Analysis of Multilayer Neural Networks for Object Recognition

Pulkit Agrawal

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-202
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-202.html

December 1, 2014

Acknowledgement

Thanks to Prof. Jitendra Malik for his guidance, support and providing me with the freedom to explore various research directions. Many ideas in this work were developed in close collaboration with Dr. Ross Girshick. I am indebted to him. Thanks to Prof. Bruno Olshaussen, Prof. Jack Gallant, Prof. Alyosha Efros and Prof. Benjamin Recht for their support and wisdom.

Special thanks to Bharath Hariharan and Saurabh Gupta, other lab-mates in Malik group and floor mates on 7th floor of SDH. Its been a pleasure working with you all.

Thanks to Mayur Mudigonda for always being there. A big thank you to all the supporting staff including Xuan Quach, Judy Tam and Angie Abbatecola. Thanks to Fulbright Fellowship for funding my research and NVIDIA for generously donating GPUs.

Analysis of Multilayer Neural Networks for Object Recognition

by

Pulkit Agrawal

B.Tech., Indian Institute of Technology Kanpur (2011)

A thesis submitted in partial satisfaction of the
requirements for the degree of

Master of Science

in the

Department of Electrical Engineering and Computer Science

at

University of California, Berkeley

Committee in charge:

Professor Jitendra Malik
Professor Alexei A. Efros

Spring 2014

Analysis of Multilayer Neural Networks for Object Recognition

by Pulkit Agrawal

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Jitendra Malik
Research Advisor

(Date)

* * * * * * *

Professor Alexei A. Efros
Second Reader

(Date)

Acknowledgements

I would like to thank Professor Jitendra Malik for his guidance, support and providing me with the freedom to explore various research directions. Many ideas in this work were developed in close collaboration with Dr. Ross Girshick. Thanks to him for his patience and more than a helping hand. Thanks to Professor Bruno Olshaussen, Professor Jack Gallant and Professor Benjamin Recht for their support and wisdom. I would also like to thank Professor Alyosha Efros.

Bharath Hariharan and Saurabh Gupta, my lab-mates in Malik Group have been really helpful in providing feedback, discussing research ideas and other aspects of graduate student life. It has been a pleasure working with the group and other fellow floor mates on the 7th floor of SDH.

Thanks, to Mayur Mudigonda for always being there and supporting me during all the hard times.

Last, but not the least I would like to thank all the supporting staff including Xuan Quach, Judy Tam and Angie Abbatecola.

Abstract

In 2012 Krizhevsky et al. demonstrated that a Convolutional Neural Network (CNN) trained on large amount of data as part of the ImageNet challenge significantly outperformed traditional computer vision approaches on image classification. Subsequently, Girshick et al. (2013) exploited these features to establish the new state of the art on PASCAL object detection. This suggests that computer vision maybe in the process of a feature revolution akin to that following SIFT and HOG nearly a decade ago. It is therefore important to get more insights into features learned by these networks. Our work provides answer to the following four questions: (a) What happens during finetuning of a discriminatively pretrained network? (b) How much information is in the location and how much of it is in the magnitude of filter activation? (c) Does a multilayer CNN contain Grand-Mother Cells? How distributed is the code? (d) How does training of CNN progress over time? Does too much pre-training hurt generalization performance?

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The breakthrough work of [2] created a splash in the computer vision community by presenting a convolutional neural network model which easily surpassed all existing methods on the ImageNet ILSVRC-2012 challenge [3]. The top-5 error rates dropped by an exceptional amount to 16.4% from 26.2 % (achieved by the second best alternative based on fisher vectors). At that time, it was unclear if these networks would be useful for other computer vision tasks. The recent work of [4] demonstrated that features learnt by such networks generalize and achieve state of the art results on classification datasets such as SUN-397, Caltech-UCSD Birds and Caltech-101 among others.

In a more recent development (R-CNN)[1], features extracted using convolutional networks were successfully used to achieve results on object detection which dwarf the existing state of art by a big margin. They achieved a mAP of 54.1 (as compared to 41.7 in [5]) on the PASCAL VOC 2007 detection challenge and showed impressive results on the task of semantic segmentation. The significance of these results can be well appreciated in light of the fact that negligible progress was made over the last few iterations of the PASCAL-VOC challenge. These observations strongly suggest that we might be in the middle of a feature revolution akin to the one ushered by introduction of HOG [6] and SIFT [7] in the mid 2000s.

It is not the first time that convolutional neural networks (CNNs) have generated great interest in the computer vision community. In late 80s and early nineties LeNet [8] achieved state of art performance on the task of MNIST digit classification. By the end of nineties and throughout the last decade interest in neural networks waned. One of the main reasons behind this was the fact that a large number of parameters such as the number of layers, number of units in each layer, the learning rate needed to be manually set in order to successfully train these networks. Support Vector machines on the other hand provided an easy alternative for achieving the same performance levels with only one parameter (C) to tune. However, given the impressive performance of CNN's - the stage is all set for their second renaissance in mainstream computer vision.

We take the view that rich feature hierarchies provided by convolutional nets are very likely to emerge as the prominent feature extractor for computer vision models over the next few years. Feature extractors such as SIFT and HOG afford an intuitive interpretation of templates composed of oriented edge filters. However, currently we have little understanding of what different layers of a deep convolutional network encode and what is the

most efficient way of using this information. We believe that developing such an understanding is an interesting scientific pursuit as well as the stepping stone towards designing methods which can optimally use these features. This work is focussed on a scientific investigation of multilayer convolutional neural networks centred around answers to four questions detailed below:

For a long time proponents of multilayer networks have argued that unsupervised pretraining followed by finetuning is helpful for improving performance on discriminative tasks such as image classification [9, 10, 11]. However recent work of [4] and [1] have made a strong case for the utility of learning features using discriminative pretraining and finetuning them for a specific task at hand. This leads us to our first question,

"What happens during finetuning of a discriminatively pretrained network?"

Most popular computer vision models can be categorized either as a Bag of Words models or template based models. We would like to understand if the features from the conv-net could be interpreted in any of these ways. More concretely we wish to understand,

"How much information is in the location and how much of it is in the magnitude of filter activation?"

Existence of Grand-Mother cells (units tuned to a very specific visual entity) has been a hotly debated topic in neuroscience [12] and has been fairly discussed in papers such as [9] proposing multilayer architectures for object recognition. We explore this question in the form of:

"Does a multilayer CNN contain Grand-Mother Cells? Or, in other words, how distributed is the learned representation?"

The last and the final question we address is,

"How does the training of CNN progress over time? Is too much pre-training bad for generalization?"

The next chapters in this thesis provide answers to these questions.

1.1 Network-Architecture and Nomenclature

For all our experiments we closely follow the architecture proposed in [2]. The first 2 layers consist of 4 sublayers each - convolution (conv), followed by rectified linear units (relu), pooling (pool) and contrast normalization (norm). Layers 3, 4 are composed of convolutional units followed by relu units. Layer 5 consists of convolutional units, followed by relu and pooling. The last two layers are fully connected (fc). In this work when we refer to a layer without referring to a particular sub-layer - then for layer 1,2,5 we mean the output of the pooling stage and for layers 3,4,6,7 we mean the output of relu units. We trained all our models using the publically available code [13] and Nvidia K40 GPUs. Our imagenet network was trained for 310000 iterations and achieves an error rate only about 2% higher on the ILSVRC validation set 2012.

We use the term Alex-Net to refer to a CNN trained using the architecture described above. FT (ft) or FT-Net refers to a finetuned network whereas as FC-FT(fc-ft) or FC-FT-Net refers to a network finetuned by setting the learning rate of convolutional layers to zero. We use the terms CNNs and conv-nets synonymously to refer to Alex-Net kind of multilayer network architectures. Terms filter/unit are used interchangeably to refer to filters of the CNN and GT-BBOX/gt-bbox stands for Ground truth bounding boxes from the PASCAL-VOC-2007 detection challenge and mAP refers to mean average precision [14].

1.2 Training Setup

Unless otherwise specified all our results for image and gt-bbox classification are obtained by training on linear SVM's on train-val sets of PASCAL-VOC-2007 [14] challenge and tested on the test set. For detection we closely follow the R-CNN approach described in [1].
For our experiments with the SUN-397 [15] dataset we use a non-standard train-test split since it is infeasible to finetune conv-nets for 10 different subsets as proposed in [15]. In particular we randomly split the dataset into 3 parts namely train-val-test using 50%,10% and 40%. The distribution of classes is uniform across all the 3 sets. We only use these results to support our investigations and not to compare with other scene-classification methods.

Fine-Tuning

For a particular task, we fine-tune conv-nets by running SGD (Stochastic Gradient) with a starting learning rate set to $\frac{1}{10}^{th}$ of the initial learning rate of the imagenet model. This choice has been made because we do not want to drastically change the parameters of the network and overfit to the training set. At every 20,000 iterations we reduce the learning rate by a factor of 10 and use a mini-batch size of 128. In order to finetune our networks for PASCAL-VOC-2007 detection setting we use region proposals for training closely following the procedure described in [1]. Unless otherwise stated, the finetuned network for PASCAL experiments is finetuned for 70K iterations whereas that for SUN is finetuned for 40K iterations.

Chapter 2

What happens during fine-tuning of discriminatively pre-trained network?

2.1 Introduction

Finetuning a network is the process of slowly updating pre-learned parameters to minimize a target loss function for a new task at hand. Since, convolutional networks have a large number of parameters they are prone to overfitting when trained on small datasets. Finetuning can be considered as a method of transfer learning and recent results from [1][4] present a strong case that this methodology boosts performance. Although, unsupervised pretraining has widely studied in the multilayer network literature [11][10], till date, there has been almost no work analysing the effect of fine-tuning on different layers of a discriminatively trained multilayer convolutional networks such as the Alex-Net.

