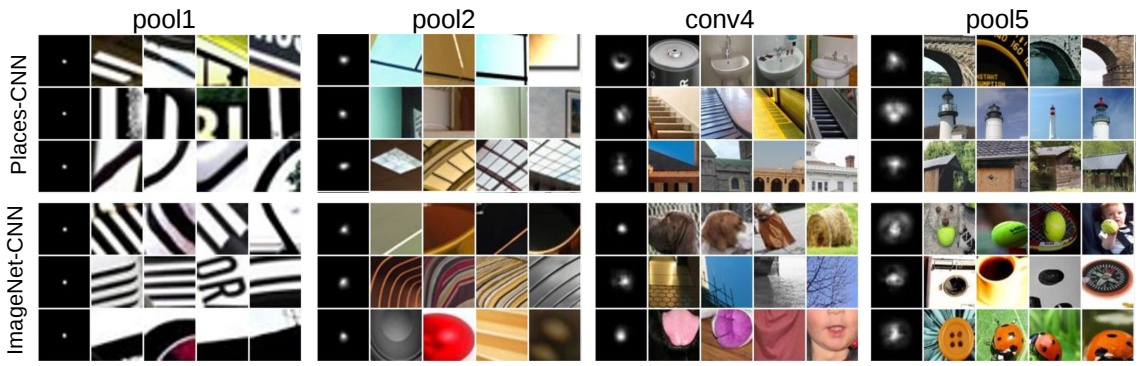


Figure 4-4: The RFs of 3 units of pool1, pool2, conv4, and pool5 layers respectively for ImageNet- and Places-CNNs, along with the image patches corresponding to the top activation regions inside the RFs.

Specifically, we generate occluders in a dense grid with a stride of 3. This results in about 5000 occluded images per original image. We now feed all the occluded images into the same network and record the change in activation as compared to using the original image. If there is a large discrepancy, we know that the given patch is important and vice versa. This allows us to build a discrepancy map for each image.

Finally, to consolidate the information from the K images, we center the discrepancy map around the spatial location of the unit that caused the maximum activation for the given image. Then we average the re-centered discrepancy maps to generate the final RF.

In Fig. 4-4 we visualize the RFs for units from 4 different layers of the Places-CNN and ImageNet-CNN, along with their highest scoring activation regions inside the RF. We observe that, as the layers go deeper, the RF size gradually increases and the activation regions become more semantically meaningful. Further, as shown in Fig. 4-5, we use the RFs to segment images using the feature maps of different units. Lastly, in Table 4.2, we compare the theoretical and empirical size of the RFs at different layers. As expected, the actual size of the RF is much smaller than the theoretical size, especially in the later layers. Overall, this analysis allows us to better understand each unit by focusing precisely on the important regions of each image.

Table 4.2: Comparison of the theoretical and empirical sizes of the RFs for Places-CNN and ImageNet-CNN at different layers. Note that the RFs are assumed to be square shaped, and the sizes reported below are the length of each side of this square, in pixels.

	pool1	pool2	conv3	conv4	pool5
Theoretic size	19	67	99	131	195
Places-CNN actual size	17.8± 1.6	37.4± 5.9	52.1±10.6	60.0± 13.7	72.0± 20.0
ImageNet-CNN actual size	17.9± 1.6	36.7± 5.4	51.1±9.9	60.4± 16.0	70.3± 21.6

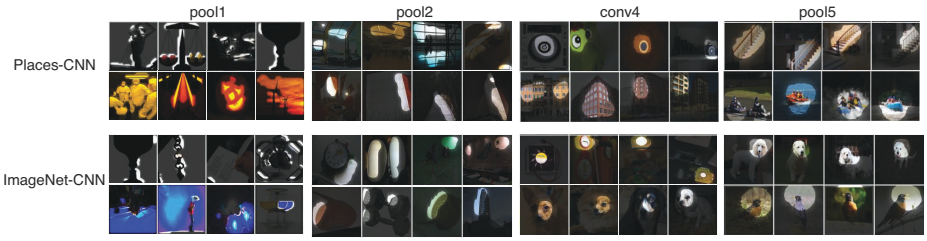


Figure 4-5: Segmentation based on RFs. Each row shows the 4 most confident images for some unit.

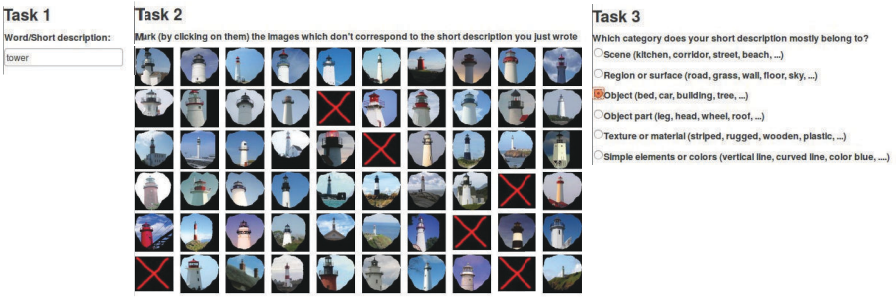


Figure 4-6: AMT interface for unit concept annotation. There are three tasks in each annotation.

4.2.3 Identifying the semantics of internal units

In Section 4.2.2, we found the exact RFs of units and observed that activation regions tended to become more semantically meaningful with increasing depth of layers. In this section, our goal is to understand and quantify the precise semantics learned by each unit.

In order to do this, we ask workers on Amazon Mechanical Turk (AMT) to identify the common theme or *concept* that exists between the top scoring segmentations for each unit. We expect the tags provided by naive annotators to reduce biases. Workers provide tags without being constrained to a dictionary of terms that could bias or limit the identification of interesting properties.

Specifically, we divide the task into three main steps as shown in Fig. 4-6. We show workers the top 60 segmented images that most strongly activate one unit and we ask them to (1) identify the concept, or semantic theme given by the set of 60 images e.g., car, blue, vertical lines, etc, (2) mark the set of images that do not fall into this theme, and (3) categorize the concept provided in (1) to one of 6 semantic groups ranging from low-level to high-level: simple elements and colors (e.g., horizontal lines, blue), materials and textures (e.g., wood, square grid), regions ans surfaces (e.g., road, grass), object parts (e.g., head, leg), objects (e.g., car, person), and scenes (e.g., kitchen, corridor). This allows us to obtain both the semantic information for each unit, as well as the level of abstraction provided by the labeled concept.

To ensure high quality of annotation, we included 3 images with high negative scores that the workers were required to identify as negatives in order to submit the task. Fig. 4-7 shows some example annotations by workers. For each unit, we measure its precision as the percentage of images that were selected as fitting the labeled concept. In Fig. 4-8.(a) we plot the average precision for ImageNet-CNN and Places-CNN for each layer.

In Fig. 4-8b-c we plot the distribution of concept categories for ImageNet-CNN and Places-CNN at each layer. For this plot we consider only units that had a precision above 75% as provided by the AMT workers. Around 60% of the units on each layer where above that threshold. For both networks, units at the early layers (pool1, pool2) have more units responsive to simple elements and colors, while those at later layers (conv4, pool5)

have more high-level semantics (responsive more to objects and scenes). Furthermore, we observe that conv4 and pool5 units in Places-CNN have higher ratios of high-level semantics as compared to the units in ImageNet-CNN.

Fig. 4-9 provides a different visualization of the same data as in Fig. 4-8b-c. This plot better reveals how different levels of abstraction emerge in different layers of both networks. The vertical axis indicates the percentage of units in each layer assigned to each concept category. ImageNet-CNN has more units tuned to simple elements and colors than Places-CNN while Places-CNN has more objects and scenes. ImageNet-CNN has more units tuned to object parts (with the maximum around conv4). It is interesting to note that Places-CNN discovers more objects than ImageNet-CNN despite having no object-level supervision.

4.3 Emergence of objects as the internal representation

As shown before, a large number of units in pool5 are devoted to detecting objects and scene-regions (Fig. 4-9). But what categories are found? Is each category mapped to a single unit or are there multiple units for each object class? Can we actually use this information to segment a scene?

4.3.1 What object classes emerge

To answer the question of why certain objects emerge from pool5, we tested ImageNet-CNN and Places-CNN on fully annotated images from the SUN database [108]. The SUN database contains 8220 fully annotated images from the same 205 place categories used to train Places-CNN. There are no duplicate images between SUN and Places. We use SUN instead of COCO [56] as we need dense object annotations to study what the most informative object classes for scene categorization are, and what the natural object frequencies in scene images are. For this study, we manually mapped the tags given by AMT workers to the SUN categories.

Fig. 4-10a shows the distribution of objects found in pool5 of Places-CNN. Some objects are detected by several units. For instance, there are 15 units that detect buildings.

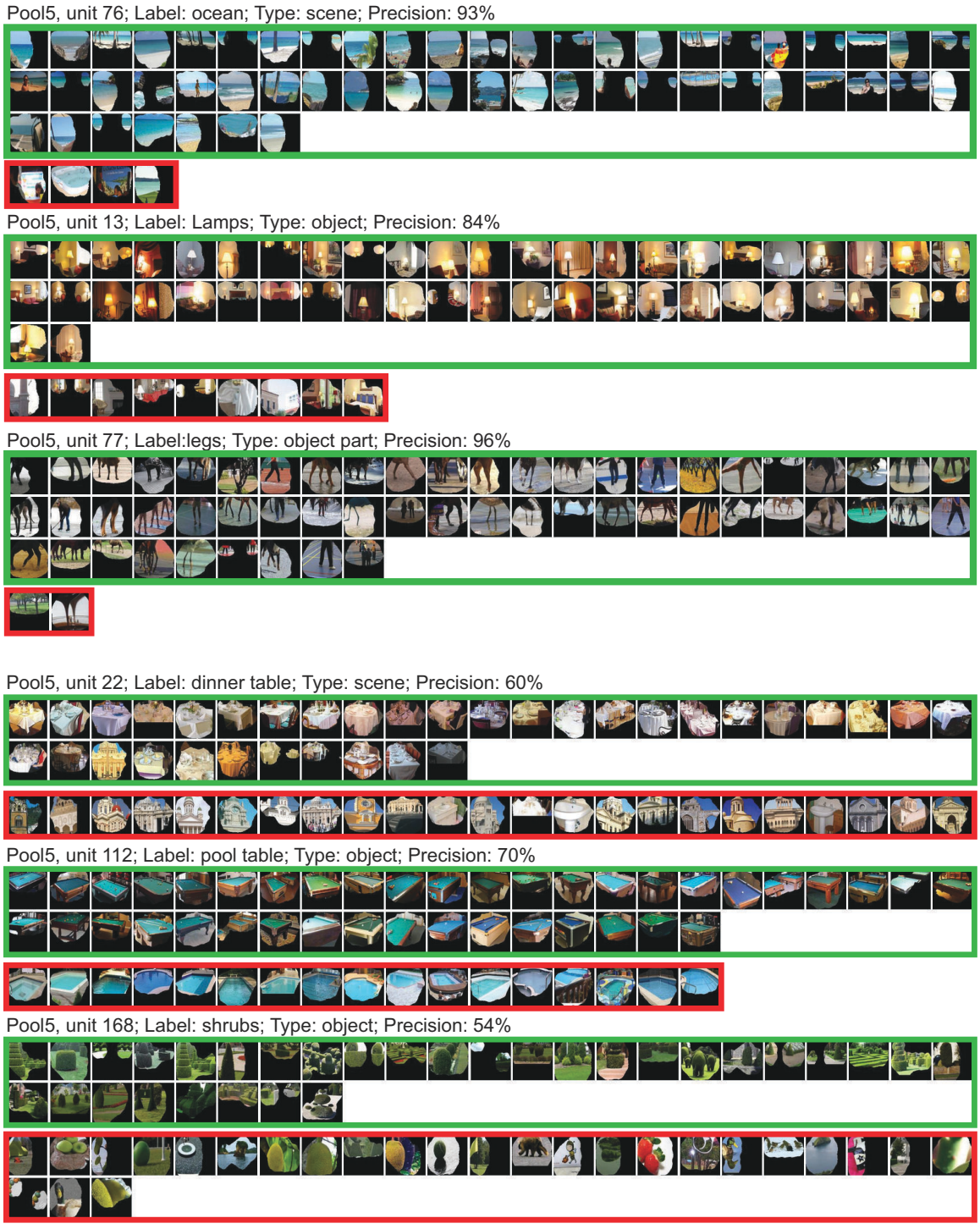


Figure 4-7: Examples of unit annotations provided by AMT workers for 6 units from pool5 in Places-CNN. For each unit the figure shows the label provided by the worker, the type of label, the images selected as corresponding to the concept (green box) and the images marked as incorrect (red box). The precision is the percentage of correct images. The top three units have high performance while the bottom three have low performance ($< 75\%$).

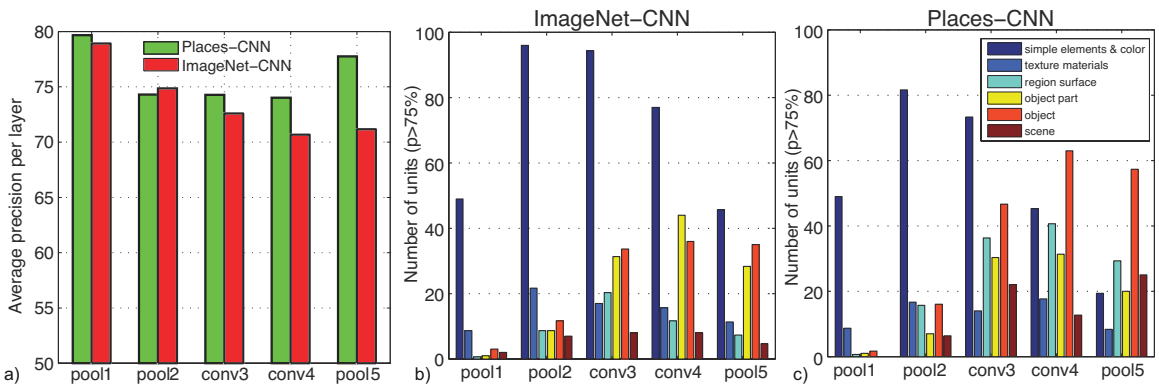


Figure 4-8: (a) Average precision of all the units in each layer for both networks as reported by AMT workers. (b) and (c) show the number of units providing different levels of semantics for ImageNet-CNN and Places-CNN respectively.

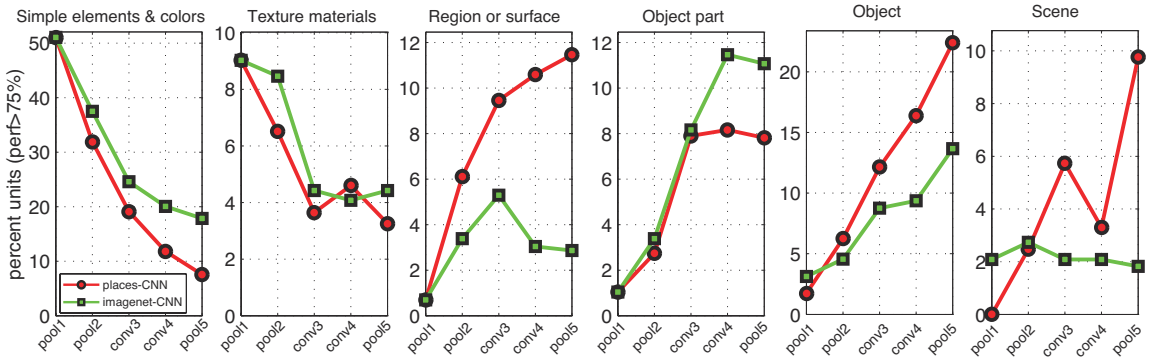


Figure 4-9: Distribution of semantic types found for all the units in both networks. From left to right, each plot corresponds to the distribution of units in each layer assigned to simple elements or colors, textures or materials, regions or surfaces, object parts, objects, and scenes. The vertical axis is the percentage of units with each layer assigned to each type of concept.

Fig. 4-11 shows some units from the Places-CNN grouped by the type of object class they seem to be detecting. Each row shows the top five images for a particular unit that produce the strongest activations. The segmentation shows the regions of the image for which the unit is above a certain threshold. Each unit seems to be selective to a particular appearance of the object. For instance, there are 6 units that detect lamps, each unit detecting a particular type of lamp providing finer-grained discrimination; there are 9 units selective to people, each one tuned to different scales or people doing different tasks.

Fig. 4-10b shows the distribition of objects found in pool5 of ImageNet-CNN. ImageNet has an abundance of animals among the categories present: in the ImageNet-CNN,

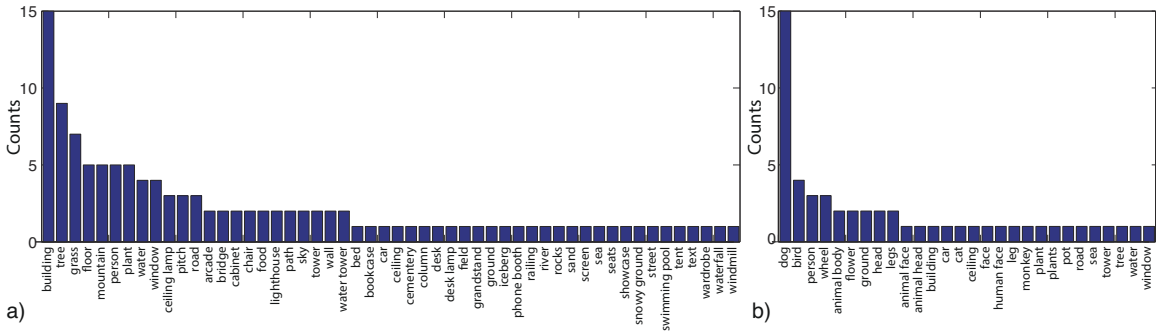


Figure 4-10: Object counts of CNN units discovering each object class for (a) Places-CNN and (b) ImageNet-CNN.

out of the 256 units in pool5, there are 15 units devoted to detecting dogs and several more detecting parts of dogs (body, legs, ...). The categories found in pool5 tend to follow the target categories in ImageNet.

Why do those objects emerge? One possibility is that the objects that emerge in pool5 correspond to the most frequent ones in the database. Fig. 4-12a shows the sorted distribution of object counts in the SUN database which follows Zipf's law. Fig. 4-12b shows the counts of units found in pool5 for each object class (same sorting as in Fig. 4-12a). The correlation between object frequency in the database and object frequency discovered by the units in pool5 is 0.54. Another possibility is that the objects that emerge are the objects that allow discriminating among scene categories. To measure the set of discriminant objects we used the ground truth in the SUN database to measure the classification performance achieved by each object class for scene classification. Then we count how many times each object class appears as the most informative one. This measures the number of scene categories a particular object class is the most useful for. The counts are shown in Fig. 4-12c. Note the similarity between Fig. 4-12b and Fig. 4-12c. The correlation is 0.84 indicating that the network is automatically identifying the most discriminative object categories to a large extent.

Note that there are 115 units in pool5 of Places-CNN not detecting objects. This could be due to incomplete learning or a complementary texture-based or part-based representation of the scenes. Therefore, although objects seem to be a key part of the representation learned by the network, we cannot rule out other representations being used in combination with objects.

