Visualizing and Understanding Convolutional Networks

©Kuang

Graduate School of Information, Production and Systems
早稲田大学　大学院情報生産システム研究科

Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." In *European conference on computer vision*, pp. 818-833. Springer, Cham, 2014.

Contents

Introduction

Several factors are responsible for the dramatic improvement in image processing 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
(iii) better model regularization strategies, such as Dropout.

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.

Introduction

In this paper we introduce a visualization technique

- reveals the input stimuli that excite individual feature maps at any layer in the model.
- allows 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. [1], 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.

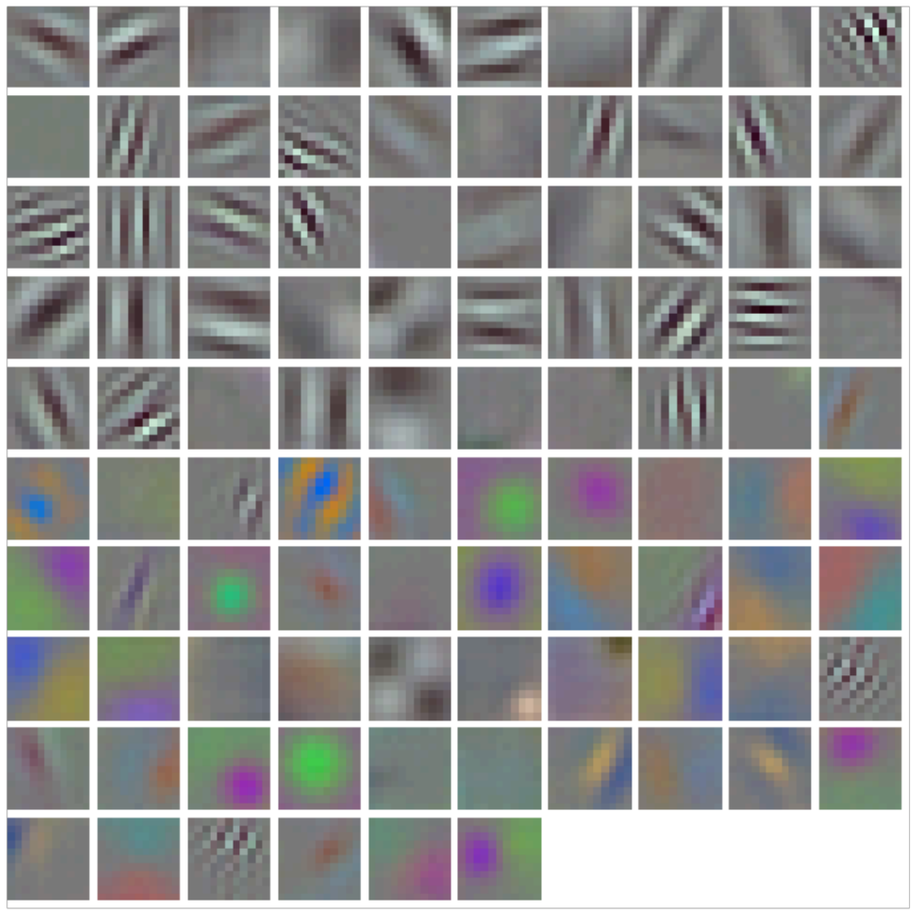
---

[1] Zeiler, M., Taylor, G., Fergus, R.: Adaptive deconvolutional networks for mid and high level feature learning. In: ICCV (2011)

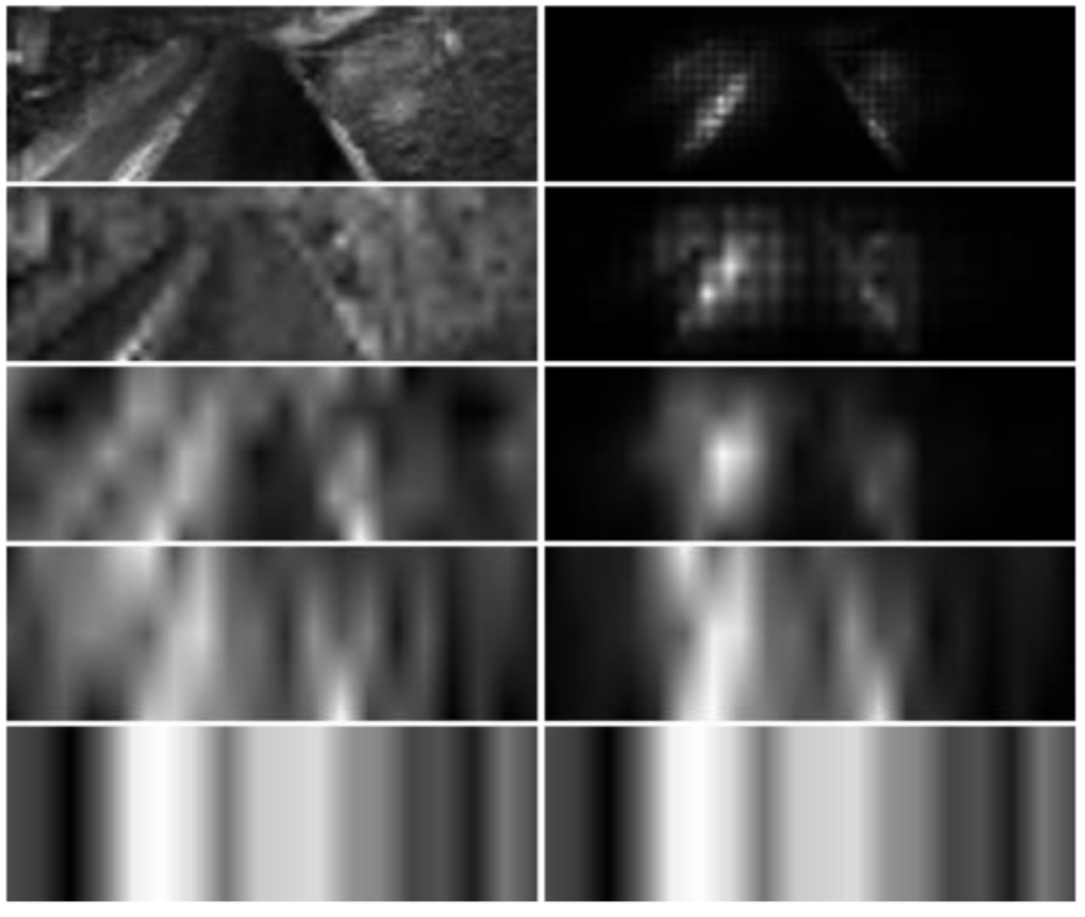
Introduction

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.



[1]



[2]

---

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

[2] Bojarski, Mariusz, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. "Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car." *arXiv preprint arXiv: 1704.07911* (2017).

Approach

We use standard fully supervised convnet models throughout the paper, as defined by LeCun et al. [1] and Krizhevsky et al. [2]. These models map a color input image *xi*, via a series of layers, to a probability vector *yi* 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 inp 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.

---

[1] 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)
[2] Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)

Approach

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.

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.

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.

