Visualizing and Understanding Convolutional Networks

Matthew D. Zeiler and Rob Fergus

Dept. of Computer Science,
New York University, USA
{zeiler,fergus}@cs.nyu.edu

Abstract. Large Convolutional Network models have recently demonstrated impressive classification performance on the ImageNet benchmark Krizhevsky *et al.* [18]. However there is no clear understanding of why they perform so well, or how they might be improved. In this paper we explore both issues. We introduce a novel visualization technique that gives insight into the function of intermediate feature layers and the operation of the classifier. Used in a diagnostic role, these visualizations allow us to find model architectures that outperform Krizhevsky *et al.* on the ImageNet classification benchmark. We also perform an ablation study to discover the performance contribution from different model layers. We show our ImageNet model generalizes well to other datasets: when the softmax classifier is retrained, it convincingly beats the current state-of-the-art results on Caltech-101 and Caltech-256 datasets.

1 Introduction

Since their introduction by LeCun *et al.* [20] in the early 1990's, Convolutional Networks (convnets) have demonstrated excellent performance at tasks such as hand-written digit classification and face detection. In the last 18 months, several papers have shown that they can also deliver outstanding performance on more challenging visual classification tasks. Ciresan *et al.* [4] demonstrate state-of-the-art performance on NORB and CIFAR-10 datasets. Most notably, Krizhevsky *et al.* [18] show record beating performance on the ImageNet 2012 classification benchmark, with their convnet model achieving an error rate of 16.4%, compared to the 2nd place result of 26.1%. Following on from this work, Girshick *et al.* [10] have shown leading detection performance on the PASCAL VOC dataset. Several factors are responsible for this dramatic improvement in performance: (i) the availability of much larger training sets, with millions of labeled examples; (ii) powerful GPU implementations, making the training of very large models practical and (iii) better model regularization strategies, such as Dropout [14].

Despite this encouraging progress, there is still little insight into the internal operation and behavior of these complex models, or how they achieve such good performance. From a scientific standpoint, this is deeply unsatisfactory. Without clear understanding of how and why they work, the development of better models is reduced to trial-and-error. In this paper we introduce a visualization

D. Fleet et al. (Eds.): ECCV 2014, Part I, LNCS 8689, pp. 818–833, 2014.
© Springer International Publishing Switzerland 2014

technique that reveals the input stimuli that excite individual feature maps at any layer in the model. It also allows us to observe the evolution of features during training and to diagnose potential problems with the model. The visualization technique we propose uses a multi-layered Deconvolutional Network (deconvnet), as proposed by Zeiler *et al.* [29], to project the feature activations back to the input pixel space. We also perform a sensitivity analysis of the classifier output by occluding portions of the input image, revealing which parts of the scene are important for classification.

Using these tools, we start with the architecture of Krizhevsky *et al.* [18] and explore different architectures, discovering ones that outperform their results on ImageNet. We then explore the generalization ability of the model to other datasets, just retraining the softmax classifier on top. As such, this is a form of supervised pre-training, which contrasts with the unsupervised pre-training methods popularized by Hinton *et al.* [13] and others [1,26].

1.1 Related Work

Visualization: Visualizing features to gain intuition about the network is common practice, but mostly limited to the 1st layer where projections to pixel space are possible. In higher layers alternate methods must be used. [8] find the optimal stimulus for each unit by performing gradient descent in image space to maximize the unit's activation. This requires a careful initialization and does not give any information about the unit's invariances. Motivated by the latter's short-coming, [19] (extending an idea by [2]) show how the Hessian of a given unit may be computed numerically around the optimal response, giving some insight into invariances. The problem is that for higher layers, the invariances are extremely complex so are poorly captured by a simple quadratic approximation. Our approach, by contrast, provides a non-parametric view of invariance, showing which patterns from the training set activate the feature map. Our approach is similar to contemporary work by Simonyan *et al.* [23] who demonstrate how saliency maps can be obtained from a convnet by projecting back from the fully connected layers of the network, instead of the convolutional features that we use. Girshick *et al.* [10] show visualizations that identify patches within a dataset that are responsible for strong activations at higher layers in the model. Our visualizations differ in that they are not just crops of input images, but rather top-down projections that reveal structures within each patch that stimulate a particular feature map.

Feature Generalization: Our demonstration of the generalization ability of convnet features is also explored in concurrent work by Donahue *et al.* [7] and Girshick *et al.* [10]. They use the convnet features to obtain state-of-the-art performance on Caltech-101 and the Sun scenes dataset in the former case, and for object detection on the PASCAL VOC dataset, in the latter.