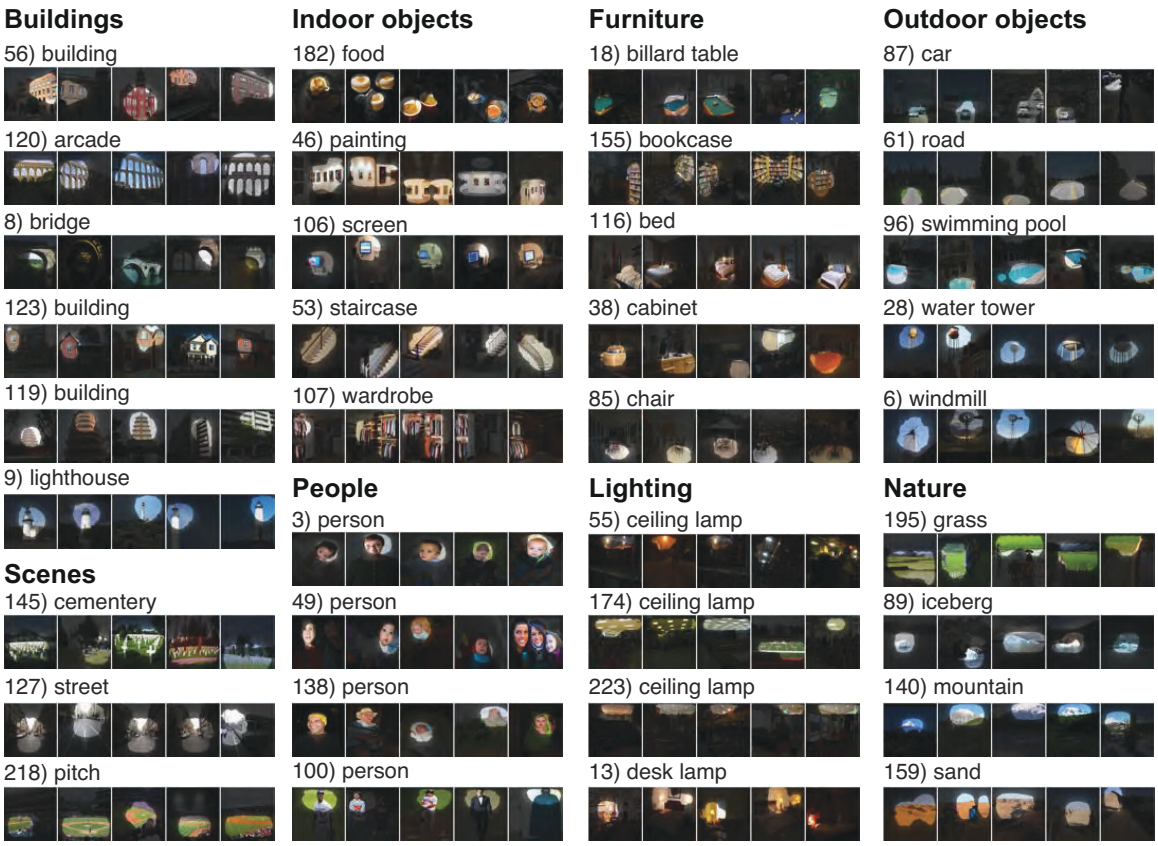


Figure 4-11: Segmentations using pool5 units from Places-CNN. Many classes are encoded by several units covering different object appearances. Each row shows the 5 most confident images for each unit. The number represents the unit number in pool5.

4.3.2 Object Localization within the Inner Layers

Places-CNN is trained to do scene classification using the output of the final layer of logistic regression and achieves state-of-the-art performance. From our analysis above, many of the units in the inner layers could perform interpretable object localization. Thus we could use this single Places-CNN with the annotation of units to do both scene recognition and object localization in a single forward-pass. Fig. 4-13 shows an example of the output of different layers of the Places-CNN using the tags provided by AMT workers. Bounding boxes are shown around the areas where each unit is activated within its RF above a certain threshold.

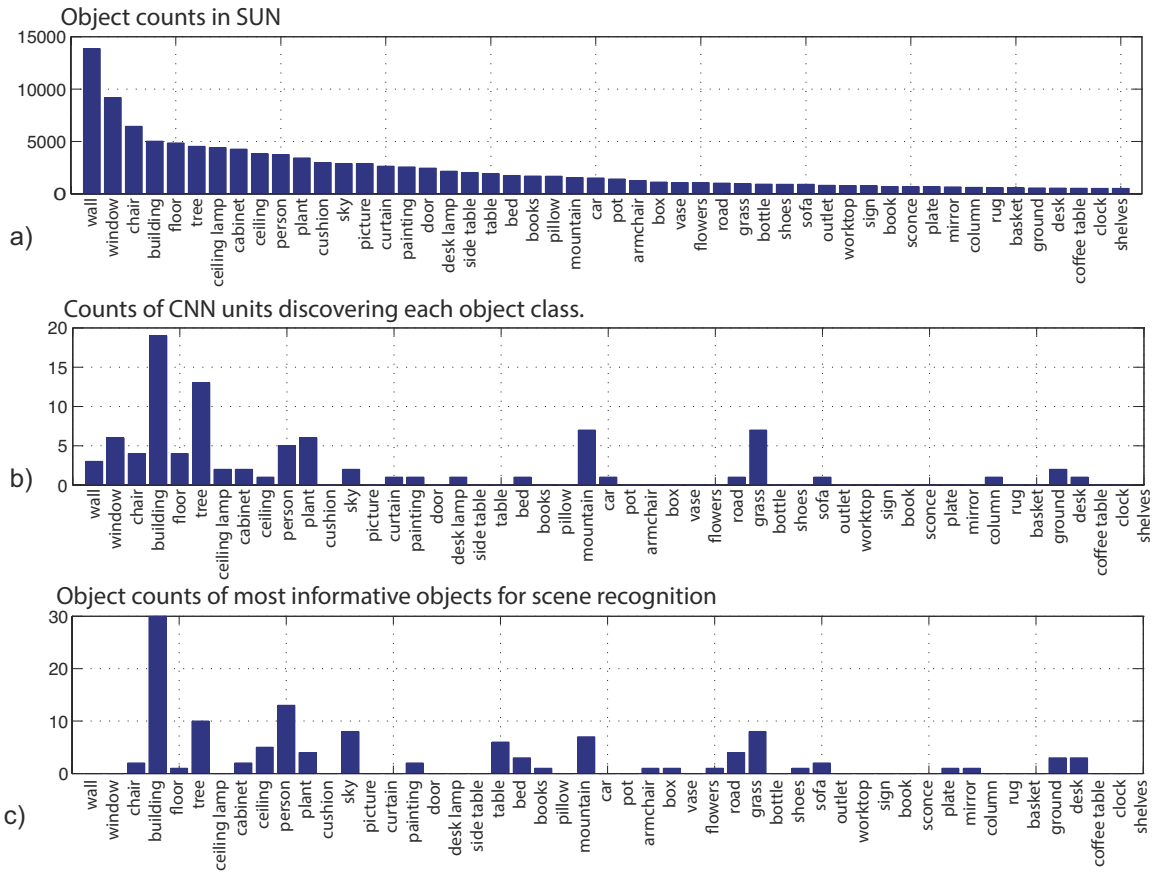


Figure 4-12: (a) Object frequency in SUN (only top 50 objects shown), (b) Counts of objects discovered by pool5 in Places-CNN. (c) Frequency of most informative objects for scene classification.

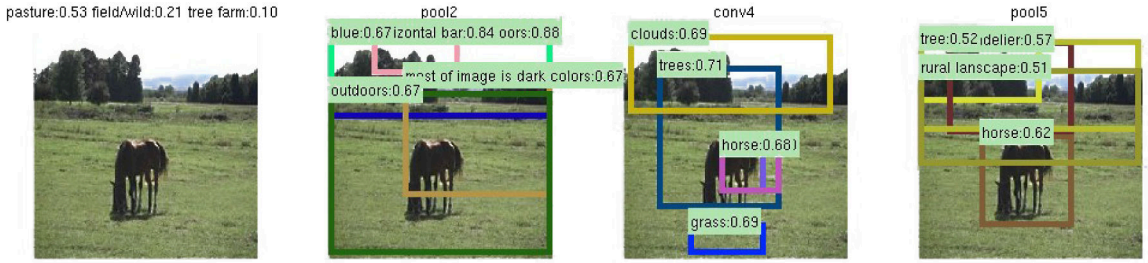


Figure 4-13: Interpretation of a picture by different layers of the Places-CNN using the tags provided by AMT workers. The first shows the final layer output of Places-CNN. The other three show detection results along with the confidence based on the units' activation and the semantic tags.

Chapter 5

Quantifying the Interpretability of Deep Visual Representations

Observations of hidden units in large deep neural networks have revealed that human-interpretable concepts sometimes emerge as individual latent variables within those networks. For example, object detector units emerge within networks trained to recognize places [119]; part detectors emerge in object classifiers [32]; and object detectors emerge in generative video networks [100]. This internal structure has appeared in situations where the networks are not constrained to decompose problems in any interpretable way.

The emergence of interpretable structure present in previous chapter suggests that deep networks may be learning disentangled representations spontaneously. While it is commonly understood that a network can learn an efficient encoding that makes economical use of hidden variables to distinguish the input, the appearance of a disentangled representation is not well understood. A disentangled representation aligns its variables with a meaningful factorization of the underlying problem structure, and encouraging disentangled representations is a significant area of research [7]. If the internal representation of a deep network is partly disentangled, one possible path for understanding its mechanisms is to detect disentangled structure, and simply read out the human interpretable factors.

In this chapter we address the following three key issues about the deep visual representations:

- What is a disentangled representation of neural networks, and how can its factors be

quantified and detected?

- Do interpretable hidden units reflect a special alignment of feature space?
- What differences in network architectures, data sources, and training conditions lead to the internal representations with greater or lesser entanglement?

To examine these issues, we propose a general analytic framework, *Network Dissection*, for interpreting deep visual representations and quantifying their interpretability. Using Broden, a broadly and densely labeled dataset, our framework identifies hidden units' semantics for any given CNN, then aligns them with human-interpretable concepts.

Building upon the preliminary result published at [5], we begin with a detailed description of the methodology of Network Dissection. We use the method to interpret a variety of deep visual representations trained with different network architectures (AlexNet, VGG, GoogLeNet, ResNet, DenseNet) and supervisions (supervsied training on ImageNet for object recognition and on Places for scene recognition, along with various self-taught supervision tasks). We show that interpretability is an axis-aligned property of a representation that can be destroyed by rotation without affecting discriminative power. We further examine how interpretability is affected by different training datasets. Our experiments reveal that units emerge as semantic detectors in the intermediate layers of most deep visual representations, while the degree of interpretability can vary widely across changes in architecture and training. We conclude that representations learned by deep networks are more interpretable than previously thought, and that measurements of interpretability provide insights about the structure of deep visual representations that that are not revealed by their classification power alone[1].

5.1 Framework of Network Dissection

The notion of a disentangled representation rests on the human perception of what it means for a concept to be mixed up. Therefore we define the *interpretability* of deep visual representation in terms of the degree of alignment with a set of human-interpretable concepts.

[1]Code, data, and more dissection results are available at the project page `http://netdissect.csail.mit.edu/`.

Our quantitative measurement of interpretability for deep visual representations proceeds in three steps:

- Identify a broad set of human-labeled visual concepts.
- Gather the response of the hidden variables to known concepts.
- Quantify alignment of hidden variable−concept pairs.

This three-step process of *network dissection* is reminiscent of the procedures used by neuroscientists to understand similar representation questions in biological neurons [77]. Since our purpose is to measure the level to which a representation is disentangled, we focus on quantifying the correspondence between a single latent variable and a visual concept.

In a fully interpretable local coding such as a one-hot-encoding, each variable will match exactly with one human-interpretable concept. Although we expect a network to learn partially nonlocal representations in interior layers [7], and past experience shows that an emergent concept will often align with a combination of a several hidden units [3,32], our present aim is to assess how well a representation is disentangled. Therefore we measure the alignment between single units and single interpretable concepts. This does not gauge the discriminative power of the representation; rather it quantifies its disentangled interpretability. As we will show in Sec. 5.2.2, it is possible for two representations of perfectly equivalent discriminative power to have very different levels of interpretability.

To assess the interpretability of any given CNN, we draw concepts from a new broadly and densely labeled image dataset that unifies labeled visual concepts from a heterogeneous collection of labeled data sources, described in Sec. 5.1.1. We then measure the alignment of each hidden unit of the CNN with each concept by evaluating the feature activation of each individual unit as a segmentation model for each concept. To quantify the interpretability of a layer as a whole, we count the number of distinct visual concepts that are aligned with a unit in the layer, as detailed in Sec. 5.1.2.

5.1.1 Broden: Broadly and Densely Labeled Dataset

To be able to ascertain alignment with both low-level concepts such as colors and higher-level concepts such as objects, we have assembled a new heterogeneous dataset.

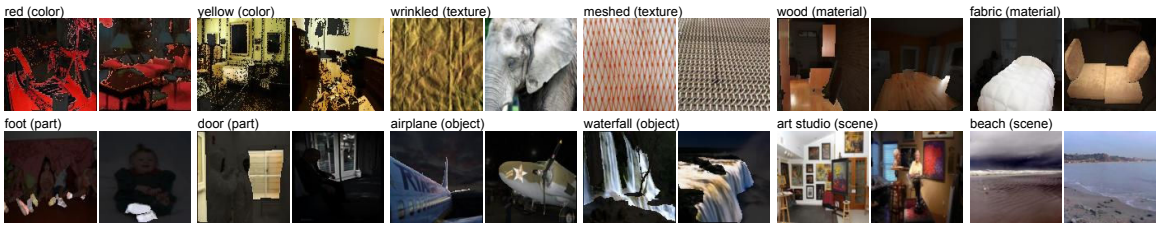


Figure 5-1: Samples from the Broden Dataset. The ground truth for each concept is a pixel-wise dense annotation.

Table 5.1: Statistics of each label type included in the dataset.

Category	Classes	Sources	Avg sample
scene	468	ADE [124]	38
object	584	ADE [124], Pascal-Context [65]	491
part	234	ADE [124], Pascal-Part [12]	854
material	32	OpenSurfaces [6]	1,703
texture	47	DTD [13]	140
color	11	Generated	59,250

The **Bro**adly and **Den**sely Labeled Dataset (**Broden**) unifies several densely labeled image datasets: ADE [124], OpenSurfaces [6], Pascal-Context [65], Pascal-Part [12], and the Describable Textures Dataset [13]. These datasets contain examples of a broad range of objects, scenes, object parts, textures, and materials in a variety of contexts. Most examples are segmented down to the pixel level except textures and scenes, which are given for full images. In addition, every image pixel in the dataset is annotated with one of the eleven common color names according to the human perceptions classified by van de Weijer [99]. Samples of the types of labels in the Broden dataset are shown in Fig. 5-1.

The purpose of Broden is to provide a ground truth set of exemplars for a broad set of visual concepts. The concept labels in Broden are normalized and merged from their original datasets so that every class corresponds to an English word. Labels are merged based on shared synonyms, disregarding positional distinctions such as 'left' and 'top' and avoiding a blacklist of 29 overly general synonyms (such as 'machine' for 'car'). Multiple Broden labels can apply to the same pixel: for example, a black pixel that has the Pascal-Part label 'left front cat leg' has three labels in Broden: a unified 'cat' label representing cats across datasets; a similar unified 'leg' label; and the color label 'black'. Only labels with at least 10 image samples are included. Table 5.1 shows the number of classes per dataset and the average number of image samples per label class. Totally there are 1197

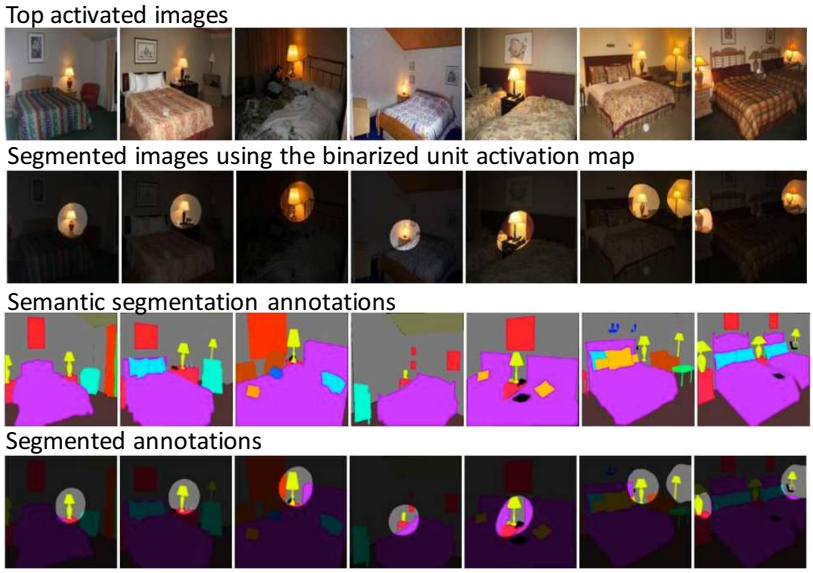


Figure 5-2: Scoring unit interpretability by evaluating the unit for semantic segmentation.

visual concept classes included.

5.1.2 Scoring Unit Interpretability

The proposed network dissection method evaluates every individual convolutional unit in a CNN as a solution to a binary segmentation task to every visual concept in Broden, as illustrated in Fig. 5-3. Our method can be applied to any CNN using a forward pass without the need for training or backpropagation.

For every input image $\mathbf{x}$ in the Broden dataset, the activation map $A_k(\mathbf{x})$ of every internal convolutional unit k is collected. Then the distribution of individual unit activations a_k is computed. For each unit k, the top quantile level T_k is determined such that $P(a_k > T_k) = 0.005$ over every spatial location of the activation map in the dataset.

To compare a low-resolution unit's activation map to the input-resolution annotation mask L_c for some concept c, the activation map is scaled up to the mask resolution $S_k(\mathbf{x})$ from $A_k(\mathbf{x})$ using bilinear interpolation, anchoring interpolants at the center of each unit's receptive field.

$S_k(\mathbf{x})$ is then thresholded into a binary segmentation: $M_k(\mathbf{x}) \equiv S_k(\mathbf{x}) \geq T_k$, selecting all regions for which the activation exceeds the threshold T_k. These segmentations are evaluated against every concept c in the dataset by computing intersections $M_k(\mathbf{x}) \cap L_c(\mathbf{x})$,

for every (k, c) pair.

The score of each unit k as segmentation for concept c is reported as a the intersection over union score across all the images in the dataset,

$$IoU_{k,c} = \frac{\sum |M_k(\mathbf{x}) \cap L_c(\mathbf{x})|}{\sum |M_k(\mathbf{x}) \cup L_c(\mathbf{x})|}, \tag{5.1}$$

where $|\cdot|$ is the cardinality of a set. Because the dataset contains some types of labels which are not present on some subsets of inputs, the sums are computed only on the subset of images that have at least one labeled concept of the same category as c. The value of $IoU_{k,c}$ is the accuracy of unit k in detecting concept c; we consider one unit k as a detector for concept c if $IoU_{k,c}$ exceeds a threshold. Our qualitative results are insensitive to the IoU threshold: different thresholds denote different numbers of units as concept detectors across all the networks but relative orderings remain stable. For our comparisons we report a detector if $IoU_{k,c} > 0.04$. Note that one unit might be the detector for multiple concepts; for the purpose of our analysis, we choose the top ranked label. To quantify the interpretability of a layer, we count the number unique concepts aligned with units. We call this the number of *unique detectors*.

Figure 5-2 summarizes the whole process of scoring unit interpretability: By segmenting the annotation mask using the receptive field of units for the top activated images, we compute the IoU for each concept. The IoU evaluating the quality of the segmentation of a unit is an objective confidence score for interpretability that is *comparable across networks*. Thus this score enables us to compare interpretability of different representations and lays the basis for the experiments below. Note that network dissection works only as well as the underlying dataset: if a unit matches a human-understandable concept that is absent in Broden, then it will not score well for interpretability. Future versions of Broden will be expanded to include more kinds of visual concepts.

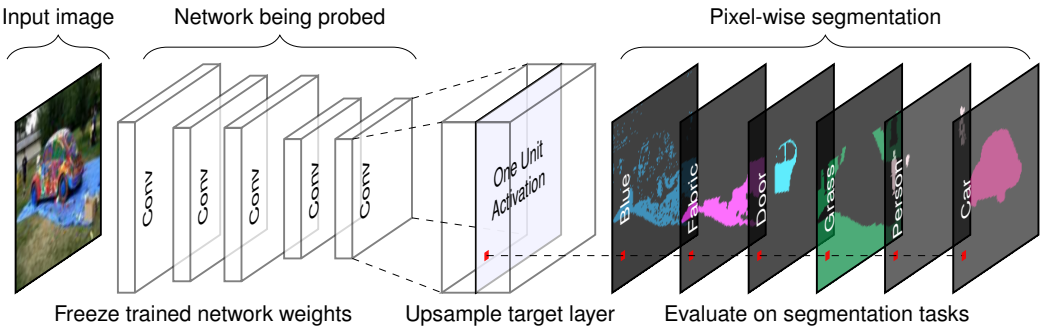


Figure 5-3: Illustration of network dissection for measuring semantic alignment of units in a given CNN. Here one unit of the last convolutional layer of a given CNN is probed by evaluating its performance on various segmentation tasks. Our method can probe any convolutional layer.