Approach

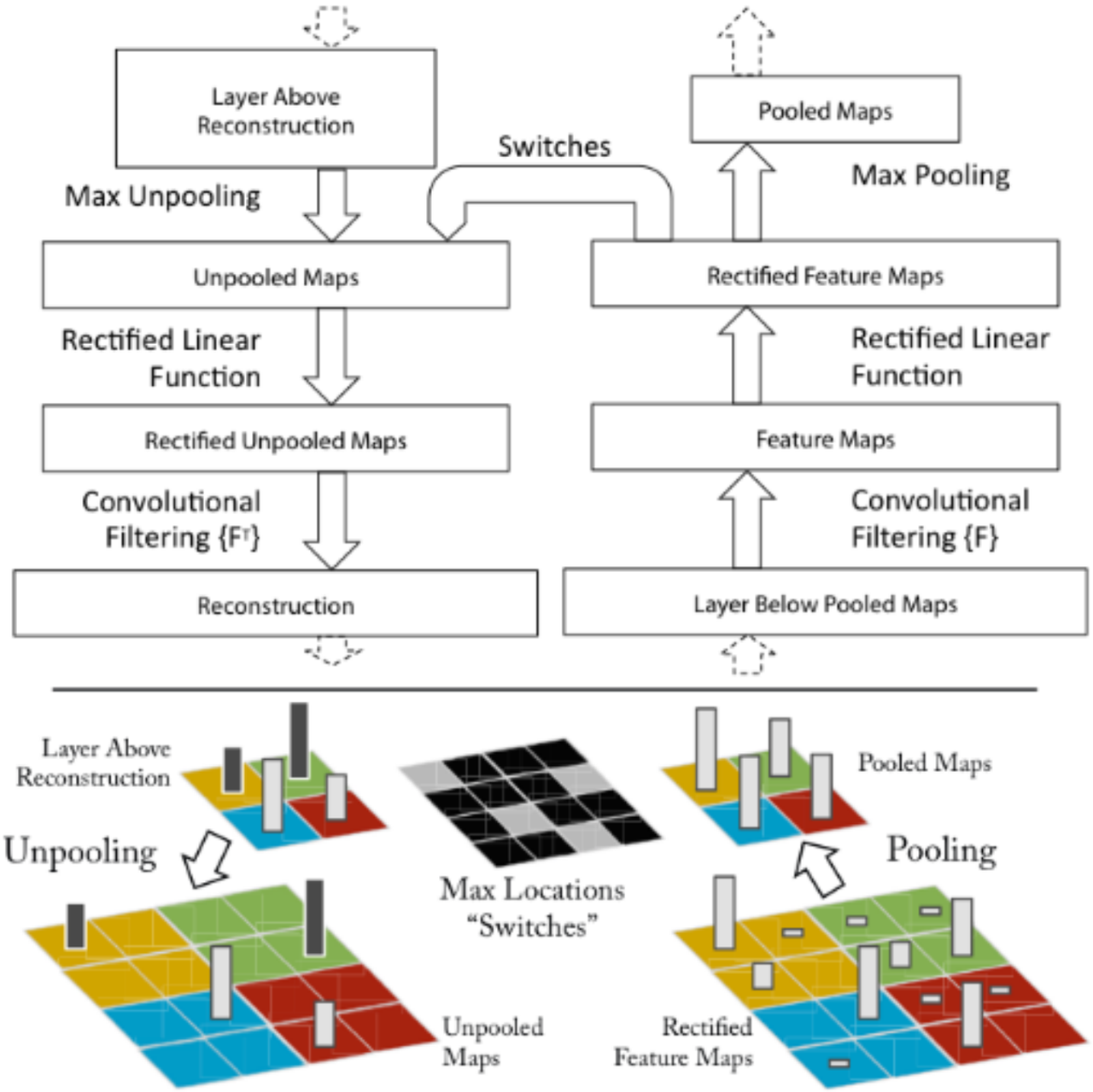


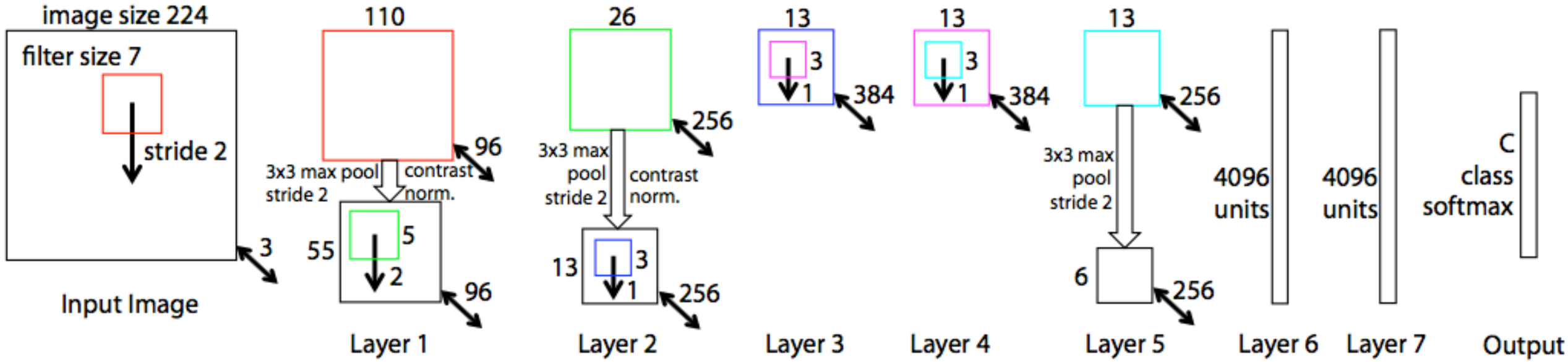
Fig. 1. Top: A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath. Bottom: An illustration of the unpooling operation in the deconvnet, using *switches* which record the location of the local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

Training Details

The architecture, shown in figure below, is similar to that used by [1] 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.

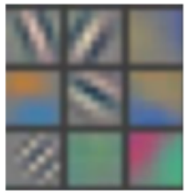
We stopped training after 70 epochs, which took around 12 days on a single GTX580 GPU, using an implementation based on [1].



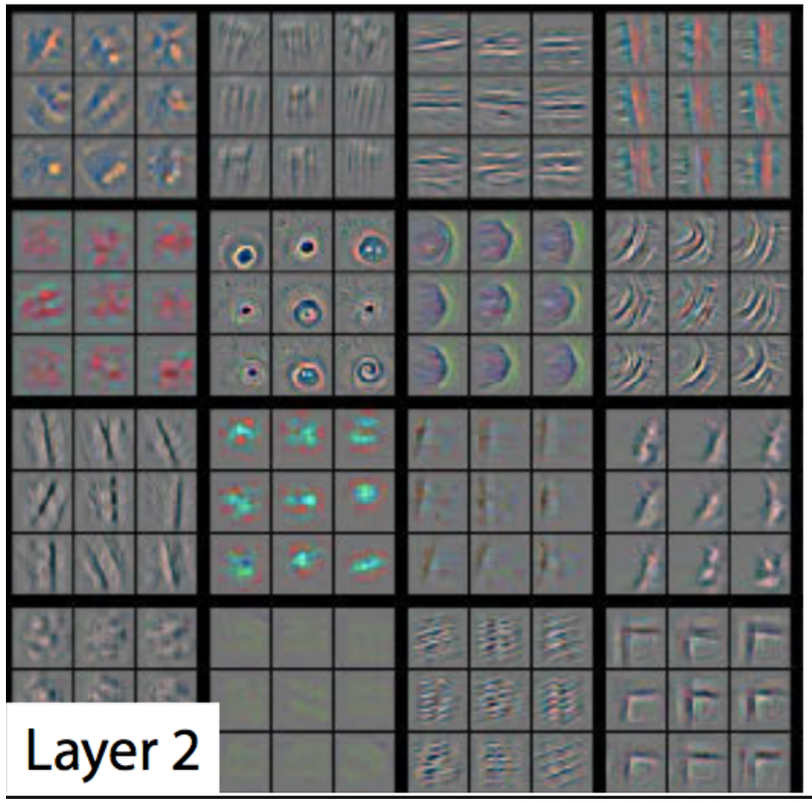
[1] Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)