2 Approach

We use standard fully supervised convnet models throughout the paper, as defined by LeCun *et al.* [20] and Krizhevsky *et al.* [18]. These models map a color

Autonomous Ex. 2007 p. 2 of 16
Tesla v. Autonomous
IPR2024-00745

2D input image x_i, via a series of layers, to a probability vector $\hat{y}_i$ over the C different classes. Each layer consists of (i) convolution of the previous layer output (or, in the case of the 1st layer, the input image) with a set of learned filters; (ii) passing the responses through a rectified linear function ($relu(x) = \max(x, 0)$); (iii) [optionally] max pooling over local neighborhoods and (iv) [optionally] a local contrast operation that normalizes the responses across feature maps. For more details of these operations, see [18] and [16]. The top few layers of the network are conventional fully-connected networks and the final layer is a softmax classifier. Fig. 3 shows the model used in many of our experiments.

We train these models using a large set of N labeled images $\{x, y\}$, where label y_i is a discrete variable indicating the true class. A cross-entropy loss function, suitable for image classification, is used to compare $\hat{y}_i$ and y_i. The parameters of the network (filters in the convolutional layers, weight matrices in the fully-connected layers and biases) are trained by back-propagating the derivative of the loss with respect to the parameters throughout the network, and updating the parameters via stochastic gradient descent. Details of training are given in Section 3.

2.1 Visualization with a Deconvnet

Understanding the operation of a convnet requires interpreting the feature activity in intermediate layers. We present a novel way to *map these activities back to the input pixel space*, showing what input pattern originally caused a given activation in the feature maps. We perform this mapping with a Deconvolutional Network (deconvnet) Zeiler *et al.* [29]. A deconvnet can be thought of as a convnet model that uses the same components (filtering, pooling) but in reverse, so instead of mapping pixels to features does the opposite. In Zeiler *et al.* [29], deconvnets were proposed as a way of performing unsupervised learning. Here, they are not used in any learning capacity, just as a probe of an already trained convnet.

To examine a convnet, a deconvnet is attached to each of its layers, as illustrated in Fig. 1(top), providing a continuous path back to image pixels. To start, an input image is presented to the convnet and features computed throughout the layers. To examine a given convnet activation, we set all other activations in the layer to zero and pass the feature maps as input to the attached deconvnet layer. Then we successively (i) unpool, (ii) rectify and (iii) filter to reconstruct the activity in the layer beneath that gave rise to the chosen activation. This is then repeated until input pixel space is reached.

Unpooling: In the convnet, the max pooling operation is non-invertible, however we can obtain an approximate inverse by recording the locations of the maxima within each pooling region in a set of *switch* variables. In the deconvnet, the unpooling operation uses these switches to place the reconstructions from the layer above into appropriate locations, preserving the structure of the stimulus. See Fig. 1(bottom) for an illustration of the procedure.

Rectification: The convnet uses *relu* non-linearities, which rectify the feature maps thus ensuring the feature maps are always positive. To obtain valid

feature reconstructions at each layer (which also should be positive), we pass the reconstructed signal through a *relu* non-linearity[1].

Filtering: The convnet uses learned filters to convolve the feature maps from the previous layer. To approximately invert this, the deconvnet uses transposed versions of the same filters (as other autoencoder models, such as RBMs), but applied to the rectified maps, not the output of the layer beneath. In practice this means flipping each filter vertically and horizontally.

Note that we do not use any contrast normalization operations when in this reconstruction path. Projecting down from higher layers uses the switch settings generated by the max pooling in the convnet on the way up. As these switch settings are peculiar to a given input image, the reconstruction obtained from a single activation thus resembles a small piece of the original input image, with structures weighted according to their contribution toward to the feature activation. Since the model is trained discriminatively, they implicitly show which parts of the input image are discriminative. Note that these projections are *not* samples from the model, since there is no generative process involved. The whole procedure is similar to backpropping a single strong activation (rather than the usual gradients), i.e. computing $\frac{\partial h}{\partial X_n}$, where h is the element of the feature map with the strong activation and X_n is the input image. However, it differs in that (i) the the *relu* is imposed independently and (ii) contrast normalization operations are not used. A general shortcoming of our approach is that it only visualizes a single activation, not the joint activity present in a layer. Nevertheless, as we show in Fig. 6, these visualizations are accurate representations of the input pattern that stimulates the given feature map in the model: when the parts of the original input image corresponding to the pattern are occluded, we see a distinct drop in activity within the feature map.

3 Training Details

We now describe the large convnet model that will be visualized in Section 4. The architecture, shown in Fig. 3, is similar to that used by Krizhevsky *et al.* [18] for ImageNet classification. One difference is that the sparse connections used in Krizhevsky's layers 3,4,5 (due to the model being split across 2 GPUs) are replaced with dense connections in our model. Other important differences relating to layers 1 and 2 were made following inspection of the visualizations in Fig. 5, as described in Section 4.1.