We start our analysis by investigating how the entropy of filters across different layers changes as a result of discriminative fine-tuning (see sec 2.3). Since, entropy of a filter can be evaluated at different threshold level of activations we propose the metric of Area under the Entropy curve (AuE) to judge changes in filter selectivity. Our main finding is that most of the learning during finetuning happens only in the top two fully connected layers. Motivated by this observation, we finetune networks for PASCAL detection and SUN-397 scene classification task by setting non-zero learning rates only in the top 2 layers (see sec2.3.1). We find this results in a negligible drop in performance and allows for 2x speed-up in finetuning time. Other conclusions are presented in the sec 2.4.

2.2 Entropy Computation

The choice of using entropy as a metric is motivated by the fact that entropy naturally measures how selective a particular feature/filter is across different classes and not just individual classes. Consequently, entropy has been extensively used in a lot of computer vision applications based on decision trees such as [16] [17].

We define the entropy of a filter with respect to a given set of image-label pairs in the following way. Each image, when passed through the convolutional neural network

produces a $p \times p$ heatmap of filter responses. (eg p = 6, for a layer 5 filter). Further assume that there are C classes.

Label Entropy

We vectorize the heatmap into a vector of scores of length p^2 and with each element of this vector we associate the class label of the image. Thus, for each image we have a score vector and a label vector of length p^2 each. Next, we concatenate score vectors and label vectors from N images into a giant score vector and a giant label vector of size Np^2 each. Now for every score threshold (t) we consider all the labels which have an associated score $\geq t$. We construct a normalized histogram of label counts from this set of labels. The entropy of this histogram is the entropy of the filter at this threshold. As this threshold changes, entropy traces out a curve which we call as the entropy curve.

Weighted Label Entropy

When computing the label histogram as described above, instead of the label count we use the sum of the scores associated with the labels to construct the histogram. (Note: Since we are using outputs from relu or rectified linear units, all scores are ≥ 0.)

Spatial-Max (spMax) Label Entropy

From the heatmap, we chose the element which has the maximum value and associate with it the label of the image class. Thus, for each image we have a score vector and a label vector of length 1 each. Next, we concatenate score vectors and label vectors from N images into a score vector and a label vector of size N each. Now for every score threshold (t) we consider all the labels which have an associated score $\geq t$. Next, we compute the entropy as in the case of Label-Entropy.

2.2.1 Mean Cumulative Area Under Entropy Curve (MCAuE Index)

In order to compute entropy of the layer, here we introduce the notions of Area Under the Entropy Curve and Mean Cumulative Area Under Entropy Curve (MCAuE) analogous to the notion of computing area under Mean Average-Precision ([14]).

For each filter in the layer we first calculate the area under entropy (AuE) curve. Next, filters are sorted according to their AuE. Then, we calculate the Cumulative Area under Entropy (CAuE) by computing the cumulative sum of sorted list of AuE of all filters of this layer. Finally, to normalize for the number of filters we divide the i^{th} entry of the CAuE list by i. This results into a number which we call as the Mean Cumulative Area Under the Entropy Curve (MCAuE). In order to compare different layers, we compare MCAuE at 30 threshold points. The k^{th} threshold point for layer l corresponds to MCAuE of top $N_l(k/30)$ filters, where N_l is number of filters in layer l. Note, that a lower value of MC-AuE means that a layer is more selective/discriminative.

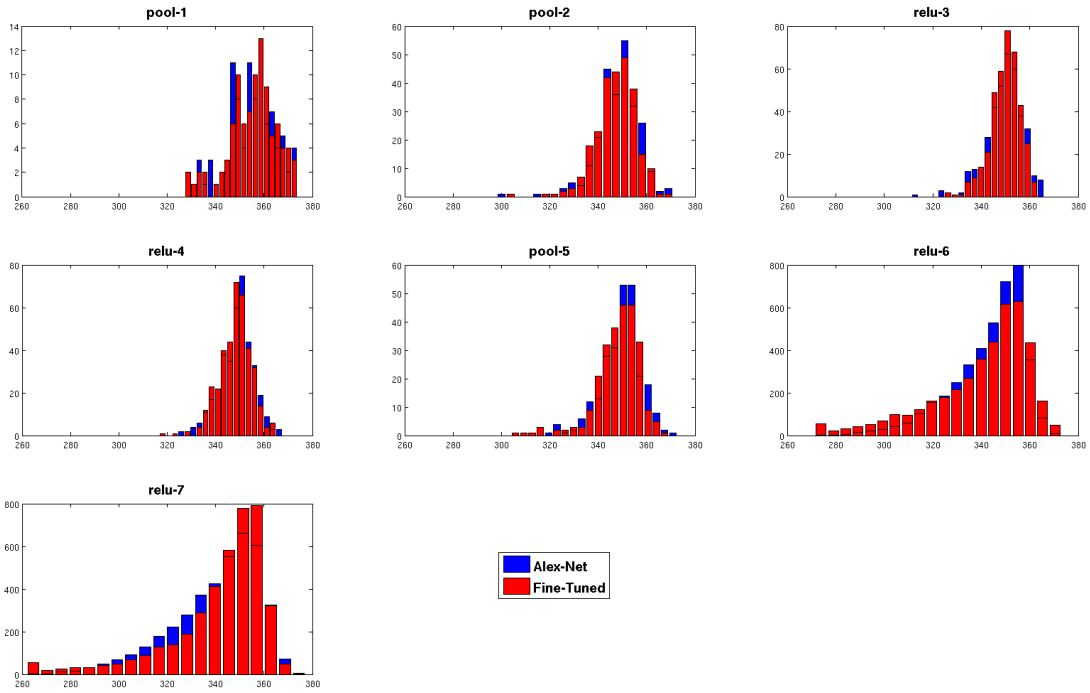

Figure 2-1: Distribution of AuE for different layers in Alex-Net and FT-Net. X-axis is the entropy and the Y-axis is the number of filters. Notice that the left tail for relu 6 and 7 becomes heavier after finetuning. This indicates that finetuning makes these filters more discriminative.

2.3 Analysis

We compute the AuE for each filter in a layer and corresponding MCAuE curves (using the definition of entropy described in sec 2.2) for all layers of Alex-Net and FT-Net using the ground-truth bounding boxes taken from the VOC-2007 test-set to the study the effect of fine-tuning.

The distribution of AuE for all filters across the seven layers of the CNN is illustrated in fig 2-1. The change in MCAuE for different layers as measured using various definitions of layer entropy can be seen in fig 2-2. A quantitative measure of change in entropy after finetuning is provided in table 2.1. The percentage change in MCAuE is calculated as:

$$\text{Percent Decrease} = 100 \times \frac{MCAuE_{untuned} - MCAuE_{fine}}{MCAuE_{untuned}} \tag{2.1}$$

where, $MCAuE_{fine}$ is for fine-tuned network and $MCAuE_{untuned}$ is for network trained on imagenet only.

From figures 2-1 and 2-2, we draw 2 conclusions:

1. Although the entropy of filters decreases as we move to higher layers, the magnitude of change in entropy between layer 1 and layer 5 is small as compared to the change which happens when going from layer 5 to layer 6.

