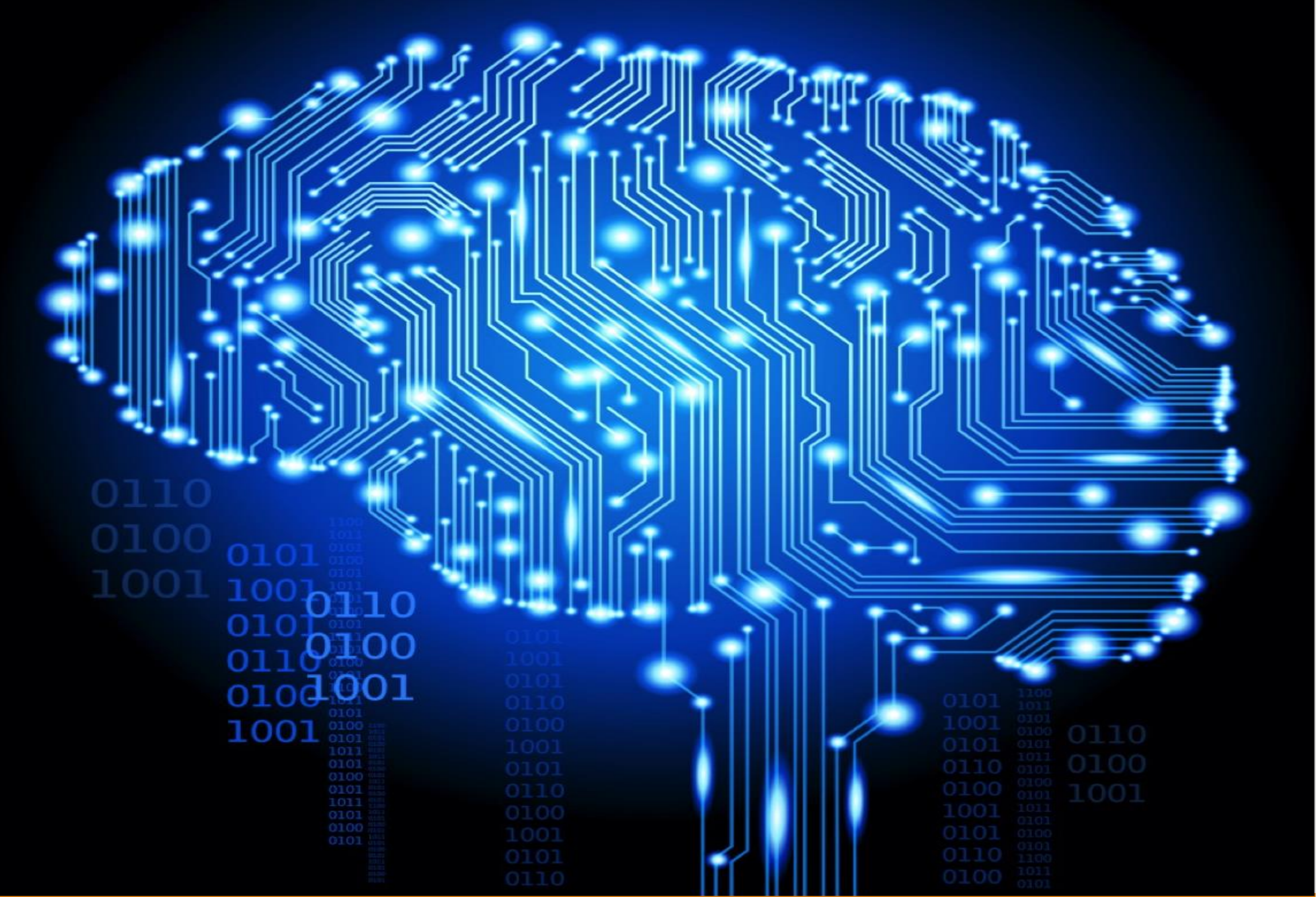


Lecture 5: Understanding Convnets and Knowledge Transfer

Deep Learning @ UvA

Previous Lecture

- What are the Convolutional Neural Networks?
- Why are they important in Computer Vision?
- Differences from standard Neural Networks
- How to train a Convolutional Neural Network?

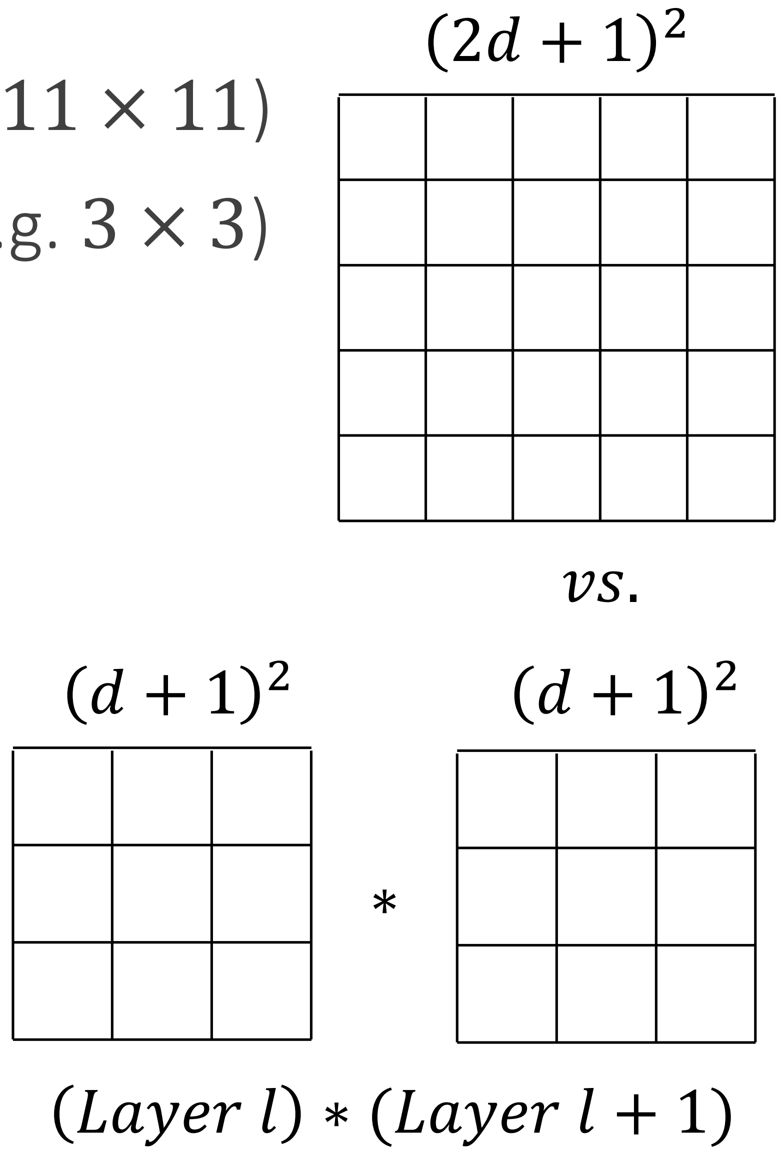
Lecture Overview

- What do convolutions look like?
- Build on the visual intuition behind Convnets
- Deep Learning Feature maps
- Transfer Learning

Understanding convnets

How large filters?