The model was trained on the ImageNet 2012 training set (1.3 million images, spread over 1000 different classes) [6]. Each RGB image was preprocessed by resizing the smallest dimension to 256, cropping the center 256x256 region, subtracting the per-pixel mean (across all images) and then using 10 different sub-crops of size 224x224 (corners + center with(out) horizontal flips). Stochastic gradient descent with a mini-batch size of 128 was used to update the parameters, starting with a learning rate of 10^{-2}, in conjunction with a momentum term of 0.9. We

[1] We also tried rectifying using the binary mask imposed by the feed-forward *relu* operation, but the resulting visualizations were significantly less clear.

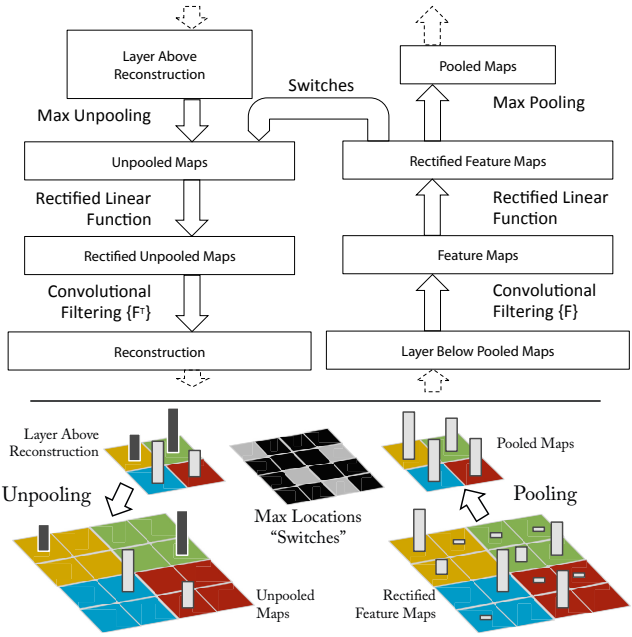

Fig. 1. Top: A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath. Bottom: An illustration of the unpooling operation in the deconvnet, using *switches* which record the location of the local max in each pooling region (colored zones) during pooling in the convnet. The black/white bars are negative/positive activations within the feature map.

anneal the learning rate throughout training manually when the validation error plateaus. Dropout [14] is used in the fully connected layers (6 and 7) with a rate of 0.5. All weights are initialized to 10^{-2} and biases are set to 0.

Visualization of the first layer filters during training reveals that a few of them dominate. To combat this, we renormalize each filter in the convolutional layers whose RMS value exceeds a fixed radius of 10^{-1} to this fixed radius. This is crucial, especially in the first layer of the model, where the input images are roughly in the [-128,128] range. As in Krizhevsky *et al.* [18], we produce multiple different crops and flips of each training example to boost training set size. We stopped training after 70 epochs, which took around 12 days on a single GTX580 GPU, using an implementation based on [18].

4 Convnet Visualization

Using the model described in Section 3, we now use the deconvnet to visualize the feature activations on the ImageNet validation set.

Feature Visualization: Fig. 2 shows feature visualizations from our model once training is complete. For a given feature map, we show the top 9 activations, each projected separately down to pixel space, revealing the different

structures that excite that map and showing its invariance to input deformations. Alongside these visualizations we show the corresponding image patches. These have greater variation than visualizations which solely focus on the discriminant structure within each patch. For example, in layer 5, row 1, col 2, the patches appear to have little in common, but the visualizations reveal that this particular feature map focuses on the grass in the background, not the foreground objects.

The projections from each layer show the hierarchical nature of the features in the network. Layer 2 responds to corners and other edge/color conjunctions. Layer 3 has more complex invariances, capturing similar textures (e.g. mesh patterns (Row 1, Col 1); text (R2,C4)). Layer 4 shows significant variation, and is more class-specific: dog faces (R1,C1); bird's legs (R4,C2). Layer 5 shows entire objects with significant pose variation, e.g. keyboards (R1,C11) and dogs (R4).

Feature Evolution during Training: Fig. 4 visualizes the progression during training of the strongest activation (across all training examples) within a given feature map projected back to pixel space. Sudden jumps in appearance result from a change in the image from which the strongest activation originates. The lower layers of the model can be seen to converge within a few epochs. However, the upper layers only develop develop after a considerable number of epochs (40-50), demonstrating the need to let the models train until fully converged.