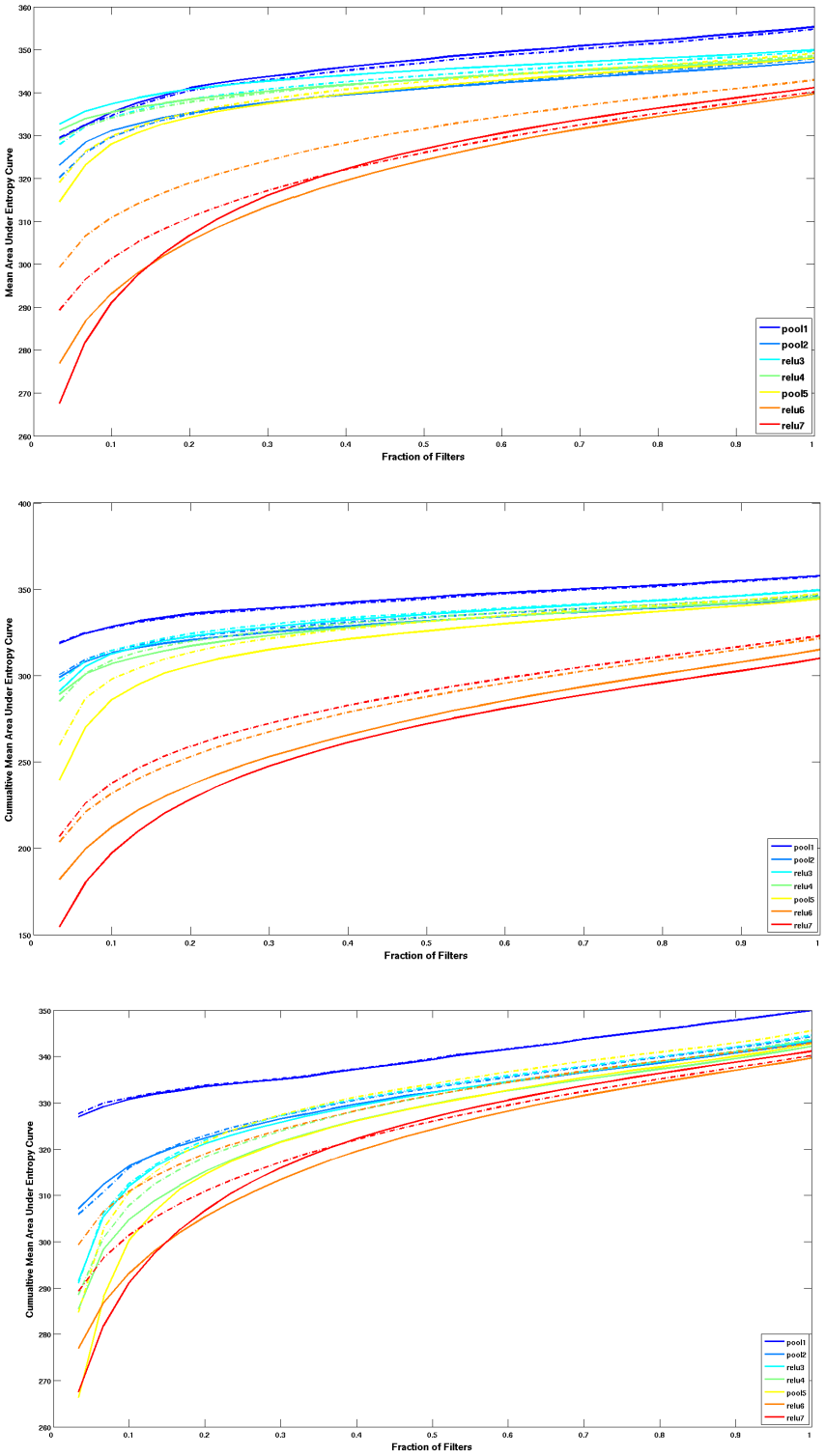

Figure 2-2: Mean Cumulative AuE plotted as fraction of filters for all layers of the Conv-Net. Dash-Dot Line: Alex-Net, Solid Line: Fine-Tuned Network. The top plot shows entropy calculated using Label-Entropy Method, the middle plot uses Weighted-Label-Entropy Method, whereas in the bottom plot entropy calculated using spMax-Label-Entropy Method. (see sec 2.2 for method definitions.)

Table 2.1: This table lists percentage decrease in MCAuE as a result of finetuning when only 0.1, 0.25, 0.50 and 1.00 fraction of all the filters were used for computing MCAuE. A lower MCAuE indicates that filters in a layer are more selective/class specific. The 0.1 fraction includes the top 10% most selective filters, 0.25 is top 25% of most selective filters. Consequently, comparing MCAuE at different fraction of filters gives a better sense of how selective the "most" selective filters have become to how selective all the filters have become. A negative value in the tabel below indicates increase in entropy. Note that for all the metrics maximum decrease in entropy takes place while moving from layer 5 to layer 7. Also, note that for relu-6 and relu-7 the values in Label-Entropy and spMax-Label-Entropy are same as relu-6 and 7 have spatial maps of size 1.

Layer	Label-Entropy				Weighted-Label-Entropy				spMax-Label-Entropy			
	0.1	0.25	0.5	1.0	0.1	0.25	0.5	1	0.1	0.25	0.5	1.0
pool-1	−0.02	−0.14	−0.19	−0.19	0.06	−0.13	−0.16	−0.16	0.19	0.10	0.07	0.04
pool-2	−0.71	−0.31	−0.14	0.01	0.41	0.53	0.58	0.57	−0.39	−0.03	0.11	0.23
relu-3	−1.14	−0.86	−0.67	−0.44	1.11	0.66	0.52	0.32	0.14	0.20	0.32	0.33
relu-4	−0.54	−0.31	−0.19	−0.05	−0.10	0.55	0.64	0.57	0.93	0.97	0.80	0.65
pool-5	0.97	0.55	0.43	0.36	5.84	3.53	2.66	1.85	4.87	3.05	2.31	1.62
relu-6	6.52	5.06	3.92	2.64	9.59	7.55	6.08	4.27	6.52	5.06	3.92	2.64
relu-7	5.17	2.66	1.33	0.44	20.58	14.75	11.12	7.78	5.17	2.66	1.33	0.44

Table 2.2: Comparison in performance on of Alex-Net, Finetuned Network(ft-net) and a network with only fc layers finetuned (fc-ft).

Layer	detection(mAP)			sun-397 (accuracy)		
	alex-net	ft-net	fc-ft	alex-net	ft-net	fc-ft
relu-7	45.5	54.1	53.3	53.1	56.8	56.2

2. Finetuning mostly reduces the entropy of filters in the fully connected layers, there are small changes in layer 5 and negligible changes in other layers.

It is noteworthy, that the label-entropy metric indicates that pool-5 undergoes negligible change after fine-tuning, whereas the weighted-label and spMax-label-entropy metrics indicate that changes are comparable to changes in relu-6 and relu-7 (see table 2.1). Thus, we can either conclude either that layers 1-5 are generic or layers 1-4 are generic depending on the choice of the metric. From the results presented in sec 2.3.1 it will become clear to the reader that layers 1-5 are indeed generic.

2.3.1 Is finetuning only the fully-connected layers sufficient?

The observations made in 2.3 indicate that finetuning the convolutional layers in the CNN may not be critical for achieving good-performance on novel datasets. We test this hypothesis on the 2 challenging tasks of object detection(PASCAL) and scene classification (SUN-397) by comparing the performance of a fully finetuned network with a network finetuned by only updating weights in the fully-connected (fc) layers. Our results are summarized in table 2.2.

Table 2.3: Evaluation of effect finetuning towards the task of object detection. (l5, l6, l7: layers 5, 6 and 7 of Alex Net)

layer	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
l5	51.9	61.1	36.8	28.4	23.7	52.3	60.8	48.4	24.9	47.1	47.5	42.1	55.6	58.7	42.5	24.5	46.9	39.3	52.0	55.4	45.0
l5-ft	57.8	63.9	38.8	28.0	29.0	54.8	66.9	51.3	30.5	52.1	45.2	43.2	57.3	58.8	46.0	27.2	51.2	39.3	53.3	56.6	47.6
l6-ft	63.5	66.3	48.7	38.1	30.6	61.4	70.9	60.3	34.8	57.8	47.6	53.6	59.8	63.5	52.5	29.8	54.6	48.2	58.5	62.2	53.1
l6-fc-ft	61.4	63.9	44.2	36.2	29.0	59.9	66.0	55.3	31.1	57.6	49.5	49.4	59.4	63.7	50.8	29.5	54.1	43.2	57.4	58.8	51.0
l7	57.6	57.2	41.4	31.2	25.6	52.4	58.8	50.9	25.2	50.4	42.7	47.1	52.2	55.6	44.5	23.9	48.0	38.1	51.5	56.6	45.5
l7-ft	64.3	69.6	50.1	41.8	32.0	62.6	71.0	60.6	32.8	58.5	46.4	56.0	60.0	66.9	54.2	31.5	52.7	48.8	57.7	64.7	54.1
l7-fc-ft	62.9	65.2	47.5	39.0	30.3	63.1	68.4	59.7	34.2	58.5	52.0	53.8	60.7	65.3	53.0	30.2	55.5	46.3	57.7	62.2	53.3

We find that indeed it is the case that the final performance in the detection setup only drops by 0.8 points and by 0.6 points for scene-classification. In our experiments we also noted accuracy of image classification on PASCAL is almost untouched by finetuning. This is suggestive of the fact finetuning is a task specific operation and finetuning for detection does not necessarily leads to an increase in classification performance, even though the classes and images are shared across PASCAL classification and detection challenges.

A detailed layer-wise analysis of detection performance for all PASCAL classes and the 3 network configurations is presented in table 2.3. Notice that for both the finetuned networks there is big jump in the performance while going from layer 5 to 6 and a rather small jump from layer 6 to 7. For Alex-Net, the performance is virtually the same for layers 5 and 7. It is also notable, that although the performance of the FT-net is better by 2.6 points at layer 5, but FC-FT net is able to compensate for this and the performance difference between layer 7 of FC-FT and FT nets are very close.

2.4 Discussion

Since, the change in performance between the FT and FC-FT networks is small, it suggests that layers 1-5 are contain generic features. Although, performance can be improved slightly by finetuning all layers as opposed to only finetuning the fully connected layers - finetuning the full network may not be practical for all applications. Also, finetuning only the last 2 layers leads to a 2x speedup in training time using publically available software [**?**] for training.

It should also be noted that the proposed entropy metric captures how the performance of each unit changes individually. The MCAuE index is similar to the average entropy of the filters in a layer. This metrics does-not account for interaction between filters. Thus, it is possible that filters individually donot become discriminative but groups of filters become discriminative. The purpose of using entropy as a measure of discrimination was purely used to develop an intuition. The validity of the intuition comes from the result that FT and FC-FT networks have similar performance.

Finetuning seems to be task-specific. For example, finetuning the network for the task of detection has little effect on classification (Our FC network for PASCAL VOC-2007 detection does as well on classification as the Alex-Net). Further, although FT and FC-FT networks lead to similar performance - they may have different error modes. This is

Table 2.4: mAP for gt-bbox classification(FT: Fine-Tuned,A-Net: Alex-Net). Note, that FT network's performance when using features from layers 4-7 is better that network without finetuning. Please see the text for an elaborate discussion.

Layer	A-Net	FT	Layer	A-Net	FT	Layer	A-Net	FT	Layer	A-Net	FT	FC-FT
pool-1	43.2	43.2	relu-3	73.3	73.7	pool-5	79.1	82.2	relu-6	83.4	85.4	81.96
pool-2	67.1	67.7	relu-4	75.5	77.8	-	-	-	relu-7	84.0	87.1	83.44

suggested by the fact that, after full finetuning the performance of layer 5 actually improves - but it is compensated for at the layer 7 when comparing the difference between FT and FC-FT networks. For the purpose of clarity, when we say features are generic we do not mean that they will not change at all during finetuning for a particular task. What we mean is that the performance of a system using these features and finetuning only the top 2 layers will be very close to finetuning all the layers -i.e. these features can be employed by a classifier (like the neural net on top of layer 5) to achieve good performance for different tasks.