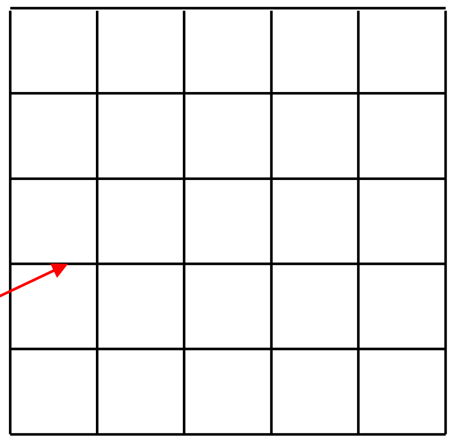
- Traditionally, medium sized filters (smaller than 11×11)
- Modern architectures prefer small filter sizes (e.g. 3×3)
- We lose frequency resolution
- Fewer parameters to train
- Deeper networks of cascade filters
  - Still, the same output dimensionalities



How large filters?

- Traditionally, medium sized filters (smaller than 11×11)
- Modern architectures prefer small filter sizes (e.g. 3×3)
- We lose frequency resolution
- Fewer parameters to train
- Deeper networks of cascade filters
  - Still, the same output dimensionalities

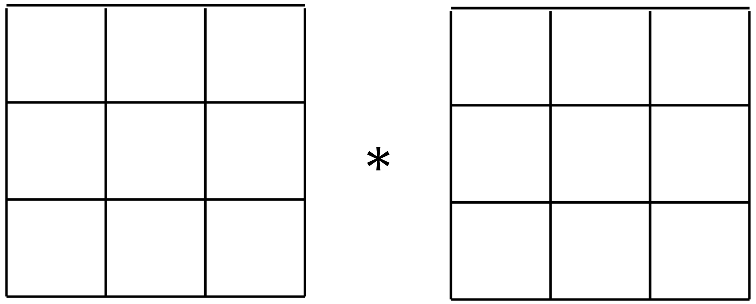
$(2d+1)^2$



For stride 1 the first feature map has dimensionality $\frac{H-2d-1}{1}+1=H-2d$

$vs.$

$(d+1)^2$ $\quad$ $(d+1)^2$

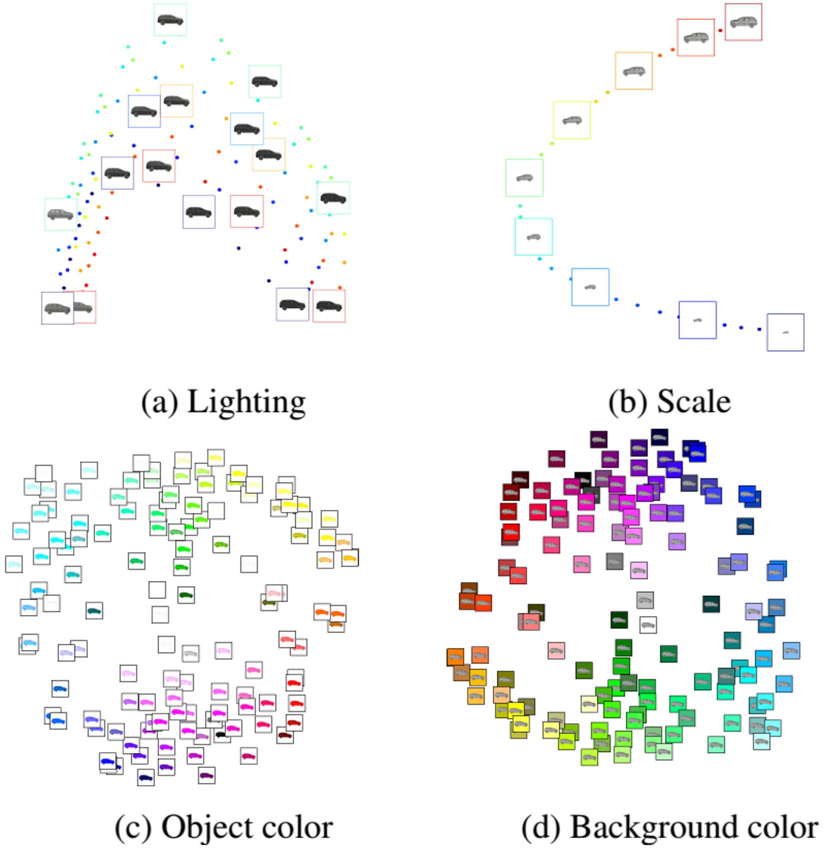


For stride 1 the first feature map has dimensionality $H-d$, the second image $\frac{H-d-d-1}{1}+1=H-2d$

$(Layer\ l) * (Layer\ l+1)$

Filter invariance and equivariance

- Filters learn how different variances affect appearance
- Different layers and different hierarchies focus on different transformations
- For different objects filters reproduce different behaviors



Aubry et al., Understanding deep features with computer-generated imagery , 2015]