Table 5.2: Tested CNN Models

Training	**Network**	**dataset or task**
none	AlexNet	random
Supervised	AlexNet	ImageNet, Places205, Places365, Hybrid.
	GoogLeNet	ImageNet, Places205, Places365.
	VGG-16	ImageNet, Places205, Places365, Hybrid.
	ResNet-152	ImageNet, Places365.
	DenseNet-161	ImageNet, Places365.
Self	AlexNet	`context`, `puzzle`, `egomotion`, `tracking`, `moving`, `videoorder`, `audio`, `crosschannel`,`colorization`. `objectcentric`, `transinv`.

5.2 Interpreting Deep Visual Representations

For testing we prepare a collection of CNN models with different network architectures and supervision of primary tasks, as listed in Table 5.2. The network architectures include AlexNet [48], GoogLeNet [94], VGG [90], ResNet [37], and DenseNet [41]. For supervised training, the models are trained from scratch (i.e., not pretrained) on ImageNet [79], Places205 [122], and Places365 [121]. ImageNet is an object-centric dataset, which contains 1.2 million images from 1000 classes. Places205 and Places365 are two subsets of the Places Database, which is a scene-centric dataset with categories such as kitchen, living room, and coast. Places205 contains 2.4 million images from 205 scene categories, while Places365 contains 1.6 million images from 365 scene categories. "Hybrid" refers to a combination of ImageNet and Places365. For self-supervised training tasks, we select several recent models trained on predicting context (`context`) [17], solving puzzles (`puzzle`) [68], predicting ego-motion (`egomotion`) [43], learning by moving (`moving`) [2], predicting video frame order (`videoorder`) [62] or tracking (`tracking`) [105], detecting object-centric alignment (`objectcentric`) [29], colorizing images (`colorization`) [116], inpainting (`contextencoder`) [72], predicting cross-channel (`crosschannel`) [117], predicting ambient sound from frames (`audio`) [71], and tracking invariant patterns in videos (`transinv`) [106]. The self-supervised models we analyze are comparable to each other in that they all use AlexNet or an AlexNet-derived architecture, with one exception model `transinv` [106], which uses VGG as the base network.

In the following experiments, we begin by validating our method using human evaluation. Then, we use random unitary rotations of a learned representation to test whether interpretability of CNNs is an axis-independent property; we find that it is not, and we conclude that interpretability is not an inevitable result of the discriminative power of a representation. Next, we analyze all the convolutional layers of AlexNet as trained on ImageNet [48] and as trained on Places [122], and confirm that our method reveals detectors for higher-level concepts at higher layers and lower-level concepts at lower layers; and that more detectors for higher-level concepts emerge under scene training. Then, we

show that different network architectures such as AlexNet, VGG, and ResNet yield different interpretability, while differently supervised training tasks and self-supervised training tasks also yield a variety of levels of interpretability. Additionally we show the impact of different training conditions, examine the relationship between discriminative power and interpretability, and investigate a possible way to improve the interpretability of CNNs by increasing their width. Finally we utilize the interpretable units as explanatory factors to the prediction given by a CNN.

5.2.1 Human Evaluation of Interpretations

Using network dissection, we analyze the interpretability of units within all the convolutional layers of Places-AlexNet and ImageNet-AlexNet, then compare with human interpretation. Places-AlexNet is trained for scene classification on Places205 [122], while ImageNet-AlexNet is the identical architecture trained for object classification on ImageNet [48].

Our evaluation was done by raters on Amazon Mechanical Turk (AMT). As a baseline description of unit semantics, we used human-written descriptions of each unit from [119]. These descriptions were collected by asking raters to write words or short phrases to describe the common meaning or pattern selected by each unit, based on a visualization of the top image patches. Three descriptions and a confidence were collected for each unit. As a canonical description we chose the most common description of a unit (when raters agreed), and the highest-confidence description (when raters did not agree). Some units may not be interpretable. To identify these, raters were shown the canonical descriptions of visualizations and asked whether they were descriptive. Units with validated descriptions are taken as the set of interpretable units.

To compare these baseline descriptions to network-dissection-derived labels, we ran the following experiment. Raters were shown a visualization of top images patches for an interpretable unit, along with a word or short phrase description, and they were asked to vote (yes/no) whether the given phrase was descriptive of most of the image patches. The baseline human-written descriptions were randomized with the labels derived using

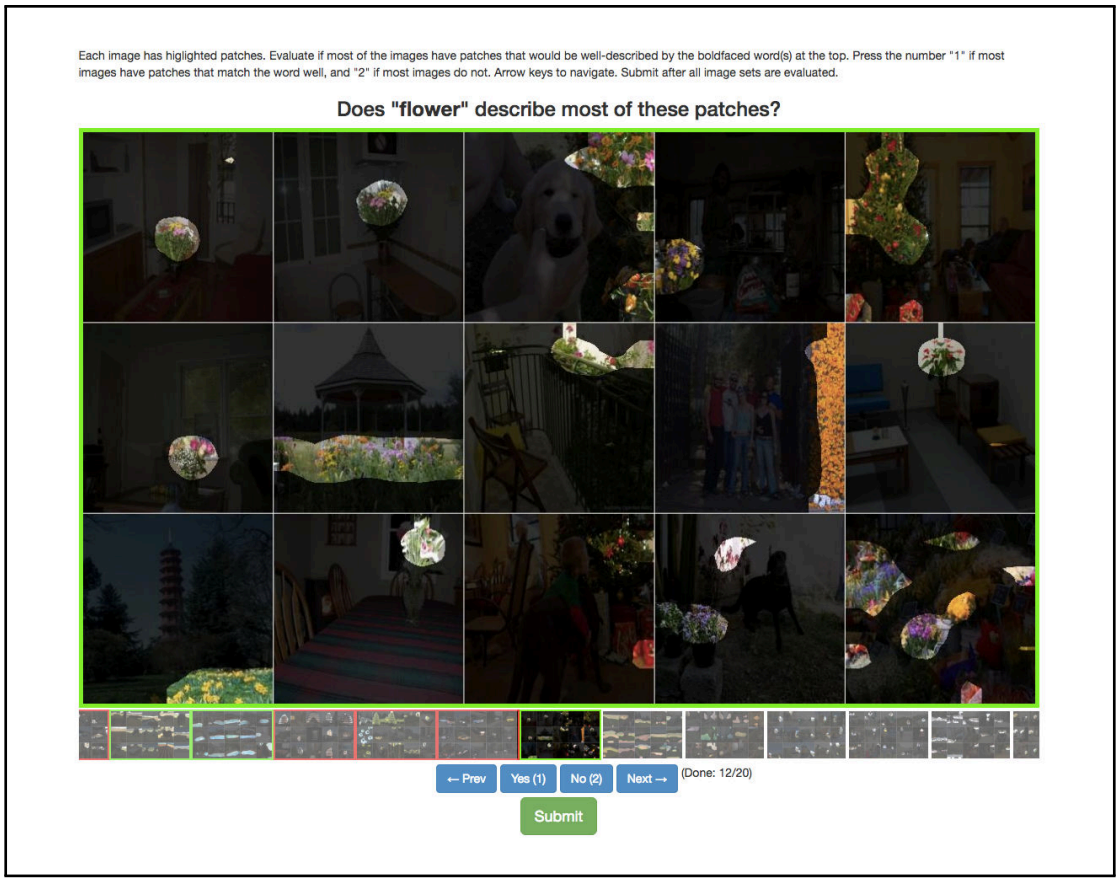


Figure 5-4: The annotation interface used by human raters on Amazon Mechanical Turk. Raters are shown descriptive text in quotes together with fifteen images, each with highlighted patches, and must evaluate whether the quoted text is a good description for the highlighted patches.

net dissection, and the origin of the labels was not revealed to the raters.

Table 5.3 summarizes the results. The number of interpretable units is shown for each layer, and average positive votes for descriptions of interpretable units are shown, both for human-written labels and network-dissection-derived labels. Human labels are most highly consistent for units of `conv5`, suggesting that humans have no trouble identifying high-level visual concept detectors, while lower-level detectors are more difficult to label. Similarly, labels given by network dissection are best at `conv5`, and are found to be less descriptive for lower layers.

Comparison of the human interpretation and the labels predicted by network dissection is plotted in Fig. 5-5. A sample of units is shown together with both automatically inferred interpretations and manually assigned interpretations taken from [119]. We can see that

Table 5.3: Human evaluation of our Network Dissection approach.

	conv1	conv2	conv3	conv4	conv5
Interpretable units	57/96	126/256	247/384	258/384	194/256
Human consistency	82%	76%	83%	82%	91%
Network Dissection	37%	56%	54%	59%	71%

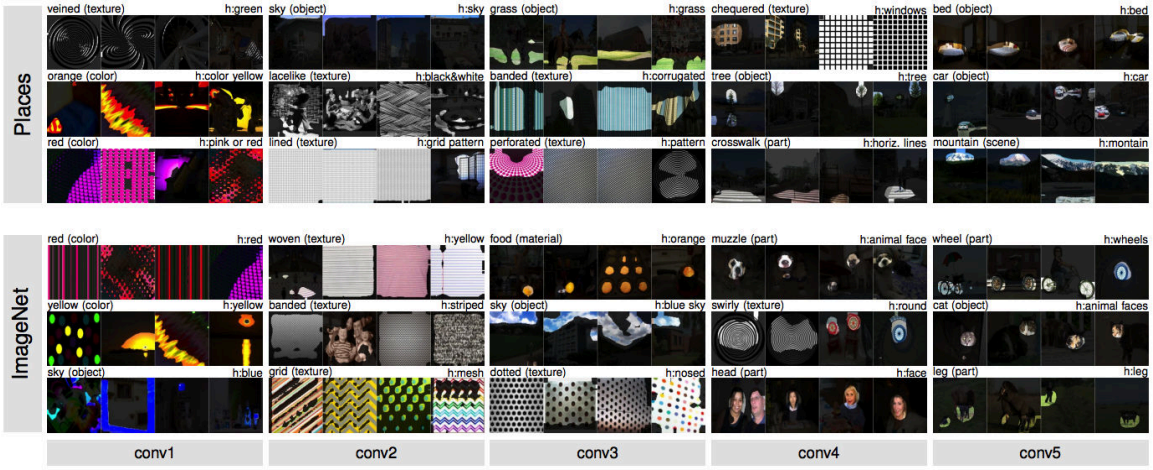


Figure 5-5: Comparison of the interpretability of all five convolutional layers of AlexNet, as trained on classification tasks for Places (top) and ImageNet (bottom).Four examples of units in each layer are shown with identified semantics. The segmentation generated by each unit is shown on the three Broden images with highest activation. Top-scoring labels are shown above to the left, and human-annotated labels are shown above to the right. Some disagreement can be seen for the dominant judgment of meaning. For example, human annotators mark the first `conv4` unit on Places as a 'windows' detector, while the algorithm matches the 'chequered' texture.

the predicted labels match the human annotation well, though sometimes they capture a different description of a visual concept, such as the 'crosswalk' predicted by the algorithm compared to 'horizontal lines' given by the human for the third unit in `conv4` of Places-AlexNet in Fig. 5-5. Confirming intuition, color and texture concepts dominate at lower layers `conv1` and `conv2` while more object and part detectors emerge in `conv5`.

5.2.2 Measurement of Axis-aligned Interpretability

We conduct an experiment to determine whether it is meaningful to assign an interpretable concept to an individual unit. Two possible hypotheses can explain the emergence of interpretability in individual hidden layer units:

Hypothesis 1. Interpretability is a property of the representation as a whole, and individual interpretable units emerge because interpretability is a generic property of typical directions of representation space. Under this hypothesis, projecting to *any* direction would typically reveal an interpretable concept, and interpretations of single units in the natural basis would not be more meaningful than interpretations that can be found in any other direction.

Hypothesis 2. Interpretable alignments are unusual, and interpretable units emerge because learning converges to a special basis that aligns explanatory factors with individual units. In this model, the natural basis represents a meaningful decomposition learned by the network.

Hypothesis 1 is the default assumption: in the past it has been found [95] that with respect to interpretability "there is no distinction between individual high level units and random linear combinations of high level units."

Network dissection allows us to re-evaluate this hypothesis. We apply random changes in basis to a representation learned by AlexNet. Under hypothesis 1, the overall level of interpretability should not be affected by a change in basis, even as rotations cause the specific set of represented concepts to change. Under hypothesis 2, the overall level of interpretability is expected to drop under a change in basis.

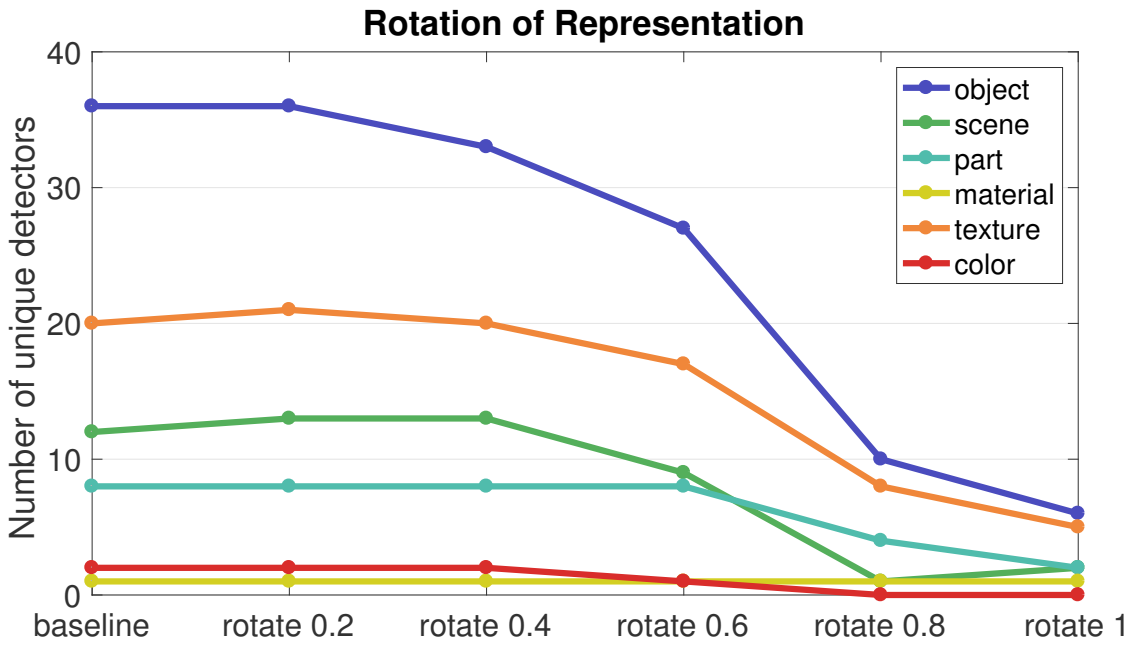


Figure 5-6: Interpretability over changes in basis of the representation of AlexNet `conv5` trained on Places. The vertical axis shows the number of unique interpretable concepts that match a unit in the representation. The horizontal axis shows α, which quantifies the degree of rotation.

We begin with the representation of the 256 convolutional units of AlexNet `conv5` trained on Places205 and examine the effect of a change in basis. To avoid any issues of conditioning or degeneracy, we change basis using a random orthogonal transformation Q. The rotation Q is drawn uniformly from $SO(256)$ by applying Gram-Schmidt on a normally-distributed $QR = A \in \mathbf{R}^{256^2}$ with positive-diagonal right-triangular R, as described by [16]. Interpretability is summarized as the number of unique visual concepts aligned with units, as defined in Sec. 5.1.2.

Denoting AlexNet `conv5` as $f(x)$, we find that the number of unique detectors in $Qf(x)$ is 80% fewer than the number of unique detectors in $f(x)$. Our finding is inconsistent with hypothesis 1 and consistent with hypothesis 2.

We also test smaller perturbations of basis using Q^α for $0 \leq \alpha \leq 1$, where the fractional powers $Q^\alpha \in SO(256)$ are chosen to form a minimal geodesic gradually rotating from I to Q; these intermediate rotations are computed using a Schur decomposition. Fig. 5-6 shows that interpretability of $Q^\alpha f(x)$ decreases as larger rotations are applied. Fig. 5-7 shows some examples of the linearly combined units.

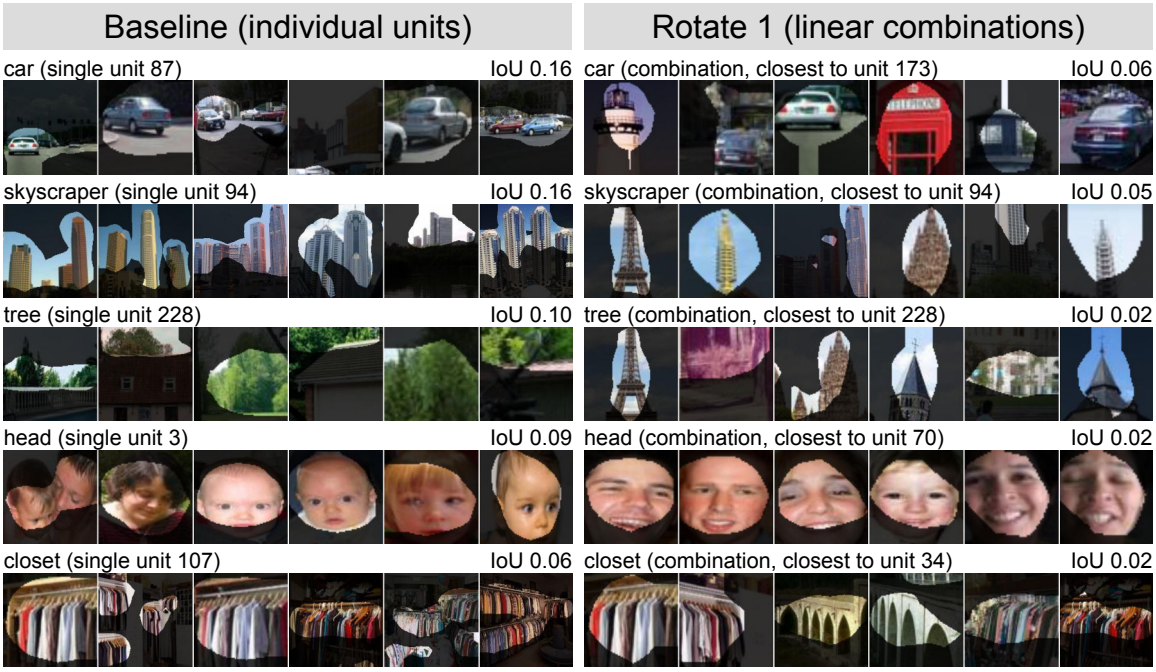


Figure 5-7: Visualizations of the best single-unit concept detectors of five concepts taken from individual units of AlexNet `conv5` trained on Places (left), compared with the best linear-combination detectors of the same concepts taken from the same representation under a random rotation (right). For most concepts, both the IoU and the visualization of the top activating image patches confirm that individual units match concepts better than linear combinations. In other cases, (e.g. head detectors) visualization of a linear combination appears highly consistent, but the IoU reveals lower consistency when evaluated over the whole dataset.

Each rotated representation has exactly the same discriminative power as the original layer. Writing the original network as $g(f(x))$, note that $g'(r) \equiv g(Q^T r)$ defines a neural network that processes the rotated representation $r = Qf(x)$ exactly as the original g operates on $f(x)$. We conclude that interpretability is neither an inevitable result of discriminative power, nor is it a prerequisite to discriminative power. Instead, we find that interpretability is a different quality that must be measured separately to be understood.

We repeat the complete rotation ($\alpha = 1$) on Places365 and ImageNet 10 times, the result is shown in Fig. 5-8. We observe the drop of interpretability for both of the network, while it drops more for the AlexNet on Places365. It is because originally the interpretability of AlexNet on Places365 is higher than AlexNet on ImageNet thus the random rotation damages more.