It is interesting to observe the results of a network finetuned for detection on classification of ground truth bounding boxes (see table 2.4). Note, that even though the FC-FT network does as well the FT network on detection, on classifying ground-truth bounding boxes it is slightly worse than the Alex-Net itself. The FT network on the other hand does significantly better on classifying ground truth boxes. This illustrates that FT and FC-FT network end up learning different representations and fine-tuning may only help increase performance of specific task (detection in this case) and not other related tasks (like GT-BBox classification). One possible explanation for this discrepancy in performance between detection and GT-BBOX task is that for good detection, the network additionally needs to learn the concept of background which is not required foe classifying bounding-boxes.

Note, that features in layers 1-5 being generic suggests that the units in these layers maybe form a distributed code rather than forming class-specific units. We will discuss this point in great detail in sec 3.1.

Chapter 3

Are there Grand-Mother Cells in CNN's?

3.1 Introduction

How information is represented in deep architectures such as convolutional neural networks is an open question. We know that the first layer ends up learning gabor like edge detectors and that units in the final layers are very class specific. However, we are far from understanding the representations learned in the middle layers. Aprioiri it is unclear whether discriminative information about a certain class is encoded in distribution of a group of filter activations or whether there are specific units tuned to specific classes.

Some recent work such as [24], [25] have addressed this question by coming up with visualization techniques to understand tuning of various units. [24] present a deconvolution strategy for back-projecting into the image the regions which maximally activate a certain filter. [25], on the other hand pose filter-tuning in an optimization framework and estimate optimal stimuli for a given filter. [9] train a massive non-convolutional network and argue about the presence of cat and people specific filters in their network. Although visualizations are helpful, we feel they do not convey the full story. They are subjective and it is unclear what conclusions one might draw. In particular, visualizing the tuning of a few filters tells us very little about what the other filters might be doing. To best of our knowledge there is little work which tries to find an objective answer to this question.

Originally, in the neuroscience literature Grand-Mother Cells were described as "hypothetical" neurons in the brain which only respond to very specific visual stimulus [26, 12]. Some recent work on visualization techniques [9] has attempted to identify presence of such units in multi-layer networks by estimating the optimal stimulus for a single unit in the network. Mathematically, this idea can be expressed as,

$$\text{argmax}_s Prob(\text{Filter Activity}|s) \tag{3.1}$$

where s is the stimulus. Since, multi-layer networks are prone to local minima any work attempting at identifying the optimal stimuli must report results for multiple restarts of the optimization with different initial values. Previous, work such as [9, 25] only show one optimal s for a unit. From such visualizations, it is hard to interpret the tuning of a unit.

Instead of finding the optimal stimuli, one may instead be interested in finding which object class maximally activates a particular unit. One way of estimating this is by computing the class cl which maximises,

$$\text{argmax}_{cl} Prob(\text{Filter Activity} \geq th | cl) \tag{3.2}$$

for a high threshold (th). This metric, just like estimating the optimal stimuli is an incomplete way for determining how selective a certain filter is. A hypothetical filter which has a high activation for all classes will have high probabilities at high thresholds for all the classes.

If our goal is to evaluate whether a certain filter is very selective (aka Grand-Mother Cell as proposed in [26]), then it translates into finding filters with high precision, i.e. finding units which have a high value of,

$$Prob(cl | \text{Filter Activity} \geq th) \tag{3.3}$$

Although, presence/absence of Grand-Mother is a scientifically interesting question to pursue - for recognition, only filters with high precision are not sufficient. We also require high recall. Consequently, we divide our analysis into three parts. In sec 3.2 we try to address if there exist filters with very high precision (aka Grand-Mother cells), in sec 3.3 we explore are there units with high mAP and in sec 3.4 we answer how many filters are required to discriminate a class.

3.2 Are there Grand-Mother like units in layer 5?

The precision for each unit in layer 5 is independently for computed for every class in the PASCAL VOC 2007 challenge. The method of computation is analogous to entropy computation described in 2.2 with the difference being that instead of calculating entropy we compute precision as defined by 3.3. To account for a range of thresholds, the selectivity of a filter is described as the area under the precision curve. We use ground truth bounding boxes from PASCAL VOC-2007 test challenge for this computation.

The precision curves for top 5 filters for all the 20 PASCAL classes are shown in fig 3-1. It is clear, that for some classes such as persons and bicycles there are indeed some very high precision filters, but for a lot of classes like sofa and horses no such filters exist.

3.3 How discriminative are individual units in layer 5?

We explored the discriminative ability of each individual units based on their mAP on classifying ground-truth bounding boxes. Figure 3-2 plots precision-recall curves of 5 filters with highest AP for each of the PASCAL classes. This figure indicates that even at modest recalls, for most of the classes precision degrades quite fast. Its only for a few classes such as Person, Bicycle, Cars that we have filters which have relatively high AP values. It should be noted that fine-tuning does makes filters slightly more discriminative.

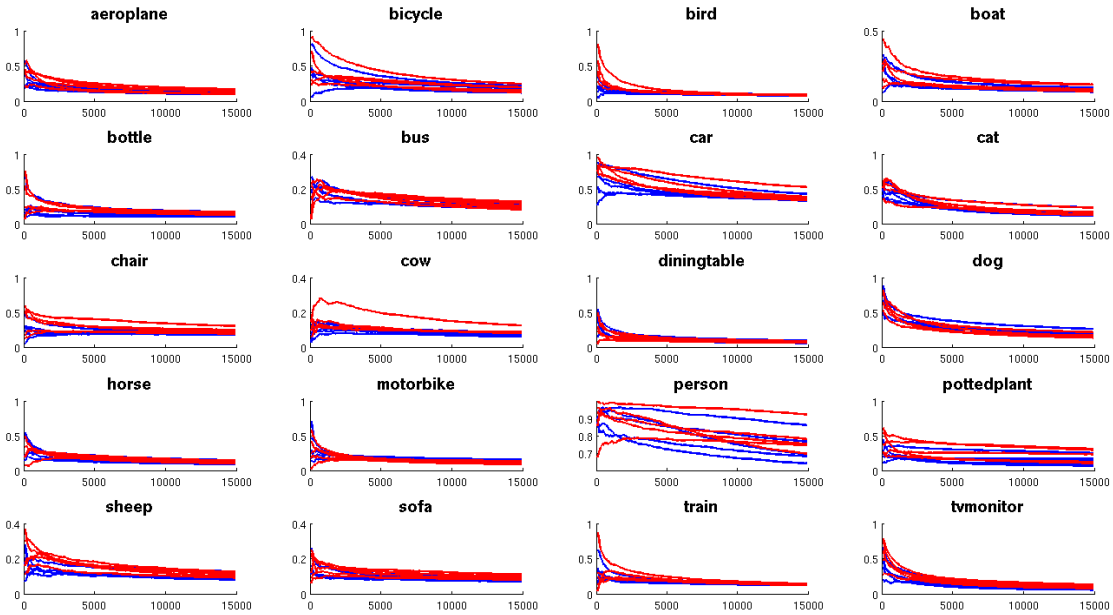

Figure 3-1: This plot shows the precision curve for the top 5 most selective filters taken from Alex-Net (Blue) and FT-Net(Red) for all PASCAL classes. Y-axis is the precision and X-axis is number of examples.

Also, generally speaking classes which have high precision filters also have filters with better discrimination power.

3.4 How many filters are required for discrimination?

In order to answer this question we train linear a svm for each class using only a subset of 256 pool-5 filters. In particular we construct subsets of size k, where k takes the values - [1,2,3,5,10,15,20,25,30,35,40,45, 50,80, 100,128,256]. A subset of size k is constructed independently for each class using a greedy selection strategy described in figure 3-3. We use the variation in performance with the number of filters needed as a metric to evaluate how many filters are needed for each class.

The results of our analysis are summarized in fig 3-4 and table 3.1. For classes such as persons, cars, cats we require a relatively few number of filters, but for most of the classes we need to look at around 30-40 filters to achieve atleast 90% of the full performance. This also indicates, that for a few classes yes, there are grand-mother kind of neurons but for a lot of classes the representation is distributed. Also, as expected the fine-tuned network requires activations of a fewer numbers of filters to achieve the same performance but this reduction in number of filters is not large.