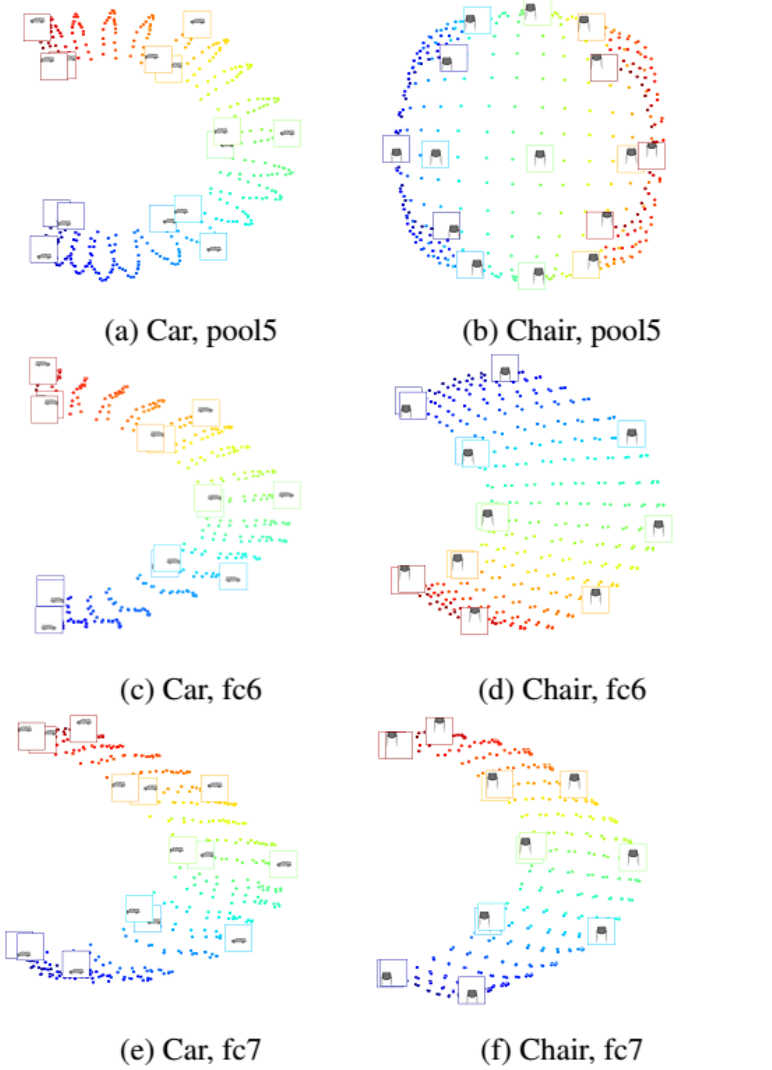
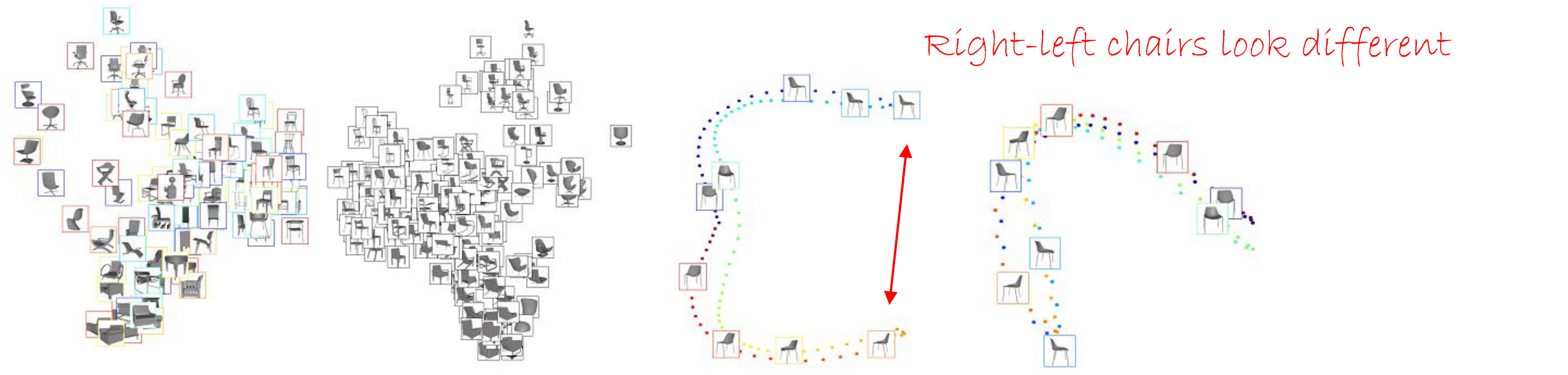


Figure 3: PCA embeddings for 2D position on AlexNet.

Filter invariance and equivariance



(a) Chair, pool5 (b) Chair, pool5, style (c) Chair, pool5, rotation (d) Chair, fc6, rotation

(e) Car, pool5 (f) Car, pool5, style (g) Car, pool5, rotation (h) Car, fc6, rotation

Figure 2: **Best viewed in the electronic version.** PCA embeddings (dims. 1,2) of AlexNet features for "chairs" (first row) and "cars" (second row). Column 1 – Direct embedding of the rendered images without viewpoint-style separation. Columns 2,3 – Embeddings associated with style (for all rotations) and rotation (for all styles). Column 4 – Rotation embedding for fc6, which is qualitatively different than pool5. Colors correspond to orientation and can be interpreted via the example images in columns 3,4. Similar results for other categories and PCA dimensions are available in the supplementary material.

The effect of the architecture

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

Depth is important

Early signs of overfitting

Overfitting

Convolution activations are “images”

- $h_I(x, y) = I(x, y) * h$
  $$= \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x - i, y - j) \cdot h(i, j)$$
- For every image pixel we compute a convolved image pixel
- After each convolution we end up with a "new image

Several interesting questions

- What do the image activations in different layers look like?
- What types of images create the strongest activations?
- What are the activations for the class "ostrich"?
- Do the activations occur in meaningful positions?

Feature maps

- The convolution activations are also called feature maps
- A deep network has several hierarchical layers
  - hence several feature maps

