Analyzing the Performance of Multilayer Neural Networks for Object Recognition

Pulkit Agrawal, Ross Girshick, Jitendra Malik
{pulkitag,rbg,malik}@eecs.berkeley.edu

University of California, Berkeley

Abstract. In the last two years, convolutional neural networks (CNNs) have achieved an impressive suite of results on standard recognition datasets and tasks. CNN-based features seem poised to quickly replace engineered representations, such as SIFT and HOG. However, compared to SIFT and HOG, we understand much less about the nature of the features learned by large CNNs. In this paper, we experimentally probe several aspects of CNN feature learning in an attempt to help practitioners gain useful, evidence-backed intuitions about how to apply CNNs to computer vision problems.

Keywords: convolutional neural networks, object recognition, empirical analysis

1 Introduction

Over the last two years, a sequence of results on benchmark visual recognition tasks has demonstrated that convolutional neural networks (CNNs) [7, 16, 20] will likely replace engineered features, such as SIFT [17] and HOG [3], for a wide variety of problems. This sequence started with the breakthrough ImageNet [4] classification results reported by Krizhevsky et al. [13]. Soon after, Donahue et al. [5] showed that the same network, trained for ImageNet classification, was an effective blackbox feature extractor. Using CNN features, they reported state-of-the-art results on several standard image classification datasets. At the same time, Girshick et al. [9] showed how the network could be applied to object detection. Their system, called R-CNN, classifies object proposals generated by a bottom-up grouping mechanism (e.g., selective search [25]). Since detection training data is limited, they proposed a transfer learning strategy in which the CNN is first pre-trained, with supervision, for ImageNet classification and then fine-tuned on the small PASCAL detection dataset [6]. Since this initial set of results, several other papers have reported similar findings on a wider range of tasks (see, for example, the outcomes reported by Razavian et al. in [19]).

Feature transforms such as SIFT and HOG afford an intuitive interpretation as histograms of oriented edge filter responses arranged in spatial blocks. However, we have little understanding of what visual features the different layers of a CNN encode. Given that rich feature hierarchies provided by CNNs are likely

to emerge as the prominent feature extractor for computer vision models over the next few years, we believe that developing such an understanding is an interesting scientific pursuit and an essential exercise that will help guide the design of computer vision methods that use CNNs. Therefore, in this paper we study several aspects of CNNs through an empirical lens.

1.1 Summary of findings

Effects of fine-tuning and pre-training. Girshick et al. [9] showed that supervised pre-training and fine-tuning are effective when training data is scarce. However, they did not investigate what happens when training data becomes more abundant. We show that it is possible to get good performance when training R-CNN from a random initialization (i.e., without ImageNet supervised pre-training) with a reasonably modest amount of detection training data (37k ground truth bounding boxes). However, we also show that in this data regime, supervised pre-training is still beneficial and leads to a large improvement in detection performance. We show similar results for image classification, as well.

ImageNet pre-training does not overfit. One concern when using supervised pre-training is that achieving a better model fit to ImageNet, for example, might lead to higher generalization error when applying the learned features to another dataset and task. If this is the case, then some form of regularization during pre-training, such as early stopping, would be beneficial. We show the surprising result that pre-training for longer yields better results, with diminishing returns, but does *not* increase generalization error. This implies that fitting the CNN to ImageNet induces a general and portable feature representation. Moreover, the learning process is well behaved and does not require ad hoc regularization in the form of early stopping.

Grandmother cells and distributed codes. We do not have a good understanding of mid-level feature representations in multilayer networks. Recent work on feature visualization, (e.g., [15, 28]) suggests that such networks might consist mainly of "grandmother" cells [1, 18]. Our analysis shows that the representation in intermediate layers is more subtle. There are a small number of grandmother-cell-like features, but most of the feature code is distributed and several features must fire in concert to effectively discriminate between classes.

Importance of feature location and magnitude. Our final set of experiments investigates what role a feature's spatial location and magnitude plays in image classification and object detection. Matching intuition, we find that spatial location is critical for object detection, but matters little for image classification. More surprisingly, we find that feature magnitude is largely unimportant. For example, binarizing features (at a threshold of 0) barely degrades performance. This shows that sparse binary features, which are useful for large-scale image retrieval [10, 26], come "for free" from the CNN's representation.

2 Experimental setup

2.1 Datasets and tasks

In this paper, we report experimental results using several standard datasets and tasks, which we summarize here.

Image classification. For the task of image classification we consider two datasets, the first of which is PASCAL VOC 2007 [6]. We refer to this dataset and task by "PASCAL-CLS". Results on PASCAL-CLS are reported using the standard average precision (AP) and mean average precision (mAP) metrics.

PASCAL-CLS is fairly small-scale with only 5k images for training, 5k images for testing, and 20 object classes. Therefore, we also consider the medium-scale SUN dataset [27], which has around 108k images and 397 classes. We refer to experiments on SUN by "SUN-CLS". In these experiments, we use a non-standard train-test split since it was computationally infeasible to run all of our experiments on the 10 standard subsets proposed by [27]. Instead, we randomly split the dataset into three parts (train, val, and test) using 50%, 10% and 40% of the data, respectively. The distribution of classes was uniform across all the three sets. We emphasize that results on these splits are only used to support investigations into properties of CNNs and not for comparing against other scene-classification methods in the literature. For SUN-CLS, we report 1-of-397 classification accuracy averaged over all classes, which is the standard metric for this dataset[1]. For select experiments we report the error bars in performance as mean $\pm$ standard deviation in accuracy over 3 runs (it was computationally infeasible to compute error bars for all experiments). For each run, a different random split of train, val, and test sets was used.

Object detection. For the task of object detection we use PASCAL VOC 2007. We train using the trainval set and test on the test set. We refer to this dataset and task by "PASCAL-DET". PASCAL-DET uses the same set of images as PASCAL-CLS. We note that it is standard practice to use the 2007 version of PASCAL VOC for reporting results of ablation studies and hyperparameter sweeps. We report performance on PASCAL-DET using the standard AP and mAP metrics. In some of our experiments we use only the ground-truth PASCAL-DET bounding boxes, in which case we refer to the setup by "PASCAL-DET-GT".