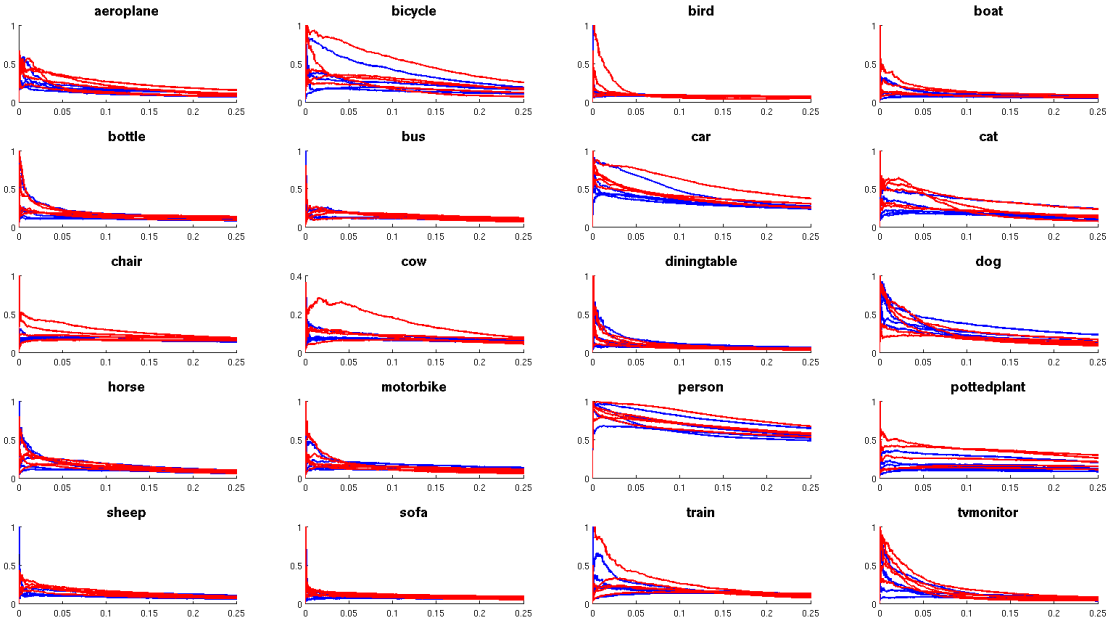

Figure 3-2: Precision-Recall Curves for 20 PASCAL Classes calculated using pool-5 filter responses on ground truth bounding boxes. Red-Curves: Fine Tuned Network, Blue-Curves: Alex-Net. The figure only displays 5 best filters for each class based on AP upto a recall threshold of 0.25. For most classes, precision drops significantly even at modest recall values.

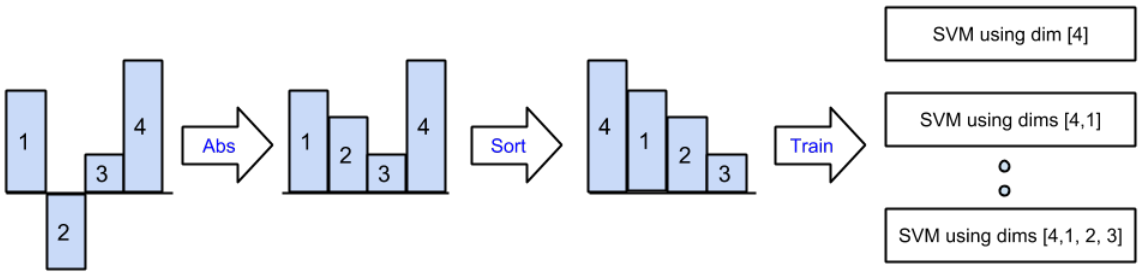

Figure 3-3: Illustration of our greedy strategy for constructing subsets of filters. For each class we first train a linear-svm using the spatial-max feature transformation described in section 5.1. Spatial-max leaves us with a 256-D vector wherein each dimension has a one to one correspondence with 256 pool-5 filters. We use the magnitude of each dimension of the learnt weight vector as a proxy for the importance of that dimension towards discriminating a given class. For the purpose of illustration we describe the procedure with a 4-D weight vector shown on the extreme left (the numbers on each bar are the "dimension"). Firstly, we take the absolute value for each dimension and then sort the dimensions based on this value. Then, we chose the top k filters/dimensions from this ranked list to construct a subset of size k.

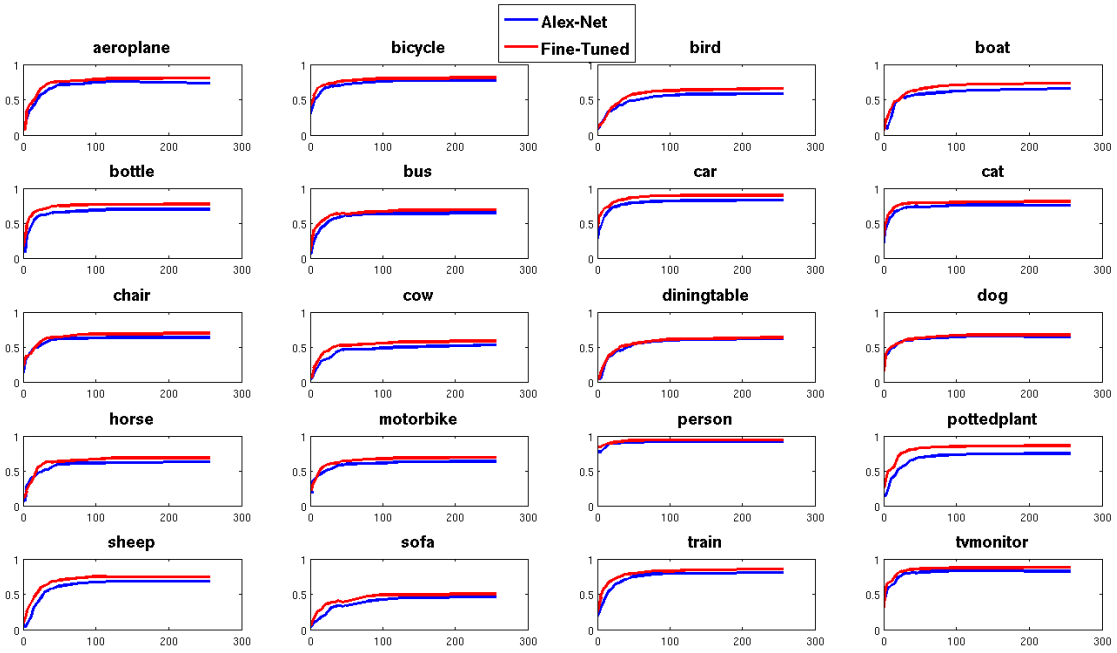

Figure 3-4: Analysis of how many filters are required to classify ground truth bounding boxes for 20 categories taken from PASCAL-2007 detection challenge. The y-axis in each of plot represents classification accuracy measured as mean-ap where as x-axis stand for the number of filters.)

Table 3.1: Number of filters required to achieve 50% ,90% of the full performance for PASCAL classes using Alex-Net(AN) and the Fine-Tuned network(FT)

Net	AP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
AN	50	15	3	15	15	10	10	3	2	5	15	15	2	10	3	1	10	20	25	10	2
FT	50	10	1	20	15	5	5	2	2	3	10	15	3	15	10	1	5	15	15	5	2
AN	90	40	35	80	80	35	40	30	20	35	100	80	30	45	40	15	45	50	100	45	25
FT	90	35	30	80	80	30	35	25	20	35	50	80	35	30	40	10	35	40	80	40	20

3.5 Discussion

We conclude, that for discriminating some classes only a few units suffice whereas for other classes quite a lot of them are required. Our results also indicate that mid-level ConvNet representations are different from widely used SIFT-based BOW models (distributed) and discriminative mid-level patches (more like grandmother cells. It seems that ConvNets are able to balance the best of both worlds. It is also interesting to note that “Person" is not a class label in imagenet, yet there are units very selective for person even in Alex-Net. This indicates, that network is learning from the the concept of people from co-occurence of “Person" class with other Imagenet categories in-spite of their being no explicit information about their presence.

We also visualized the degree to which different classes share filters for reaching a

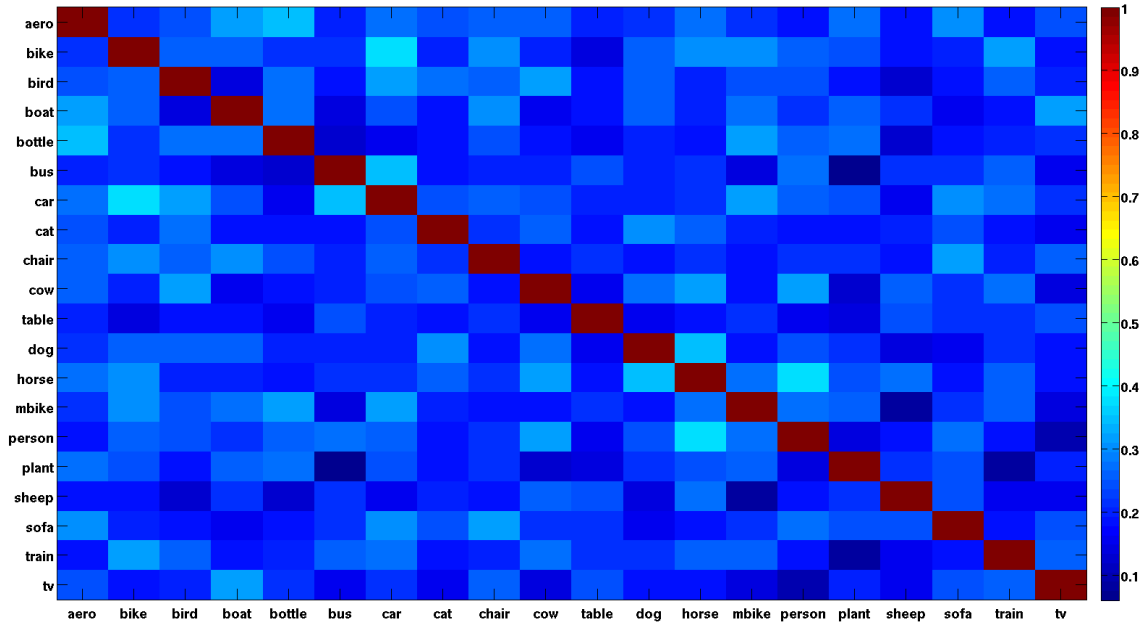

Figure 3-5: This plot depicts the degree of overlap among the top-50 filters in layer 5 of finetuned network used for classifying each category of the PASCAL 2007 challenge. Entry (i,j) of the matrix is the fraction of filters of the i^{th} class which are common with j^{th} class. This plot indicates, that there is very little overlap between filters used for different classes.

reasonable performance level. The results are shown in 3-5. This plot indicates, that there is little overlap between filters employed by different classes. This is not surprising, as the code in layer-5 is fairly distributed. We chose to consider top 50 filters, as for most classes around 50 filters lead to 90% of the performance achieved by considering all the filters.