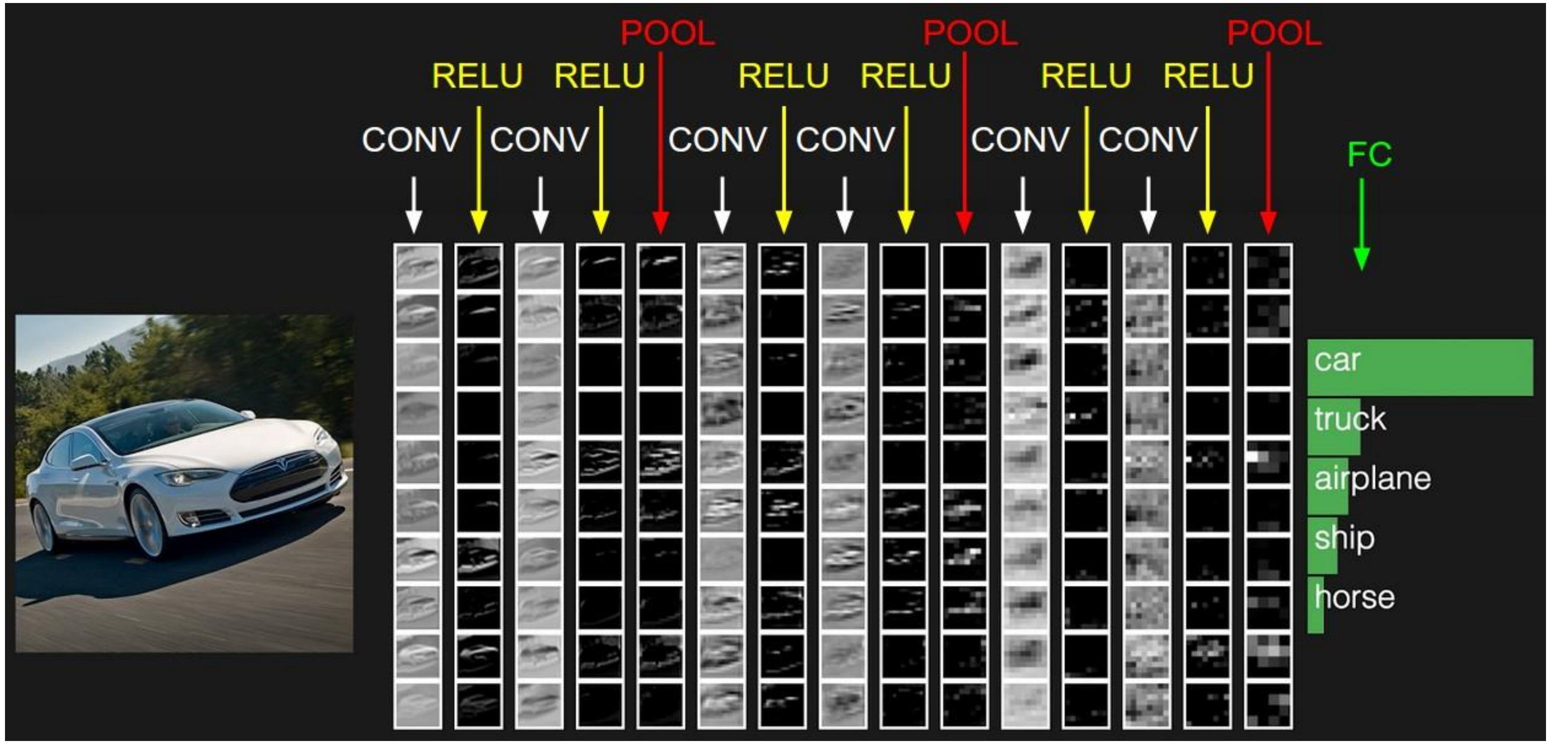


Image borrowed by A. Karpathy

Feature map strength

- Naturally, given an image some feature maps will be "stronger"
  - Contribute higher amount to the final classification score
- Why? What pixel structure excited these particular feature maps?
  - Generally, what part of the picture excites a certain feature map?

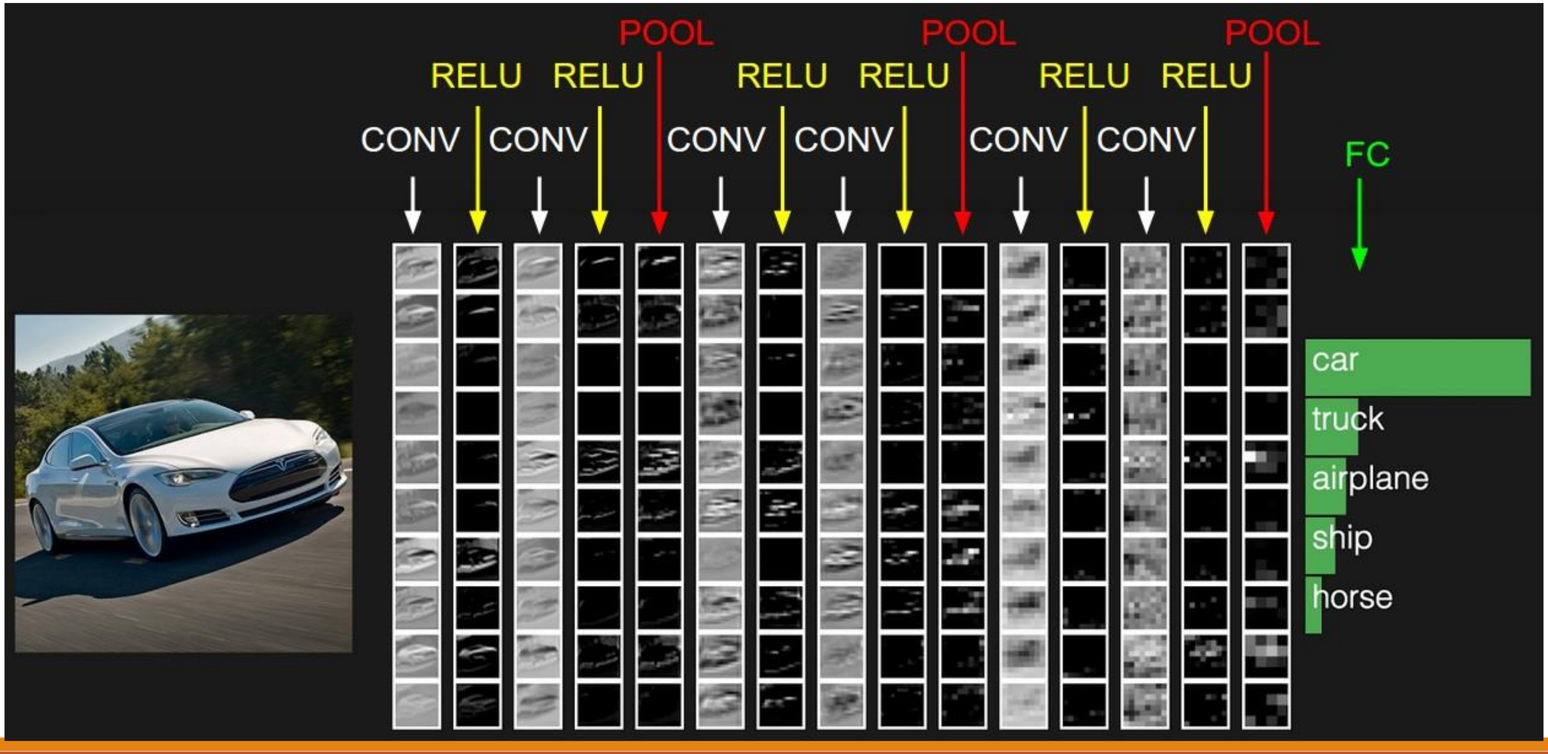


Image borrowed by A. Karpathy

Feature map strength

- Naturally, given an image some feature maps will be "stronger"
  - Contribute higher amount to the final classification score
- Why? What pixel structure excited these particular feature maps?
  - Generally, what part of the picture excites a certain feature map?

