# Analyzing The Performance of Multilayer Neural Networks for Object Recognition

Anonymous ECCV submission

Paper ID 1355

**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 paper 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? (d)How does training of CNN progress over time?



## 1 Introduction

The breakthrough work of [1] 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 [2]. The top-5 error rates dropped by an exceptional amount to 16.4% from 26.2 % (achieved by the second best alternative.) At that time, it was unclear if these networks would be useful for other computer vision tasks. The recent work of [3] 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 big development (R-CNN)[4], features extracted using convolutional networks were succesfully 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 2000's.

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 succesfully 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 heirarchies 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 paper is focussed on a scientific investigation of multilayer convolutional neural networks centered around answers to four questions detalied below:

For a long time proponents of multilayer networks have argued that unsupervised pre-training followed by finetuning is helpful for improving performance on discriminative tasks such as image classification [9], [10], [11]. However recent work of [3] and [4] 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? Do we really need 7 days?"*

The rest of the paper is organized as following: section 2 details the procedure used to train conv-nets. Sections 3, 4, 5 and 6 are devoted to the four questions and our analysis is concluded in section 7.

## 2 Training Procedure

### 2.1 Network-Architecture and Nomenclature

For all our experiments we closely follow the architecture proposed in [1]. 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].

### 2.2 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 [4].
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 intial learning rate of the imagenet model. This choice has been made because we donot

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 [4]. Unless otherwise stated, the finetuned network for PASCAL experiments is finetuned for 70K iterations whereas that for SUN is finetuned for 40K iterations.

### 2.3 Method of Entropy Computation

We define the entropy of a filter with respect to a given set of image-label paris 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). We vectorize this 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 we consider all the labels which have an associated score $\geq$ to this threshold score. The entropy of this set of labels 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.

## 3 What happens when a discriminatively pretrained network is finetuned?

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 [4][3] present a strong case that this methodology boosts performance. Although, unsupervised pretraining has widely studied in the multilayer network literatue [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 3.1). 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 fine-tuning 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 sec3.2). We find this results in a negligible drop in performance and allows for moderate speedups in finetuning time. Other conclusions are presented in the sec 3.3.

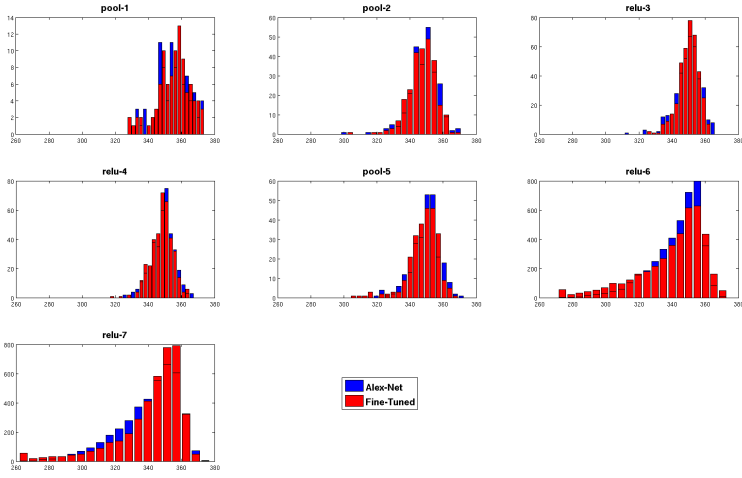


Fig. 1: Distribution of AuE for different layers in Alex-Net and FT-Net. X-axis is the entropy and the Y-axis is the number of filters. Notice that the left tail for relu 6 and 7 becomes heavier after finetuning. This indicates that finetuning makes these filters more discriminative.

### 3.1 Entropy Analysis

We compute the entropy curve of each filter (using the method described in 2.3) for all layers of Alex-Net and FT-Net using the ground-truth bounding boxes taken from the VOC-2007 test-set.

We use Area under this entropy curve (AuE) to quantify selectivity of each filter. The distribution of AuE for all filters across the seven layers of the CNN is illustrated in fig 1. Next, in order to determine the overall change in a layer's tuning we use the Cumulative AuE (C-AuE) of filters sorted in decreasing order of their individual AuE's. We normalize this metric appropriately to account for different number of filters in different layers. We call this normalized C-AuE as MC-AuE. A lower value of MC-AuE means that a layer is more selective. Fig 2 plots MC-AuE as a function of fraction of filters in each layer.

From figures 1 and 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.
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.

Table 1 summarizes the classification mean AP obtained on classifying GT-BBOX using features from different layers of a fine-tuned and a non-finetuned network. It can be seen that is indeed the case performance advantage for the finetuned network only happens for higher layers.