In order to provide a larger detection training set for certain experiments, we also make use of the "PASCAL-DET+DATA" dataset, which we define as including VOC 2007 trainval union with VOC 2012 trainval. The VOC 2007 test set is still used for evaluation. This dataset contains approximately 37k

[1] The version of this paper published at ECCV'14 contained an error in our description of the accuracy metric. That version used overall accuracy, instead of class-averaged accuracy. This version contains corrected numbers for SUN-CLS to reflect the standard accuracy metric of class-averaged accuracy.

labeled bounding boxes, which is roughly three times the number contained in PASCAL-DET.

2.2 Network architecture and layer nomenclature

All of our experiments use a single CNN architecture. This architecture is the Caffe [11] implementation of the network proposed by Krizhevsky et al. [13]. The layers of the CNN are organized as follows. The first two are subdivided into four sublayers each: convolution (conv), $\max(x, 0)$ rectifying non-linear units (ReLUs), max pooling, and local response normalization (LRN). Layers 3 and 4 are composed of convolutional units followed by ReLUs. Layer 5 consists of convolutional units, followed by ReLUs and max pooling. The last two layers are fully connected (fc). When we refer to conv-1, conv-2, and conv-5 we mean the output of the max pooling units following the convolution and ReLU operations (also following LRN when applicable).[2] For layers conv-3, conv-4, fc-6, and fc-7 we mean the output of ReLU units.

2.3 Supervised pre-training and fine-tuning

Training a large CNN on a small dataset often leads to catastrophic overfitting. The idea of supervised *pre-training* is to use a data-rich auxiliary dataset and task, such as ImageNet classification, to initialize the CNN parameters. The CNN can then be used on the small dataset, directly, as a feature extractor (as in [5]). Or, the network can be updated by continued training on the small dataset, a process called *fine-tuning*.

For fine-tuning, we follow the procedure described in [9]. First, we remove the CNN's classification layer, which was specific to the pre-training task and is not reusable. Next, we append a new randomly initialized classification layer with the desired number of output units for the target task. Finally, we run stochastic gradient descent (SGD) on the target loss function, starting from a learning rate set to 0.001 (1/10-th the initial learning rate used for training the network for ImageNet classification). This choice was made to prevent clobbering the CNN's initialization to control overfitting. At every 20,000 iterations of fine-tuning we reduce the learning rate by a factor of 10.

3 The effects of fine-tuning and pre-training on CNN performance and parameters

The results in [9] (R-CNN) show that supervised pre-training for ImageNet classification, followed by fine-tuning for PASCAL object detection, leads to large gains over directly using features from the pre-trained network (without fine-tuning). However, [9] did not investigate three important aspects of fine-tuning: (1) What happens if we train the network "from scratch" (i.e., from a random

[2] Note that this nomenclature differs slightly from [9].

Table 1: Comparing the performance of CNNs trained from scratch, pre-trained on ImageNet, and fine-tuned. PASCAL-DET+DATA includes additional data from VOC 2012 trainval. (Bounding-box regression was not used for detection results.)

SUN-CLS			**PASCAL-DET**			**PASCAL-DET+DATA**		
scratch	pre-train	fine-tune	scratch	pre-train	fine-tune	scratch	pre-train	fine-tune
35.7 ± 0.2	48.4 ± 0.1	52.2 ± 0.1	40.7	45.5	54.1	52.3	45.5	59.2

initialization) on the detection data? (2) How does the amount of fine-tuning data change the picture? and (3) How does fine-tuning alter the network's parameters? In this section, we explore these questions on object detection and image classification datasets.

3.1 Effect of fine-tuning on CNN performance

The main results of this section are presented in Table 1. First, we focus on the detection experiments, which we implemented using the open source R-CNN code. All results use features from layer fc-7.

Somewhat surprisingly, it's possible to get reasonable results (40.7% mAP) when training the CNN from scratch using only the training data from VOC 2007 trainval (13k bounding box annotations). However, this is still worse than using the pre-trained network, directly, without fine-tuning (45.5%). Even more surprising is that when the VOC 2007 trainval data is augmented with VOC 2012 data (an additional 25k bounding box annotations), we are able to achieve a mAP of 52.3% from scratch. This result is almost as good as the performance achieved by pre-training on ImageNet and then fine-tuning on VOC 2007 trainval (54.1% mAP). These results can be compared to the 30.5% mAP obtained by DetectorNet [23], a recent detection system based on the same network architecture, which was trained from scratch on VOC 2012 trainval.

Next, we ask if ImageNet pre-training is still useful in the PASCAL-DET +DATA setting? Here we see that even though it's possible to get good performance when training from scratch, pre-training still helps considerably. The final mAP when fine-tuning with the additional detection data is 59.2%, which is 5 percentage points higher than the best result reported in [9] (both without bounding-box regression). This result suggests that R-CNN performance is not data saturated and that simply adding more detection training data without any other changes may substantially improve results.

We also present results for SUN image classification. Here we observe a similar trend: reasonable performance is achievable when training from scratch, however initializing from ImageNet and then fine-tuning yields significantly better performance.

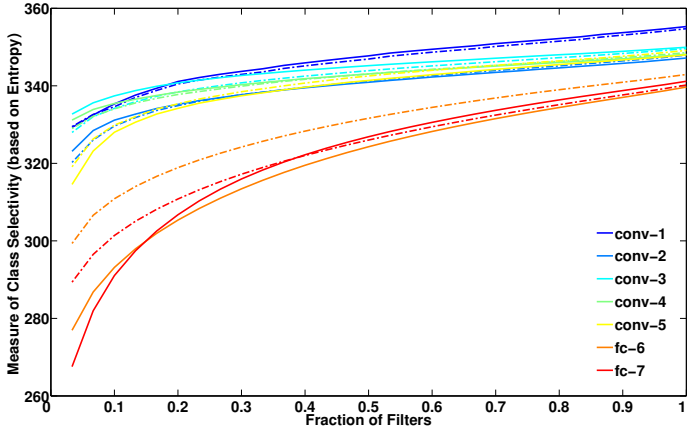


Fig. 1: PASCAL object class selectivity plotted against the fraction of filters, for each layer, before fine-tuning (dash-dot line) and after fine-tuning (solid line). A lower value indicates greater class selectivity. Although layers become more discriminative as we go higher up in the network, fine-tuning on limited data (PASCAL-DET) only significantly affects the last two layers (fc-6 and fc-7).