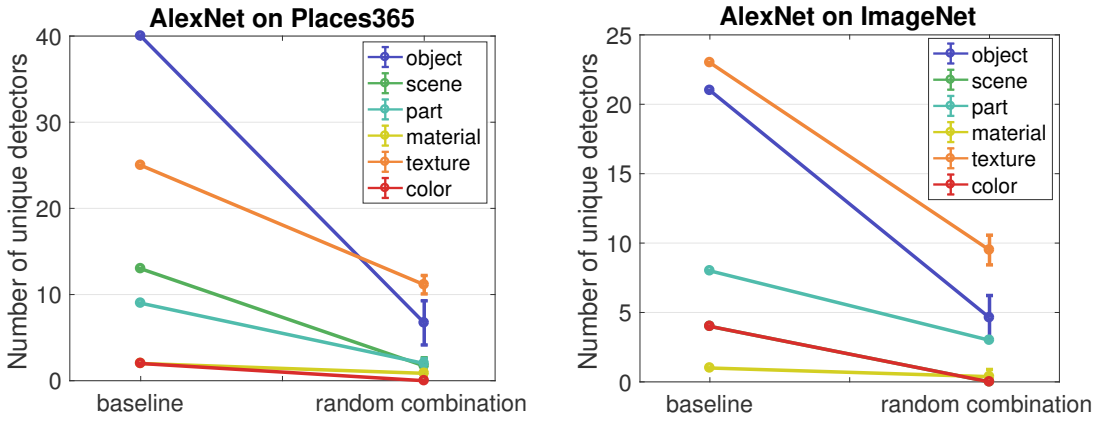


Figure 5-8: Complete rotation ($\alpha = 1$) repeated on AlexNet trained on Places365 and ImageNet respectively. Rotation reduces the interpretability significantly for both of the networks.

5.2.3 Network Architectures with Supervised Learning

How do different network architectures affect disentangled interpretability of the learned representations? We apply network dissection to evaluate a range of network architectures trained on ImageNet and Places. For simplicity, the following experiments focus on the last convolutional layer of each CNN, where semantic detectors emerge most.

Results showing the number of unique detectors that emerge from various network architectures trained on ImageNet and Places are plotted in Fig. 5-9. In terms of network architecture, we find that interpretability of ResNet $>$ DenseNet $>$ VGG $>$ GoogLeNet $>$ AlexNet. Deeper architectures usually appear to allow greater interpretability, though individual layer structure is different across architecture. Comparing training datasets, we find Places $>$ ImageNet. As discussed in [119], one scene is composed of multiple objects, so it may be beneficial for more object detectors to emerge in CNNs trained to recognize scenes.

Fig. 5-10 shows the histogram of object detectors identified inside ResNet and DenseNet trained on ImageNet and Places respectively. DenseNet161-Places365 has the largest number of unique object detectors among all the networks. The emergent detectors differ across both training data source and architecture. The most frequent object detectors in the two networks trained on ImageNet are dog detectors, because there are more than 100 dog categories out of the 1000 classes in the ImageNet training set.

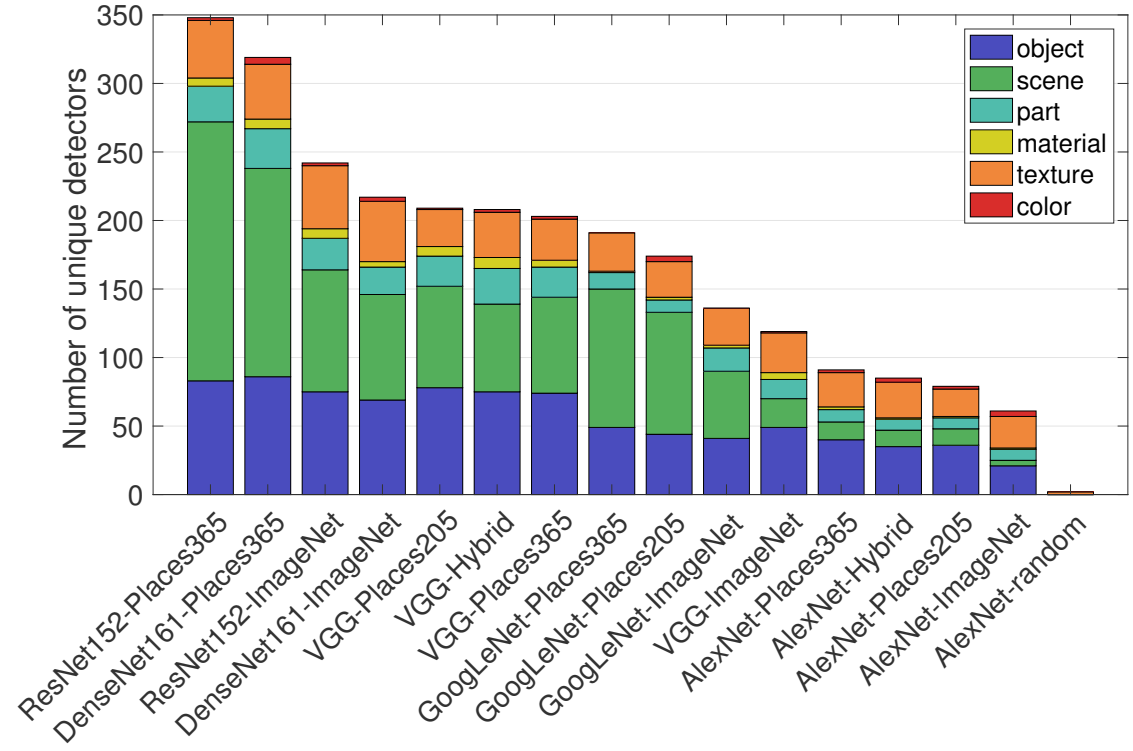


Figure 5-9: Interpretability across different architectures trained on ImageNet and Places.

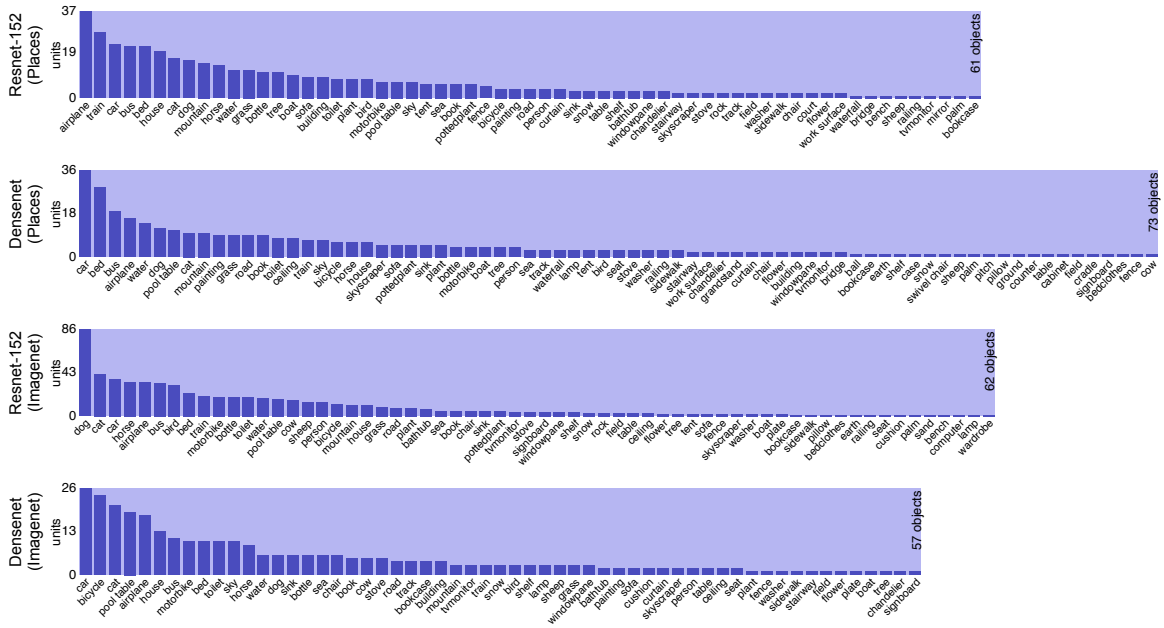


Figure 5-10: Histogram of the object detectors from the ResNet and DenseNet trained on ImageNet and Places respectively.

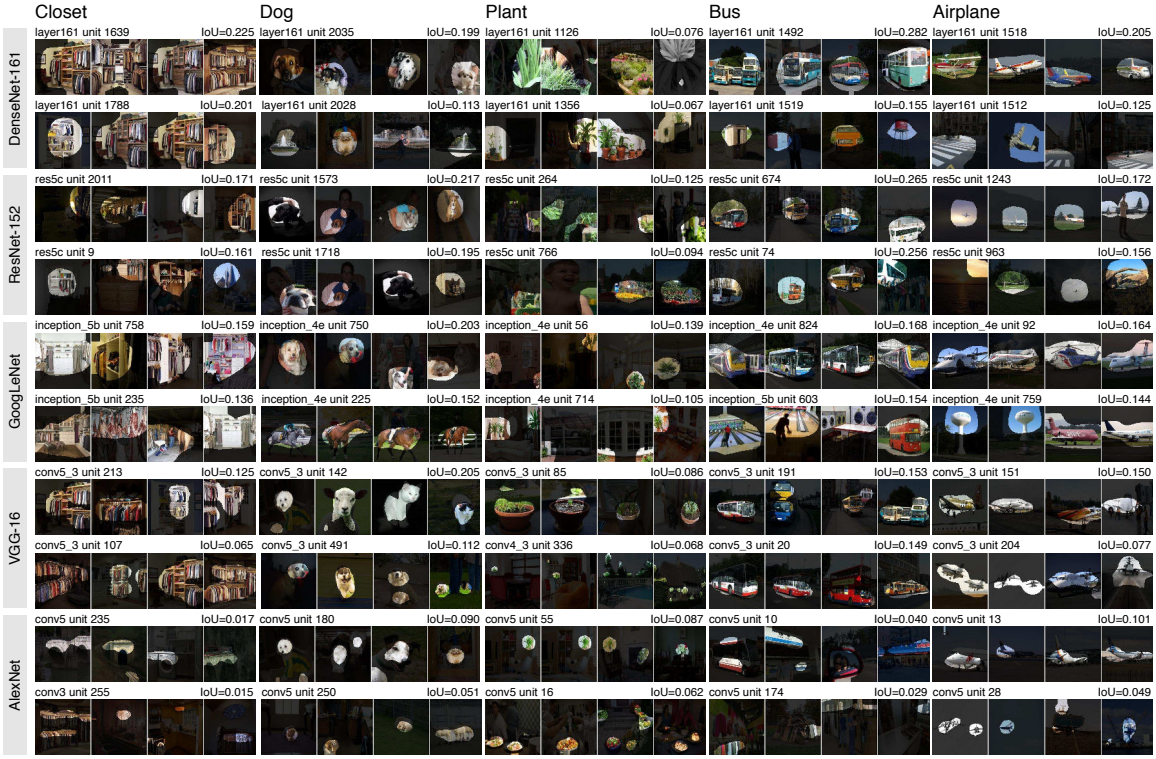


Figure 5-11: Comparison of several visual concept detectors identified by network dissection in DenseNet, ResNet, GoogLeNet, VGG, and AlexNet. Each network is trained on Places365. The two highest-IoU matches among convolutional units of each network is shown. The segmentation generated by each unit is shown on the four maximally activating Broden images. Some units activate on concept generalizations, e.g., GoogLeNet 4e's unit 225 on horses and dogs, and 759 on white ellipsoids and jets.

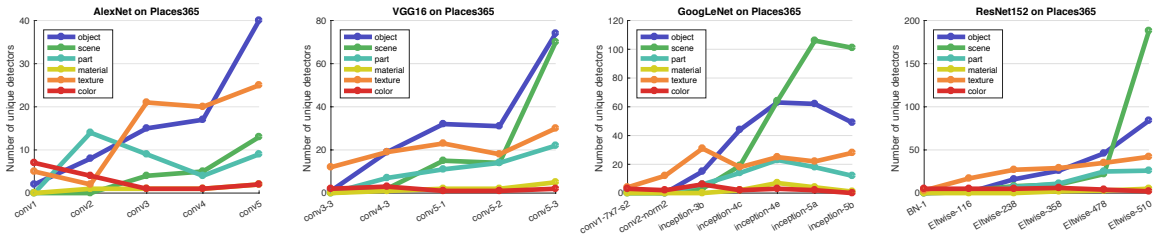


Figure 5-12: Comparison of interpretability of the layers for AlexNet, VGG16, GoogLeNet, and ResNet152 trained on Places365. All five conv layers of AlexNet and the selected layers of VGG, GoogLeNet, and ResNet are included.

Fig. 5-11 shows the examples of object detectors grouped by object categories. For the same object category, the visual appearance of the unit as detector varies not only within the same network but also across different networks. DenseNet and ResNet has such good detectors for bus and airplane with IoU > 0.25.

Fig. 5-12 shows the unique interpretable detectors over different layers for different network architectures trained on Places365. We observe that more object and scene detectors emerge at the higher layers across all architectures: AlexNet, VGG, GoogLeNet, and ResNet. This suggests that representation ability increases over layer depth. Because of the compositional structure of the CNN layers, the deeper layers should have higher capacity to represent concepts with larger visual complexity such as objects and scene parts. Our measurements confirm this, and we conclude that higher network depth encourages the emergence of visual concepts with higher semantic complexity.

5.2.4 Representations from Self-supervised Learning

Recently many work have explored a novel paradigm for unsupervised learning of CNNs without using millions of annotated images, namely self-supervised learning. For example, [17] trains deep CNNs to predict the neighborhoods of two image patches, while [116] trains networks by colorizing images. Totally we investigate 12 networks trained for different self-supervised learning tasks. How do different supervisions affect those internal representations?

Here we compare the interpretability of the deep visual representations resulting from self-supervised learning and supervised learning. We keep the network architecture the

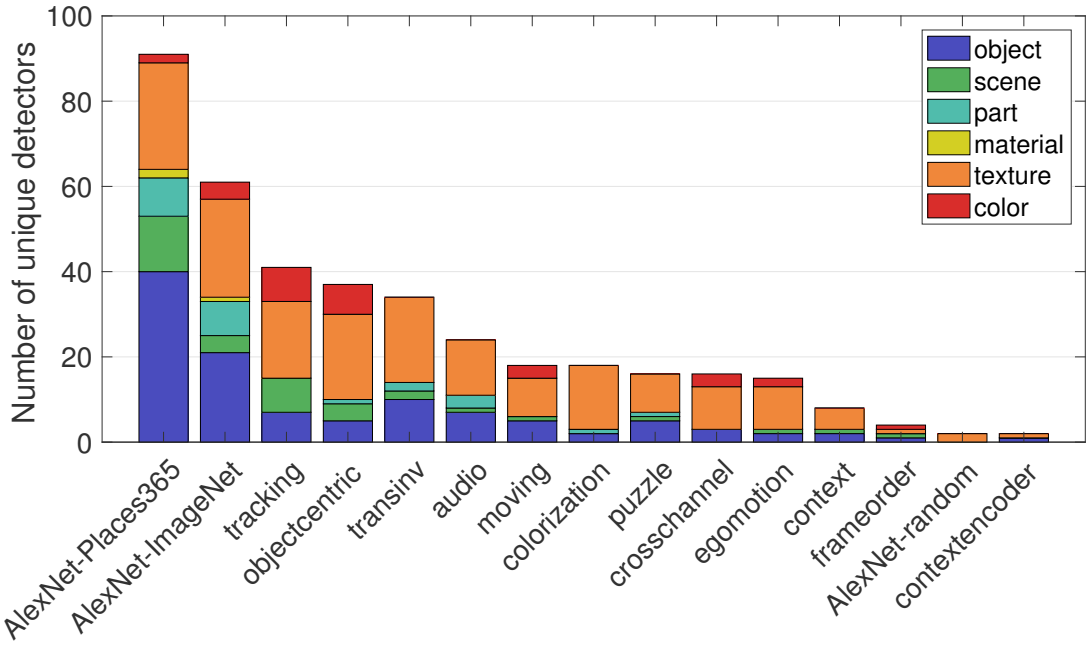


Figure 5-13: Semantic detectors emerge across different supervision of the primary training task. All these models use the AlexNet architecture and are tested at `conv5`.

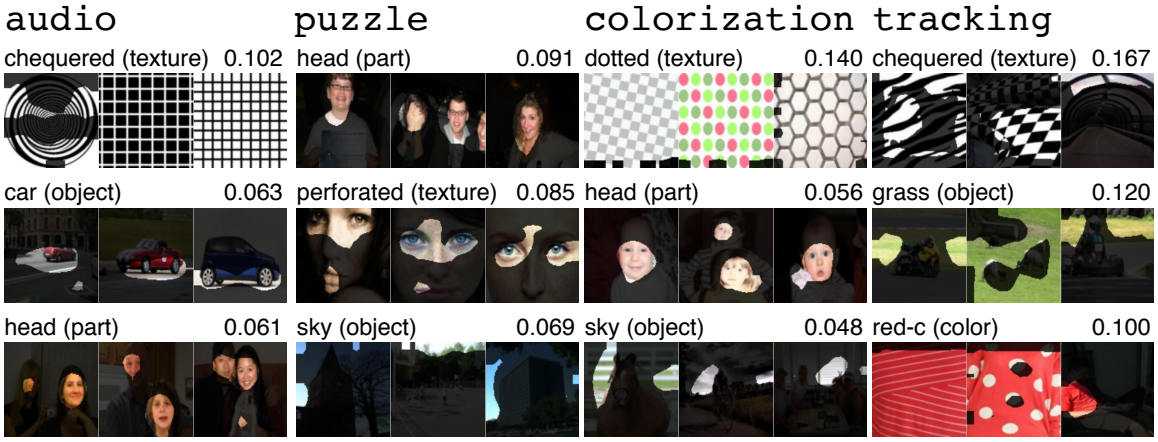


Figure 5-14: The top ranked concepts in the three top categories in four self-supervised networks. Some object and part detectors emerge in `audio`. Detectors for person heads also appear in `puzzle` and `colorization`. A variety of texture concepts dominate models with self-supervised training.

same as AlexNet for each model (one exception is the recent model `transinv` which uses VGG as the base network). Results are shown in Fig. 5-13. We observe that training on Places365 creates the largest number of unique detectors. Self-supervised models create many texture detectors but relatively few object detectors; apparently, supervision from a self-taught primary task is much weaker at inferring interpretable concepts than supervised training on a large annotated dataset. The form of self-supervision makes a difference: for example, the colorization model is trained on colorless images, and almost no color detection units emerge. We hypothesize that emergent units represent concepts required to solve the primary task.

Fig. 5-14 shows some typical visual detectors identified in the self-supervised CNN models. For the models `audio` and `puzzle`, some object and part detectors emerge. Those detectors may be useful for CNNs to solve the primary tasks: the `audio` model is trained to associate objects with a sound source, so it may be useful to recognize people and cars; while the `puzzle` model is trained to align the different parts of objects and scenes in an image. For `colorization` and `tracking`, recognizing textures might be good enough for the CNN to solve primary tasks such as colorizing a desaturated natural image; thus it is unsurprising that the texture detectors dominate.

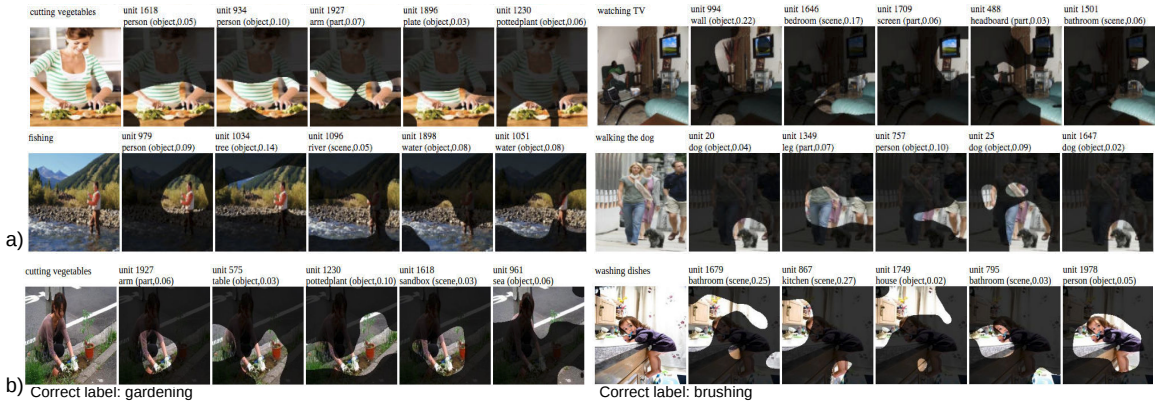


Figure 5-15: Segmenting images using top activated units weighted by the class label from ResNet152-Places365 deep feature. a) the correctly predicted samples. b) the incorrectly predicted samples.

5.2.5 Explaining the Predictions for the Deep Features

After we interpret the units inside the deep visual representation, we show that the unit activation along with the interpreted label can be used to explain the prediction given by the deep features. Previous work [120] uses the weighted sum of the unit activation maps to highlight which image regions are most informative to the prediction, here we further decouple at individual unit level to segment the informative image regions.