Convnet Visualization

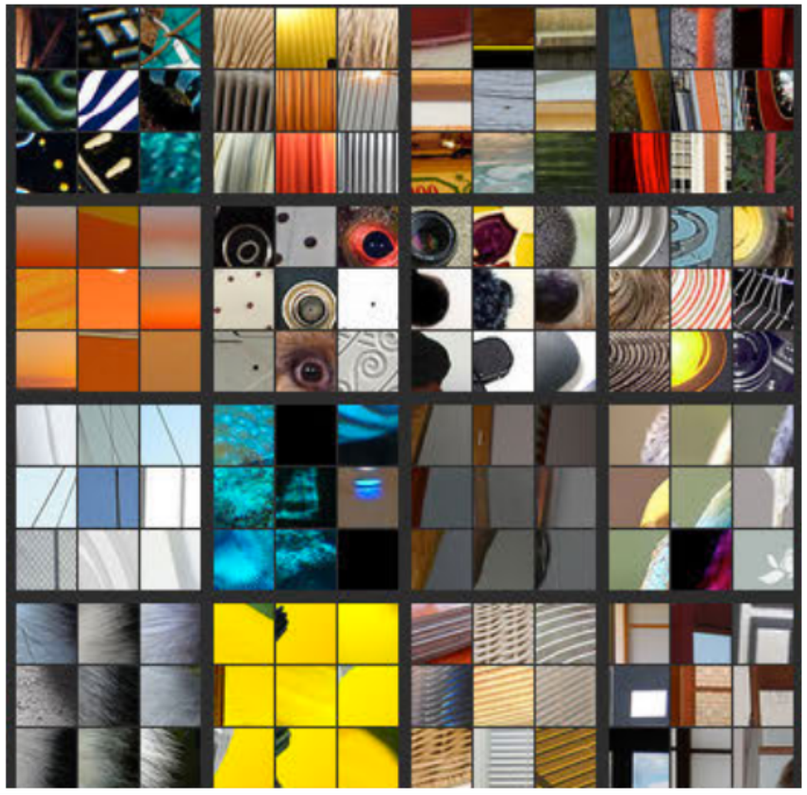
1.Feature Visualization:

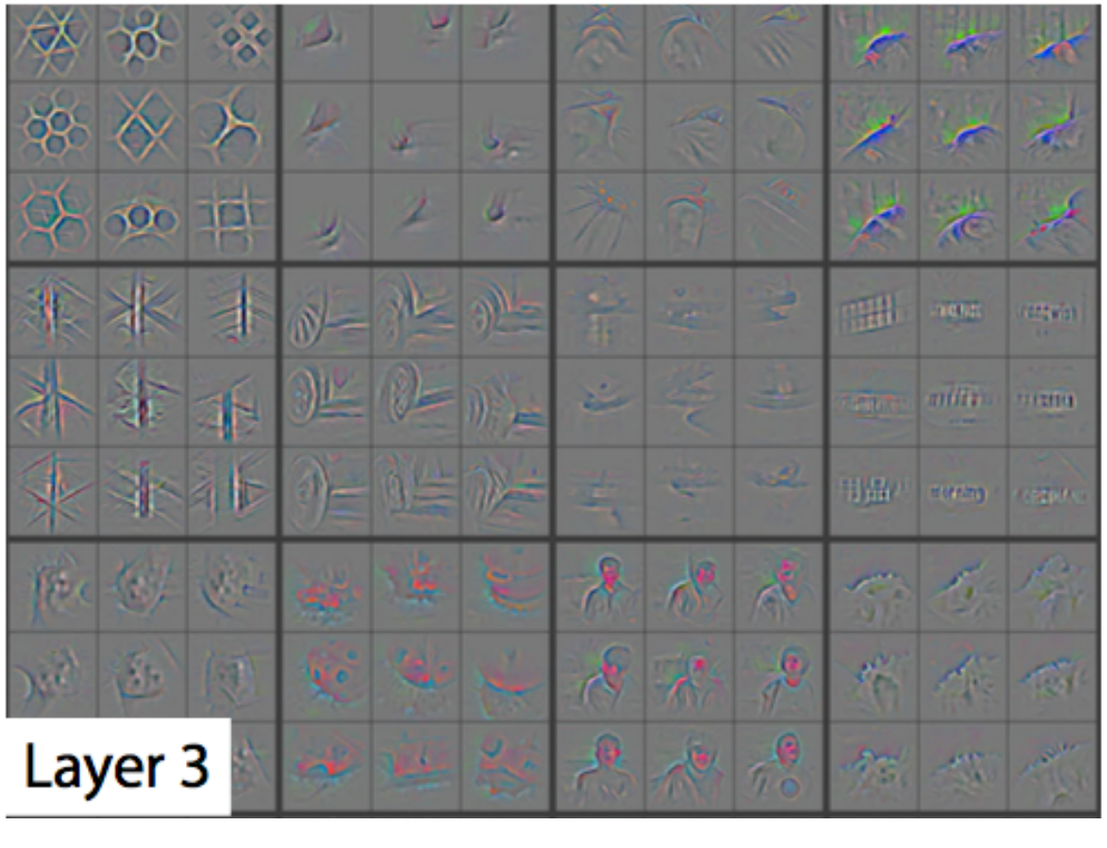


Layer 1

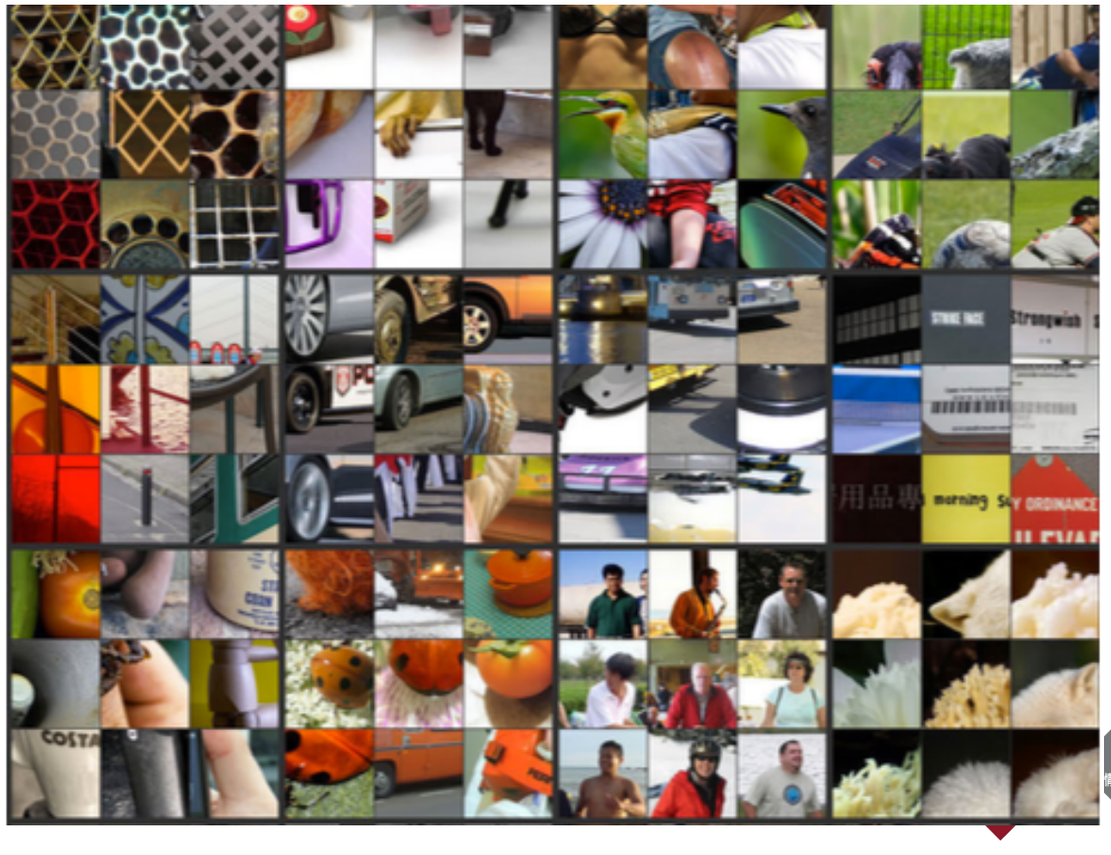


Layer 2



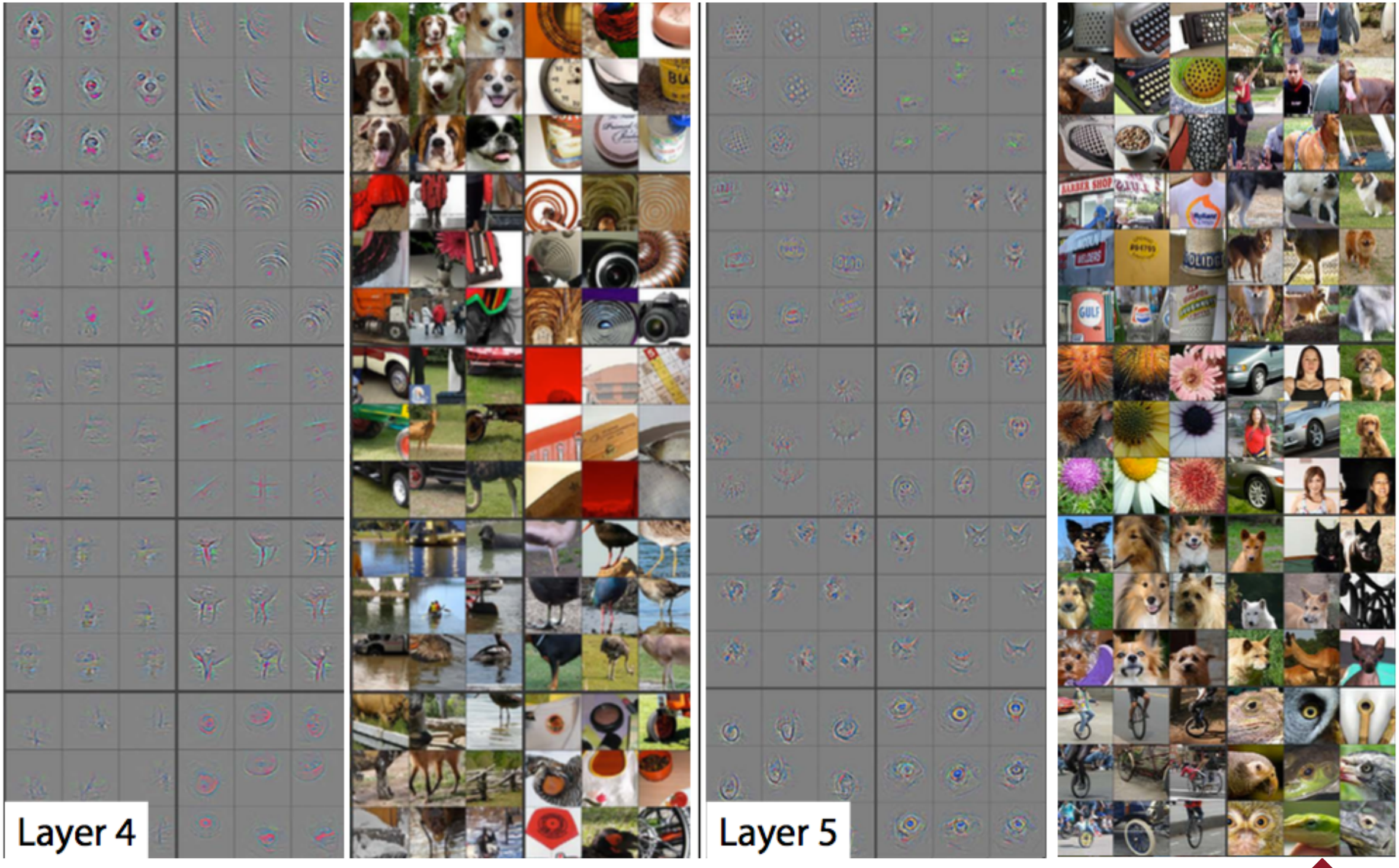


Layer 3



Convnet Visualization

1.Feature Visualization:



Convnet Visualization

2.Feature Evolution during Training:

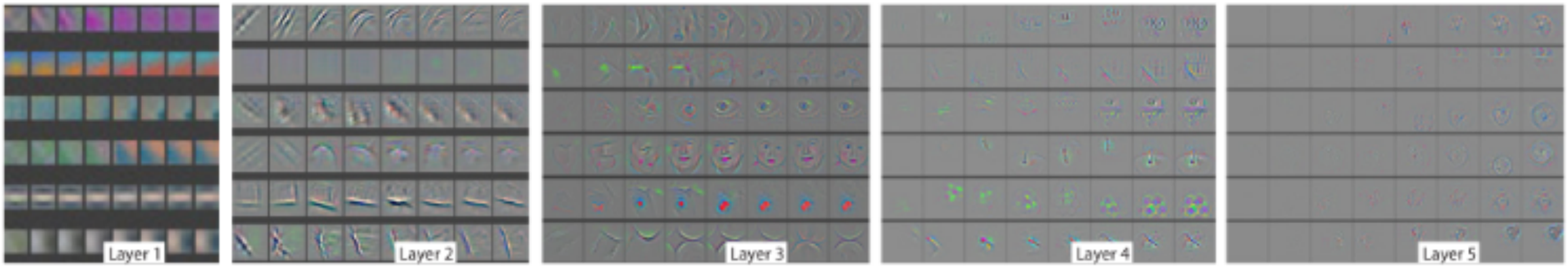


Fig. 4. Evolution of a randomly chosen subset of model features through training. Each layer's features are displayed in a different block. Within each block, we show a randomly chosen subset of features at epochs [1,2,5,10,20,30,40,64]. The visualization shows the strongest activation (across all training 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.