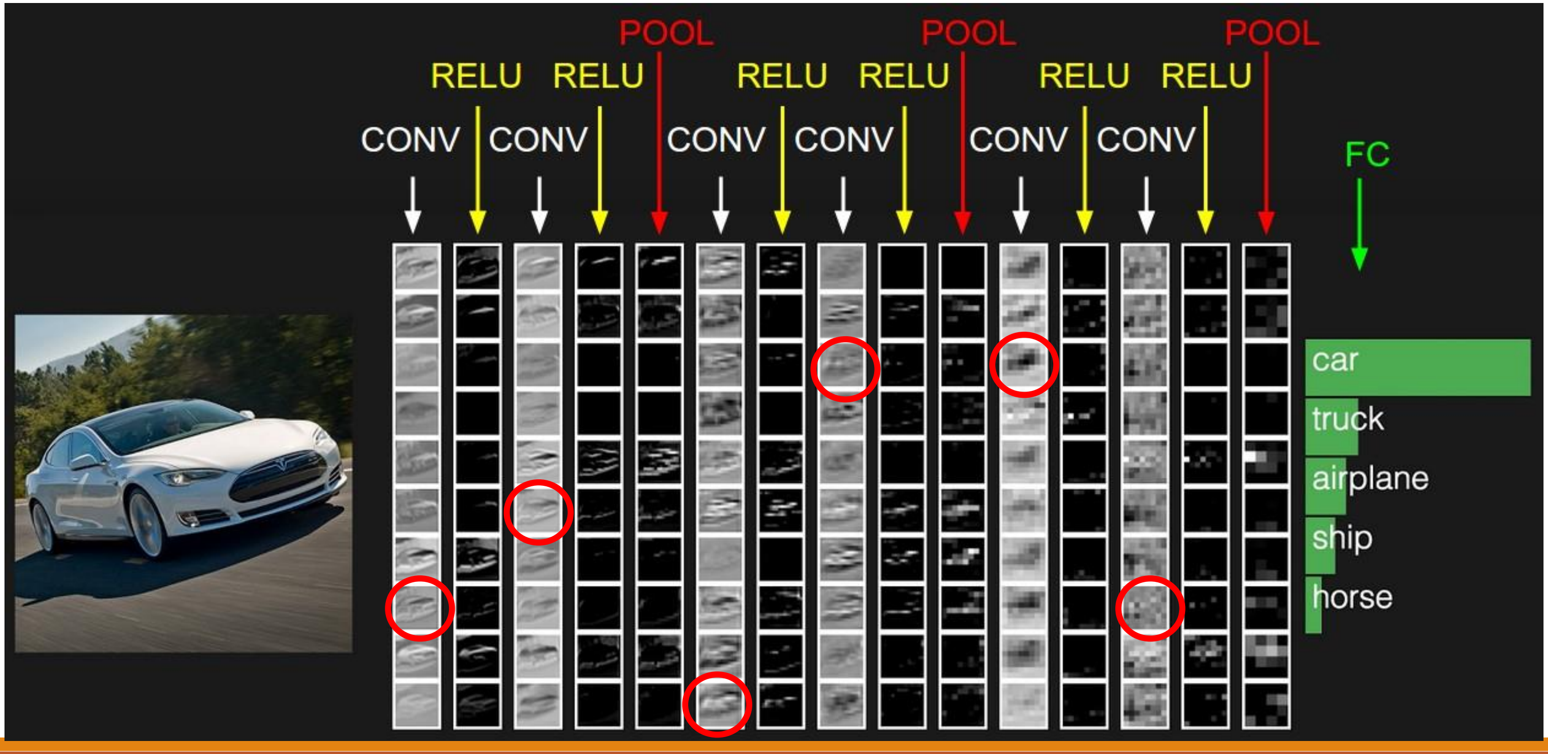
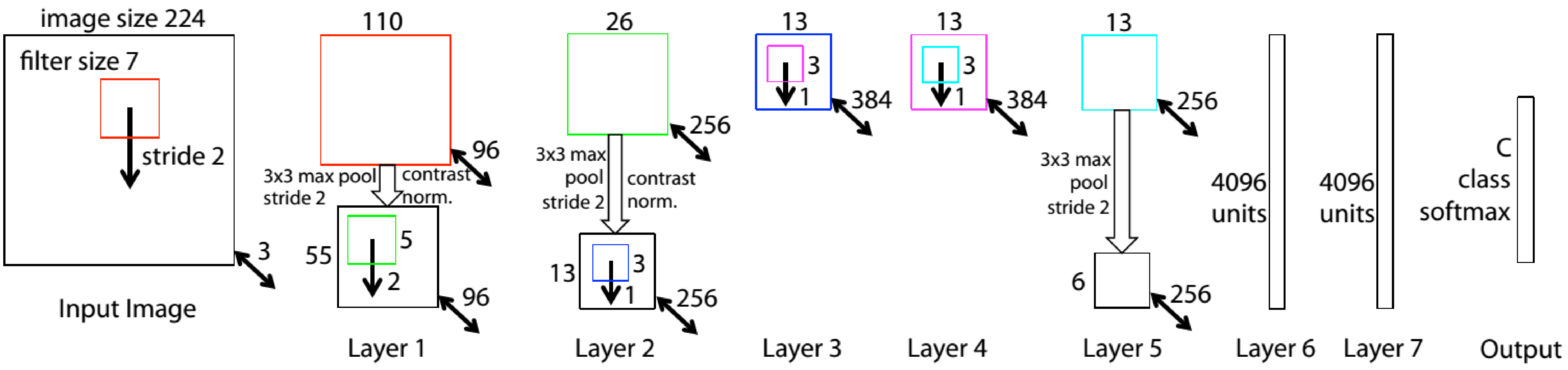