We first plot the *Class-specific units*. After the linear SVM is trained, we can rank the elements of the feature according to their SVM weights to obtain the elements of the deep features which contribute most to that class. Those elements are units that act as explanatory factors, and we call those top ranked units associated with each output class *class-specific units*. Fig. 5-16 shows the class-specific units of ResNet152-ImageNet and ResNet152-Places365 for one class from action40 and sun397 respectively. For example, for the *Walking the dog* class from action40, the top three class-specific units from ResNet152-ImageNet are two dog detection unit and one person detection unit; for the *Picnic area* class from sun397, the top three class-specific units from ResNet152-Places365 are plant detection unit, grass detection unit, and fence detection unit. The intuitive match between visual detectors and the classes they explain suggests that visual detectors from CNNs behave as the bag-of-semantic-words visual features.

We further use the individual units identified as concept detectors to build an explanation of the individual image prediction given by a classifier. The procedure is as follows: Given any image, let the unit activation of the deep feature (for ResNet the `GAP` activation) be $[x_1, x_2, ..., x_N]$, where each x_n represents the value summed up from the activation map of unit n. Let the top prediction's SVM response be $s = \sum_n w_n x_n$, where $[w_1, w_2, ..., w_N]$ is the SVM's learned weight. We get the top ranked units in Figure 5-15 by ranking $[w_1 x_1, w_2 x_2, ..., w_N x_N]$, which are the unit activations weighted by the SVM weight for the top predicted class. Then we simply upsample the activation map of the top ranked unit to segment the image.

Image segmentations using individual unit activation are plotted in Fig. 5-15a. The unit segmentation explain the prediction explicitly. For example, the prediction for the first

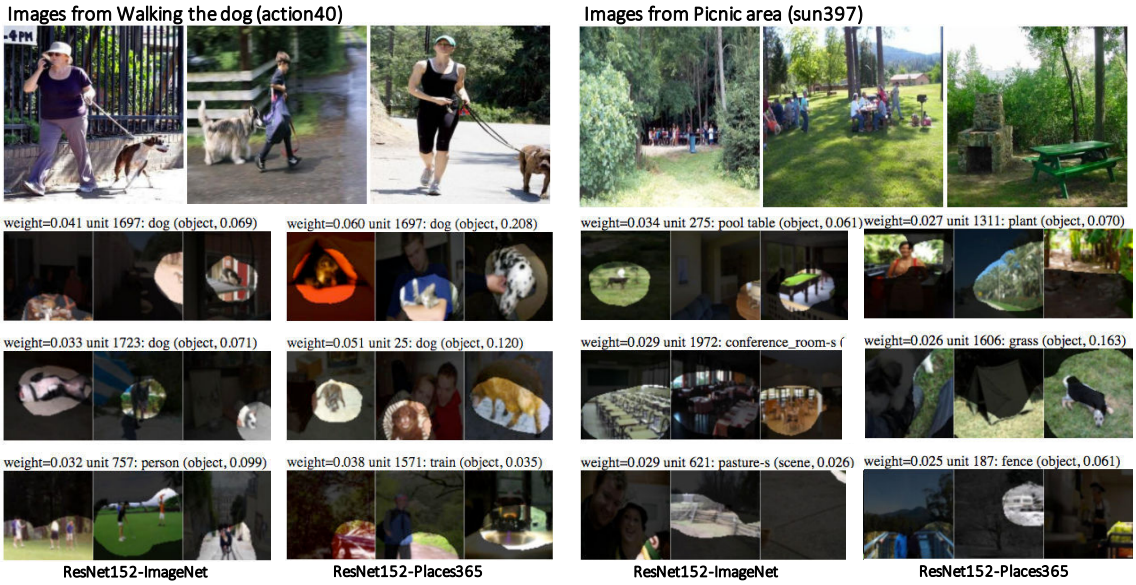


Figure 5-16: Class-specific units from ResNet152-ImageNet and ResNet152-Places365 on one class from action40 and sun397. For each class, we show three sample images, followed by the top 3 units from ResNet152-ImageNet and ResNet152-Places365 ranked by the class weight of linear SVM to predict that class. SVM weight, detected concept name and the IoU score are shown above each unit.

image is *Gardening*, and the explanatory units detect plant, grass, person, flower, and pot. The prediction for the second image is *Riding a horse*, the explanatory units detect horse, fence and dog. We also plot some incorrectly predicted samples in Figure 5-15b. The segmentation gives the intuition as to why the classifier made mistakes. For example, for the first image the classifier predicts *cutting vegetables* rather than the true label *gardening*, because the second unit incorrectly considers the ground as table.

Chapter 6

Explaining the Deep Neural Networks via Class Activation Mapping

In previous chapters, we have shown that the convolutional units of various layers of convolutional neural networks (CNNs) actually behave as object detectors despite no supervision on the location of the object was provided. Despite having this remarkable ability to localize objects in the convolutional layers, this ability is lost when fully-connected layers are used for classification. On the other hand, when the CNN produces a prediction for an image, it lacks the explanation about why the model makes such a decision.

Some popular fully-convolutional neural networks such as the Network in Network (NIN) [55] and GoogLeNet [93] have been proposed to avoid the use of fully-connected layers to minimize the number of parameters while maintaining high performance. [55] uses *global average pooling* which acts as a structural regularizer, preventing overfitting during training. In our experiments, we found that the advantages of this global average pooling layer extend beyond simply acting as a regularizer - In fact, with a little tweaking, the network can retain its remarkable localization ability until the final layer. This tweaking allows identifying easily the informative image regions in a single forward-pass for a wide variety of tasks, even those that the network was not originally trained for. As shown in Figure 6-1a, a CNN trained on object categorization is successfully able to localize the most informative regions for classifying the action as the objects that the humans are interacting with rather than the humans themselves.

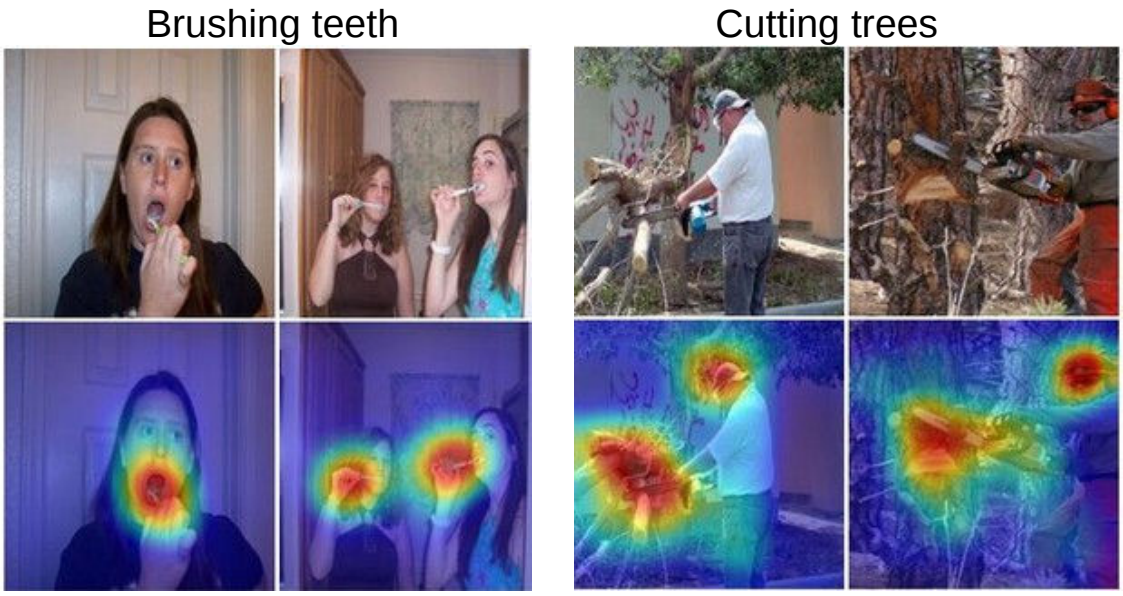


Figure 6-1: A simple modification of the global average pooling layer combined with our class activation mapping (CAM) technique allows the classification-trained CNN to both classify the image and localize class-specific image regions in a single forward-pass e.g., the toothbrush for *brushing teeth* and the chainsaw for *cutting trees*. It produces a visual explanation for the decision made by the CNN.

Despite the apparent simplicity of our approach, for the weakly supervised object localization on ILSVRC benchmark [79], our best network achieves 37.1% top-5 test error, which is rather close to the 34.2% top-5 test error achieved by fully supervised AlexNet [48]. Furthermore, we demonstrate that the localizability of the deep features in our approach can be easily transferred to other recognition datasets for generic classification, localization, and concept discovery.[1].

6.1 Class Activation Mapping

In this section, we describe the procedure for generating *class activation maps* (CAM) using global average pooling (GAP) in CNNs. A class activation map for a particular category indicates the discriminative image regions used by the CNN to identify that category. The procedure for generating these maps is illustrated in Fig. 6-2.

We use a network architecture similar to Network in Network [55] and GoogLeNet [93]

[1]The demo code is available at: http://cnnlocalization.csail.mit.edu

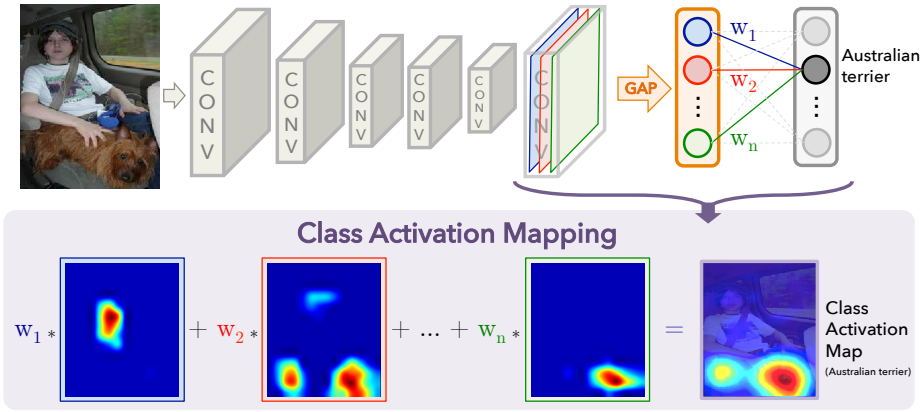


Figure 6-2: Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

- the network largely consists of convolutional layers, and just before the final output layer (softmax in the case of categorization), we perform global average pooling on the convolutional feature maps and use those as features for a fully-connected layer that produces the desired output (categorical or otherwise). Given this simple connectivity structure, we can identify the importance of the image regions by projecting back the weights of the output layer on to the convolutional feature maps, a technique we call class activation mapping.

As illustrated in Fig. 6-2, global average pooling outputs the spatial average of the feature map of each unit at the last convolutional layer. A weighted sum of these values is used to generate the final output. Similarly, we compute a weighted sum of the feature maps of the last convolutional layer to obtain our class activation maps. We describe this more formally below for the case of softmax. The same technique can be applied to regression and other losses.

For a given image, let $f_k(x, y)$ represent the activation of unit k in the last convolutional layer at spatial location (x, y). Then, for unit k, the result of performing global average pooling, F^k is $\sum_{x,y} f_k(x, y)$. Thus, for a given class c, the input to the softmax, S_c, is $\sum_k w_k^c F_k$ where w_k^c is the weight corresponding to class c for unit k. Essentially, w_k^c indicates the *importance* of F_k for class c. Finally the output of the softmax for class c, P_c is given by $\frac{\exp(S_c)}{\sum_c \exp(S_c)}$. Here we ignore the bias term: we explicitly set the input bias of the softmax to 0 as it has little to no impact on the classification performance.

By plugging $F_k = \sum_{x,y} f_k(x, y)$ into the class score, S_c, we obtain

$$\begin{aligned} S_c &= \sum_k w_k^c \sum_{x,y} f_k(x, y) \\ &= \sum_{x,y} \sum_k w_k^c f_k(x, y). \end{aligned} \tag{6.1}$$

We define M_c as the class activation map for class c, where each spatial element is given by

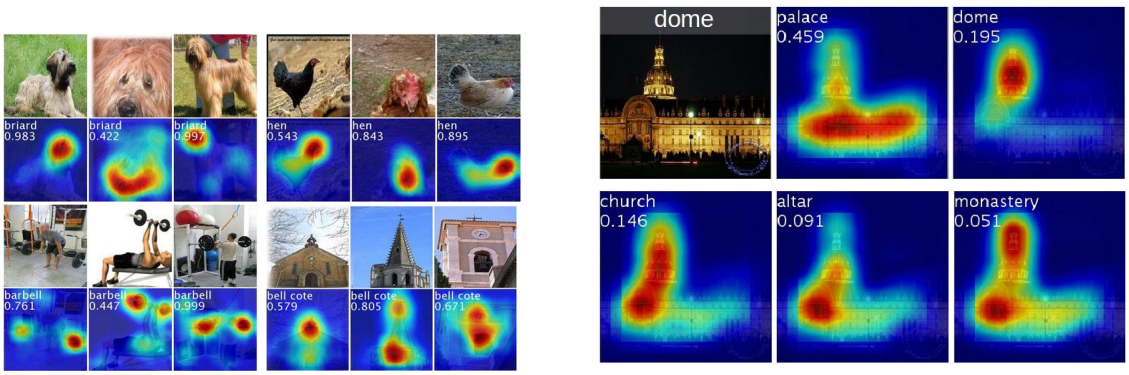
$$M_c(x, y) = \sum_k w_k^c f_k(x, y). \tag{6.2}$$

Thus, $S_c = \sum_{x,y} M_c(x, y)$, and hence $M_c(x, y)$ directly indicates the importance of the activation at spatial grid (x, y) leading to the classification of an image to class c.

Intuitively, based on prior works [112,119], we expect each unit to be activated by some visual pattern within its receptive field. Thus f_k is the map of the presence of this visual pattern. The class activation map is simply a weighted linear sum of the presence of these visual patterns at different spatial locations. By simply upsampling the class activation map to the size of the input image, we can identify the image regions most relevant to the particular category.

In Fig. 6-3a, we show some examples of the CAMs output using the above approach. We can see that the discriminative regions of the images for various classes are highlighted. In Fig. 6-3b we highlight the differences in the CAMs for a single image when using different classes c to generate the maps. We observe that the discriminative regions for different categories are different even for a given image. This suggests that our approach works as expected. We demonstrate this quantitatively in the sections ahead.

Global average pooling (GAP) vs global max pooling (GMP): Given the prior work [70] on using GMP for weakly supervised object localization, we believe it is important to highlight the intuitive difference between GAP and GMP. We believe that GAP loss encourages the network to identify the extent of the object as compared to GMP which encourages it to identify just one discriminative part. This is because, when doing the average of a map, the value can be maximized by finding *all* discriminative parts of an object as all



(a) (b)

Figure 6-3: (a) The CAMs of four classes from ImageNet. The maps highlight the discriminative image regions used for image classification e.g., the head of the animal for *briard* and *hen*, the plates in *barbell*, and the bell in *bell cote*. (b) Examples of the CAMs generated from the top 5 predicted categories for the given image with ground-truth as dome. The predicted class and its score are shown above each class activation map. We observe that the highlighted regions vary across predicted classes e.g., *dome* activates the upper round part while *palace* activates the lower flat part of the compound.

low activations reduce the output of the particular map. On the other hand, for GMP, low scores for all image regions except the most discriminative one do not impact the score as you just perform a max. We verify this experimentally on ILSVRC dataset in Sec. 6.2: while GMP achieves similar classification performance as GAP, GAP outperforms GMP for localization.

6.2 Weakly-supervised Object Localization

In this section, we evaluate the localization ability of CAM when trained on the ILSVRC 2014 benchmark dataset [79]. We first describe the experimental setup and the various CNNs used in Sec. 6.2.1. Then, in Sec. 6.2.2 we verify that our technique does not adversely impact the classification performance when learning to localize and provide detailed results on weakly-supervised object localization.

6.2.1 Setup

For our experiments we evaluate the effect of using CAM on the following popular CNNs: AlexNet [48], VGGnet [90], and GoogLeNet [93]. In general, for each of these networks

we remove the fully-connected layers before the final output and replace them with GAP followed by a fully-connected softmax layer.

We found that the localization ability of the networks improved when the last convolutional layer before GAP had a higher spatial resolution, which we term the *mapping resolution*. In order to do this, we removed several convolutional layers from some of the networks. Specifically, we made the following modifications: For AlexNet, we removed the layers after `conv5` (i.e., `pool5` to `prob`) resulting in a mapping resolution of 13×13. For VGGnet, we removed the layers after `conv5-3` (i.e., `pool5` to `prob`), resulting in a mapping resolution of 14×14. For GoogLeNet, we removed the layers after `inception4e` (i.e., `pool4` to `prob`), resulting in a mapping resolution of 14×14. To each of the above networks, we added a convolutional layer of size 3×3, stride 1, pad 1 with 1024 units, followed by a GAP layer and a softmax layer. Each of these networks were then fine-tuned[2] on the 1.3M training images of ILSVRC [79] for 1000-way object classification resulting in our final networks AlexNet-GAP, VGGnet-GAP and GoogLeNet-GAP respectively.

For classification, we compare our approach against the original AlexNet [48], VGGnet [90], and GoogLeNet [93], and also provide results for Network in Network (NIN) [55]. For localization, we compare against the original GoogLeNet[3], NIN and using backpropagation [88] instead of CAMs. Further, to compare average pooling against max pooling, we also provide results for GoogLeNet trained using global max pooling (GoogLeNet-GMP).

We use the same error metrics (top-1, top-5) as ILSVRC for both classification and localization to evaluate our networks. For classification, we evaluate on the ILSVRC validation set, and for localization we evaluate on both the validation and test sets.

6.2.2 Results

We first report results on object classification to demonstrate that our approach does not significantly hurt classification performance. Then we demonstrate that our approach is effective at weakly-supervised object localization.

Classification: Tbl. 6.1 summarizes the classification performance of both the origi-

[2]Training from scratch also resulted in similar performances.
[3]This has a lower mapping resolution than GoogLeNet-GAP.

nal and our GAP networks. We find that in most cases there is a small performance drop of $1 - 2\%$ when removing the additional layers from the various networks. We observe that AlexNet is the most affected by the removal of the fully-connected layers. To compensate, we add two convolutional layers just before GAP resulting in the AlexNet*-GAP network. We find that AlexNet*-GAP performs comparably to AlexNet. Thus, overall we find that the classification performance is largely preserved for our GAP networks. Further, we observe that GoogLeNet-GAP and GoogLeNet-GMP have similar performance on classification, as expected. Note that it is important for the networks to perform well on classification in order to achieve a high performance on localization as it involves identifying both the object category and the bounding box location accurately.

Localization: In order to perform localization, we need to generate a bounding box and its associated object category. To generate a bounding box from the CAMs, we use a simple thresholding technique to segment the heatmap. We first segment the regions of which the value is above 20% of the max value of the CAM. Then we take the bounding box that covers the largest connected component in the segmentation map. We do this for each of the top-5 predicted classes for the top-5 localization evaluation metric. Fig. 6-5(a) shows some example bounding boxes generated using this technique. The localization performance on the ILSVRC validation set is shown in Tbl. 6.2, and example outputs in Fig. 6-4.

We observe that our GAP networks outperform all the baseline approaches with GoogLeNet-GAP achieving the lowest localization error of 43% on top-5. This is remarkable given that this network was not trained on a single annotated bounding box. We observe that our CAM approach significantly outperforms the backpropagation approach of [88] (see Fig. 6-5(b) for a comparison of the outputs). Further, we observe that GoogLeNet-GAP significantly outperforms GoogLeNet on localization, despite this being reversed for classification. We believe that the low mapping resolution of GoogLeNet (7×7) prevents it from obtaining accurate localizations. Last, we observe that GoogLeNet-GAP outperforms GoogLeNet-GMP by a reasonable margin illustrating the importance of average pooling over max pooling for identifying the extent of objects.

To further compare our approach with the existing weakly-supervised [88] and fully-

Table 6.1: Classification error on the ILSVRC validation set.