4.1 Architecture Selection

While visualization of a trained model gives insight into its operation, it can also assist with selecting good architectures in the first place. By visualizing the first and second layers of Krizhevsky *et al.* 's architecture (Fig. 5(a) & (c)), various problems are apparent. The first layer filters are a mix of extremely high and low frequency information, with little coverage of the mid frequencies. Additionally, the 2nd layer visualization shows aliasing artifacts caused by the large stride 4 used in the 1st layer convolutions. To remedy these problems, we (i) reduced the 1st layer filter size from 11x11 to 7x7 and (ii) made the stride of the convolution 2, rather than 4. This new architecture retains much more information in the 1st and 2nd layer features, as shown in Fig. 5(b) & (d). More importantly, it also improves the classification performance as shown in Section 5.1.

4.2 Occlusion Sensitivity

With image classification approaches, a natural question is if the model is truly identifying the location of the object in the image, or just using the surrounding context. Fig. 6 attempts to answer this question by systematically occluding different portions of the input image with a grey square, and monitoring the output of the classifier. The examples clearly show the model is localizing the objects within the scene, as the probability of the correct class drops significantly when the object is occluded. Fig. 6 also shows visualizations from the strongest feature map of the top convolution layer, in addition to activity in this map (summed over spatial locations) as a function of occluder position. When the

Fig. 2. Visualization of features in a fully trained model. For layers 2-5 we show the top 9 activations in a random subset of feature maps across the validation data, projected down to pixel space using our deconvolutional network approach. Our reconstructions are *not* samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii) exaggeration of discriminative parts of the image, e.g. eyes and noses of dogs (layer 4, row 1, cols 1). Best viewed in electronic form. The compression artifacts are a consequence of the 30Mb submission limit, not the reconstruction algorithm itself.

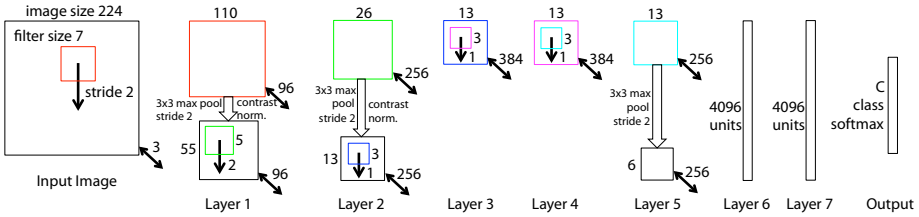

Fig. 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

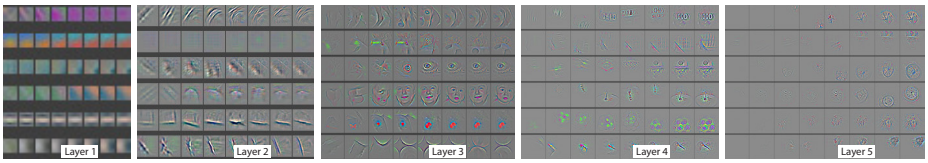

Fig. 4. Evolution of a randomly chosen subset of model features through training. Each layer's features are displayed in a different block. Within each block, we show a randomly chosen subset of features at epochs [1,2,5,10,20,30,40,64]. The visualization shows the strongest activation (across all training examples) for a given feature map, projected down to pixel space using our deconvnet approach. Color contrast is artificially enhanced and the figure is best viewed in electronic form.

occluder covers the image region that appears in the visualization, we see a strong drop in activity in the feature map. This shows that the visualization genuinely corresponds to the image structure that stimulates that feature map, hence validating the other visualizations shown in Fig. 4 and Fig. 2.

5 Experiments

5.1 ImageNet 2012

This dataset consists of 1.3M/50k/100k training/validation/test examples, spread over 1000 categories. Table 1 shows our results on this dataset.

Using the exact architecture specified in Krizhevsky *et al.* [18], we attempt to replicate their result on the validation set. We achieve an error rate within 0.1% of their reported value on the ImageNet 2012 validation set.

Next we analyze the performance of our model with the architectural changes outlined in Section 4.1 (7×7 filters in layer 1 and stride 2 convolutions in layers

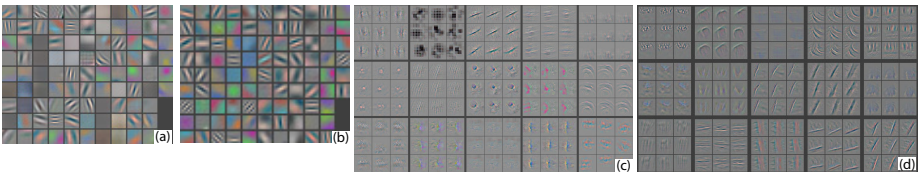

Fig. 5. (a): 1st layer features without feature scale clipping. Note that one feature dominates. (b): 1st layer features from Krizhevsky *et al.* [18]. (c): Our 1st layer features. The smaller stride (2 vs 4) and filter size (7x7 vs 11x11) results in more distinctive features and fewer "dead" features. (d): Visualizations of 2nd layer features from Krizhevsky *et al.* [18]. (e): Visualizations of our 2nd layer features. These are cleaner, with no aliasing artifacts that are visible in (d).