Convnet Visualization

3.Feature Invariance:

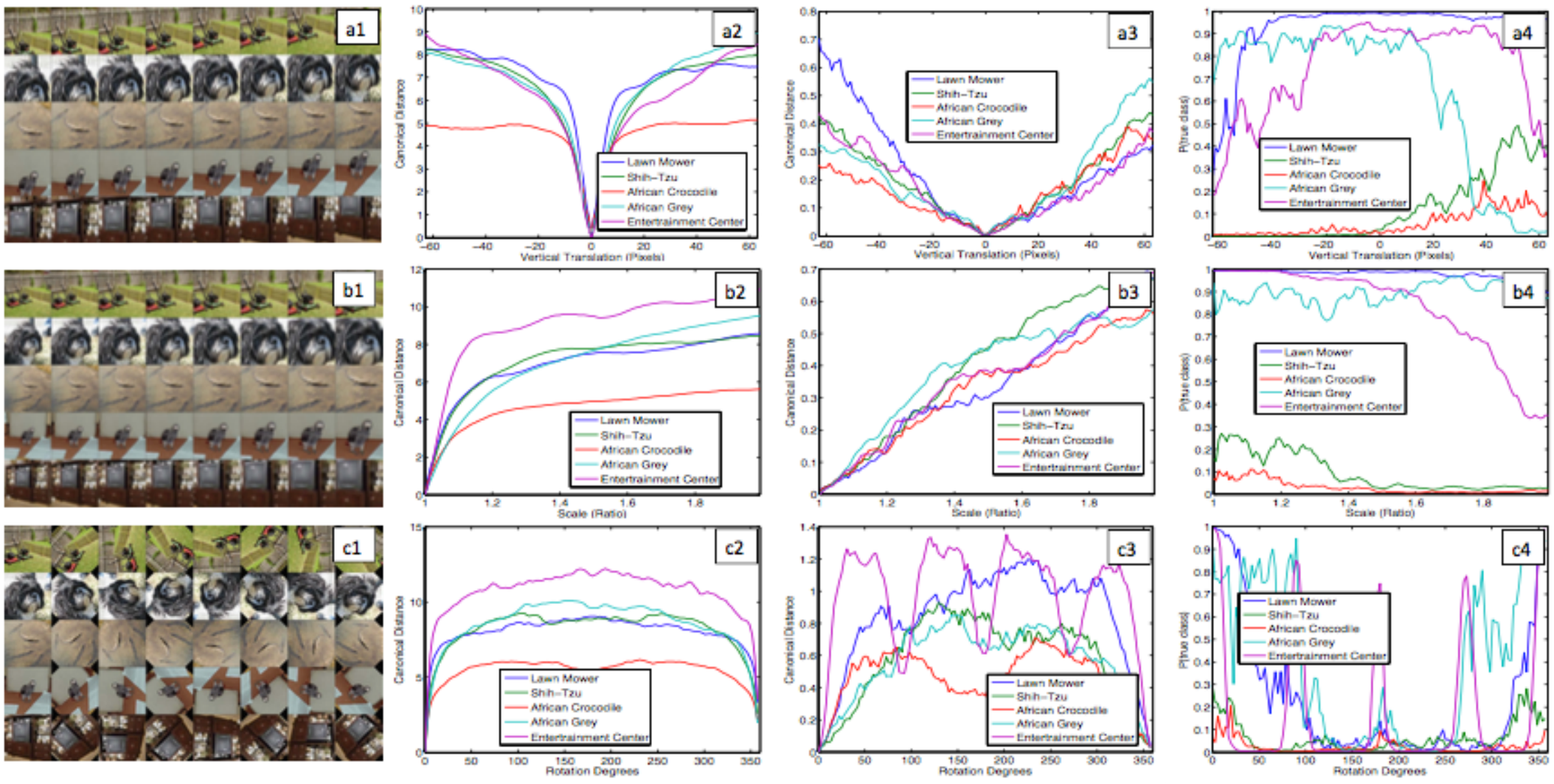


Figure 5. Analysis of vertical translation, scale, and rotation invariance within the model (rows a-c respectively). Col 1: 5 example images undergoing the transformations. Col 2 & 3: Euclidean distance between feature vectors from the original and transformed images in layers 1 and 7 respectively. Col 4: the probability of the true label for each image, as the image is transformed.