3.2 Effect of fine-tuning on CNN parameters

We have provided additional evidence that fine-tuning a discriminatively pre-trained network is very effective in terms of task performance. Now we look inside the network to see how fine-tuning changes its parameters.

To do this, we define a way to measure the class selectivity of a set of filters. Intuitively, we use the class-label entropy of a filter given its activations, above a threshold, on a set of images. Since this measure is entropy-based, a low value indicates that a filter is highly class selective, while a large value indicates that a filter fires regardless of class. The precise definition of this measure is given in the Appendix.

In order to summarize the class selectivity for a *set* of filters, we sort them from the most selective to least selective and plot the average selectivity of the first k filters while sweeping k down the sorted list. Figure 1 shows the class selectivity for the sets of filters in layers 1 to 7 before and after fine-tuning (on VOC 2007 trainval). Selectivity is measured using the ground truth boxes from PASCAL-DET-GT instead of a whole-image classification task to ensure that filter responses are a direct result of the presence of object categories of interest and not correlations with image background.

Figure 1 shows that class selectivity increases from layer 1 to 7 both with and without fine-tuning. It is interesting to note that entropy changes due to fine-tuning are only significant for layers 6 and 7. This observation indicates that fine-tuning only layers 6 and 7 may suffice for achieving good performance when fine-tuning data is limited. We tested this hypothesis on SUN-CLS and PASCAL-DET by comparing the performance of a fine-tuned network (ft) with

Table 2: Comparison in performance when fine-tuning the entire network (ft) versus only fine-tuning the fully-connected layers (fc-ft).

SUN-CLS		**PASCAL-DET**		**PASCAL-DET+DATA**	
ft	fc-ft	ft	fc-ft	ft	fc-ft
52.2 ± 0.1	51.6 ± 0.1	54.1	53.3	59.2	56.0

Table 3: Performance variation (% mAP) on PASCAL-CLS as a function of pre-training iterations on ImageNet. The error bars for all columns are similar to the one reported in the 305k column.

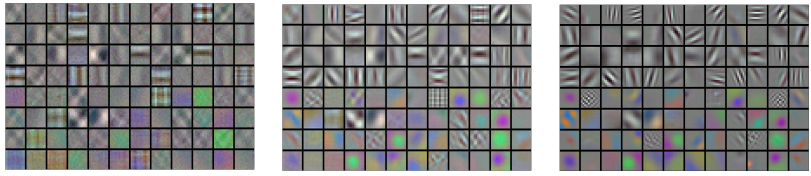
layer	5k	15k	25k	35k	50k	95k	105k	195k	205k	305k
conv-1	23.0	24.3	24.4	24.5	24.3	24.8	24.7	24.4	24.4	24.4 ± 0.5
conv-2	33.7	40.4	40.9	41.8	42.7	43.2	44.0	45.0	45.1	45.1 ± 0.7
conv-3	34.2	46.8	47.0	48.2	48.6	49.4	51.6	50.7	50.9	50.5 ± 0.6
conv-4	33.5	49.0	48.7	50.2	50.7	51.6	54.1	54.3	54.4	54.2 ± 0.7
conv-5	33.0	53.4	55.0	56.8	57.3	59.2	63.5	64.9	65.5	65.6 ± 0.3
fc-6	34.2	59.7	62.6	62.7	63.5	65.6	69.3	71.3	71.8	72.1 ± 0.3
fc-7	30.9	61.3	64.1	65.1	65.9	67.8	71.8	73.4	74.0	74.3 ± 0.3

a network which was fine-tuned by only updating the weights of fc-6 and fc-7 (fc-ft). These results, in Table 2, show that with small amounts of data, fine-tuning amounts to "rewiring" the fully connected layers. However, when more fine-tuning data is available (PASCAL-DET+DATA), there is still substantial benefit from fine-tuning all network parameters.

3.3 Effect of pre-training on CNN parameters

There is no single image dataset that fully captures the variation in natural images. This means that all datasets, including ImageNet, are biased in some way. Thus, there is a possibility that pre-training may eventually cause the CNN to overfit and consequently hurt generalization performance [24]. To understand if this happens, in the specific case of ImageNet pre-training, we investigated the effect of pre-training time on generalization performance both with and without fine-tuning. We find that pre-training for longer improves performance. This is surprising, as it shows that fitting more to ImageNet leads to better performance when moving to the other datasets that we evaluated.

We report performance on PASCAL-CLS as a function of pre-training time, without fine-tuning, in Table 3. Notice that more pre-training leads to better performance. By 15k and 50k iterations all layers are close to 80% and 90% of their final performance (5k iterations is only $\sim$1 epoch). This indicates that training required for generalization takes place quite quickly. Figure 2 shows conv-1 filters after 5k, 15k, and 305k iterations and reinforces this observation.



(a) 5k Iterations (b) 15k Iterations (c) 305k Iterations

Fig. 2: Evolution of conv-1 filters with time. After just 15k iterations, these filters closely resemble their converged state.