The ”extent" to which the code is distributed is likely to be a function of number of filters. If there are a few filters we will expect the code to be more distributed whereas if there are a large number of filters we expect to find more Grandmother kind of cells. The other important tradeoffs to consider are accuracy and training time as a function of number of filters in each layer. We as a community have only had a chance to experiment with a few network architectures out of the exponentially large number of possibilities. Although, it is beyond the scope of the current work determining the optimal number of filters in each layer is an open important question which needs to be addressed!

Chapter 4

How do the different layers of a CNN train over time?

4.1 Introduction

Convolutional neural networks take a long time to train. For achieving state of art accuracy on the imagenet challenge these networks are often trained on high-end GPUs for more than 7 days. Even our implementation of fine-tuning following the approach proposed in [1] takes more than 12 hours on a Nvidia Tesla-K40. Any speeds up in training will allow for a rich exploration of network architectures and parameters which is currently not possible.

4.2 Analysis

As a first step towards addressing this problem, we looked at the evolution of training loss and validation accuracy as the training progresses (fig 4-1.) The top-1 accuracy on the imagenet validation set at 15K iterations is at 29.5 % and 38.13% at 50K iterations (compared to 57.4 % at 310K iterations). The training loss rapidly increases initially and then there is a slow sluggish decay except at the stage where learning rate is decimated by a factor of 10 at 100K iterations.

The first question we ask is - Is there is an insightful interpretation of the fast initial drop in training loss? Towards this end, we visualized layer 1 filters at different time instances. Surprisingly, we found that within 15K iterations these filters look pretty similar to what they would be by the end of the training (See fig 4-1). This naturally leads us to ask the following questions:

- How fast do different layers train? Does the training proceed in a layerwise manner?
- Given that discriminative pre-training is helpful, is it the case that there exists a critical point after which learning on imagenet is not helpful for generalizing to other datasets/tasks? A “yes" answer to this question would imply that we can indeed speed-up finetuning.

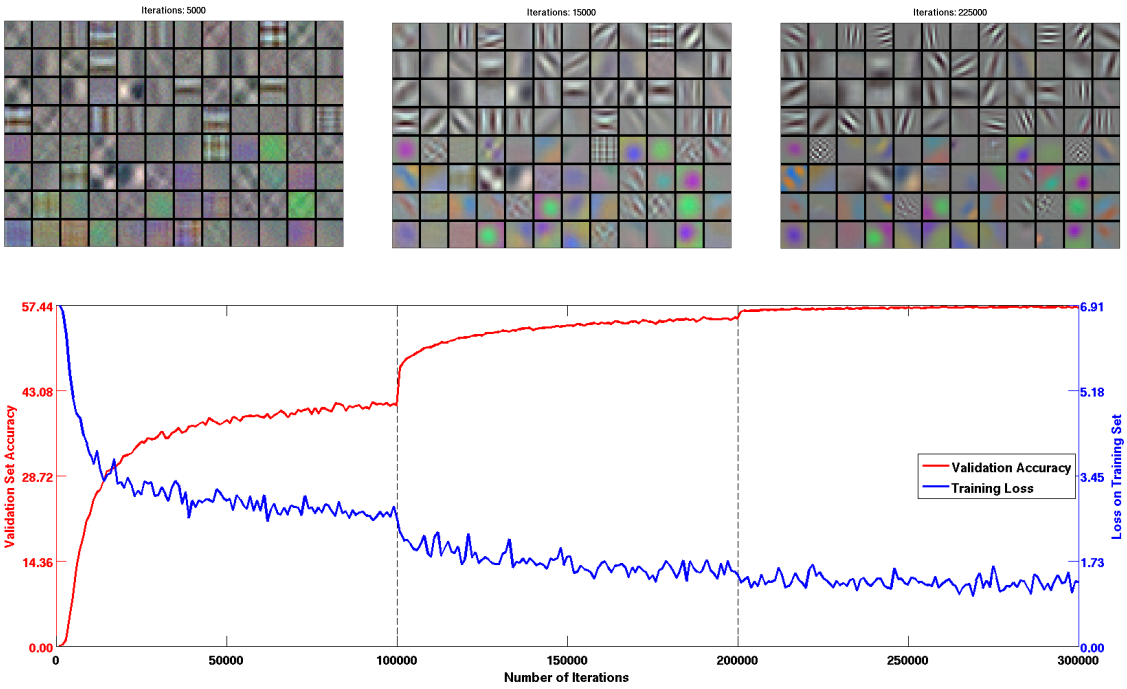

Figure 4-1: The first row shows conv-1 filters at 5K, 15K and 225K training iteration. The second row shows the evolution of training loss and top-1 accuracy on imagenet ilsvrc-2012 validation set as a function of number of iterations.

For developing a better understanding, we evaluated performance of a linear SVM classifier learned on features extracted from individual layers on Pascal 2007 classification challenge. The results are summarized in table 4.1. It is quite surprising to note that by 15K iterations all layers are within 80% and at 45K iterations within 90% of there final performance. This strongly indicates that a great portion of training required for generalization happens quite early on.

Motivated by these observations, we trained a 50-50 network (50K iterations on imagenet and finetuned for 50K iterations using the procedure described in sec. 1.2) and evaluated its performance on the Pascal 2007 detection challenge (see table 4.2 for results). Consistent with our earlier results we find that this network achieves a surprising performance of 50.5 mean AP points compared to 54.1 achieved by pre-training for 310K iterations.

Further, if we train for 100K iteration on imagenet, (100-50 network) the performance goes up to 52.6 mAP points. Due to dataset bias, one might expect that too much pre-training can hurt generalization performance. Instead, we find that although pre-training for 50K iterations gets us close to the final performance, pre-training more improves performance. This is surprising! It says that fitting better to ImageNet allows lower generalization error when moving to other datasets.

Table 4.1: Variation in classification accuracy (mean-AP) on PASCAL VOC 2007 challenge using features extracted from different layers of Alex-Net as a function of number of iterations. The black dotted lines indicate the iterations at which the learning rate is reduced by a factor of 10. Notice that training loss reduces very slowly after the first 100K iterations.

Layer	5K	15K	25K	35K	45K	95K	105K	310K
pool-1	23.0	24.3	24.4	24.5	24.6	24.8	24.7	25.1
pool-2	33.7	40.4	40.9	41.8	42.0	43.2	44.0	45.0
conv-3	34.2	46.8	47.0	48.2	48.5	49.4	51.6	50.1
conv-4	33.5	49.0	48.7	50.2	50.6	51.6	54.1	54.2
pool-5	33.0	53.4	55.0	56.8	57.4	59.2	63.5	65.6
relu-6	34.2	59.7	62.6	62.7	64.1	65.6	69.3	70.6
relu-7	30.9	61.3	64.1	65.1	65.8	67.8	71.8	73.2

Table 4.2: Performance of 50-50 network for detection on pascal-voc-2007 challenge. (l5 is pool-5 and l7 is relu-7)

Feature	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
l5(50-50)	52.5	59.0	35.9	28.6	21.5	55.6	62.2	48.7	23.8	46.9	39.0	41.0	53.1	56.5	42.8	24.2	45.1	37.8	51.4	56.0	44.1
l5 (full)	57.8	63.9	38.8	28.0	29.0	54.8	66.9	51.3	30.5	52.1	45.2	43.2	57.3	58.8	46.0	27.2	51.2	39.3	53.3	56.6	47.6
l7(50-50)	60.8	65.3	47.0	39.9	26.8	60.1	66.2	55.8	30.3	51.9	44.7	51.9	56.6	61.3	50.6	26.4	54.2	44.9	54.3	61.2	50.5
l7(full)	64.3	69.6	50.1	41.8	32.0	62.6	71.0	60.6	32.8	58.5	46.4	56.0	60.0	66.9	54.2	31.5	52.7	48.8	57.7	64.7	54.1

4.3 Discussion

The above analysis indicates that a lot of learning happens in the first few epochs of going through the training examples. In hindsight, this results is not very surprising and is something which can be expected due to the convergence properties of stochastic gradient descent. At the same time, it is interesting to observe the similarity between training progress of ConvNets and latent-svms during hard-negative mining. Maybe some ideas from this literature can be used to speedup ConvNet training. Although, we do-not propose a concrete method, we hope that this exposition will act as an invitation for researchers to actively explore such ideas.

The observation which is striking is that for the task of transfer learning - i.e. using a pretrained network for recognition tasks on a different dataset - more pretraining does not hurt, but counter-intuitively leads to better performance.

A small note on implementation: All finetuning was done with an initial learning rate of 0.001 and decreasing the learning rate by a factor of 10 after every 20K iterations. Using lower initial learning rates lead to slightly worse performance.

Chapter 5

Is the information in the location or in the magnitude of filter activation?

Recent CNN based solutions for classification, detection and localization ([20], [4],[1]) have more or less treated CNN's as a black-box feature extractor. Such work has been limited to using either linear svm's, regression or a soft-max on top of vectorized layer features in contrast to the rich history of using structured multi-scale models in computer vision ([21], [22],[23]). In order to move away from black-box style of using CNN's and fully exploiting the potential which this feature hierarchy provides us with - it is instructive to tease apart what aspects of filter activation are important and what aspects can be ignored.

In this work we focus on 2 particular questions:

- How important is the location where a filter fires?
- How much information is contained in the magnitude of filter activations?