1 & 2). This model, shown in Fig. 3, significantly outperforms the architecture of Krizhevsky *et al.* [18], beating their single model result by 1.7% (test top-5). When we combine multiple models, we obtain a test error of 14.8%, an improvement of 1.6%. This result is close to that produced by the data-augmentation approaches of Howard [15], which could easily be combined with our architecture. However, our model is some way short of the winner of the 2013 Imagenet classification competition [28].

Table 1. ImageNet 2012/2013 classification error rates. The $*$ indicates models that were trained on both ImageNet 2011 and 2012 training sets.

Error %	Val Top-1	Val Top-5	Test Top-5
Gunji *et al.* [12]	-	-	26.2
DeCAF [7]	-	-	19.2
Krizhevsky *et al.* [18], 1 convnet	40.7	18.2	−−
Krizhevsky *et al.* [18], 5 convnets	38.1	16.4	16.4
Krizhevsky *et al.* *[18], 1 convnets	39.0	16.6	−−
Krizhevsky *et al.* *[18], 7 convnets	36.7	15.4	15.3
Our replication of Krizhevsky *et al.* , 1 convnet	40.5	18.1	−−
1 convnet as per Fig. 3	38.4	16.5	−−
5 convnets as per Fig. 3 – (a)	36.7	15.3	15.3
1 convnet as per Fig. 3 but with layers 3,4,5: 512,1024,512 maps – (b)	37.5	16.0	16.1
6 convnets, (a) & (b) combined	36.0	14.7	14.8
Howard [15]	-	-	13.5
Clarifai [28]	-	-	11.7

Varying ImageNet Model Sizes: In Table 2, we first explore the architecture of Krizhevsky *et al.* [18] by adjusting the size of layers, or removing them entirely. In each case, the model is trained from scratch with the revised architecture. Removing the fully connected layers (6,7) only gives a slight increase in error (in

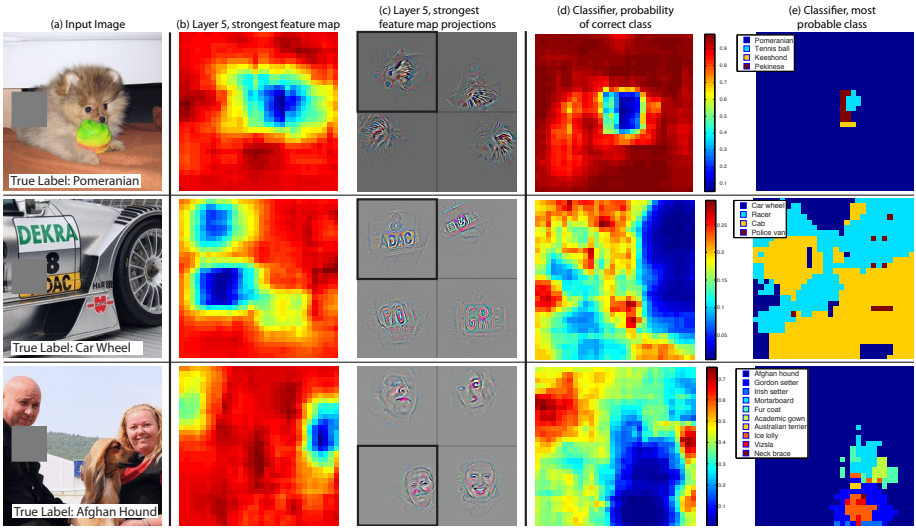

Fig. 6. Three test examples where we systematically cover up different portions of the scene with a gray square (1st column) and see how the top (layer 5) feature maps ((b) & (c)) and classifier output ((d) & (e)) changes. (b): for each position of the gray scale, we record the total activation in one layer 5 feature map (the one with the strongest response in the unoccluded image). (c): a visualization of this feature map projected down into the input image (black square), along with visualizations of this map from other images. The first row example shows the strongest feature to be the dog's face. When this is covered-up the activity in the feature map decreases (blue area in (b)). (d): a map of correct class probability, as a function of the position of the gray square. E.g. when the dog's face is obscured, the probability for "pomeranian" drops significantly. (e): the most probable label as a function of occluder position. E.g. in the 1st row, for most locations it is "pomeranian", but if the dog's face is obscured but not the ball, then it predicts "tennis ball". In the 2nd example, text on the car is the strongest feature in layer 5, but the classifier is most sensitive to the wheel. The 3rd example contains multiple objects. The strongest feature in layer 5 picks out the faces, but the classifier is sensitive to the dog (blue region in (d)), since it uses multiple feature maps.

the following, we refer to top-5 validation error). This is surprising, given that they contain the majority of model parameters. Removing two of the middle convolutional layers also makes a relatively small difference to the error rate. However, removing both the middle convolution layers and the fully connected layers yields a model with only 4 layers whose performance is dramatically worse. This would suggest that the overall depth of the model is important for obtaining good performance. We then modify our model, shown in Fig. 3. Changing the size of the fully connected layers makes little difference to performance (same for model of Krizhevsky *et al.* [18]). However, increasing the size of the middle convolution layers goes give a useful gain in performance. But increasing these, while also enlarging the fully connected layers results in over-fitting.