Networks	top-1 val. error	top-5 val. error
VGGnet-GAP	33.4	12.2
GoogLeNet-GAP	35.0	13.2
AlexNet*-GAP	44.9	20.9
AlexNet-GAP	51.1	26.3
GoogLeNet	31.9	11.3
VGGnet	31.2	11.4
AlexNet	42.6	19.5
NIN	41.9	19.6
GoogLeNet-GMP	35.6	13.9

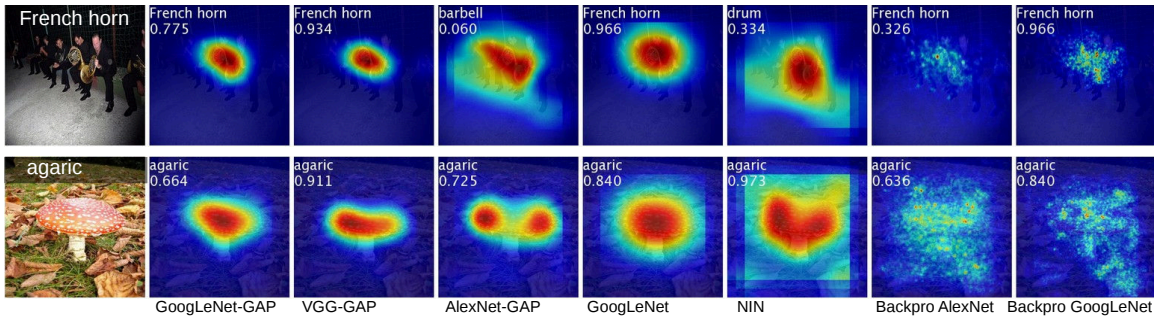


Figure 6-4: Class activation maps from CNN-GAPs and the class-specific saliency map from the backpropagation methods.

supervised [82, 93, 93] CNN methods, we evaluate the performance of GoogLeNet-GAP on the ILSVRC test set. We follow a slightly different bounding box selection strategy here: we select two bounding boxes (one tight and one loose) from the class activation map of the top 1st and 2nd predicted classes and one loose bounding boxes from the top 3rd predicted class. We found that this heuristic was helpful to improve performances on the validation set. The performances are summarized in Tbl. 6.3. GoogLeNet-GAP with heuristics achieves a top-5 error rate of 37.1% in a weakly-supervised setting, which is surprisingly close to the top-5 error rate of AlexNet (34.2%) in a fully-supervised setting. While impressive, we still have a long way to go when comparing the fully-supervised networks with the same architecture (i.e., weakly-supervised GoogLeNet-GAP vs fully-supervised GoogLeNet) for the localization.

Table 6.2: Localization error on the ILSVRC validation set. *Backprop* refers to using [88] for localization instead of CAM.

Method	top-1 val.error	top-5 val. error
GoogLeNet-GAP	**56.40**	**43.00**
VGGnet-GAP	57.20	45.14
GoogLeNet	60.09	49.34
AlexNet*-GAP	63.75	49.53
AlexNet-GAP	67.19	52.16
NIN	65.47	54.19
Backprop on GoogLeNet	61.31	50.55
Backprop on VGGnet	61.12	51.46
Backprop on AlexNet	65.17	52.64
GoogLeNet-GMP	57.78	45.26

Table 6.3: Localization error on the ILSVRC test set for various weakly- and fully- supervised methods.

Method	supervision	top-5 test error
GoogLeNet-GAP (heuristics)	weakly	**37.1**
GoogLeNet-GAP	weakly	42.9
Backprop [88]	weakly	46.4
GoogLeNet [93]	full	26.7
OverFeat [82]	full	29.9
AlexNet [93]	full	34.2

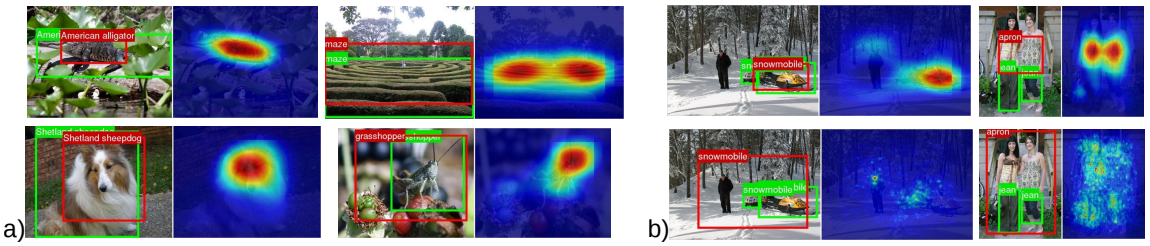


Figure 6-5: a) Examples of localization from GoogleNet-GAP. b) Comparison of the localization from GooleNet-GAP (upper two) and the backpropagation using AlexNet (lower two). The ground-truth boxes are in green and the predicted bounding boxes from the class activation map are in red.

6.3 Deep Features for Generic Localization

The responses from the higher-level layers of CNN (e.g., `fc6`, `fc7` from AlexNet) have been shown to be very effective generic features with state-of-the-art performance on a variety of image datasets [18, 78, 122]. Here, we show that the features learned by our GAP CNNs also perform well as generic features, and as bonus, identify the discriminative image regions used for categorization, despite not having being trained for those particular tasks. To obtain the weights similar to the original softmax layer, we simply train a linear SVM [25] on the output of the GAP layer.

First, we compare the performance of our approach and some baselines on the following scene and object classification benchmarks: SUN397 [108], MIT Indoor67 [76], Scene15 [50], SUN Attribute [73], Caltech101 [26], Caltech256 [35], Stanford Action40 [109], and UIUC Event8 [53]. The experimental setup is the same as in [122]. In Tbl. 6.5, we compare the performance of features from our best network, GoogLeNet-GAP, with the `fc7` features from AlexNet, and `ave pool` from GoogLeNet.

As expected, GoogLeNet-GAP and GoogLeNet significantly outperform AlexNet. Also, we observe that GoogLeNet-GAP and GoogLeNet perform similarly despite the former having fewer convolutional layers. Overall, we find that GoogLeNet-GAP features are competitive with the state-of-the-art as generic visual features.

More importantly, we want to explore whether the localization maps generated using our CAM technique with GoogLeNet-GAP are informative even in this scenario. Fig. 6-7 shows some example maps for various datasets. We observe that the most discriminative regions tend to be highlighted across all datasets. Overall, our approach is effective for generating localizable deep features for generic tasks.

In Sec. 6.3.1, we explore fine-grained recognition of birds and demonstrate how we evaluate the generic localization ability and use it to further improve performance. In Sec. 6.3.2 we demonstrate how GoogLeNet-GAP can be used to identify generic visual patterns from images.

Table 6.4: Fine-grained classification performance on CUB200 dataset. GoogLeNet-GAP can successfully localize important image crops, boosting classification performance.

Methods	Train/Test Anno.	Accuracy
GoogLeNet-GAP on full image	n/a	63.0%
GoogLeNet-GAP on crop	n/a	67.8%
GoogLeNet-GAP on BBox	BBox	70.5%
Alignments [30]	n/a	53.6%
Alignments [30]	BBox	67.0%
DPD [115]	BBox+Parts	51.0%
DeCAF+DPD [18]	BBox+Parts	65.0%
PANDA R-CNN [114]	BBox+Parts	76.4%

6.3.1 Fine-grained Recognition

In this section, we apply our generic localizable deep features to identifying 200 bird species in the CUB-200-2011 [107] dataset. The dataset contains 11,788 images, with 5,994 images for training and 5,794 for test. We choose this dataset as it also contains bounding box annotations allowing us to evaluate our localization ability. Tbl. 6.4 summarizes the results.

We find that GoogLeNet-GAP performs comparably to existing approaches, achieving an accuracy of 63.0% when using the full image without any bounding box annotations for both train and test. When using bounding box annotations, this accuracy increases to 70.5%. Now, given the localization ability of our network, we can use a similar approach as Sec. 6.2.2 (i.e., thresholding) to first identify bird bounding boxes in both the train and test sets. We then use GoogLeNet-GAP to extract features again from the crops inside the bounding box, for training and testing. We find that this improves the performance considerably to 67.8%. This localization ability is particularly important for fine-grained recognition as the distinctions between the categories are subtle and having a more focused image crop allows for better discrimination.

Further, we find that GoogLeNet-GAP is able to accurately localize the bird in 41.0% of the images under the 0.5 intersection over union (IoU) criterion, as compared to a chance performance of 5.5%. We visualize some examples in Fig. 6-6. This further validates the localization ability of our approach.

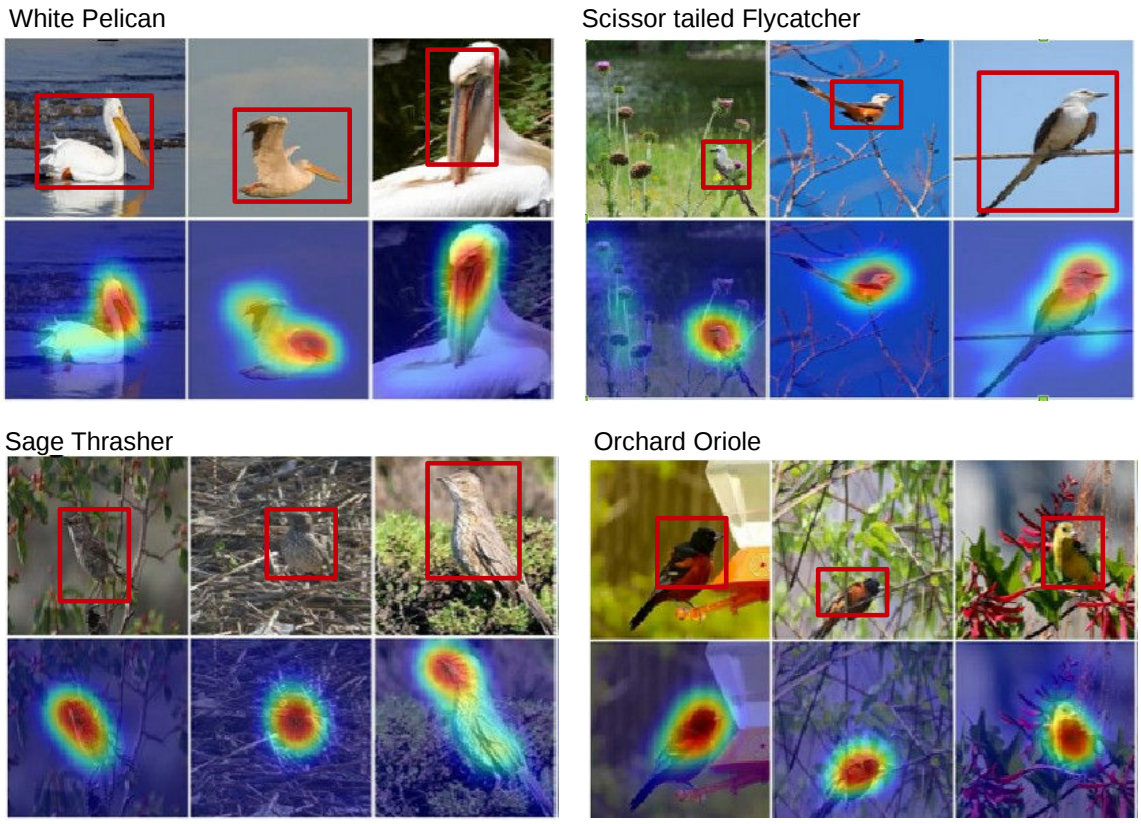


Figure 6-6: CAMs and the inferred bounding boxes (in red) for selected images from four bird categories in CUB200. In Sec. 6.3.1 we quantitatively evaluate the quality of the bounding boxes (41.0% accuracy for 0.5 IoU). We find that extracting GoogLeNet-GAP features in these CAM bounding boxes and re-training the SVM improves bird classification accuracy by about 5% (Tbl. 6.4).

Table 6.5: Classification accuracy on representative scene and object datasets for different deep features.

	SUN397	MIT Indoor67	Scene15	SUN Attribute	Caltech101
`fc7` from AlexNet	42.61	56.79	84.23	84.23	87.22
`ave pool` from GoogLeNet	51.68	66.63	88.02	92.85	92.05
`gap` from GoogLeNet-GAP	51.31	66.61	88.30	92.21	91.98

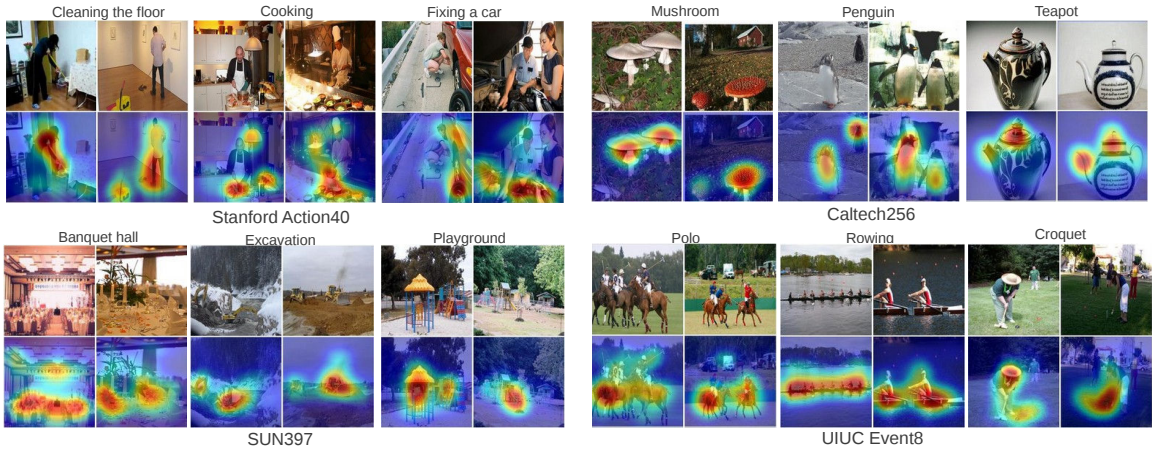


Figure 6-7: Generic discriminative localization using our GoogLeNet-GAP deep features (which have been trained to recognize objects). We show 2 images each from 3 classes for 4 datasets, and their class activation maps below them. We observe that the discriminative regions of the images are often highlighted e.g., in Stanford Action40, the mop is localized for *cleaning the floor*, while for *cooking* the pan and bowl are localized and similar observations can be made in other datasets. This demonstrates the generic localization ability of our deep features.

6.3.2 Pattern Discovery

In this section, we explore whether our technique can identify common elements or patterns in images beyond objects, such as text or high-level concepts. Given a set of images containing a common concept, we want to identify which regions our network recognizes as being important and if this corresponds to the input pattern. We follow a similar approach as before: we train a linear SVM on the GAP layer of the GoogLeNet-GAP network and apply the CAM technique to identify important regions. We conducted three pattern discovery experiments using our deep features. The results are summarized below. Note that in this case, we do not have train and test splits – we just use our CNN for visual pattern discovery.

Discovering informative objects in the scenes: We take 10 scene categories from the SUN dataset [108] containing at least 200 fully annotated images, resulting in a total of 4675 fully annotated images. We train a one-vs-all linear SVM for each scene category and compute the CAMs using the weights of the linear SVM. In Fig. 6-8 we plot the CAM for the predicted scene category and list the top 6 objects that most frequently overlap with the high CAM activation regions for two scene categories. We observe that the high activation

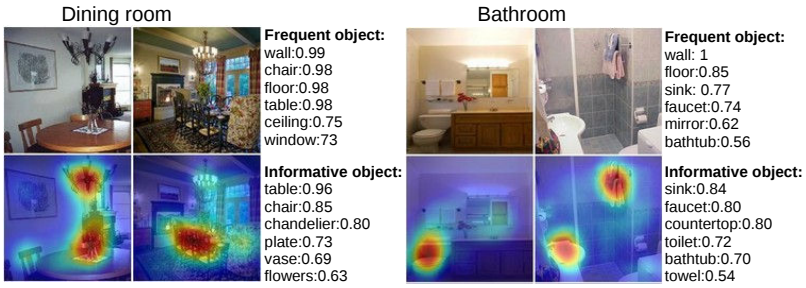


Figure 6-8: Informative objects for two scene categories. For the dining room and bathroom categories, we show examples of original images (top), and list of the 6 most frequent objects in that scene category with the corresponding frequency of appearance. At the bottom: the CAMs and a list of the 6 objects that most frequently overlap with the high activation regions.

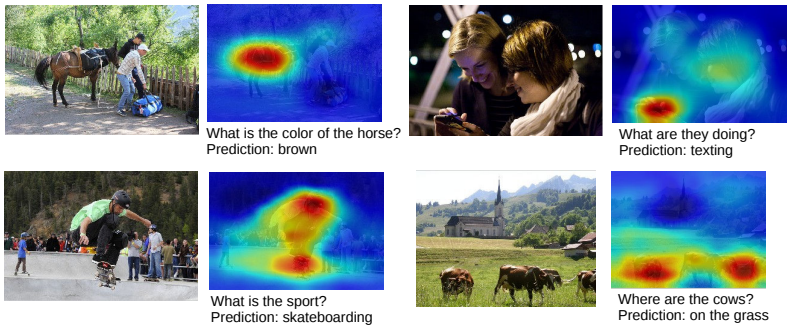


Figure 6-9: Examples of highlighted image regions for the predicted answer class in the visual question answering.

regions frequently correspond to objects indicative of the particular scene category.

Interpreting visual question answering: We use our approach and localizable deep feature in the baseline proposed in [123] for visual question answering. It has overall accuracy 55.89% on the test-standard in the Open-Ended track. As shown in Fig. 6-9, our approach highlights the image regions relevant to the predicted answers.

Chapter 7

Efficient Deep Visual Representations for Activity Recognition

Going beyond image recognition, activity recognition in videos has been one of the core topics in computer vision. However, it remains difficult due to the ambiguity of describing activities at appropriate timescales [85]. Many video datasets, such as UCF101 [92], Sport1M [45], and THUMOS [33], include many activities that can be recognized without reasoning about the long-term temporal relations: still frames and optical flow are sufficient to identify many of the labeled activities. Indeed, the classical two-stream Convolutional Neural Network [89] and the recent I3D Network [9], both based on frames and optical flow, perform activity recognition very well on these datasets.

However, convolutional neural networks still struggle in situations where data and observations are limited, or where the underlying structure is characterized by transformations and temporal relations, rather than the appearance of certain entities [49, 81]. It remains remarkably challenging for convolutional neural networks to reason about temporal relations and to anticipate what transformations are happening to the observations. Fig.7-1 shows such examples. The networks are required to discover visual common sense knowledge over time beyond the appearance of objects in the frames and the optical flow.

The ability to reason about the relations between entities over time is crucial for intelligent decision-making. Temporal relational reasoning allows intelligent species to analyze the current situation relative to the past and formulate hypotheses on what may happen next.

Figure 7-1: What takes place between two observations? (see answer below the first page). Humans can easily infer the temporal relations and transformations between these observations, but this task remains difficult for neural networks.

For example (Fig.7-1), given two observations of an event, people can easily recognize the temporal relation between two states of the visual world and deduce what has happened between the two frames of a video[1].

In this chapter, we propose a simple and interpretable network module called Temporal Relation Network (TRN) [118] that enables temporal relational reasoning in neural networks. This module is inspired by the relational network proposed in [81], but instead of modeling the spatial relations, TRN aims to describe the temporal relations between observations in videos. Thus, TRN can learn and discover possible temporal relations at multiple time scales. TRN is a general and extensible module that can be used in a plug-and-play fashion with any existing CNN architecture. We apply TRN-equipped networks on three recent video datasets (Something-Something [34], Jester [1], and Charades [86]), which are constructed for recognizing different types of activities such as human-object interactions and hand gestures, but all depend on temporal relational reasoning. The TRN-equipped networks achieve very competitive results even given only discrete RGB frames, bringing significant improvements over baselines. Thus TRN provides a practical solution for standard neural networks to solve activity recognition tasks using temporal relational reasoning.

[1] Answer: a) Poking a stack of cans so it collapses; b) Stack something; c) Tidying up a closet; d) Thumb up.