Table 4: Performance variation on SUN-CLS and PASCAL-DET using features from a CNN pre-trained for different numbers of iterations and fine-tuned for a fixed number of iterations (40k for SUN-CLS and 70k for PASCAL-DET)

	50k	105k	205k	305k
SUN-CLS	48.5 ± 0.1	50.0 ± 0.2	51.8 ± 0.3	51.9 ± 0.3
PASCAL-DET	50.2	52.6	55.3	55.4[3]

Further, notice from Table 3 that conv-1 trains first and the higher the layer is the more time it takes to converge. This suggests that a CNN, trained with backpropagation, converges in a layer-by-layer fashion. Table 4 shows the interaction between varied amounts of pre-training time and fine-tuning on SUN-CLS and PASCAL-DET. Here we also see that more pre-training prior to fine-tuning leads to better performance.

4 Are there grandmother cells in CNNs?

Neuroscientists have conjectured that cells in the human brain which only respond to very specific and complex visual stimuli (such as the face of one's grandmother) are involved in object recognition. These neurons are often referred to as *grandmother cells* (GMC) [1, 18]. Proponents of artificial neural networks have shown great interest in reporting the presence of GMC-like filters for specific object classes in their networks (see, for example, the cat filter reported in [15]). The notion of GMC like features is also related to standard feature encodings for image classification. Prior to the work of [13], the dominant approaches for image and scene classification were based on either representing images as a bag of local descriptors (BoW), such as SIFT (e.g., [14]), or by first finding a set of mid-level patches [12, 22] and then encoding images in terms of them. The problem of finding good mid-level patches is often posed as a search for a set of high-recall discriminative templates. In this sense, mid-level patch discovery

[3] A network pre-trained from scratch, which was different from the one used in Section 3.1, was used to obtain these results. The difference in performance is not significant.

is the search for a set of GMC templates. The low-level BoW representation, in contrast, is a *distributed code* in the sense that a single feature by itself is not discriminative, but a group of features taken together is. This makes it interesting to investigate the nature of mid-level CNN features such as conv-5.

For understanding these feature representations in CNNs, [21, 28] recently presented methods for finding locally optimal visual inputs for individual filters. However, these methods only find the best, or in some cases top-k, visual inputs that activate a filter, but do not characterize the *distribution* of images that cause an individual filter to fire above a certain threshold. For example, if it is found that the top-10 visual inputs for a particular filter are cats, it remains unclear what is the response of the filter to other images of cats. Thus, it is not possible to make claims about presence of GMC like filters for cat based on such analysis. A GMC filter for the cat class, is one that fires strongly on *all* cats and nothing else. This criteria can be expressed as a filter that has high *precision* and high *recall*. That is, a GMC filter for class C is a filter that has a high average precision (AP) when tasked with classifying inputs from class C versus inputs from all other classes.

First, we address the question of finding GMC filters by computing the AP of individual filters (Section 4.1). Next, we measure how distributed are the feature representations (Section 4.2). For both experiments we use features from layer conv-5, which consists of responses of 256 filters in a 6×6 spatial grid. Using max pooling, we collapse the spatial grid into a 256-D vector, so that for each filter we have a single response per image (in Section 5.1 we show that this transformation causes only a small drop in task performance).

4.1 Finding Grandmother Cells

For each filter, its AP value is calculated for classifying images using class labels and filter responses to object bounding boxes from PASCAL-DET-GT. Then, for each class we sorted filters in decreasing order of their APs. If GMC filters for this class exist, they should be the top ranked filters in this sorted list. The precision-recall curves for the top-five conv-5 filters are shown in Figure 3. We find that GMC-like filters exist for only for a few classes, such as bicycle, person, cars, and cats.

4.2 How distributed are the feature representations?

In addition to visualizing the AP curves of individual filters, we measured the number of filters required to recognize objects of a particular class. Feature selection was performed to construct nested subsets of filters, ranging from a single filter to all filters, using the following greedy strategy. First, separate linear SVMs were trained to classify object bounding boxes from PASCAL-DET-GT using conv-5 responses. For a given class, the 256 dimensions of the learnt weight vector (w) is in direct correspondence with the 256 conv-5 filters. We used the magnitude of the i-th dimension of w to rank the importance of the i-th conv-5 filter for discriminating instances of this class. Next, all filters were sorted using

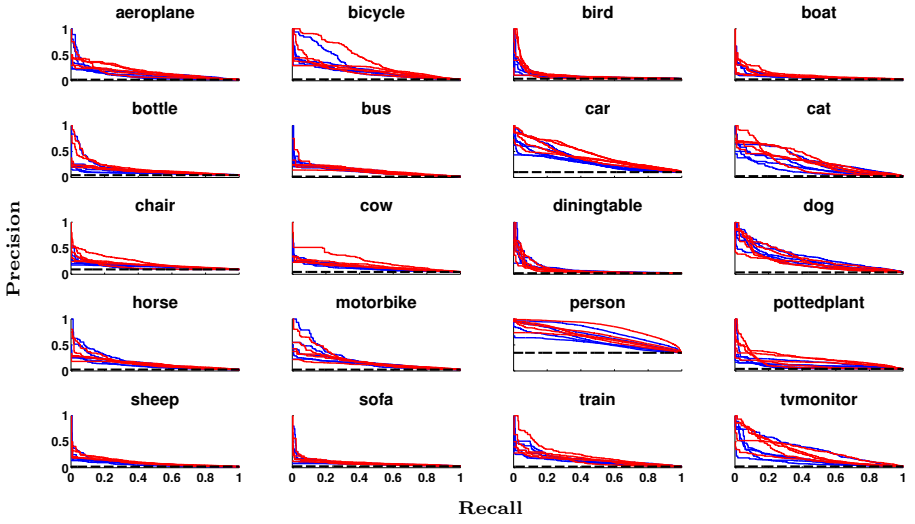


Fig. 3: The precision-recall curves for the top five (based on AP) conv-5 filter responses on PASCAL-DET-GT. Curves in red and blue indicate AP for fine-tuned and pre-trained networks, respectively. The dashed black line is the performance of a random filter. For most classes, precision drops significantly even at modest recall values. There are GMC filters for classes such as bicycle, person, car, cat.