Related Work

4.1. Architecture Selection

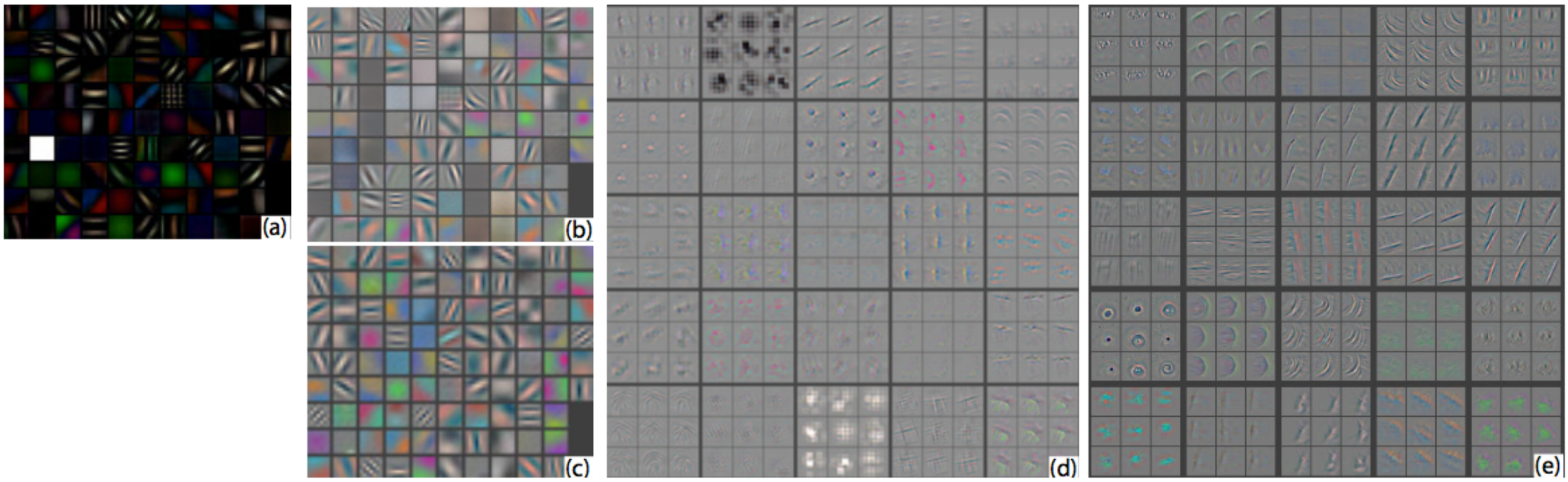
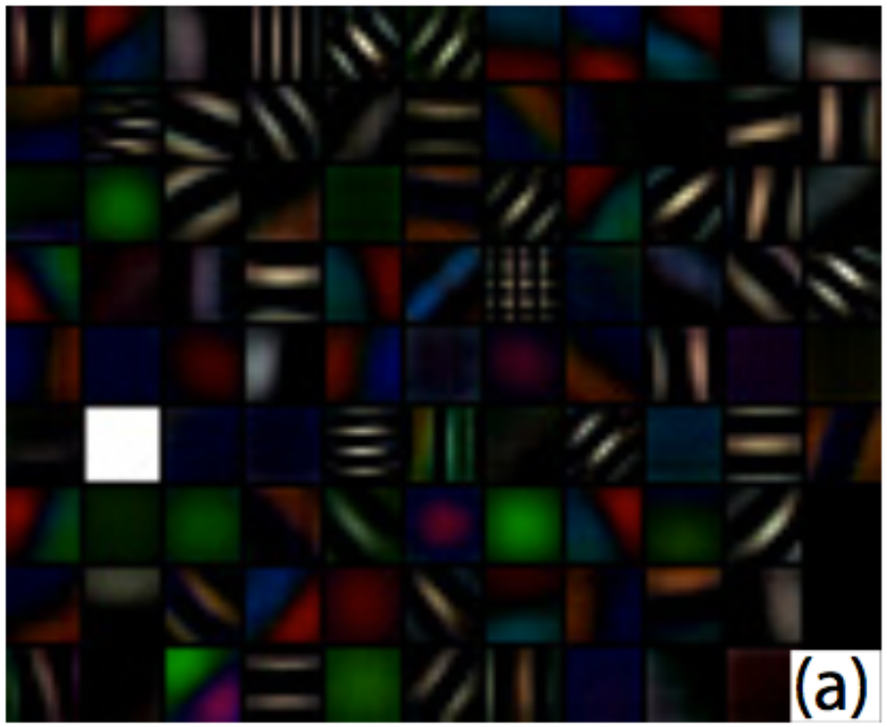
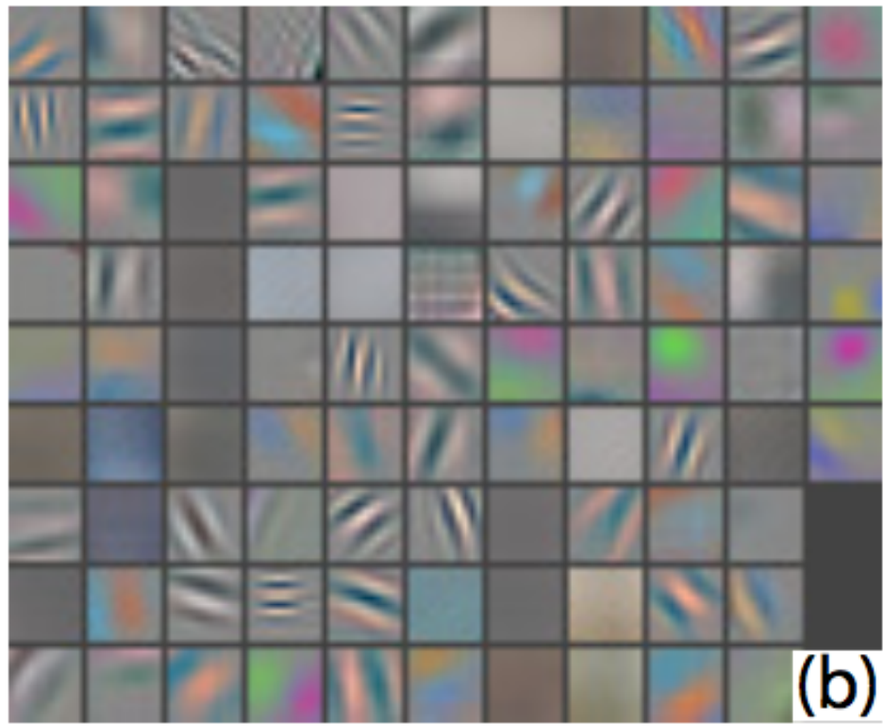


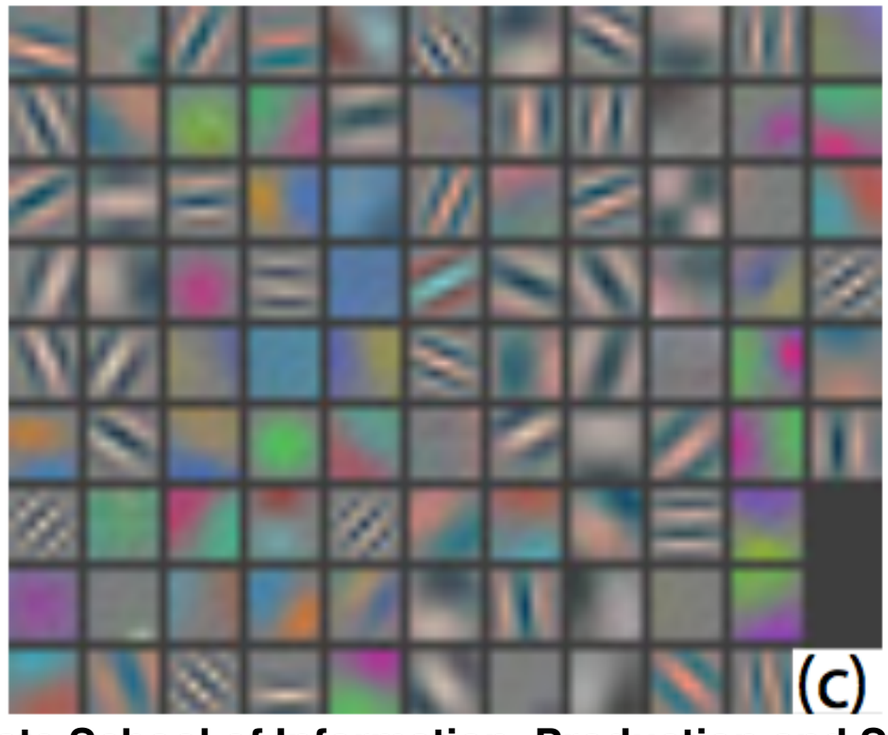
Figure 6. (a): 1st layer features without feature scale clipping. Note that one feature dominates. (b): 1st layer features from (Krizhevsky et al., 2012). (c): Our 1st layer features. The smaller stride (2 vs 4) and filter size (7x7 vs 11x11) results in more distinctive features and fewer "dead" features. (d): Visualizations of 2nd layer features from (Krizhevsky et al., 2012). (e): Visualizations of our 2nd layer features. These are cleaner, with no aliasing artifacts that are visible in (d).