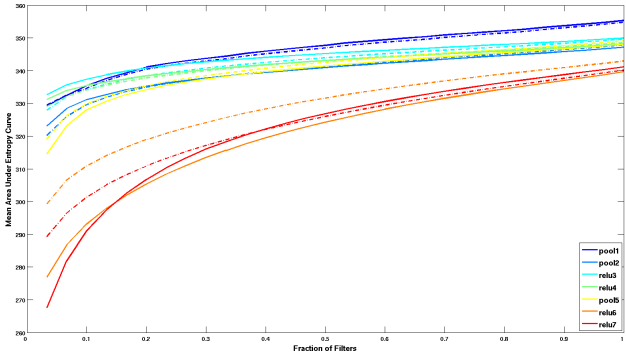


Fig. 2: Mean Cumulative AuE plotted as fraction of filters (see sec 3.1). (Dash-Dot Line :Alex-Net, Solid Line: Fine-Tuned Network).

Table 1: mAP for gt-bbox classification(FT: Fine-Tuned,A-Net: Alex-Net).

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

Table 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 | 56.1 | 55.5 |

### 3.2 Is finetuning only the fully-connected layers sufficient?

The observations made in 3.1 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.

We find that indeed it is the case that the final performace 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 doesnot necessarily leads to an increase in classification performance, even though the classes and images are shared across PASCAL classification and detection challenges.

Table 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 |

A detailed layerwise analysis of detection performance for all PASCAL classes and the 3 network configurations is presented in table 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 for FT-net is better by 2.6 points at layer 5 - the performance is virtually the same at layer 7.

### 3.3 Discussion

Since layers 1-5 change a little over the course finetuning, this suggests that these are generic features. Although, one could always improve performance by a few points by finetuning the full network - for a lot of applications this may not be practical. It is also notable to point out that the more or less generic representations learnt in layer 4 and 5 are in contrast with some of the mid-level feature learning work such as [16] [17] wherein the problem of finding good mid-level parts is often posed as a greedy search for high recall discriminative templates.

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

Recent CNN based solutions for classification, detection and localization ( [18], [3],[4]) 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 vectorzed layer features in contrast to the rich history of using structured multiscale models in computer vision ([19], [20],[21]). 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?

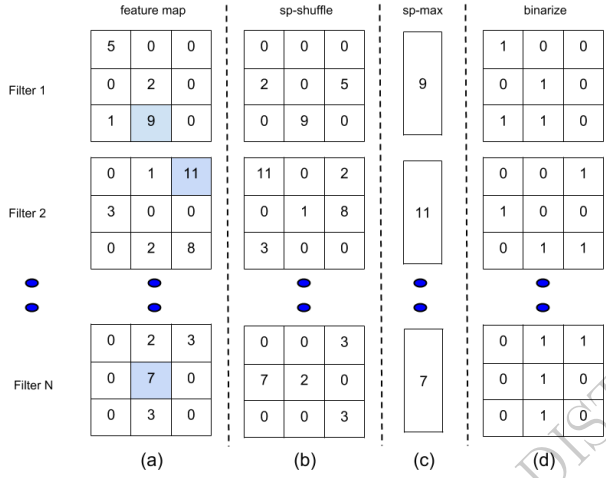


Fig. 3: Description of feature ablations: For the purpose of illustration, consider each 3*3 block in (a) as the spatial activation of individual filters. Each block in (a) is a seperate filter from the same layer. Ablation 1: *spatial-shuffle-(b)* (sp-shuffle), involves applying an independent spatial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

– 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 3) 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 4) and object detection using pool-5 features (see table5).

### 4.1 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) is disastrous and the performance drops by around 50%.

### 4.2 How important is the magnitude of activation ?

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

Table 4: Feature ablation analysis on PASCAL Image Classification (see fig 3)

| 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: Effect of various feature ablations on object detection using R-CNN[4].

| 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 |

than the location for classification. Rather, surprisingly binarization has little effect on detection performance.

### 4.3 Discussion

Our experiments indicate that if the goal is classification layer 5 features can be treated as bag of filters 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 doesnot really effect classification is layer 5 features are used. On the other hand while binarization has little effect on layer-1 features it results into a drop of performance for layer-5. This is a good thing to know when designing systems which use features from multiple layers.

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

## 5 Are there Grand-Mother Cells in CNN's? How distributed is the code?

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 [22], [23] have addressed this question by coming up with visualization techniques to understand tuning of various units. [22] present a deconvolution strategy for backprojecting into the image the regions which maximally activate a certain filter. [23], on the other hand pose tuning as an optimization problem and try to 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 donot 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.