Table 5: Number of filters required to achieve 50% or 90% of the complete performance on PASCAL-DET-GT using a CNN pre-trained on ImageNet and fine-tuned for PASCAL-DET using conv-5 features.

	perf.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
pre-train	50%	15	3	15	15	10	10	3	2	5	15	15	2	10	3	1	10	20	25	10	2
fine-tune	50%	10	1	20	15	5	5	2	2	3	10	15	3	15	10	1	5	15	15	5	2
pre-train	90%	40	35	80	80	35	40	30	20	35	100	80	30	45	40	15	45	50	100	45	25
fine-tune	90%	35	30	80	80	30	35	25	20	35	50	80	35	30	40	10	35	40	80	40	20

these magnitude values. Each subset of filters was constructed by taking the top-k filters from this list.[4] For each subset, a linear SVM was trained using only the responses of filters in that subset for classifying the class under consideration.

The variation in performance with the number of filters is shown in Figure 2. Table 10 lists the number of filters required to achieve 50% and 90% of the complete performance. For classes such as persons, cars, and cats relatively few filters are required, but for most classes around 30 to 40 filters are required to achieve at least 90% of the full performance. This indicates that the conv-5 feature representation is distributed and there are GMC-like filters for only a few classes. Results using layer fc-7 are presented in the the supplementary material.

[4] We used values of $k \in \{1, 2, 3, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 80, 100, 128, 256\}$.

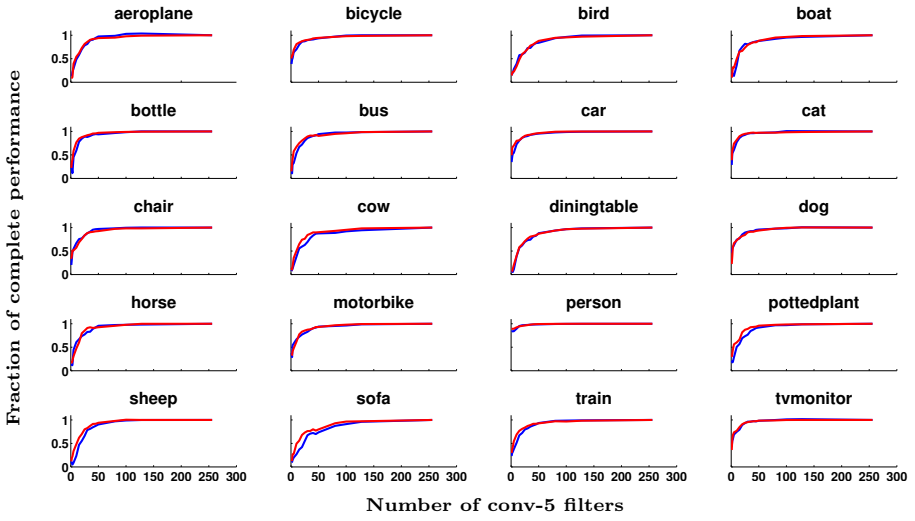


Fig. 4: The fraction of complete performance on PASCAL-DET-GT achieved by conv-5 filter subsets of different sizes. Complete performance is the AP computed by considering responses of all the filters. Notice, that for a few classes such as person and bicycle only a few filters are required, but for most classes substantially more filters are needed, indicating a distributed code.

We also find that after fine-tuning, slightly fewer filters are required to achieve performance levels similar to a pre-trained network.

Next, we estimated the extent of overlap between the filters used for discriminating between different classes. For each class i, we selected the 50 most discriminative filters (out of 256) and stored the selected filter indices in the set S_i. The extent of overlap between class i and j was evaluated by $|S_i \cap S_j|/N$, where $N = |S_i| = |S_j| = 50$. The results are visualized in Figure 5. It can be seen that different classes use different subsets of conv-5 filters and there is little overlap between classes. This further indicates that intermediate representations in the CNN are distributed.

5 Untangling feature magnitude and location

The convolutional layers preserve the coarse spatial layout of the network's input. By layer conv-5, the original 227×227 input image has been progressively downsampled to 6×6. This feature map is also sparse due to the $\max(x, 0)$ non-linearities used in the network (conv-5 is roughly 27% non-zero; sparsity statistics for all layers are given in Table 6). Thus, a convolutional layer encodes information in terms of (1) which filters have non-zero responses, (2) the magnitudes of those responses, and (3) their spatial layout. In this section, we experimentally analyze the role of filter response magnitude and spatial location by looking at ablation studies on classification and detection tasks.

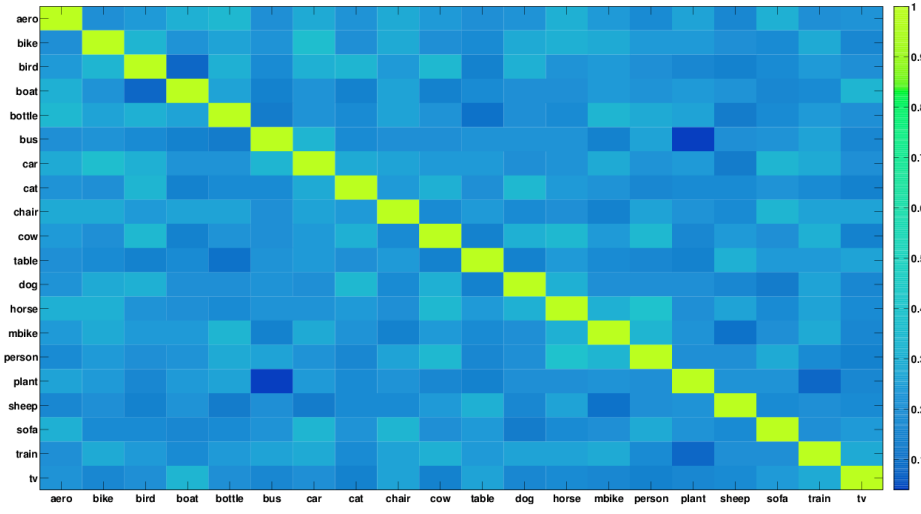


Fig. 5: The set overlap between the 50 most discriminative conv-5 filters for each class determined using PASCAL-DET-GT. Entry (i, j) of the matrix is the fraction of top-50 filters class i has in common with class j (Section 4.2). Chance is 0.195. There is little overlap, but related classes are more likely to share filters.