Related Work

4.1. Architecture Selection

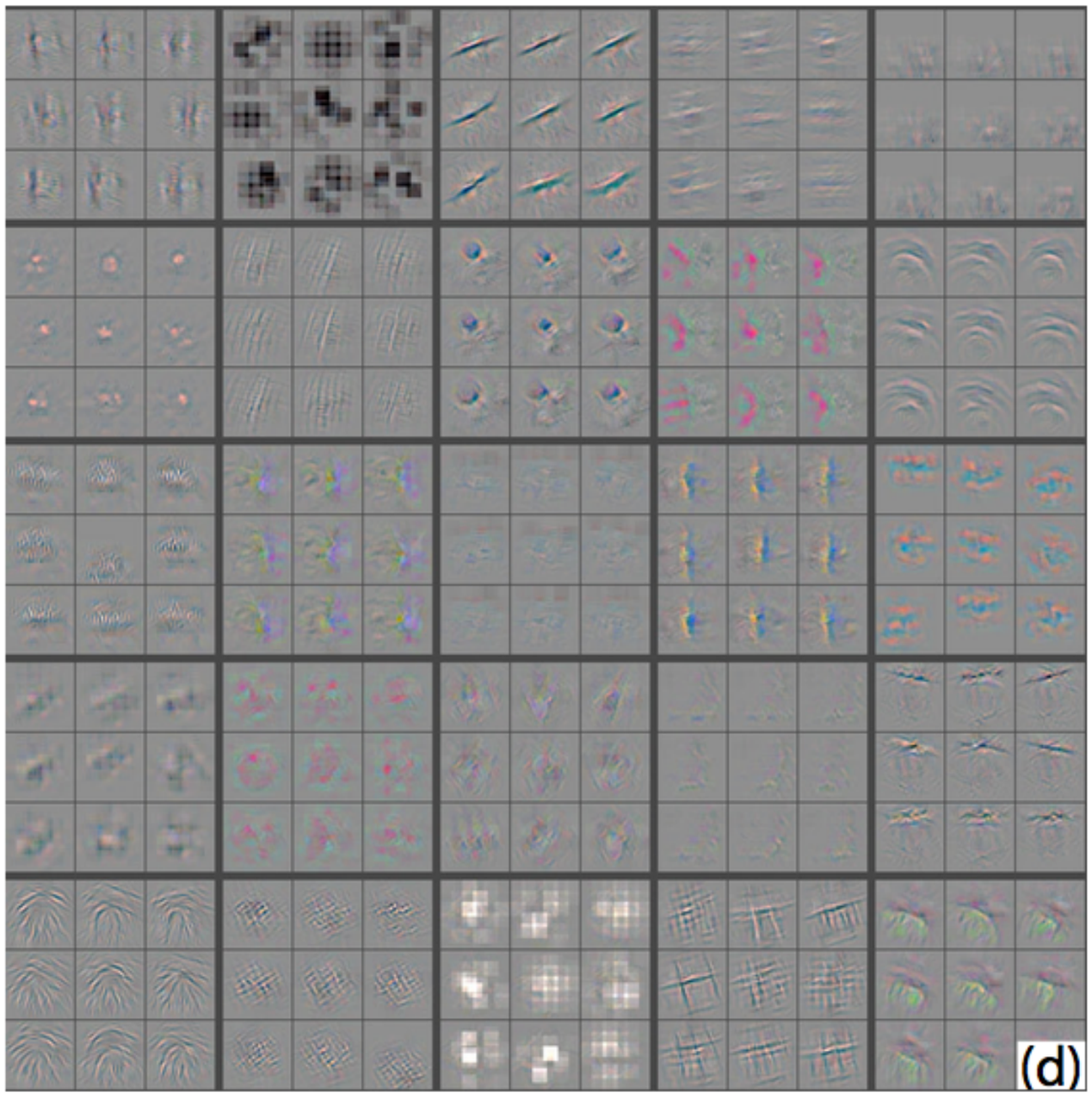


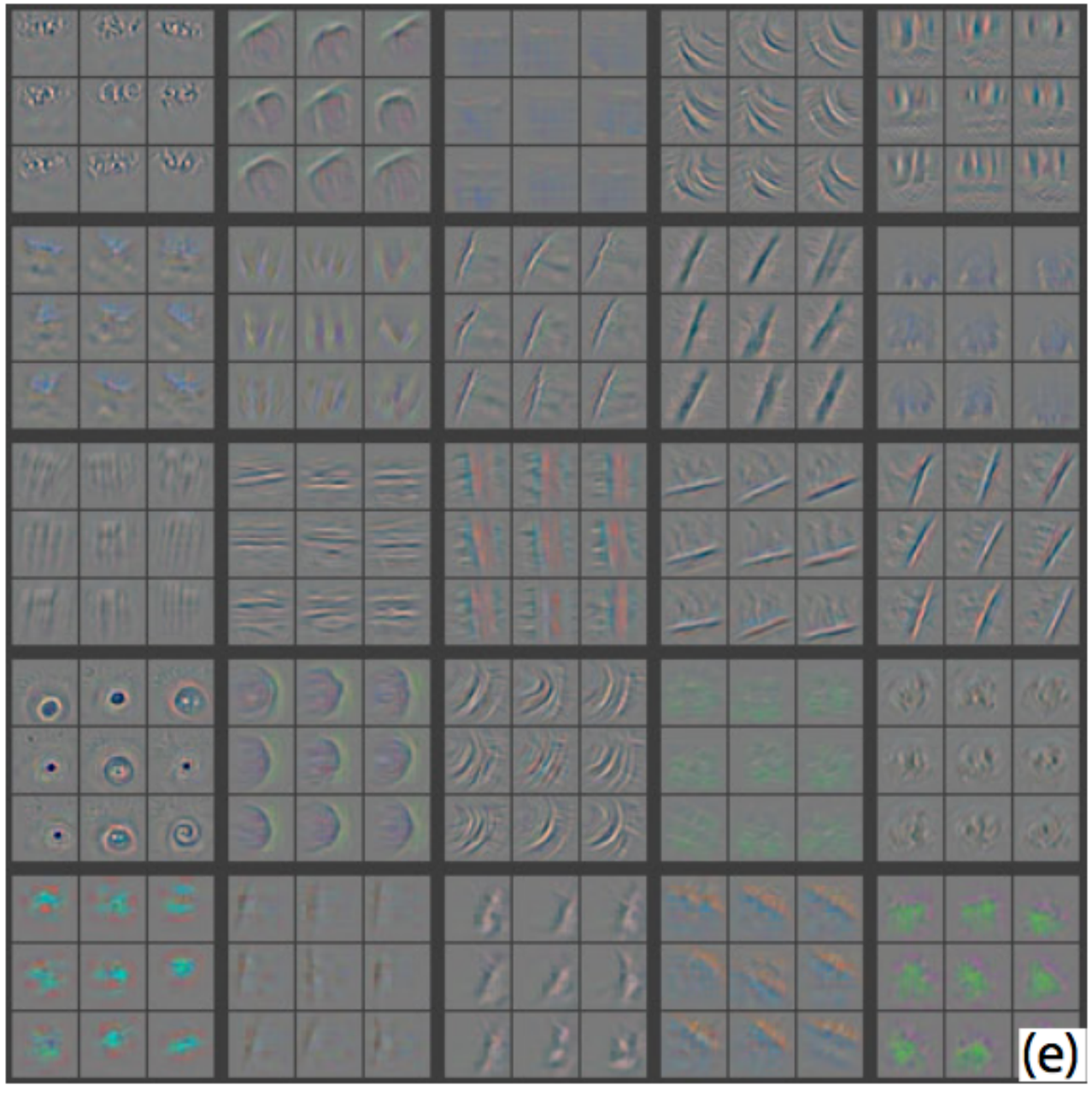




Related Work

4.1. Architecture Selection





Proposed Method

4.2 Occlusion Sensitivity

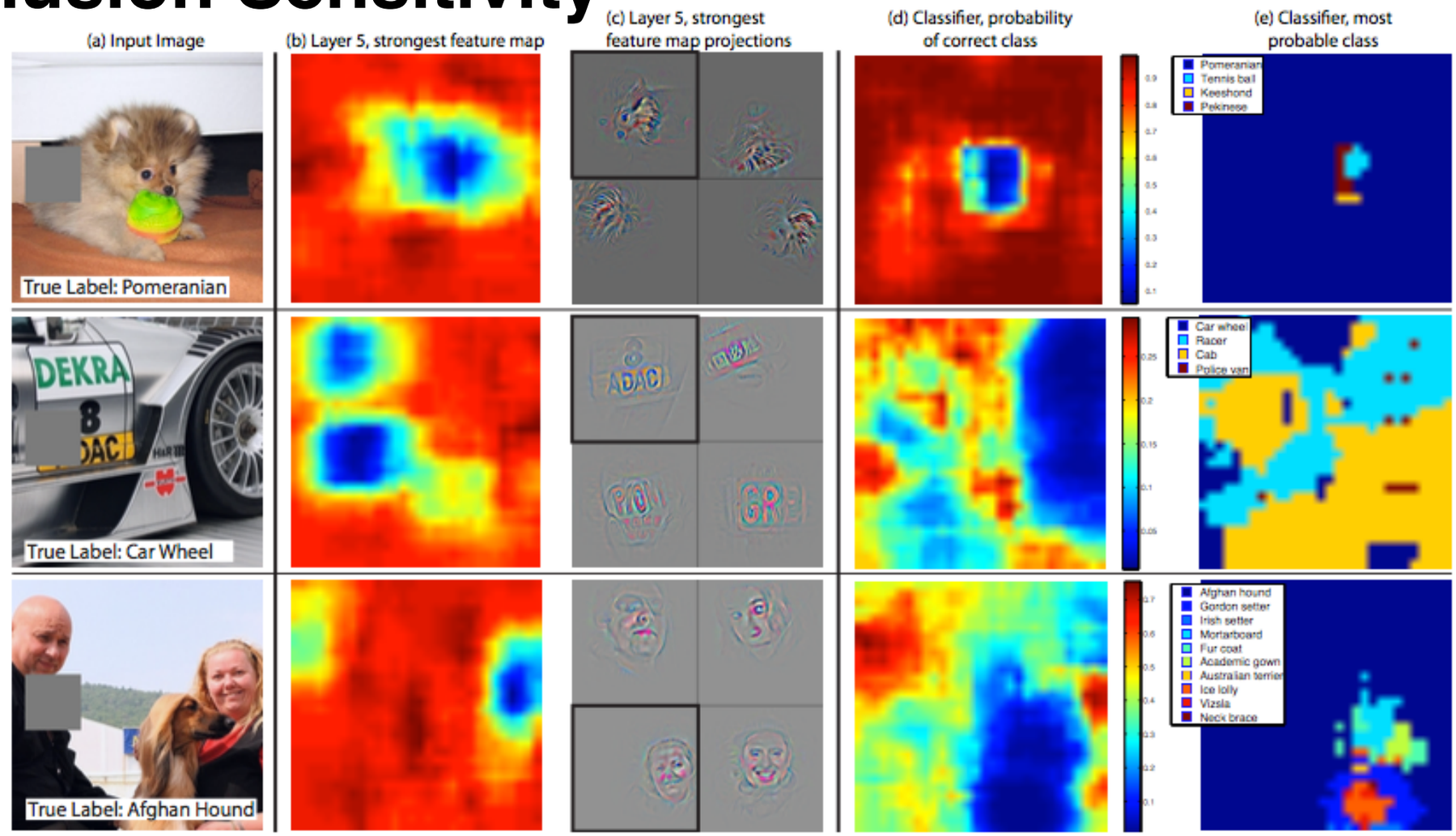


Figure 7. Three test examples where we systematically cover up different portions of the scene with a gray square (1st column) and see how the top (layer 5) feature maps ((b) & (c)) and classifier output ((d) & (e)) changes. (b): for each position of the gray scale, we record the total activation in 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.

Proposed Method

4.3 Correspondence Analysis

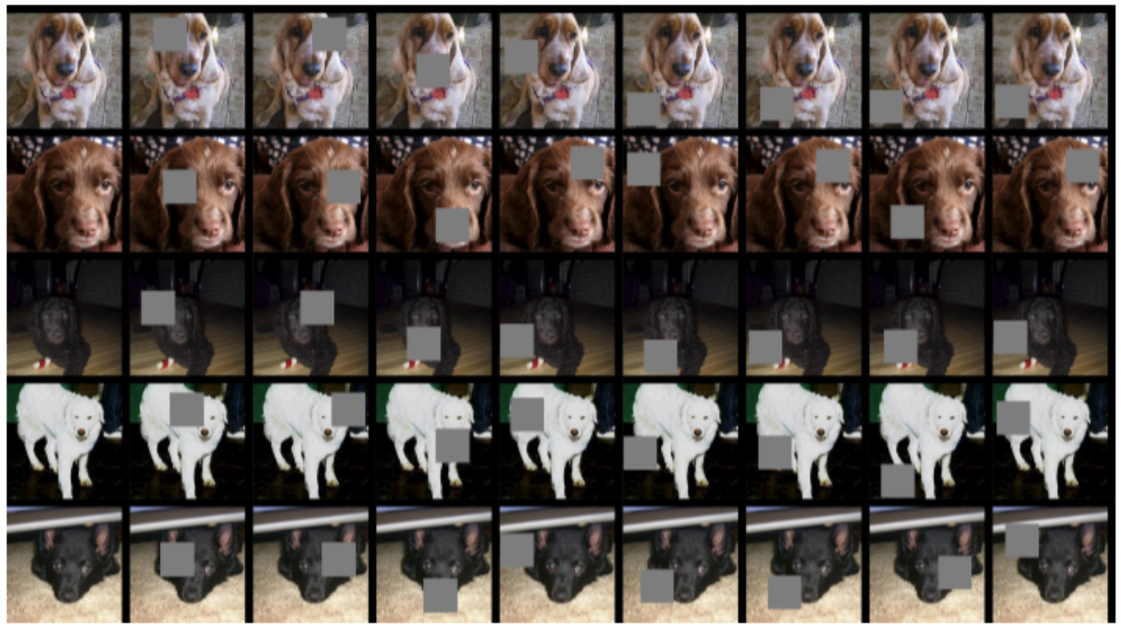


Figure 8. Images used for correspondence experiments. Col 1: Original image. Col 2,3,4: Occlusion of the right eye, left eye, and nose respectively. Other columns show examples of random occlusions.

Occlusion Location	Mean Feature Sign Change Layer 5	Mean Feature Sign Change Layer 7
Right Eye	0.067 ± 0.007	0.069 ± 0.015
Left Eye	0.069 ± 0.007	0.068 ± 0.013
Nose	0.079 ± 0.017	0.069 ± 0.011
Random	0.107 ± 0.017	0.073 ± 0.014

Table 1. Measure of correspondence for different object parts in 5 different dog images. The lower scores for the eyes and nose (compared to random object parts) show the model implicitly establishing some form of correspondence of parts at layer 5 in the model. At layer 7, the scores are more similar, perhaps due to upper layers trying to discriminate between the different breeds of dog.

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 showed how these visualization can be used to debug problems with the model to obtain better results, for example improving on Krizhevsky et al. ’s (Krizhevsky et al., 2012) 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.