Most of the above mentioned work can be mathematically treated either as estimating Prob(Filter Activity$\geq$ thresh | Class) or finding the optimal stimulus for a specific unit. We argue that this by itself is an incomplete metric for interpreting how selective a certain filter is. (A hypothetical filter which has the same activation for all classes will score high on this measure.) The right metric to evaluate if a certain filter is very selective (aka Grand-Mother Cell [24]) is precision defined as Prob(Class |Filter Activity $\geq$thresh)

We divide our analysis into two parts. In sec 5.1 we try to address if there exist filters with very high precision (aka Grand-Mother cells) and in sec 5.2 we answer how many filters are required to discriminate a class.

### 5.1 Are there high precision filters in layer 5?

Precision for each filter from layer 5 is computed in a way analogous to entropy computation described in 2.3. In the final step instead of computing the entropy, we compute Prob(Class | Filter Activity $\geq$ threshold). We define the selectivity of a filter as the area under the precision curve. For this computation we use ground truth bounding boxes taken from PASCAL VOC-2007 test challenge. From figure 4 it is clear that for some classes such 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.

### 5.2 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 constucted independently for each class using a greedy selection strategy described in figure 5. 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 6 and table 6. 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

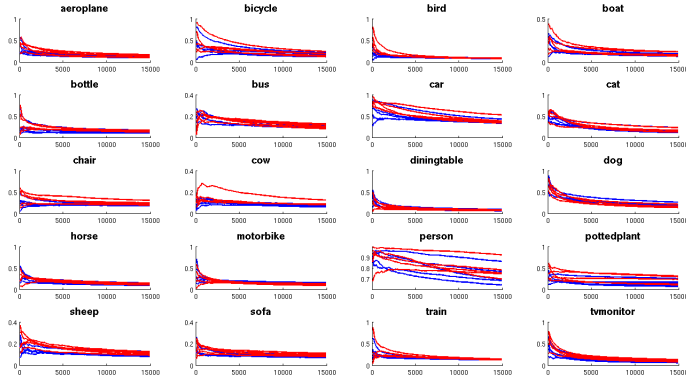


Fig. 4: This plot shows the precision curve for the top 5 most selective filters taken from Alex-Net (Blue) and FT-Net(Red) for all PASCAL classes. Y-axis is the precision and X-axis is number of examples.

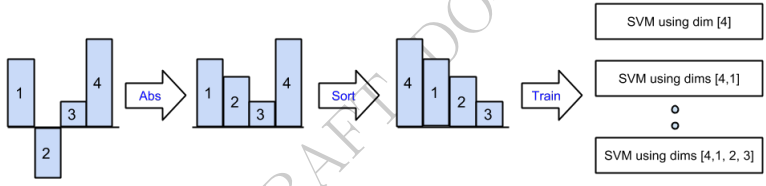


Fig. 5: Illustration of our greedy strategy for constructing subsets of filters. For each class we first train a linear-svm using the spatial-max feature transformation described in section 4.1. Spatial-max leaves us with a 256-D vector wherein each dimension has a one to one correspondence with 256 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.

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.

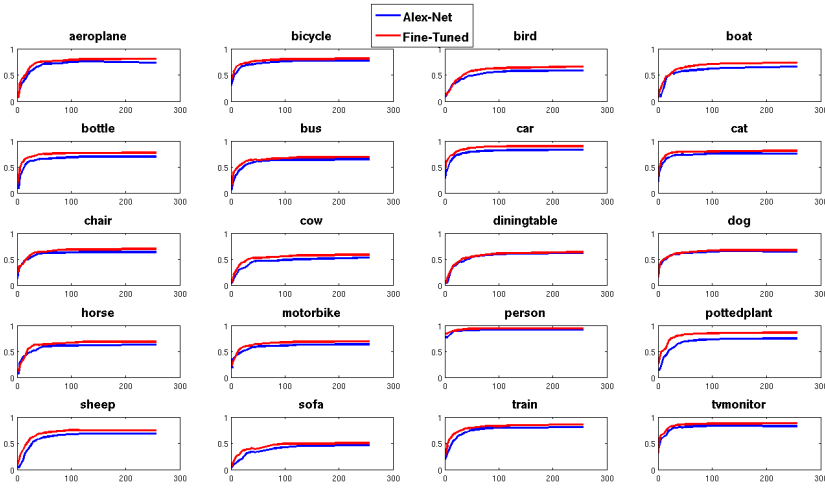


Fig. 6: Analysis of how many filters are required to classify ground truth bounding boxes for 20 categories taken from PASCAL-2007 detection challenge. The y-axis in each of plot represents classification accuracy measured as mean-ap where as x-axis stand for the number of filters.)