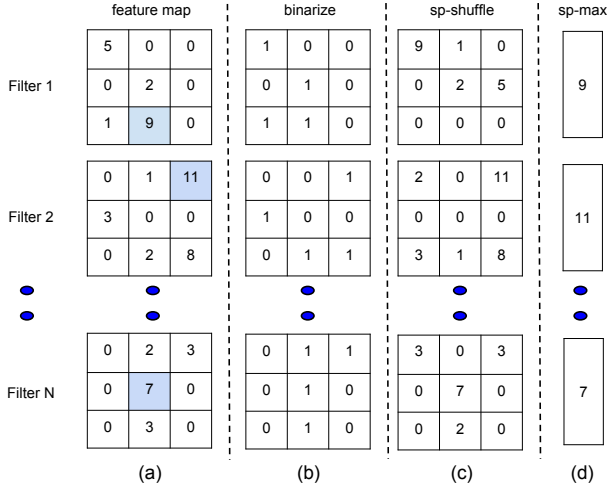


Fig. 6: Illustrations of ablations of feature activation spatial and magnitude information. See Sections 5.1 and 5.2 for details.

5.1 How important is filter response magnitude?

We can asses the importance of magnitude by setting each filter response x to 1 if $x > 0$ and to 0 otherwise. This binarization is performed prior to using the re-

Table 6: Percentage non-zeros (sparsity) in filter responses of CNN.

conv-1	conv-2	conv-3	conv-4	conv-5	fc-6	fc-7
87.5 ± 4.4	44.5 ± 4.4	31.8 ± 2.4	32.0 ± 2.7	27.7 ± 5.0	16.1 ± 3.0	21.6 ± 4.9

sponses as features in a linear classifier and leads to loss of information contained in the magnitude of response while still retaining information about which filters fired and where they fired. In Tables 7 and 8 we show that binarization leads to a negligible performance drop for both classification and detection.

For the fully-connected layers (fc-6 and fc-7) PASCAL-CLS performance is nearly identical before and after binarization. This is a non-trivial property since transforming traditional computer vision features into short (or sparse) binary codes is an active research area. Such codes are important for practical applications in large-scale image retrieval and mobile image analysis [10, 26]. Here we observe that sparse binary codes come essentially "for free" when using the representations learned in the fully-connected layers.

5.2 How important is response location?

Now we remove spatial information from filter responses while retaining information about their magnitudes. We consider two methods for ablating spatial information from features computed by the convolutional layers (the fully-connected layers do not contain *explicit* spatial information).

The first method ("sp-max") simply collapses the $p \times p$ spatial map into a single value per feature channel by max pooling. The second method ("sp-shuffle") retains the original distribution of feature activation values, but scrambles spatial correlations between columns of feature channels. To perform sp-shuffle, we permute the spatial locations in the $p \times p$ spatial map. This permutation is performed independently for each network input (i.e., different inputs undergo different permutations). Columns of filter responses in the same location move together, which preserves correlations between features within each (shuffled) spatial location. These transformations are illustrated in Figure 6.

For image classification, damaging spatial information leads to a large difference in performance between original and spatially-ablated conv-1 features, but with a gradually decreasing difference for higher layers (Table 7). In fact, the performance of conv-5 after sp-max is close to the original performance. This indicates that a lot of information important for classification is encoded in the activation of the filters and not necessarily in the spatial pattern of their activations. Note, this observation is not an artifact of small number of classes in PASCAL-CLS. On ImageNet validation data, conv-5 features and conv-5 after sp-max result into accuracy of 43.2 and 41.5 respectively. However, for detection sp-max leads to a large drop in performance. This may not be surprising since detection requires spatial information for precise localization.

Table 7: Effect of location and magnitude feature ablations on PASCAL-CLS.

layer	no ablation (mAP)	binarize (mAP)	sp-shuffle (mAP)	sp-max (mAP)
conv-1	25.1 ± 0.5	17.7 ± 0.2	15.1 ± 0.3	25.4 ± 0.5
conv-2	45.3 ± 0.5	43.0 ± 0.6	32.9 ± 0.7	40.1 ± 0.3
conv-3	50.7 ± 0.6	47.2 ± 0.6	41.0 ± 0.8	54.1 ± 0.5
conv-4	54.5 ± 0.7	51.5 ± 0.7	45.2 ± 0.8	57.0 ± 0.5
conv-5	65.6 ± 0.6	60.8 ± 0.7	59.5 ± 0.4	62.5 ± 0.6
fc-6	71.7 ± 0.3	71.5 ± 0.4	-	-
fc-7	74.1 ± 0.3	73.7 ± 0.4	-	-

Table 8: Effect of location and magnitude feature ablations on PASCAL-DET.

	no ablation (mAP)	binarize (mAP)	sp-max (mAP)
conv-5	47.6	45.7	25.4

6 Conclusion

To help researchers better understand CNNs, we investigated pre-training and fine-tuning behavior on three classification and detection datasets. We found that the large CNN used in this work can be trained from scratch using a surprisingly modest amount of data. But, importantly, pre-training significantly improves performance and pre-training for longer is better. We also found that some of the learnt CNN features are grandmother-cell-like, but for the most part they form a distributed code. This supports the recent set of empirical results showing that these features generalize well to other datasets and tasks.

Acknowledgments. This work was supported by ONR MURI N000141010933. Pulkit Agrawal is partially supported by a Fulbright Science and Technology fellowship. We thank NVIDIA for GPU donations. We thank Bharath Hariharan, Saurabh Gupta and João Carreira for helpful suggestions.