Table 2. ImageNet 2012 classification error rates with various architectural changes to the model of Krizhevsky *et al.* [18] and our model (see Fig. 3)

Error %	Train Top-1	Val Top-1	Val Top-5
Our replication of Krizhevsky *et al.* [18], 1 convnet	35.1	40.5	18.1
Removed layers 3,4	41.8	45.4	22.1
Removed layer 7	27.4	40.0	18.4
Removed layers 6,7	27.4	44.8	22.4
Removed layer 3,4,6,7	71.1	71.3	50.1
Adjust layers 6,7: 2048 units	40.3	41.7	18.8
Adjust layers 6,7: 8192 units	26.8	40.0	18.1
Our Model (as per Fig. 3)	33.1	38.4	16.5
Adjust layers 6,7: 2048 units	38.2	40.2	17.6
Adjust layers 6,7: 8192 units	22.0	38.8	17.0
Adjust layers 3,4,5: 512,1024,512 maps	18.8	**37.5**	**16.0**
Adjust layers 6,7: 8192 units and Layers 3,4,5: 512,1024,512 maps	**10.0**	38.3	16.9

5.2 Feature Generalization

The experiments above show the importance of the convolutional part of our ImageNet model in obtaining state-of-the-art performance. This is supported by the visualizations of Fig. 2 which show the complex invariances learned in the convolutional layers. We now explore the ability of these feature extraction layers to generalize to other datasets, namely Caltech-101 [9], Caltech-256 [11] and PASCAL VOC 2012. To do this, we keep layers 1-7 of our ImageNet-trained model fixed and train a new softmax classifier on top (for the appropriate number of classes) using the training images of the new dataset. Since the softmax contains relatively few parameters, it can be trained quickly from a relatively small number of examples, as is the case for certain datasets.

The experiments compare our feature representation, obtained from ImageNet, with the hand-crafted features used by other methods. In both our approach and existing ones the Caltech/PASCAL training data is only used to train the classifier. As they are of similar complexity (ours: softmax, others: linear SVM), the feature representation is crucial to performance. It is important to note that both representations were built using images beyond the Caltech and PASCAL training sets. For example, the hyper-parameters in HOG descriptors were determined through systematic experiments on a pedestrian dataset [5].

We also try a second strategy of training a model from scratch, i.e. resetting layers 1-7 to random values and train them, as well as the softmax, on the training images of the PASCAL/Caltech dataset.

One complication is that some of the Caltech datasets have some images that are also in the ImageNet training data. Using normalized correlation, we

identified these few "overlap" images[2] and removed them from our Imagenet training set and then retrained our Imagenet models, so avoiding the possibility of train/test contamination.

Caltech-101: We follow the procedure of [9] and randomly select 15 or 30 images per class for training and test on up to 50 images per class reporting the average of the per-class accuracies in Table 3, using 5 train/test folds. Training took 17 minutes for 30 images/class. The pre-trained model beats the best reported result for 30 images/class from [3] by 2.2%. Our result agrees with the recently published result of Donahue *et al.* [7], who obtain 86.1% accuracy (30 imgs/class). The convnet model trained from scratch however does terribly, only achieving 46.5%, showing the impossibility of training a large convnet on such a small dataset.

Table 3. Caltech-101 classification accuracy for our convnet models, against two leading alternate approaches

# Train	Acc % 15/class	Acc % 30/class
Bo *et al.* [3]	–	81.4 ± 0.33
Yang *et al.* [17]	73.2	84.3
Non-pretrained convnet	22.8 ± 1.5	46.5 ± 1.7
ImageNet-pretrained convnet	$\mathbf{83.8 \pm 0.5}$	$\mathbf{86.5 \pm 0.5}$

Caltech-256: We follow the procedure of [11], selecting 15, 30, 45, or 60 training images per class, reporting the average of the per-class accuracies in Table 4. Our ImageNet-pretrained model beats the current state-of-the-art results obtained by Bo *et al.* [3] by a significant margin: 74.2% vs 55.2% for 60 training images/class. However, as with Caltech-101, the model trained from scratch does poorly. In Fig. 7, we explore the "one-shot learning" [9] regime. With our pretrained model, just 6 Caltech-256 training images are needed to beat the leading method using 10 times as many images. This shows the power of the ImageNet feature extractor.