Image borrowed by A. Karpathy

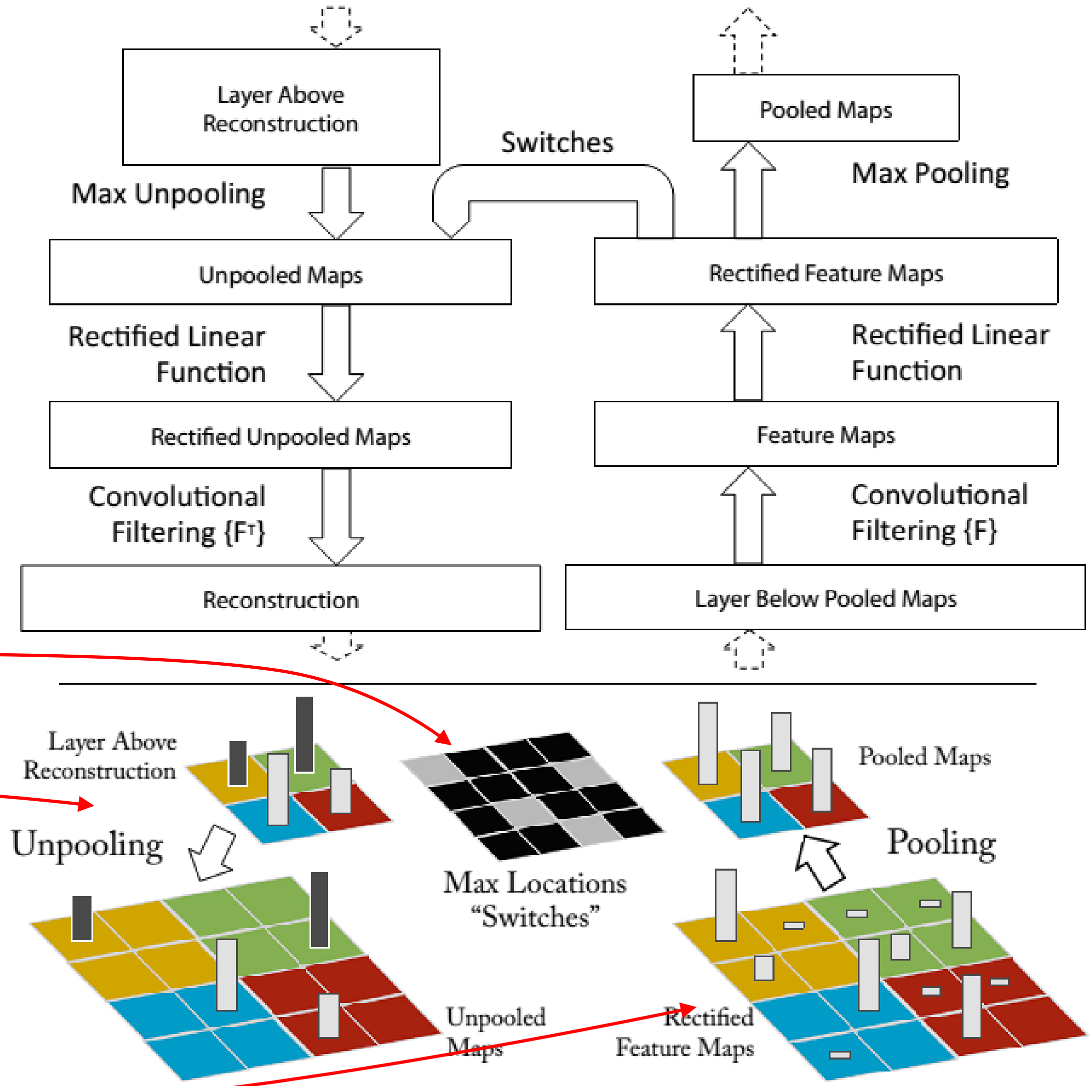
Starting from a convnet [Zeiler2014]



Visualization with a Deconvnet [Zeiler2014]

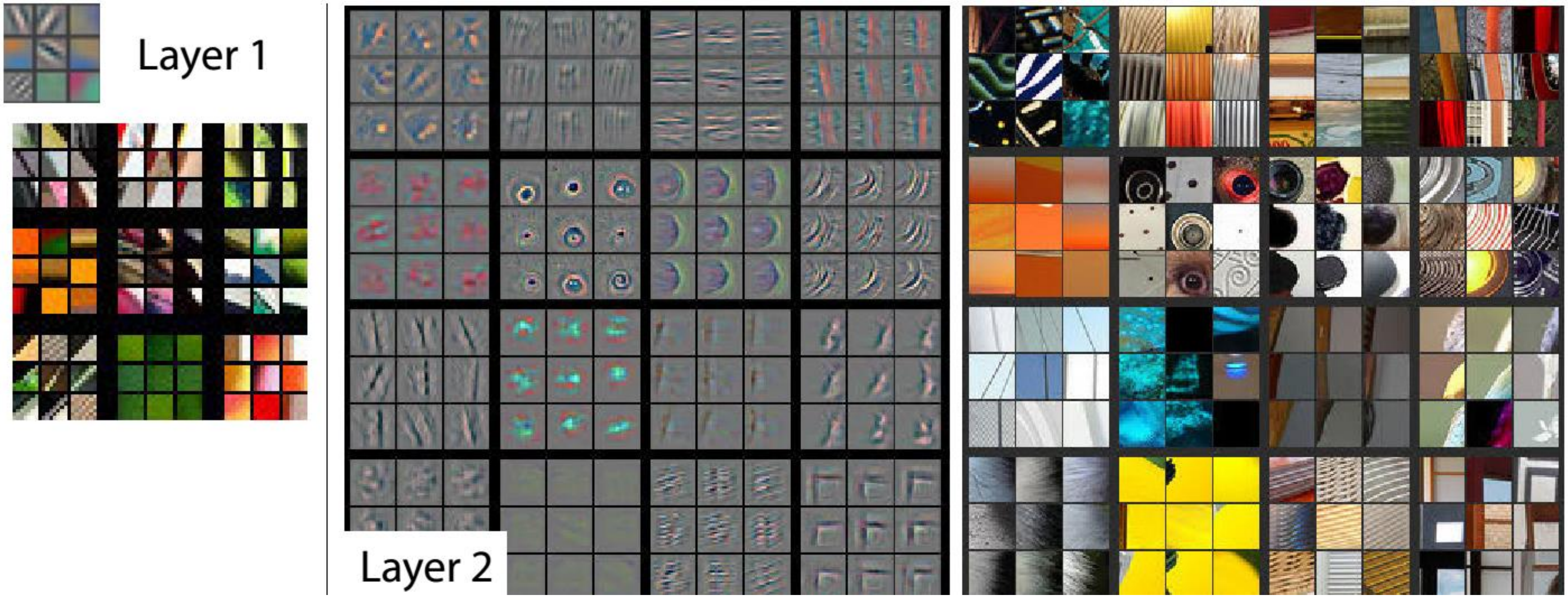
- Same as convent, but in reverse
  - Instead of mapping pixels to activations, mapping activations to pixels
- Attach a deconvnet layer on every convnet layer
- Unpooling via switches that remember where max pooling was activated
- Deconv filtering equals to "reverse conv filtering"

$$F = \begin{bmatrix} 1 & 5 \\ 6 & 3 \end{bmatrix} \qquad F^T = \begin{bmatrix} 3 & 6 \\ 5 & 1 \end{bmatrix}$$