In order to answer these questions we constructed a series of ablation experiments (see fig 5-1) and we use the difference in performance from the un-ablated feature representation (vectorized layer features) as a measure of how important each of these aspects is. We study these questions under the settings of image classification (see table 5.1) and object detection using pool-5 features (see table5.2).

5.1 How important is where a filter activates?

The two transformations which devoid the feature vector of any information about the location of where the filter activates (namely sp-max and sp-shuffle) seem to have a little effect on classification performance using pool-5 features. In contrast, for detection taking a spatial max (table 5.2) is disastrous and the performance drops by around 50%.

5.2 How important is the magnitude of activation ?

Binarization is the ablation which deprives features from their magnitude of activation. For classification, this results in a drop in performance greater than that due to sp-max. This

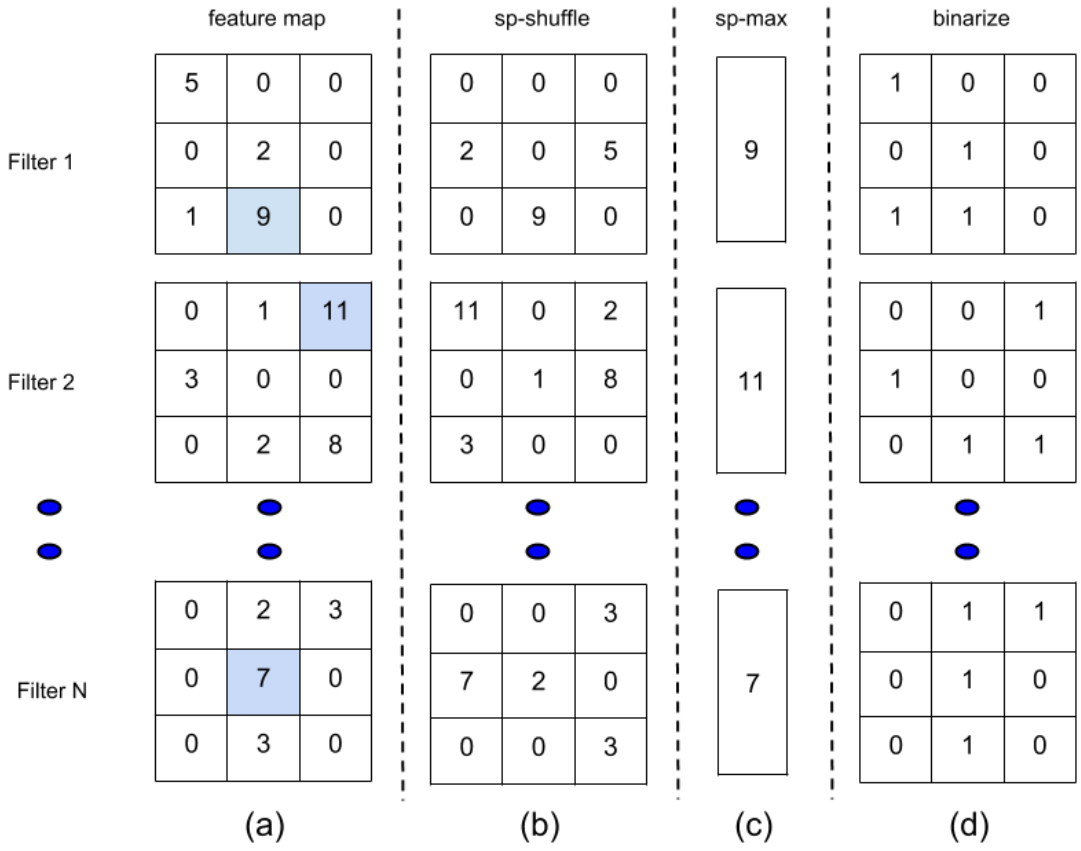

Figure 5-1: Description of feature ablations: For the purpose of illustration, consider each 3*3 block in (a) as the spatial activation of individual filters. Each block in (a) is a separate filter from the same layer. Ablation 1: *spatial-shuffle-(b)* (sp-shuffle), involves applying an independent spatial random permutation to each feature map (different images see different permutations). Ablation 2: *spatial max*-(c) (sp-max), select the max activation value in each feature map. Ablation 3: *binarization*-(d) (bin). Ablation 4: *sp-max-bin* - binarized sp-max.

Table 5.1: Feature ablation analysis on PASCAL Image Classification (see fig 5-1)

layer	unroll	sp-shuffle	sp-max	sp-max-bin	bin
pool-1	25.1	15.0	19.5	9.0	25.4
pool-2	44.9	33.5	40.2	-	43.0
conv-3	50.1	40.4	54.2	-	47.0
conv-4	54.2	45.3	57.0	-	51.3
pool-5	65.6	59.6	62.6	27.2	60.5
relu-6	70.6	-	-	-	71.4
relu-7	73.6	-	-	-	74.1

Table 5.2: Effect of various feature ablations on object detection using R-CNN[1].

Feature	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
sp-max	35.0	38.7	17.3	16.9	13.9	38.4	45.6	29.2	11.0	20.2	21.0	23.5	27.2	37.0	20.5	7.0	30.3	13.4	28.3	32.9	25.4
bin	57.9	61.3	32.6	24.7	27.5	55.0	64.7	49.8	25.3	47.4	44.5	40.3	54.6	56.4	43.6	27.1	48.4	41.6	54.3	57.6	45.7
pool-5	57.8	63.9	38.8	28.0	29.0	54.8	66.9	51.3	30.5	52.1	45.2	43.2	57.3	58.8	46.0	27.2	51.2	39.3	53.3	56.6	47.6

suggests that it is more important to retain the magnitude than the location for classification. Rather, surprisingly binarization has little effect on detection performance.

5.3 Discussion

Our experiments indicate that if the goal is classification, spatial location of layer 5 units can be ignored without much loss in performance. We also find that different ablations effect different layers in different ways. For example, where as spatial-max is brutal for layer-1, it does not really effect classification if layer 5 features are used. This maybe partially explained by the fact that receptive field of layer 5 units are quite large and have substantial overlaps as compared to much smaller receptive fields of layer 1. Although, it is noteworthy that for detection preserving the location turns out to be critical. This may be due to the fact, that detection requires precise localisation which is not the case with classification.

It is also interesting to observe as we increase the number of classes for the sp-Max ablation. For this, we computed the accuracies on Imagenet Validation set (top-1) using pool5 and sp-Max ablation. The experiment involved freezing layers 1-5 and training a 1000 unit classification layer on top of pool5 and pool5 transformed under the spMax ablation. The performance of pool-5 after 75K iterations was at 43.2, whereas that for the spMax was at 39.4.

Binarization has little effect on layer-1 features it results into a greater drop of performance in layer-5 and negligible effect on the performance due to layer 7. A lot of work in image retrieval focuses on finding sparse binary features - our results indicate that one can get these 'for free' from ConvNet features. Further, using binary features reduces feature cache size, which can turn out to be very handy as we scale to larger datasets.

Lastly, for classification although sp-max and binarization by themselves are not detrimental - their combination (sp-max-bin) is catastrophic to performance.

Chapter 6

The gold which did not glitter!

It is a common practice in research to report results which either establish a certain fact, disproof an existing belief or achieve state of art performance. However, all researchers undergo the process of experimentation and somethings work and others don't. As a matter of fact, most things don't work. I believe there is merit in reporting intuitions and methods which didnot work out as expected. Hopefully, some reader may be able to shed some new light or re-formulate the problem in novel ways which may lead to success! Although, I could write pages and pages in this section - I will talk about some of the promising directions - that just didn't stand the test of experimentation!

6.1 SVM-Nets

Until recently, unsupervised learning had been a popular trend in deep learning. However, the work of [2] suggests that discriminative pre-training may be more helpful than unsupervised pre-training. One of the major concerns about ConvNets is their long training time and lack of an obvious way to parallelize SGD. In order to tackle these issues, we hypothesized that SVMs can be used to learn convolutional layer filters. We assumed that that entropy of individual filters in each layer gradually decreases as we move from layer 1 to layer 7. Based on this, we experimented with the following scheme:

- For a layer N, use K-Means on patches extracted from the outputs of layer N-1 to identify potential filters.
- Train a SVM, by considering all patches belonging to a cluster center as positive and rest as negative.
- Now, in order to make each filter discriminative, compute the entropy of all examples which are predicted be positive by the SVM classifier. Identify the classes which have least average positive score and reassign all patches belonging to these classes as negatives. Re-train the SVM. This step, will lead to decrease in entropy of each filter.
- Keep stacking layers following this procedure.