PASCAL 2012: We used the standard training and validation images to train a 20-way softmax on top of the ImageNet-pretrained convnet. This is not ideal, as PASCAL images can contain multiple objects and our model just provides a single exclusive prediction for each image. Table 5 shows the results on the test set, comparing to the leading methods: the top 2 entries in the competition and concurrent work from Oquab *et al.* [21] who use a convnet with a more appropriate classifier. The PASCAL and ImageNet images are quite different in nature, the former being full scenes unlike the latter. This may explain our mean

[2] For Caltech-101, we found 44 images in common (out of 9,144 total images), with a maximum overlap of 10 for any given class. For Caltech-256, we found 243 images in common (out of 30,607 total images), with a maximum overlap of 18 for any given class.

Table 4. Caltech 256 classification accuracies

# Train	Acc % 15/class	Acc % 30/class	Acc % 45/class	Acc % 60/class
Sohn *et al.* [24]	35.1	42.1	45.7	47.9
Bo *et al.* [3]	40.5 ± 0.4	48.0 ± 0.2	51.9 ± 0.2	55.2 ± 0.3
Non-pretr.	9.0 ± 1.4	22.5 ± 0.7	31.2 ± 0.5	38.8 ± 1.4
ImageNet-pretr.	$\mathbf{65.7 \pm 0.2}$	$\mathbf{70.6 \pm 0.2}$	$\mathbf{72.7 \pm 0.4}$	$\mathbf{74.2 \pm 0.3}$

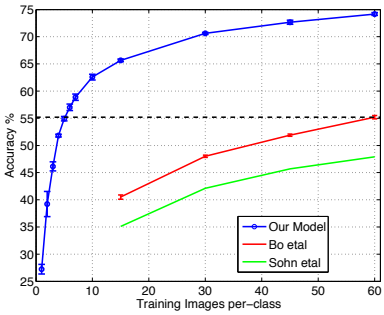

Fig. 7. Caltech-256 classification performance as the number of training images per class is varied. Using only 6 training examples per class with our pre-trained feature extractor, we surpass best reported result by Bo *et al.* [3].

Table 5. PASCAL 2012 classification results, comparing our Imagenet-pretrained convnet against the leading two methods and the recent approach of Oquab *et al.* [21]

Acc %	[22]	[27]	[21]	Ours	Acc %	[22]	[27]	[21]	Ours
Airplane	92.0	**97.3**	94.6	96.0	Dining table	63.2	**77.8**	69.0	67.7
Bicycle	74.2	**84.2**	82.9	77.1	Dog	68.9	83.0	**92.1**	87.8
Bird	73.0	80.8	88.2	**88.4**	Horse	78.2	87.5	**93.4**	86.0
Boat	77.5	85.3	60.3	**85.5**	Motorbike	81.0	**90.1**	88.6	85.1
Bottle	54.3	**60.8**	60.3	55.8	Person	91.6	95.0	**96.1**	90.9
Bus	85.2	**89.9**	89.0	85.8	Potted plant	55.9	57.8	**64.3**	52.2
Car	81.9	**86.8**	84.4	78.6	Sheep	69.4	79.2	**86.6**	83.6
Cat	76.4	89.3	90.7	**91.2**	Sofa	65.4	**73.4**	62.3	61.1
Chair	65.2	**75.4**	72.1	65.0	Train	86.7	**94.5**	91.1	91.8
Cow	63.2	77.8	**86.8**	74.4	Tv	77.4	**80.7**	79.8	76.1
Mean	74.3	82.2	**82.8**	79.0	# won	0	**11**	6	3

performance being 3.2% lower than the leading competition result [27], however we do beat them on 5 classes, sometimes by large margins.

5.3 Feature Analysis

We explore how discriminative the features in each layer of our Imagenet-pretrained model are. We do this by varying the number of layers retained from the ImageNet model and place either a linear SVM or softmax classifier on top. Table 6 shows results on Caltech-101 and Caltech-256. For both datasets, a steady improvement can be seen as we ascend the model, with best results being obtained by using all layers. This supports the premise that as the feature hierarchies become deeper, they learn increasingly powerful features.

Table 6. Analysis of the discriminative information contained in each layer of feature maps within our ImageNet-pretrained convnet. We train either a linear SVM or softmax on features from different layers (as indicated in brackets) from the convnet. Higher layers generally produce more discriminative features.