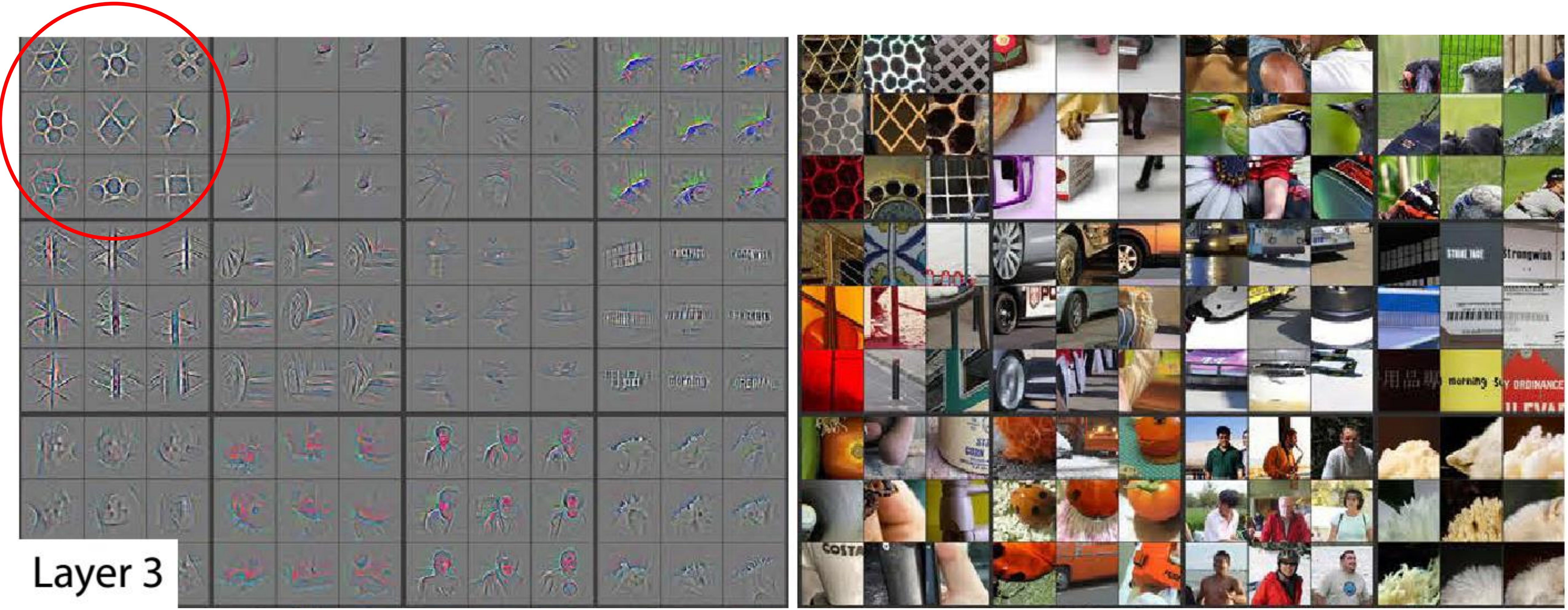
What excites feature maps?

- *"Given a random feature map what are the top 9 activations?"*



What excites feature maps?

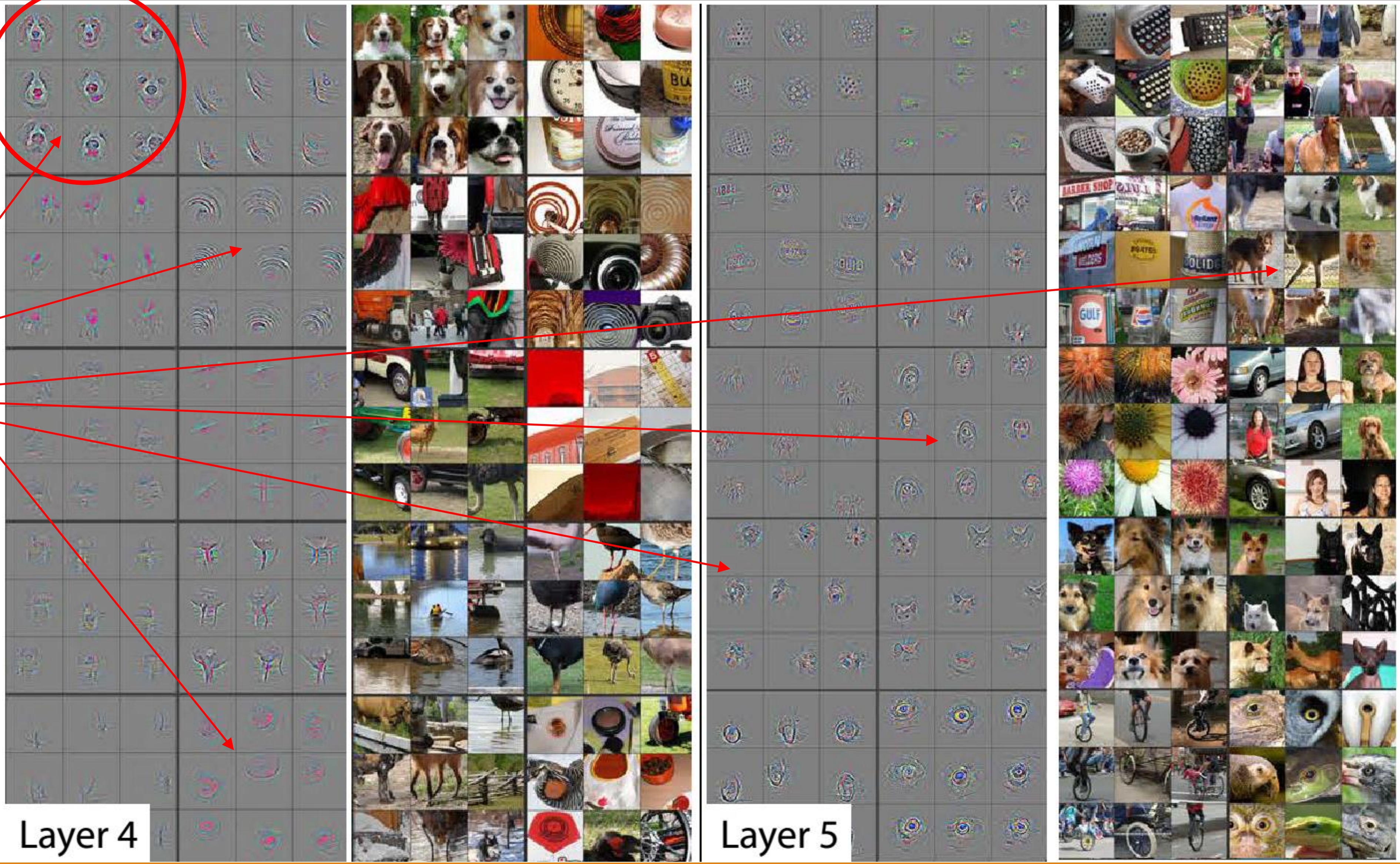
Similar activations from lower level visual patterns