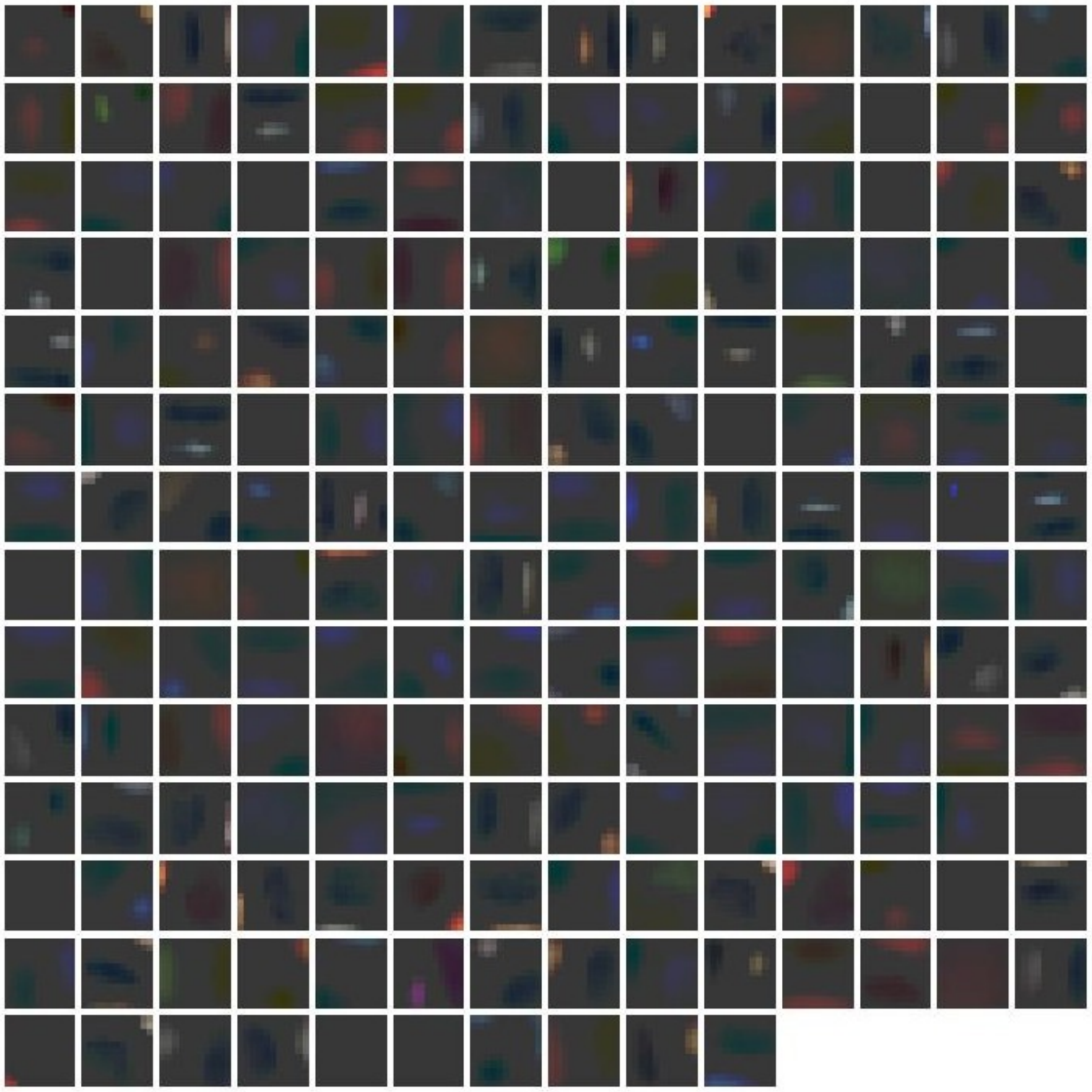

Figure 6-1: SVM-Nets layer 1 filters. Note, they are qualitatively different from the filters learnt by layer 1 of ConvNets.

Layer 1 of SVM-Net learned edge-like filters (fig 6-1) as expected and achieved slightly better performance of 27.1 mAP v/s 25.1 mAP achieved by layer 1 of ConvNet on the task of image classification on PASCAL. But, the performance of layer 2 turned out to be significantly worse. One possible reason for failure of this approach is that our initial intuition about entropy of filters decreasing gradually is actually not true. (See sec 2.1).

6.2 Non-Linear Classifiers to Replace FC-Layers

Neural Networks comprise one family of non-linear classifiers. There are other families such as decision trees, kernel methods etc. Currently, it is unclear if other methods can

achieve the same performance as a multi-layer neural network. This is an interesting and an important scientific question. Futher, [4] report that computations in FC-layers account for more than 50% of the computation time of the full network. Consequently, if one can find a fast non-linear classifier using pool-5 features to achieve comparable performance - the training time can be reduced upto a factor of 2.

In an attempt to answer this, we trained SVMs with different kernel maps and decision trees with different weak features extracted from Pool-5. However, the performance of our systems was no where even close to that of the FC layers.

One interesting way to use decision forests in this context is as following: One can look at Pool-5 features in a way [16] looked at digits. In sec 5.2 we showed that binarization only leads to a small decrease in classification performance. Thus, each of the binarized feature channels can be viewed as an answer to a yes/no question indicating if a certain "token" (as defined in [16]) is present at a certain location. One, can extend the analogy and build Amit-Geman style trees to capture spatial relations between different filters of pool-5. Unfortunately, even this approach barely made it to the performance of a linear SVM applied on top of pool-5 features.

6.3 Generalized Distance Transform Pooling

The max-pooling operation in the architecture proposed by [2] throws the location from where the "max" value is selected. Arguably, preserving this value can help in better localization/recognition of objects.

Let us assume, that pool-5 features are part-like in their selectivity. Now, to recognize an object - recognizing the correct spatial configuration of parts is critical. In ConvNets this intuition can be realized in the following way: At every pooling location we want the pooling unit for a certain filter to expects filter activity only at a certain (preferred) location. Since, we also want to allow for small deformations around the preferred location - we will add a penalty proportional to the distance from such a location. Mathematically, this can be expressed as:

$$max_{y \in R} g(y) = f(y) - ay^2 - by - c \tag{6.1}$$

where $f(y)$ is the filter score at location y, R defines the spatial extent of the pooling region and a, b, c define the deformation penalty. The roots of the quadratic equation $ay^2+by+c = 0$ defines the preferred location. Note, this is similar to the penalty used in [21].

In our experiments, we allowed for both scenarios: (1) A single preferred location for a certain filter across all pooling regions. (2) Different preferred locations for a certain filter across different pooling regions. Notice, that layer 4 has the spatial extent of 13×13 while the output of pool-5 is 6×6 - i.e. each filter-map is pooled at many locations (called as pooling regions.) We found that this pooling strategy didnot improve performance either for detection or classification. Even if argmax$g(y)$ is preserved and used a feature - there is negligible increase in performance.

Chapter 7

Conclusions

In this work we conducted an empirical analysis of properties of Multi-Layer Neural Networks with a special focus on object recognition. The salient points of the report can be summarized as following:

- We proposed the metric of MCAuE to measure selectivity of a layer. Next, we showed that fine-tuning only fully connected layers leads to similar performance as fine-tuning all the layers. This establishes that layers 1-5 are generic. This leads to 2x speedup during fine-tuning on a new dataset.
- Binarization of features only slightly degrades performance both for classification and detection. A lot of work in image retrieval focuses on finding sparse binary features - our results indicate that one can get these 'for free' from ConvNet features. Further, this vastly reduces feature cache size, which is handy as we scale to larger datasets.
- Spatial-max is catastrophic for detection, while spatial locations can be ignored for classification. The combination of Spatial-Max and Binarization taken together degrades performance significantly even for classification
- For a few classes there are grandmother cells, but for many classes the representation is distributed. Recently, many visualization papers have focussed on finding grandmother cells. Such papers provide interesting insights, but do not convey the full story. We establish an objective methodology for studying the question of how distributed are feature representations. Our results indicate that mid-level ConvNet representations are different from widely used SIFT-based BOW models (distributed) and discriminative mid-level patches (more like grandmother cells)
- On Transfer Learning: Due to dataset bias, one might expect that too much pre-training can hurt generalization performance. Instead, we find that although pre-training for 50K iterations gets us close to the final performance, pre-training more improves performance. This is surprising! It says that fitting better to ImageNet allows lower generalization error when moving to other datasets.
- Bulk of the ConvNet training takes place in the first few epochs. This observation can hopefully be used by future work to speed up the training process.

Bibliography

[1] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524 (2013)

[2] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012)

[3] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09. (2009)

[4] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531 (2013)

[5] Wang, X., Yang, M., Zhu, S., Lin, Y.: Regionlets for generic object detection. In: The IEEE International Conference on Computer Vision (ICCV). (December 2013)

[6] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: In CVPR. (2005) 886–893

[7] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60** (2004) 91–110

[8] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation **1**(4) (1989)

[9] Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., Ng, A.: Building high-level features using large scale unsupervised learning. In: International Conference in Machine Learning. (2012)

[10] Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. **11** (March 2010) 625–660

[11] Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7) (July 2006) 1527–1554

[12] Quiroga, R.Q., Reddy, L., Kreiman, G., Koch, C., Fried, I.: Invariant visual representation by single neurons in the human brain. Nature **435**(7045) (2005) 1102–7

[13] Jia, Y.: Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/ (2013)

[14] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. IJCV **88**(2) (2010)

[15] Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. In: CVPR, IEEE (2010) 3485–3492

[16] Geman, D., Amit, Y., Wilder, K.: Joint induction of shape features and tree classifiers. (1997)

[17] Breiman, L.: Random forests. Mach. Learn. **45**(1) (October 2001) 5–32

[18] Juneja, M., Vedaldi, A., Jawahar, C.V., Zisserman, A.: Blocks that shout: Distinctive parts for scene classification. In: Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2013)

[19] Singh, S., Gupta, A., Efros, A.A.: Unsupervised discovery of mid-level discriminative patches. In: European Conference on Computer Vision. (2012)

[20] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. CoRR **abs/1312.6229** (2013)

[21] Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. TPAMI **32**(9) (2010)

[22] Yang, Y., Ramanan, D.: Articulated pose estimation with flexible mixtures-of-parts. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition. CVPR '11, Washington, DC, USA, IEEE Computer Society (2011) 1385–1392

[23] Bourdev, L., Maji, S., Brox, T., Malik, J.: Detecting people using mutually consistent poselet activations. In: European Conference on Computer Vision (ECCV). (2010)

[24] Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. CoRR **abs/1311.2901** (2013)

[25] Simonyan, K., Vedaldi, A., Zisserman, A.: Learning local feature descriptors using convex optimisation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2014)

[26] Barlow, H.: Single units and sensations: A neuron doctrine for perceptual psychology? In: Perception. (1972)