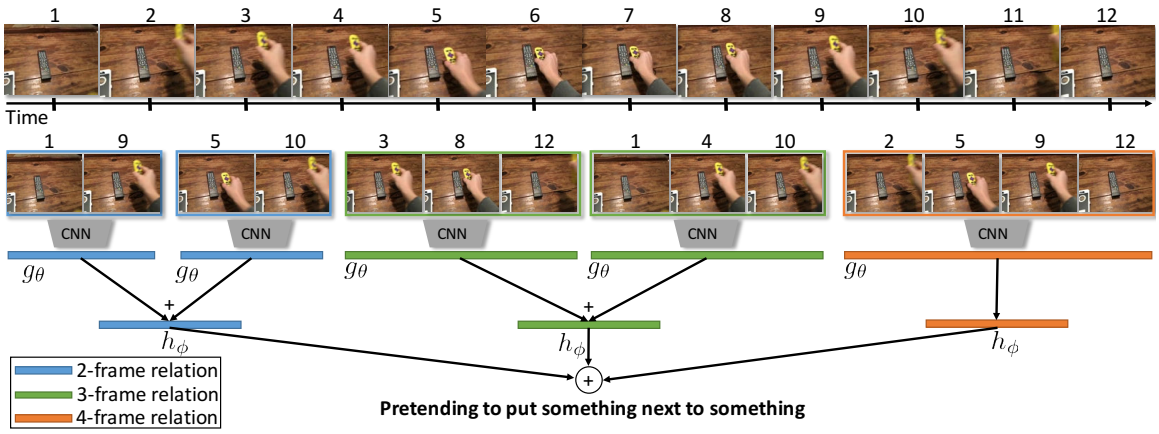


Figure 7-2: The illustration of Temporal Relation Networks. Representative frames of a video (shown above) are sampled and fed into different frame relation modules. Only a subset of the 2-frame, 3-frame, and 4-frame relations are shown, as there are higher frame relations included.

7.1 Temporal Relation Networks

In this section, we introduce the framework of Temporal Relation Networks. It is simple and can be easily plugged into any existing convolutional neural network architecture to enable temporal relational reasoning. In later experiments, we show that TRN-equipped networks discover interpretable visual common sense knowledge to recognize activities in videos.

7.1.1 Defining Temporal Relations

Inspired by the relational reasoning module for visual question answering [81], we define the pairwise temporal relation as a composite function below:

$$T_2(V) = h_\phi\Big(\sum_{i<j} g_\theta(f_i, f_j)\Big) \tag{7.1}$$

where the input is the video V with n selected ordered frames as $V = \{f_1, f_2, ..., f_n\}$, where f_i is a representation of the i^{th} frame of the video, e.g., the output activation from some standard CNN. The functions h_ϕ and g_θ fuse features of different ordered frames. Here we simply use multilayer perceptrons (MLP) with parameters ϕ and θ respectively. For

efficient computation, rather than adding all the combination pairs, we uniformly sample frames i and j and sort each pair.

We further extend the composite function of the 2-frame temporal relations to higher frame relations such as the 3-frame relation function below:

$$T_3(V) = h_\phi^{'}\Big(\sum_{i<j<k} g_\theta^{'}(f_i, f_j, f_k)\Big) \tag{7.2}$$

where the sum is again over sets of frames i, j, k that have been uniformly sampled and sorted.

7.1.2 Multi-Scale Temporal Relations

To capture temporal relations at multiple time scales, we use the following composite function to accumulate frame relations at different scales:

$$MT_N(V) = T_2(V) + T_3(V)... + T_N(V) \tag{7.3}$$

Each relation term T_d captures temporal relationships between d ordered frames. Each T_d has its own separate $h_\phi^{(d)}$ and $g_\theta^{(d)}$. Notice that for any given sample of d frames for each T_d, all the temporal relation functions are end-to-end differentiable, so they can all be trained together with the base CNN used to extract features for each video frame. The overall network framework is illustrated in Fig.7-2.

7.1.3 Efficient Training and Testing

When training a multi-scale temporal network, we could sample the sums by selecting different sets of d frames for each T_d term for a video. However, we use a sampling scheme that reduces computation significantly. First, we uniformly sample a set of N frames from the video, $V_N^* \subset V$, and we use V_N^* to calculate $T_N(V)$. Then, for each $d < N$, we choose k random subsamples of d frames $V_{kd}^* \subset V_N^*$. These are used to compute the d-frame relations

Table 7.1: Statistics of the datasets used in evaluating the TRNs.

Dataset	Classes	Videos	Type
Something	174	108,499	human-object interaction
Jester	27	148,092	human hand gesture
Charades	157	9,848	daily indoor activity

for each $T_d(V)$. This allows $O(kN^2)$ temporal relations to be sampled while evaluating the CNN on only N frames.

At testing time, we can combine the TRN-equipped network with a queue to process streaming video very efficiently. A queue is used to cache the extracted CNN features of the equidistant frames sampled from the video, then those features are further combined into different relation tuples which are further summed up to predict the activity. The CNN feature is extracted from incoming key frame only once then enqueued, thus TRN-equipped networks is able to run in real-time on a desktop to processing streaming video from a webcam.

7.2 Experiments

We evaluate the TRN-equipped networks on a variety of activity recognition tasks. For recognizing activities that depend on temporal relational reasoning, TRN-equipped networks outperform a baseline network without a TRN by a large margin. We achieve highly competitive results on the Something-Something dataset for human-interaction recognition [34] and on the Jester dataset for hand gesture recognition [1]. The TRN-equipped networks also obtain competitive results on activity classification in the Charades dataset [86], outperforming the Flow+RGB ensemble models [84, 86] using only sparsely sampled RGB frames.

The statistics of the three datasets Something-Something dataset [34], Jester dataset [1], and Charades dataset [86] are listed in Table 7.1. All three datasets are crowd-sourced, in which the videos are collected by asking the crowd-source workers to record themselves performing instructed activities. Unlike the Youtube-type videos in UCF101 and Kinetics, there is usually a clear start and end of each activity in the crowd-sourced video, emphasiz-

ing the importance of temporal relational reasoning.

7.2.1 Network Architectures and Training

The networks used for extracting image features play an important factor in visual recognition tasks [83]. Features from deeper networks such as ResNet [37] usually perform better. Our goal here is to evaluate the effectiveness of the TRN module for temporal relational reasoning in videos. Thus, we fix the base network architecture to be the same throughout all the experiments and compare the performance of the CNN model with and without the proposed TRN modules.

We adopt Inception with Batch Normalization (BN-Inception) pretrained on ImageNet used in [42] because of its balance between accuracy and efficiency. We follow the training strategies of partial BN (freezing all the batch normalization layers except the first one) and dropout after global pooling as used in [103]. We keep the network architecture of the MultiScale TRN module and the training hyper-parameters the same for training models on all the three datasets. We set $k = 3$ in the experiments as the number of accumulated relation triples in each relation module. g_ϕ is simply a two-layer MLP with 256 units per layer, while h_ϕ is a one-layer MLP with the unit number matching the class number. The CNN features for a given frame is the activation from the BN-Inception's global average pooling layer (before the final classification layer). Given the BN-Inception as the base CNN, the training can be finished in less than 24 hours for 100 training epochs on a single Nvidia Titan Xp GPU. In the Multi-Scale TRN, we include all the TRN modules from 2-frame TRN up to 8-frame TRN, as including higher frame TRNs brings marginal improvement and lowers the efficiency.

7.2.2 Results on Something-Something Dataset

Something-Something is a recent video dataset for human-object interaction recognition. There are 174 classes, some of the ambiguous activity categories are challenging, such as 'Tearing Something into two pieces' versus 'Tearing Something just a little bit', 'Turn something upside down' versus 'Pretending to turn something upside down'. We can see that the temporal relations and transformations of objects rather than the appearance of the

objects characterize the activities in the dataset.

The results on the validation set are listed in Table 7.2a, in which we compare the top1 and top5 accuracy for the base network trained on single frames randomly selected from each video, the base networks with various frame relation modules, and the multi-scale TRN-equipped network. Networks with TRNs outperform the single frame baseline by a large margin, while additional frames included in the relation bring further improvements. The Multi-Scale TRN with ten crop data augmentation achieves the best performance. To further evaluate how the contribution of optical flow improve the result, we train a TRN with

We also compare the TRN with TSN [103], to verify the importance of temporal orders. Instead of concatenating the features of temporal frames, TSN simply averages the deep features so that the model only captures the co-occurrence rather than the temporal ordering of patterns in the features. We keep all the training conditions the same, and vary the number of frames used by two models. As shown in the bottom part of Table 7.2a, our models outperform TSNs by a large margin. This result shows the importance of frame order for temporal relation reasoning.

We further evaluate how important is the optical flow information, we train a MultiScale TRN on optical flow. It gets 31.63% on the validation set. We further construct a 2-stream TRN by combining the MultiScale TRN trained on RGB images with the MultiScale TRN trained on optical flows (simply average the predicted probabilities from the the two streams for any given video). The 2-stream TRN further improves the accuracy on the validation set to 42.01%.

We submit the prediction of the MultiScale TRN and the 2-stream TRN on the test set. As shown in Table 7.2b, we achieve very competitive scores, compared to other methods at the leaderboard. Note that most of the methods submitted to the leaderboard are non-public available methods, which should be considered as concurrent work with our method.

model	Top1 acc.(%)
single frame	11.41
3-frame TRN	26.22
5-frame TRN	30.39
7-frame TRN	31.01
MultiScale TRN	34.44
3-frame TSN	17.30
5-frame TSN	18.11
7-frame TSN	18.48
2-Stream TRN	42.01

(a)

model	Top1 acc.(%)
Yana Hasson	25.55
Harrison.AI	26.38
I3D by [34]	27.23
Guillaume Berger	30.48
Besnet	31.66
MultiScale TRN	33.60
2-Stream TRN	39.46

(b)

Table 7.2: (a) Results on the validation set of Something-Something Dataset. All the models use the center cropping of the equidistant frames in the video. Multi-scale TRN with 10-crop augmentation achieves the best performance. TRN also outperforms TSN in a large margin, showing the importance of temporal order. (b) Results on the test set of the Something-Something Dataset. Comparison methods are from the official public leaderboard.

model	Top1 acc.(%)
single frame	63.60
2-frame TRN	75.65
3-frame TRN	81.45
4-frame TRN	89.38
5-frame TRN	91.40
MultiScale TRN	**95.31**

(a)

model	Top1 acc.(%)
20BN Jester System	82.34
VideoLSTM	85.86
Guillaume Berger	93.87
Ford's Gesture System	94.11
Besnet	94.23
MultiScale TRN	**94.78**

(b)

Table 7.3: Jester Dataset Results on (a) the validation set and (b) the test set.

7.2.3 Results on Jester and Charades

We further evaluate the TRN-equipped networks on the Jester dataset, which is a video dataset for hand gesture recognition with 27 classes. The results on the validation set of the Jester dataset are listed in Table 7.3a. The result on the test set and comparison with the top methods are listed in Table 7.3b. MultiScale TRN again achieves competitive performance as close to 95% Top1 accuracy.

We evaluate the MultiScale TRN on the recent Charades dataset for daily activity recognition. The results are listed in Table 7.4. Our method outperforms various methods such as 2-stream networks and C3D [86], and the recent Asynchronous Temporal Field (Temp-

Table 7.4: Results on Charades Activity Classification.

Approach	Random	C3D	AlexNet	IDT	2-Stream	TempField	Ours
mAP	5.9	10.9	11.3	17.2	14.3	22.4	**25.2**

Field) method [84].

The qualitative prediction results of the Multi-Scale TRN on the three datasets are shown in Figure 7-3. The examples in Figure 7-3 demonstrate that the TRN model is capable of correctly identifying actions for which the overall temporal ordering of frames is essential for a successful prediction. For example, the turning hand counterclockwise category would assume a different class label when shown in reverse. Moreover, the successful prediction of categories in which an individual *pretends* to carry out an action (e.g. 'pretending to put something into something' as shown in the second row) suggests that the network can capture temporal relations at multiple scales, where the ordering of several lower-level actions contained in short segments conveys crucial semantic information about the overall activity class.

This outstanding performance shows the effectiveness of the TRN for temporal relational reasoning and its strong generalization ability across different datasets.

7.2.4 Interpreting Visual Common Sense Knowledge inside the TRN

One of the distinct properties of the proposed TRNs compared to previous video classification networks such as C3D [98] and I3D [9] is that TRN has more interpretable structure. In this section, we have a more in-depth analysis to interpret the visual common sense knowledge learned by the TRNs through solving these temporal reasoning tasks. We explore the following four parts:

Representative frames of a video voted by the TRN to recognize an activity. Intuitively, a human observer can capture the essence of an action by selecting a small collection of representative frames. Does the same hold true for models trained to recognize the activity? To obtain a sequence of representative frames for each TRN, we first compute the features of the equidistant frames from a video, then randomly combine them to generate different frame relation tuples and pass them into the TRNs. Finally we rank the relation tu-

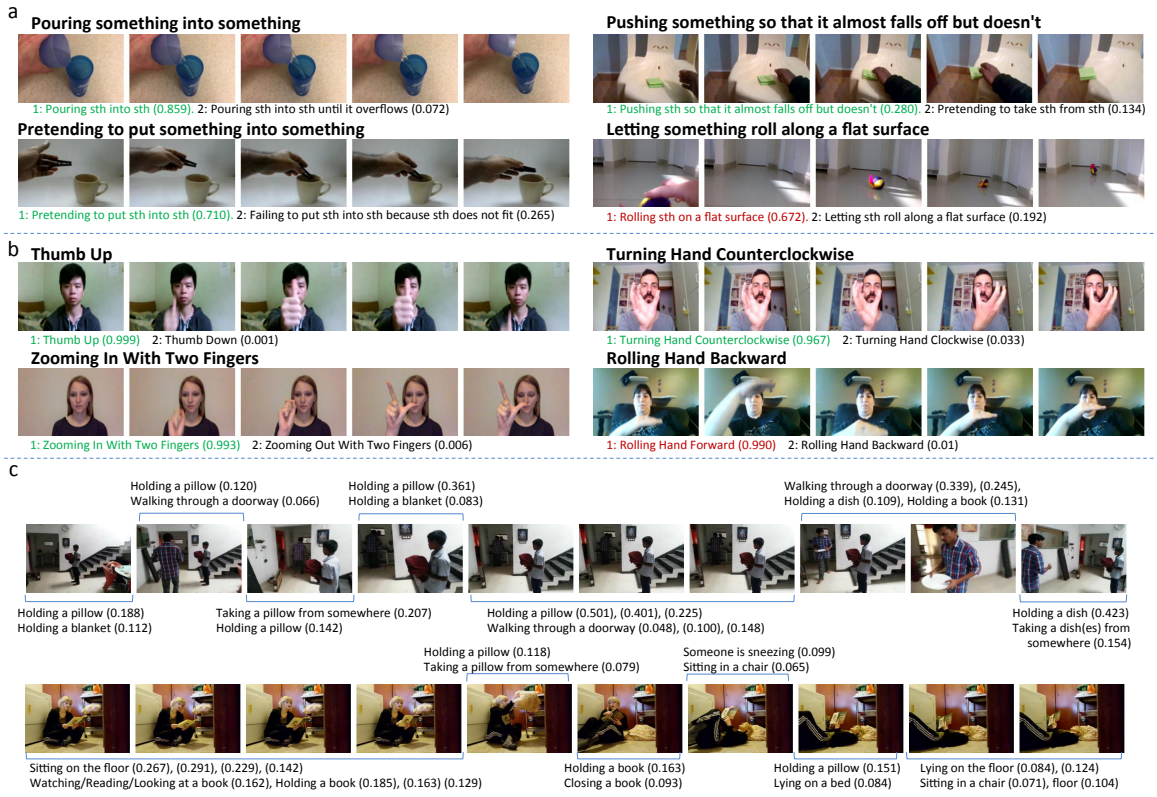


Figure 7-3: Prediction examples on a) Something-Something, b) Jester, and c) Charades. For each example drawn from Something-Something and Jester, the top two predictions with green text indicating a correct prediction and red indicating an incorrect one. Top 2 predictions are shown above Charades frames.

ples using the responses of different TRNs. Figure 7-4 shows the top representative frames voted by different TRNs to recognize an activity in the same video. We can see that the TRNs learn the temporal relations that characterize an activity. For comparatively simple actions, a single frame is sufficient to establish some degree of confidence in the correct action, but is vulnerable to mistakes when a transformation is present. 2-frame TRN picks up the two frames that best describe the transformation. Meanwhile, for more difficult activity categories such as 'Pretending to poke something', two frames are not sufficient information for even a human observer to differentiate. Similarly, the network needs additional frames in the TRNs to correctly recognize the behavior.

Thus the progression of representative frames and their corresponding class predictions inform us about how temporal relations may help the model reason about more complex behavior. One particular example is the last video in Figure 7-4: The action's context given by a single frame - a hand close to a book - is enough to narrow down the top prediction to a qualitatively plausible action, unfolding something. A similar, two-frame relation marginally increases the probability the initial prediction, although these two frames would not be sufficient for even human observers to make the correct prediction. Now, the three frame-relation begins to highlight a pattern characteristic to Something-Something's set of *pretending* categories: the initial frames closely resemble a certain action, but the later frames are inconsistent with the completion of that action as if it never happened. This relation helps the model to adjust its prediction to the correct class. Finally, the upward motion of the individual's hand in the third frame of the 4-frame relation further increases the discordance between the *anticipated* and *observed* final state of the scene; a motion resembling the action appeared to take place with no effect on the object, thus, solidifying confidence in the correct class prediction.

Temporal Alignment of Videos. The observation that the representative frames identified by the TRN are consistent across instances of an action category suggests that the TRN is well suited for the task of temporally aligning videos with one another. Here, we wish to synchronize actions across multiple videos by establishing a correspondence between their frame sequences. Given several video instances of the same action, we first select the most representative frames for each video and use their frame indices as "landmark", temporal

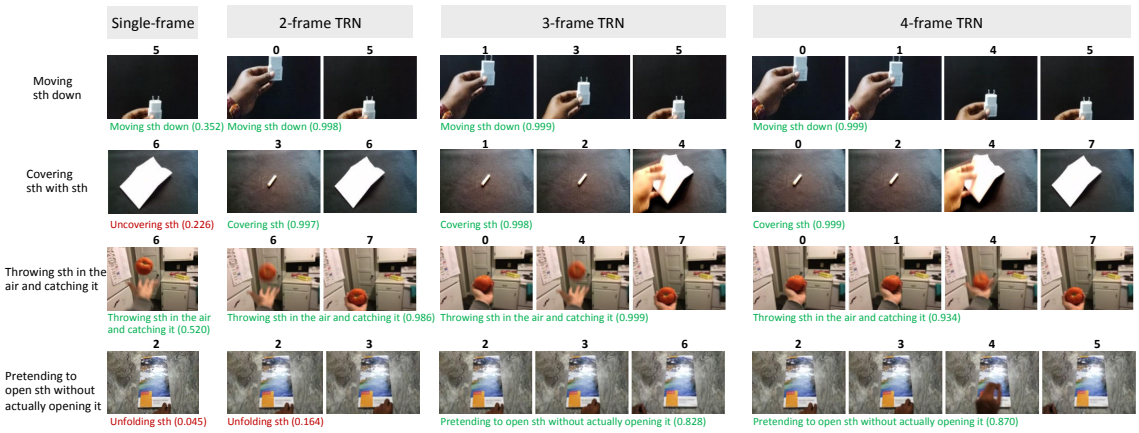


Figure 7-4: The top representative frames determined by single frame baseline network, the 2-frame TRN, 3-frame TRN, and 4-frame TRN. TRNs learn to capture the essence of an activity only given a limited number of frames. Videos are from the validation set of the Something-Something dataset

anchor points.Then, we alter the frame rate of video segments between two consecutive anchor points such that all of the individual videos arrive at the anchor points at the same time. Fig.7-5 shows the samples from the aligned videos. We can see different stages of an action are captured by the temporal relation. The temporal alignment is also an exclusive application of our TRN model, which cannot be done by previous video networks 3D convNet or two-stream networks.

Importance of temporal order for activity recognition. To verify the importance of the temporal order of frames for activity recognition, we conduct an experiment to compare the scenario with input frames in temporal order and in shuffled order when training the TRNs, as shown in Figure 7-6a. For training the shuffled TRNs, we randomly shuffle the frames in the relation modules. The significant difference on the Something-Something dataset shows the importance of the temporal order in the activity recognition. More interestingly, we repeat the same experiment on the UCF101 dataset [92] and observe no difference between the ordered frames and shuffled frames. That shows activity recognition for the Youtube-type videos in UCF101 doesn't necessarily require the temporal reasoning ability since there are not so many casual relations associated with an already on-going activity.