Table 6: 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 |

### 5.3 Discussion

Although in our analysis we find that for discriminating some classes only a few units suffice whereas for others quite a lot of them are required. 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!

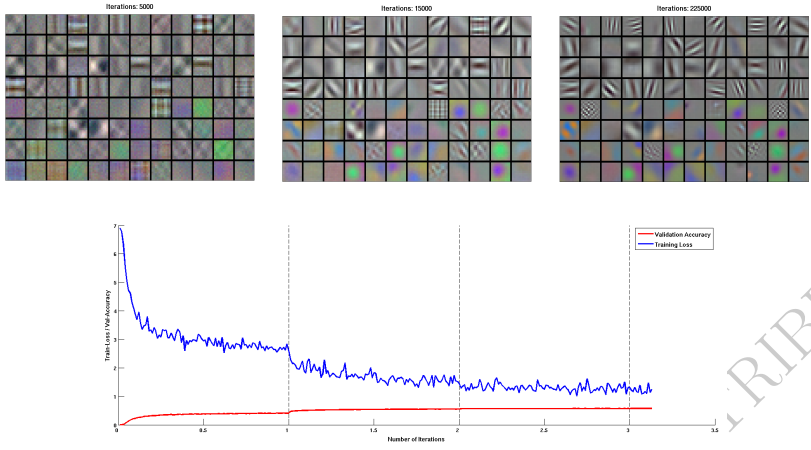


Fig. 7: The first row shows conv-1 filters at 5K, 15K and 225K training iteration. The second row shows the evolution of training loss and top-1 accuracy on imagenet ilsvrc-2012 validation set as a function of number of iterations.

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

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 [4] 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.

As a first step towards addressing this problem, we looked at the evolution of training loss and validation accuracy as the training progresses (fig 7.) 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 intepretation of the fast intial 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 almost identical to what they would be by the end of the training (See fig 7). This naturally leads us to ask the following questions: (a) How fast do different layers train? (b)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.

For 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 7.It is quite surprising to

Table 7: 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.

| 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 8: 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) | 55.2 | 58.4 | 31.0 | 28.8 | 21.0 | 53.5 | 63.6 | 41.0 | 25.4 | 44.7 | 40.9 | 34.9 | 49.5 | 56.9 | 43.8 | 25.2 | 45.3 | 31.2 | 48.7 | 54.4 | 42.7 |
| 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) | 58.7 | 64.8 | 38.2 | 34.9 | 25.9 | 59.5 | 69.5 | 46.2 | 28.7 | 52.4 | 45.2 | 44.3 | 57.3 | 63.4 | 52.4 | 28.0 | 51.5 | 34.9 | 56.0 | 59.4 | 48.6 |
| 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 |

note that by 15K iterations all layers are within 80% and at 50K iterations within 90% of there final performance. This strongly indicates that a great portion of training required for generalization happens quite quickly.

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. 2.2) and evaluated its performance on the Pascal 2007 detection challenge (see table 8 for results). Consistent with our earlier results we find that this network achieves a surprising performance of 48.6 mean AP points compared to 54.1 achieved by pre-training for 310K iterations.

The above analysis acts as an re-inforcement to our beleif that indeed majority of training required for generalization happens quite quickly as compared to the full training time of the network. This observation suggest that there may exist clever ways which can help us speed up the training.

## 7 Conclusion

In this paper we analysed different properties of convolutional neural networks with the aim of gaining insights required to efficiently exploit the rich feature hierarchies provided by such networks.

## References


1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012)
2. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09. (2009)
3. 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)
4. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524 (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.
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. 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)
17. Singh, S., Gupta, A., Efros, A.A.: Unsupervised discovery of mid-level discriminative patches. In: European Conference on Computer Vision. (2012)
18. 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)
19. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. TPAMI **32**(9) (2010)
20. 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

21. Bourdev, L., Maji, S., Brox, T., Malik, J.: Detecting people using mutually consistent poselet activations. In: European Conference on Computer Vision (ECCV). (2010)
22. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. CoRR **abs/1311.2901** (2013)
23. Simonyan, K., Vedaldi, A., Zisserman, A.: Learning local feature descriptors using convex optimisation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2014)
24. Barlow, H.: Single units and sensations: A neuron doctrine for perceptual psychology? In: Perception. (1972)