	Cal-101 (30/class)	Cal-256 (60/class)
SVM (1)	44.8 ± 0.7	24.6 ± 0.4
SVM (2)	66.2 ± 0.5	39.6 ± 0.3
SVM (3)	72.3 ± 0.4	46.0 ± 0.3
SVM (4)	76.6 ± 0.4	51.3 ± 0.1
SVM (5)	$\mathbf{86.2 \pm 0.8}$	65.6 ± 0.3
SVM (7)	$\mathbf{85.5 \pm 0.4}$	$\mathbf{71.7 \pm 0.2}$
Softmax (5)	82.9 ± 0.4	65.7 ± 0.5
Softmax (7)	$\mathbf{85.4 \pm 0.4}$	$\mathbf{72.6 \pm 0.1}$

6 Discussion

We explored large convolutional neural network models, trained for image classification, in a number ways. First, we presented a novel way to visualize the activity within the model. This reveals the features to be far from random, uninterpretable patterns. Rather, they show many intuitively desirable properties such as compositionality, increasing invariance and class discrimination as we ascend the layers. We also show how these visualization can be used to identify problems with the model and so obtain better results, for example improving on Krizhevsky *et al.* 's [18] impressive ImageNet 2012 result. We then demonstrated through a series of occlusion experiments that the model, while trained for classification, is highly sensitive to local structure in the image and is not just using broad scene context. An ablation study on the model revealed that having a minimum depth to the network, rather than any individual section, is vital to the model's performance.

Finally, we showed how the ImageNet trained model can generalize well to other datasets. For Caltech-101 and Caltech-256, the datasets are similar enough that we can beat the best reported results, in the latter case by a significant margin. Our convnet model generalized less well to the PASCAL data, perhaps

suffering from dataset bias [25], although it was still within 3.2% of the best reported result, despite no tuning for the task. For example, our performance might improve if a different loss function was used that permitted multiple objects per image. This would naturally enable the networks to tackle the object detection as well.

Acknowledgments. The authors would like to thank Yann LeCun for helpful discussions and acknowledge support from NSERC, NSF grant #1116923 and Microsoft Research.

References

1. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: NIPS, pp. 153–160 (2007)
2. Berkes, P., Wiskott, L.: On the analysis and interpretation of inhomogeneous quadratic forms as receptive fields. Neural Computation (2006)
3. Bo, L., Ren, X., Fox, D.: Multipath sparse coding using hierarchical matching pursuit. In: CVPR (2013)
4. Ciresan, D.C., Meier, J., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: CVPR (2012)
5. Dalal, N., Triggs, B.: Histograms of oriented gradients for pedestrian detection. In: CVPR (2005)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR 2009 (2009)
7. 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:1310.1531 (2013)
8. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. Technical report, University of Montreal (2009)
9. Fei-fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Trans. PAMI (2006)
10. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv:1311.2524 (2014)
11. Griffin, G., Holub, A., Perona, P.: The caltech 256. Caltech Technical Report (2006)
12. Gunji, N., Higuchi, T., Yasumoto, K., Muraoka, H., Ushiku, Y., Harada, T., Kuniyoshi, Y.: Classification entry. Imagenet Competition (2012)
13. Hinton, G.E., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. Neural Computation 18, 1527–1554 (2006)
14. Hinton, G.E., Srivastave, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. In: arXiv:1207.0580 (2012)
15. Howard, A.G.: Some improvements on deep convolutional neural network based image classification. arXiv 1312.5402 (2013)
16. Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: ICCV (2009)
17. Jianchao, Y., Kai, Y., Yihong, G., Thomas, H.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR (2009)

18. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
19. Le, Q.V., Ngiam, J., Chen, Z., Chia, D., Koh, P., Ng, A.Y.: Tiled convolutional neural networks. In: NIPS (2010)
20. 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 Comput. 1(4), 541–551 (1989)
21. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: CVPR (2014)
22. Sande, K., Uijlings, J., Snoek, C., Smeulders, A.: Hybrid coding for selective search. In: PASCAL VOC Classification Challenge 2012 (2012)
23. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv 1312.6034v1 (2013)
24. Sohn, K., Jung, D., Lee, H., Hero III, A.: Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In: ICCV (2011)
25. Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: CVPR (2011)
26. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: ICML, pp. 1096–1103 (2008)
27. Yan, S., Dong, J., Chen, Q., Song, Z., Pan, Y., Xia, W., Huang, Z., Hua, Y., Shen, S.: Generalized hierarchical matching for sub-category aware object classification. In: PASCAL VOC Classification Challenge 2012 (2012)
28. Zeiler, M.: Clarifai (2013), `http://www.image-net.org/challenges/LSVRC/2013/results.php`
29. Zeiler, M., Taylor, G., Fergus, R.: Adaptive deconvolutional networks for mid and high level feature learning. In: ICCV (2011)