Citing this paper. Please cite the paper as:

```
@inproceedings{agrawal14analyzing,
Author = {Pulkit Agrawal and Ross Girshick and Jitendra Malik},
Title = {Analyzing the Performance of Multilayer Neural Networks for Object Recognition},
Booktitle = {Proceedings of the European Conference on Computer Vision (ECCV)},
Year = {2014 } }
```

Appendix: estimating a filter's discriminative capacity

To measure the discriminative capacity of a filter, we collect filter responses from a set of N images. Each image, when passed through the CNN produces a $p \times p$

heat map of scores for each filter in a given layer (e.g., $p = 6$ for a conv-5 filter and $p = 1$ for an fc-6 filter). This heat map is vectorized into a vector of scores of length p^2. With each element of this vector we associate the image's class label. Thus, for every image we have a score vector and a label vector of length p^2 each. Next, the score vectors from all N images are concatenated into an Np^2-length score vector. The same is done for the label vectors.

Now, for a given score threshold τ, we define the *class entropy of a filter* to be the entropy of the normalized histogram of class labels that have an associated score $\geq \tau$. A low class entropy means that at scores above τ, the filter is very class selective. As this threshold changes, the class entropy traces out a curve which we call the *entropy curve*. The *area under the entropy curve* (AuE), summarizes the class entropy at all thresholds and is used as a measure of discriminative capacity of the filter. The lower the AuE value, the more class selective the filter is. The AuE values are used to sort filters in Section 1.1.

References

1. Barlow, H.: Single units and sensations: A neuron doctrine for perceptual psychology? In: Perception (1972)
2. Breiman, L.: Random forests. Mach. Learn. 45(1), 5–32 (Oct 2001), http://dx.doi.org/10.1023/A:1010933404324
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: In CVPR. pp. 886–893 (2005)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
5. 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)
6. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. IJCV 88(2) (2010)
7. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics 36(4), 193–202 (1980)
8. Geman, D., Amit, Y., Wilder, K.: Joint induction of shape features and tree classifiers (1997)
9. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
10. Gong, Y., Lazebnik, S.: Iterative quantization: A procrustean approach to learning binary codes. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 817–824. IEEE (2011)
11. Jia, Y.: Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/ (2013)
12. 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)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)

14. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. vol. 2, pp. 2169–2178. IEEE (2006)
15. 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)
16. 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)
17. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60, 91–110 (2004)
18. Quiroga, R.Q., Reddy, L., Kreiman, G., Koch, C., Fried, I.: Invariant visual representation by single neurons in the human brain. Nature 435(7045), 1102–7 (2005), http://www.biomedsearch.com/nih/Invariant-visual-representation-by-single/15973409.html
19. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: an astounding baseline for recognition. CoRR abs/1403.6382 (2014)
20. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Parallel Distributed Processing 1, 318–362 (1986)
21. Simonyan, K., Vedaldi, A., Zisserman, A.: Learning local feature descriptors using convex optimisation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2014)
22. Singh, S., Gupta, A., Efros, A.A.: Unsupervised discovery of mid-level discriminative patches. In: European Conference on Computer Vision (2012), http://arxiv.org/abs/1205.3137
23. Szegedy, C., Toshev, A., Erhan, D.: Deep neural networks for object detection. In: NIPS (2013)
24. Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 1521–1528. IEEE (2011)
25. Uijlings, J., van de Sande, K., Gevers, T., Smeulders, A.: Selective search for object recognition. IJCV (2013)
26. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances in neural information processing systems. pp. 1753–1760 (2009)
27. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. CVPR pp. 3485–3492 (2010)
28. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. CoRR abs/1311.2901 (2013)

Supplementary Material

1 Effect of fine-tuning on CNN parameters

In the main paper we provided evidence that fine-tuning a discriminatively pre-trained network is very effective in terms of task performance. We also provided insights into how fine-tuning changes its parameters. Here we describe and discuss in greater detail some metrics for determining the effect of fine-tuning.

1.1 Defining the measure of discriminative capacity of a filter

To measure the discriminative capacity of a filter, we start by collecting filter responses from a set of N images. Each image, when passed through the CNN produces a $p \times p$ heat map of scores for each filter in a given layer (e.g., $p = 6$ for a conv-5 filter and $p = 1$ for an fc-6 filter). This heat map is vectorized (`x(:)` in MATLAB) into a vector of scores of length p^2. With each element of this vector we associate the class label of the image. Thus, for every image we have a score vector and a label vector of length p^2 each. Next, the score vectors from all N images are concatenated into an Np^2-length score vector. The same is done for the label vectors.

Now, for a given score threshold τ, we define the *class entropy of a filter* to be the entropy of the normalized histogram of class labels that have an associated score $\geq \tau$. A low class entropy means that at scores above τ, the filter is very class selective. As this threshold changes, the class entropy traces out a curve which we call the *entropy curve*. The *area under the entropy curve* (AuE), summarizes the class entropy at all thresholds and is used as a measure of discriminative capacity of the filter. The lower the AuE value, the more class selective the filter is.

1.2 Defining discriminative capacity of a layer

We discuss three slightly different ways of defining the discriminative capacity of a layer.

Label Entropy The discriminative capacity of layer is computed as following: We sort the filter according to their AuE. Then, the cumulative sum of AuE values in the sorted list is calculated (called *cumulative AuE* or CAuE). The i-th entry of the CAuE list is the sum of the AuE scores of the top i most discriminative filters. The difference in the value of the i-th entry before and after fine-tuning measures the change in class selectivity of the top i most discriminative filters due to fine-tuning. For comparing results across different layers, the

CAuE values are normalized to account for different numbers of filters in each layer. Specifically, the i-th entry of the CAuE list is divided by i. This normalized CAuE is called the Mean Cumulative Area Under the Entropy Curve (MCAuE). A lower value of MCAuE indicates that the set filters is more discriminative.

Weighted Label Entropy When computing the label histogram for computing AuE, 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.