To further investigate how temporal ordering influences activity recognition in TRN, we examine and plot the categories that show the largest differences in the class accuracy

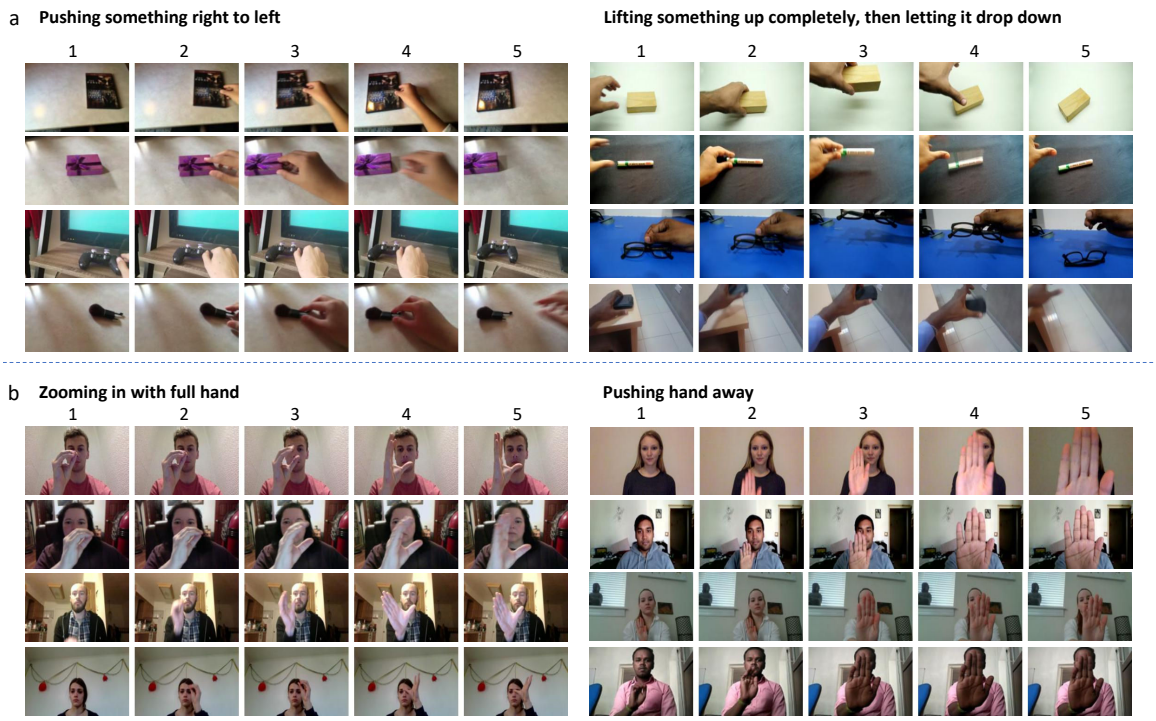
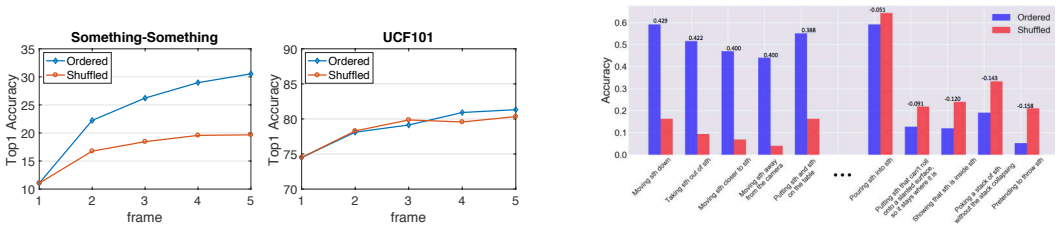


Figure 7-5: Temporal alignment of videos from the (a) Something-Something and (b) Jester datasets using the most representative frames as temporal anchor points. For each action, 4 different videos are aligned using 5 temporal anchor points.

between ordered and shuffled inputs drawn from the Something-Something dataset, in Figure 7-6b. In general, actions with strong 'directionality' and large, one-way movements, such as 'Moving something down', appear to benefit the most from preserving the correct temporal ordering. This observation aligns with the idea that the disruption of continuous motion and a potential consequence of shuffling video frames, would likely confuse a human observer, as it would go against our intuitive notions of physics.

Interestingly, the penalty for shuffling frames of relatively static actions is less severe if penalizing at all in some cases, with several categories marginally benefiting from shuffled inputs, as observed with the category 'putting something that can't roll onto a slanted surface so it stays where it is'. Here, simply learning the coincidence of frames rather than temporal transformations may be sufficient for the model to differentiate between similar activities and make the correct prediction.

Particularly in challenging ambiguous cases, for example 'Pretending to throw something' where the release point is partially or completely obscured from view, disrupting



(a) (b)

Figure 7-6: (a) Accuracy obtained using ordered frames and shuffled frames, on Something-Something and UCF101 dataset respectively. On Something-Something, the temporal order is critical for recognizing the activity. But recognizing activities in UCF101 does not necessarily require temporal relational reasoning. (b) The top 5 action categories that exhibited the largest gain and the least gain (negative) respectively between ordered and shuffled frames as inputs. Actions with directional motion appear to suffer most from shuffled inputs.

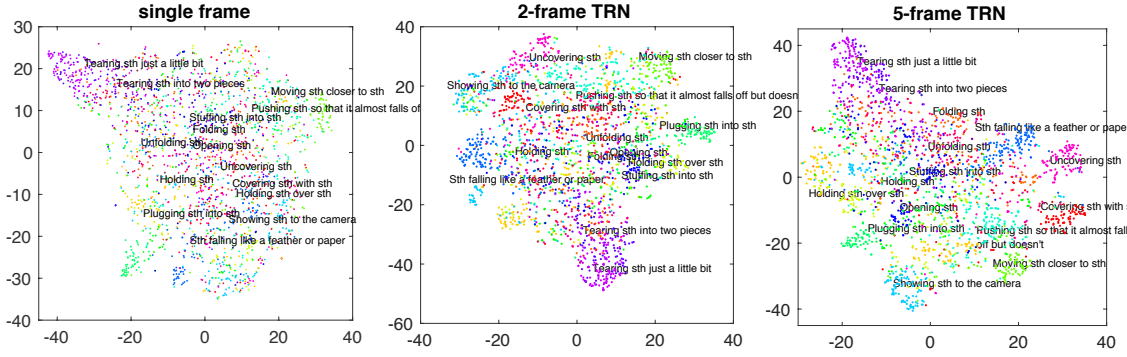


Figure 7-7: t-SNE plot of the video samples of the 15 classes using the deep features from the single-frame baseline, 2-frame TRN, and 5-frame TRN. Higher frame TRN can better differentiate activities in Something-Something dataset.

a strong 'sense of motion' may bias model predictions away from the likely alternative, 'throwing something', frequently but incorrectly selected by the ordered model, thus giving rise to a curious difference in accuracy for that action.

t-SNE visualization of activity similarity. Figure 7-7 shows the t-SNE visualization for embedding the high-level features from the single frame baseline, the 3-frame TRN, and the 5-frame TRN, for the videos of the 15 most frequent activity classes in the validation set. We can see that the features from 2-frame and 5-frame TRNs can better differentiate activity categories. We also observe the similarity among categories in the visualization map. For example, 'Tearing something into two pieces' is very similar to 'Tearing something just a little bit', and the categories 'Folding something', 'Unfolding something', 'Holding something', 'Holding something over something' are clustered together.

Table 7.5: Early activity recognition using the MultiScale TRN on Something-Something and Jester dataset. Only the first 25% and 50% of frames are given to the TRN to predict activities. Baseline is the model trained on single frames.

	Something		Jester	
Data	baseline	TRN	baseline	TRN
first 25%	9.08	11.14	27.25	34.23
first 50%	10.10	19.10	41.43	78.42
full	11.41	33.01	63.60	93.70

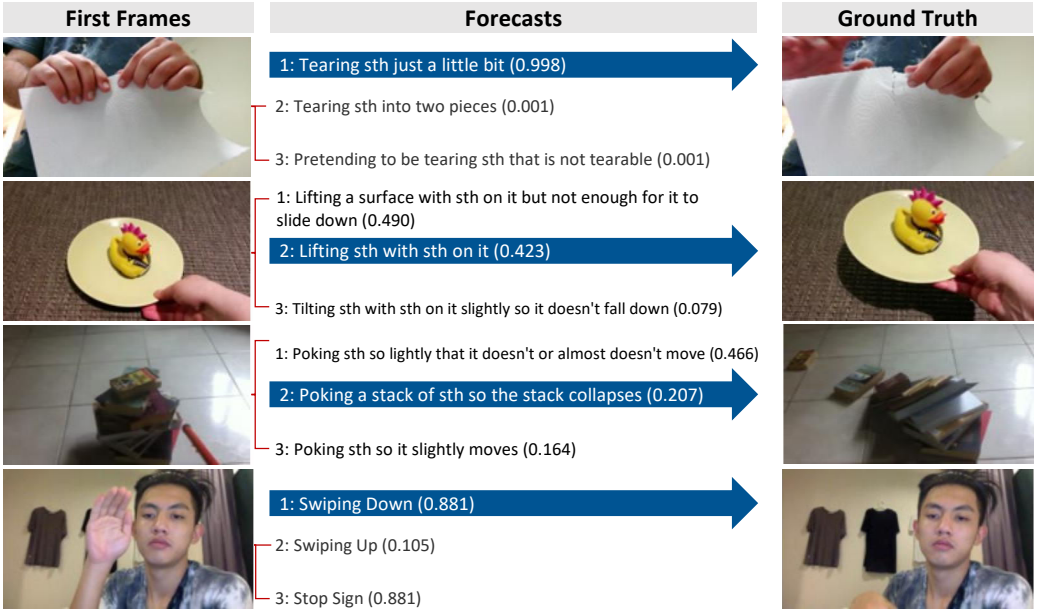


Figure 7-8: Early recognition of activity when only given the first 25% frames. The first 25% of each video, represented by the first frame shown in the left column, is used to generate the top 3 anticipated forecasts and corresponding probabilities listed in the middle column. The ground truth label is highlighted by a blue arrow which points to the last frame of the video on the right.

Early Activity Recognition. Recognizing activities early or even anticipating and forecasting activities before they happen or fully happen is a challenging yet less explored problem in activity recognition. Here we evaluate our TRN model on early recognition of activity when given only the first 25% and 50% of the frames in each validation video. Results are shown in Table 7.5. For comparison, we also include the single frame baseline. We see that TRN can use the learned temporal relations to anticipate activity. The performance increases as more ordered frames are received. Figure 7-8 shows some examples of anticipating activities using only first 25% and 50% frames of a video. A qualitative review of these examples reveals that model predictions on only initial frames do serve as very reasonable forecasts despite being given task with a high degree of uncertainty even for human observers.

Chapter 8

Discussion and Future Work

The majority of this thesis has focused on advancing the frontier of visual intelligence toward human-level scene recognition and a deeper understanding of how this level of recognition is accomplished in complex deep neural network models. I would like to conclude this thesis with the future direction. My future work involves developing interpretable and efficient intelligent systems that can effortlessly understand the visual world and interact with the environment around them. My research will progress along the following paths:

Video Scene Understanding. I aim to scale visual recognition systems from processing scenes in still images to handling the much more complex scenes that occur in videos. I have begun work in this direction by developing the Moments Dataset [63] which contains 1 million videos that have been annotated according to over 300 atomic action classes, such as "opening", "rolling" and "bouncing". Using Moments, I will explore holistic scene understanding approaches that are able to recognize objects and their spatial context, as well as dynamic interactions and events between objects. For example, *cooking* is composed of several atomic actions, such as *washing* bowls, *cutting* vegetables, and *boiling* water; my goal would be to identify each of them and their chain relations. This kind of video scene understanding will have broad real-world applications in robotics and visual analytics. I will further interpret the models of video recognition, to understand their successes and failures, and the visual knowledge learned inside, thus eventually improve their performance.

Human and Object Interactions in Scenes. To navigate inside a complex scene, a

major challenge for robot visual systems is learning to execute actions. Most actions that we take when interacting with objects are part of a complex causal chain. If a system can understand cause-and-effect relationships, it can then make a multi-step plan toward a goal. I recently started investigating efficient and interpretable models that can discover the time-dependent causalities in videos, such as inferring the ways in which objects are transforming between one timestamp and another [118]. It is crucial to build perception systems that are able not only to recognize the objects and their scene context, but also to forecast and anticipate the consequences of potential actions that may be taken in a particular scene.

Developing Interpretable and Explainable Machine Learning Models. Various network architectures and training methods have been proposed to boost model performance. However, little research has been done to improve model interpretability. Following my work on Network Dissection which is able to measure interpretability, my next step is to improve the interpretability of deep models. Some of these models have recently demonstrated better-than-human performance on various challenging tasks, including playing Go [87] and diagnosing skin cancer [21]. What knowledge has been learned by and stored within these models? How can Go players or dermatologists benefit from this knowledge? I will investigate novel approaches for extracting knowledge from such machine learning models used in visual recognition, autonomous driving and medical diagnosis. I will create new human-computer interfaces capable of translating the knowledge into understandable and useful tools for human learning. Through these efforts, I hope to push the boundaries of interpretable AI and advance multiple relevant scientific fields such as healthcare domain where it is crucial to have interpretable and explainable machine learning models.

Bibliography

[1] Twentybn jester dataset: a hand gesture dataset. *https://www.twentybn.com/datasets/jester*, 2017.

[2] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *Proc. ICCV*, 2015.

[3] Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. *Proc. ECCV*, 2014.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv:1511.00561*, 2015.

[5] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proc. CVPR*, 2017.

[6] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Trans. on Graphics (SIGGRAPH)*, 2014.

[7] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[8] Irving Biederman. *Visual object recognition*, volume 2. MIT press Cambridge, 1995.

[9] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. *arXiv preprint arXiv:1705.07750*, 2017.

[10] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *arXiv:1606.00915*, 2016.

[11] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Nam-Gyu Cho, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proc. CVPR*, 2014.

[12] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proc. CVPR*, 2014.

[13] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proc. CVPR*, 2014.

[14] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *Proc. CVPR*, 2016.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009.

[16] Persi Diaconis. What is a random matrix? *Notices of the AMS*, 52(11):1348–1349, 2005.

[17] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proc. CVPR*, 2015.

[18] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, 2014.

[19] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[20] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[21] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[22] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int'l Journal of Computer Vision*, 2010.

[23] Mark Everingham, Andrew Zisserman, Christopher KI Williams, Luc Van Gool, Moray Allan, Christopher M Bishop, Olivier Chapelle, Navneet Dalal, Thomas Deselaers, Gyuri Dorkó, et al. The pascal visual object classes challenge 2007 (voc2007) results. 2007.

[24] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. 2008.

[25] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 2008.

[26] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 2007.

[27] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Temporal localization of actions with actoms. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2782–2795, 2013.

[28] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Activity representation with motion hierarchies. *International journal of computer vision*, 107(3):219–238, 2014.

[29] Ruohan Gao, Dinesh Jayaraman, and Kristen Grauman. Object-centric representation learning from unlabeled videos. *arXiv:1612.00500*, 2016.

[30] Efstratios Gavves, Basura Fernando, Cees GM Snoek, Arnold WM Smeulders, and Tinne Tuytelaars. Local alignments for fine-grained categorization. *Int'l Journal of Computer Vision*, 2014.

[31] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[32] Abel Gonzalez-Garcia, Davide Modolo, and Vittorio Ferrari. Do semantic parts emerge in convolutional neural networks? *arXiv:1607.03738*, 2016.

[33] A Gorban, H Idrees, YG Jiang, A Roshan Zamir, I Laptev, M Shah, and R Sukthankar. Thumos challenge: Action recognition with a large number of classes. In *CVPR workshop*, 2015.

[34] Raghav Goyal, Samira Kahou, Vincent Michalski, Joanna Materzyńska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. *Proc. ICCV*, 2017.

[35] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. CVPR*, 2015.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proc. CVPR*, 2016.

[39] C.H.R. Heip, P.M.J. Herman, and K. Soetaert. Indices of diversity and evenness. *Oceanis*, 1998.

[40] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[41] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *Proc. CVPR*, 2017.

[42] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

[43] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *Proc. ICCV*, 2015.

[44] Yangqing Jia. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`, 2013.

[45] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proc. CVPR*, 2014.

[46] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[47] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *ijcv*, 2016.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[49] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, pages 1–101, 2016.

[50] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006.

[51] Quoc V Le. Building high-level features using large scale unsupervised learning. In *ICASSP*, 2013.

[52] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[53] Li-Jia Li and Li Fei-Fei. What, where and who? classifying events by scene and object recognition. In *Proc. ICCV*, 2007.

[54] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? *arXiv:1511.07543*, 2015.

[55] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[56] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[57] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 2015.

[58] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. *Proc. CVPR*, 2015.

[59] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proc. CVPR*, 2015.

[60] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[61] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*, 2001.

[62] Ishan Mikjjsra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *Proc. ECCV*, 2016.

[63] Mathew Monfort, Bolei Zhou, Sarah Adel Bargal, Alex Andonian, Tom Yan, Kandan Ramakrishnan, Lisa Brown, Quanfu Fan, Dan Gutfruend, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *arXiv preprint arXiv:1801.03150*, 2018.

[64] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proc. CVPR*, 2014.

[65] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proc. CVPR*, 2014.

[66] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *In Advances in Neural Information Processing Systems*, 2016.

[67] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proc. CVPR*, 2015.

[68] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proc. ECCV*, 2016.

[69] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int'l Journal of Computer Vision*, 2001.

[70] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free? âĂŞ weakly-supervised learning with convolutional neural networks. *Proc. CVPR*, 2015.

[71] Andrew Owens, Jiajun Wu, Josh H McDermott, William T Freeman, and Antonio Torralba. Ambient sound provides supervision for visual learning. In *Proc. ECCV*, 2016.

[72] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proc. CVPR*, 2016.

[73] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *Proc. CVPR*, 2012.

[74] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph.*, 2003.

[75] Hamed Pirsiavash and Deva Ramanan. Parsing videos of actions with segmental grammars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 612–619, 2014.

[76] A Quattoni and A Torralba. Recognizing indoor scenes. In *Proc. CVPR*, 2009.

[77] R Quian Quiroga, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–1107, 2005.

[78] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. *arXiv:1403.6382*, 2014.

[79] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int'l Journal of Computer Vision*, 2015.

[80] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *Int'l Journal of Computer Vision*, 2008.

[81] Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*, 2017.

[82] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[83] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014.

[84] Gunnar A Sigurdsson, Santosh Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. 2017.

[85] Gunnar A Sigurdsson, Olga Russakovsky, and Abhinav Gupta. What actions are needed for understanding human actions in videos? *arXiv preprint arXiv:1708.02696*, 2017.

[86] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016.

[87] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

[88] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *International Conference on Learning Representations Workshop*, 2014.

[89] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *In Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[90] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[91] Edward H Simpson. Measurement of diversity. *Nature*, 1949.

[92] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *Proc. CVPR*, 2012.

[93] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proc. CVPR*, 2015.

[94] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015.

[95] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.

[96] Keiji Tanaka. Neuronal mechanisms of object recognition. *Science*, 262(5134):685–688, 1993.

[97] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Proc. CVPR*, 2011.

[98] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proc. CVPR*, 2015.

[99] Joost Van De Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. Learning color names for real-world applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523, 2009.

[100] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *arXiv:1609.02612*, 2016.

[101] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proc. ICCV*, pages 3551–3558, 2013.

[102] Limin Wang, Yu Qiao, and Xiaoou Tang. Mofap: A multi-level representation for action recognition. *International Journal of Computer Vision*, 119(3):254–271, 2016.

[103] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proc. ECCV*, 2016.

[104] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. Actions˜ transformations. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2658–2667, 2016.

[105] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proc. CVPR*, 2015.

[106] Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. *arXiv preprint arXiv:1708.02901*, 2017.

[107] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical report, California Institute of Technology, 2010.

[108] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*, 2010.

[109] Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas Guibas, and Li Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *Proc. ICCV*, 2011.

[110] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *In Advances in Neural Information Processing Systems*, 2014.

[111] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[112] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Proc. ECCV*, 2014.

[113] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2017.

[114] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. *Proc. ECCV*, 2014.

[115] Ning Zhang, Ryan Farrell, Forrest Iandola, and Trevor Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. *Proc. ICCV*, 2013.

[116] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Proc. ECCV*. Springer, 2016.

[117] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proc. CVPR*, 2017.

[118] Bolei Zhou, Alex Andonian, and Antonio Torralba. Temporal relational reasoning in videos. *arXiv preprint arXiv:1711.08496*, 2017.

[119] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *International Conference on Learning Representations*, 2015.

[120] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proc. CVPR*, 2016.

[121] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[122] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. *In Advances in Neural Information Processing Systems*, 2014.

[123] Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167*, 2015.

[124] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proc. CVPR*, 2017.