Table 9: 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

1.3 Discussion

The change in MCAuE for different layers as measured using alternative definitions of entropy can be seen in fig 1 (Please see sec 3 of the main paper for

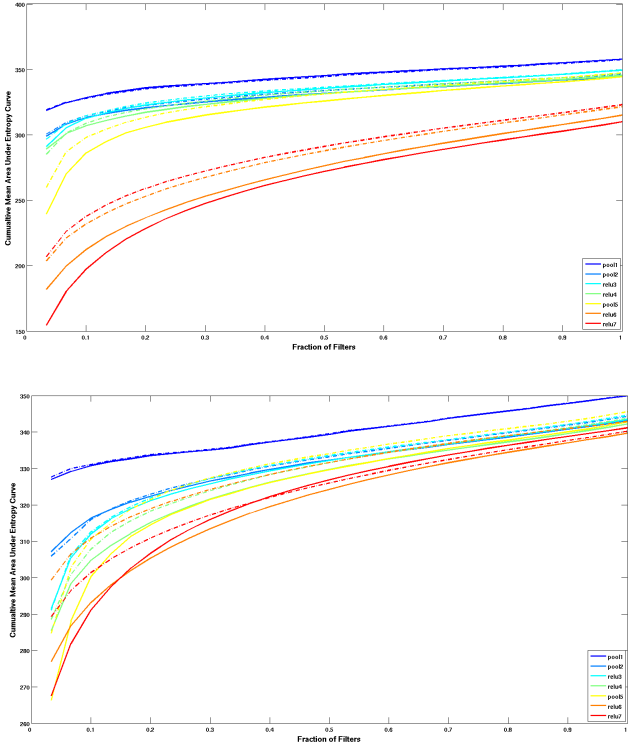


Fig. 1: Mean Cumulative AuE plotted as fraction of filters for all layers of the Conv-Net. Dash-Dot Line: Alex-Net, Solid Line: Fine-Tuned Network. The top plot shows entropy calculated using Weighted-Label-Entropy Method, whereas the bottom plot entropy calculated using spMax-Label-Entropy Method. (see sec 1.2 for method definitions.)

Table 10: Number of filters required to achieve 50% or 90% of the complete performance on PASCAL-DET-GT using a CNN pre-trained on ImageNet and fine-tuned for PASCAL-DET using fc-7 features.

	perf.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
pre-train	50%	1	1	1	2	2	3	1	1	2	6	11	2	2	2	1	3	3	5	2	1
fine-tune	50%	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	3	1	1
pre-train	90%	33	40	40	40	17	32	32	31	40	40	40	35	37	37	17	29	40	40	17	8
fine-tune	90%	6	7	11	4	5	10	2	8	19	27	32	18	9	16	2	7	7	40	3	4

more details on the procedure). A quantitative measure of change in entropy after finetuning is provided in table 9. The percentage change in MCAuE is calculated as:

$$\text{Percent Decrease} = 100 \times \frac{MCAuE_{untuned} - MCAuE_{fine}}{MCAuE_{untuned}} \tag{1}$$

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

Although, the most natural way to compute entropy is Label-Entropy as used by [2], [8], the alternative metrics also support the claim that most of the changes due to fine-tuning happen while moving from layers 5 to 7. At the same time, it is noteworthy, that for label-entropy pool-5 undergoes negligible change in entropy, whereas changes in pool-5 as measured by Weighted-Label Entropy and spMax-Label-Entropy are comparable to changes in relu-6 and relu-7.

2 Are there grandmother cells in CNNs?

In the main paper we studied the nature of representations in mid-level CNN representations given by conv-5. Here, we address the same question for layer fc-7, which is the last layer of CNN and features extracted from this lead to best performance. The results for number of filters required to achieve the same performance as all the filters taken together is presented in Figure 2. Table 10 reports the number of filters required per class to obtain 50% and 90% of the complete performance. It can be seen that like conv-5, feature representations in fc-7 are also distributed for a large number of classes. It is interesting to note, that for most classes 50% performance can be reached using a single filter, but for reaching 90% performance a lot more filters are required.

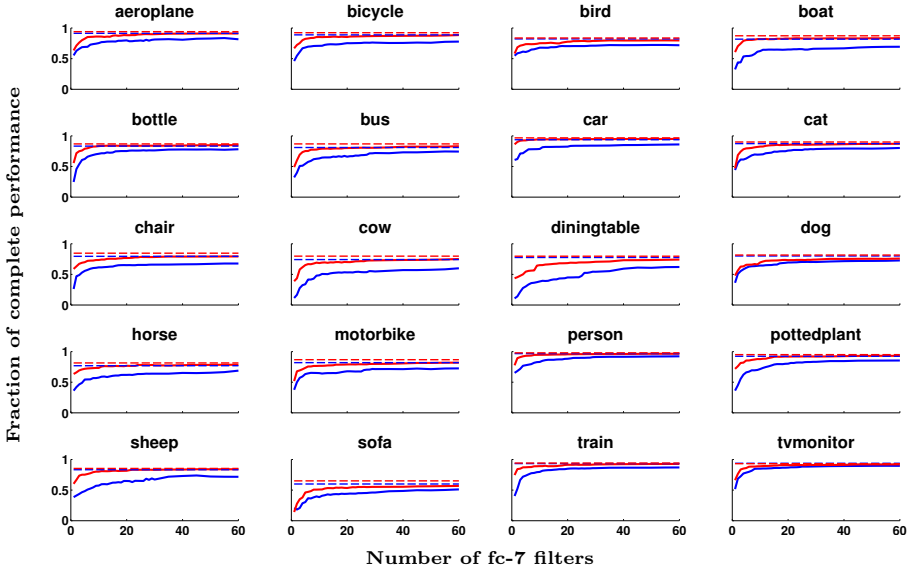


Fig. 2: The fraction of complete performance on PASCAL-DET-GT achieved by fc-7 filter subsets of different sizes. Complete performance is the AP computed by considering responses of all the filters. The solid blue and red lines are for pre-trained and fine-tuned network respectively. The dashed blue and red lines show the complete performance. Notice, that for a few classes such as person, bicycle and cars only a few filters are required, but for many classes substantially more filters are needed, indicating a distributed code.