What excites feature maps? [Zeiler2014]

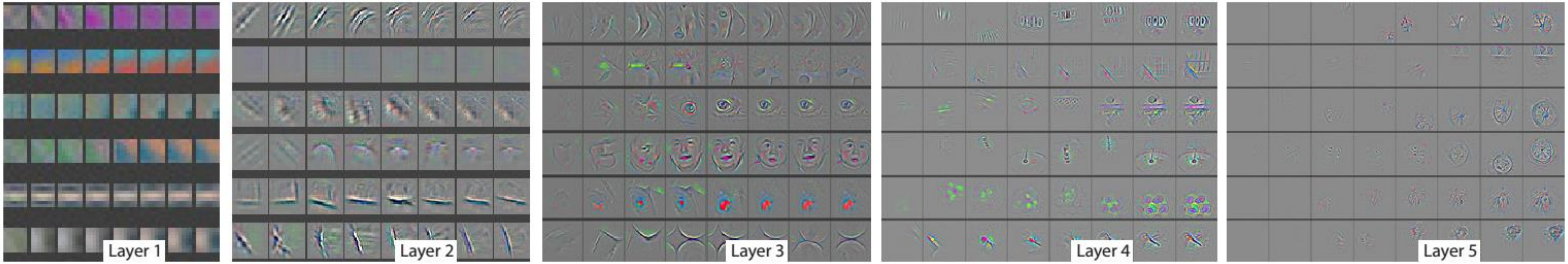
Similar activations from semantically similar pictures

Visual patterns become more and more intricate and specific (greater invariance)

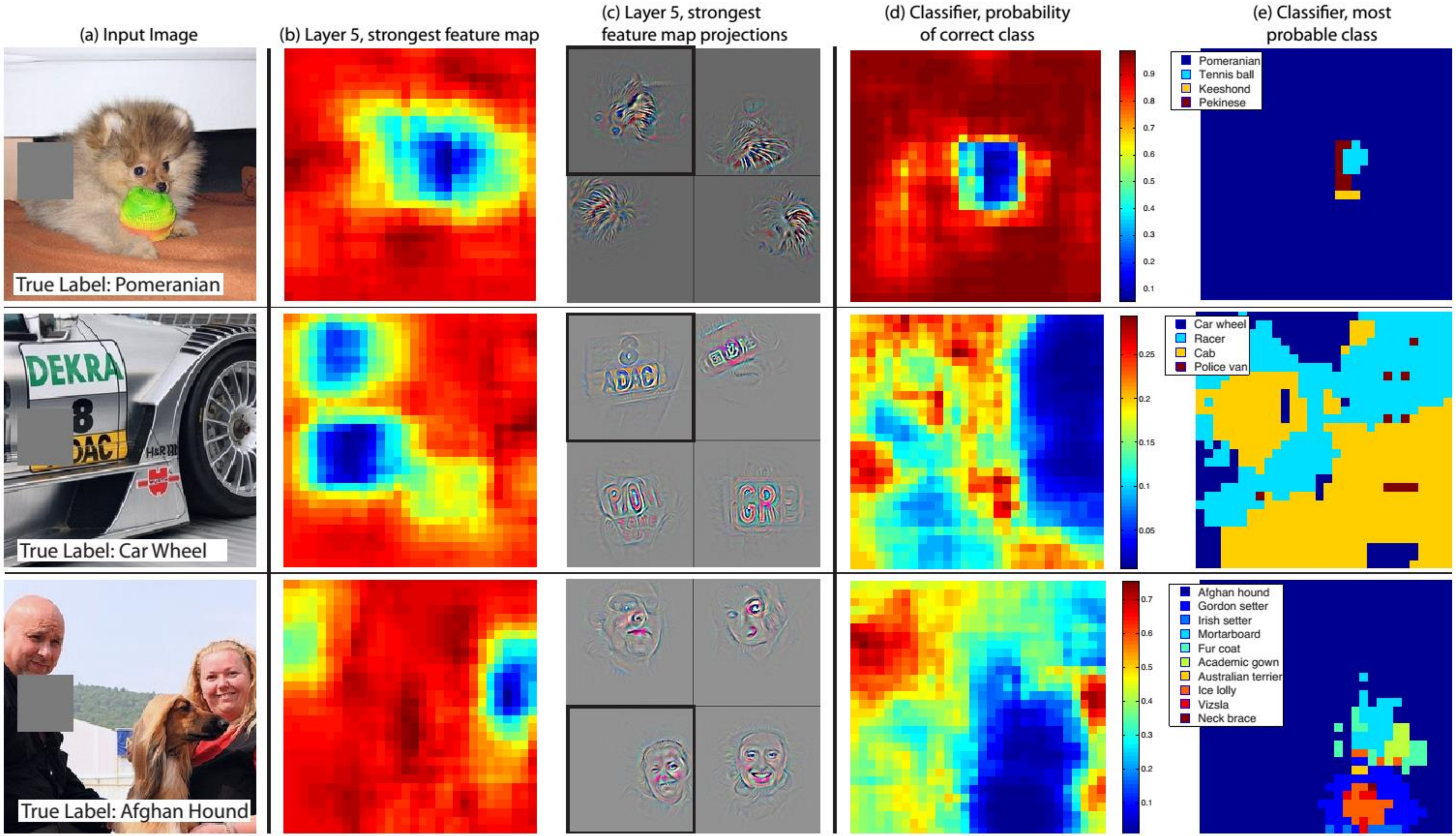


Feature evolution over training

- For a particular neuron (that generates a feature map)
- Pick the strongest activation during training
- For epochs 1, 2, 5, 10, 20, 30, 40, 64



But does a Convnet really learn the object?



What is a “Convnet dog”, however? [Simonyan2014]

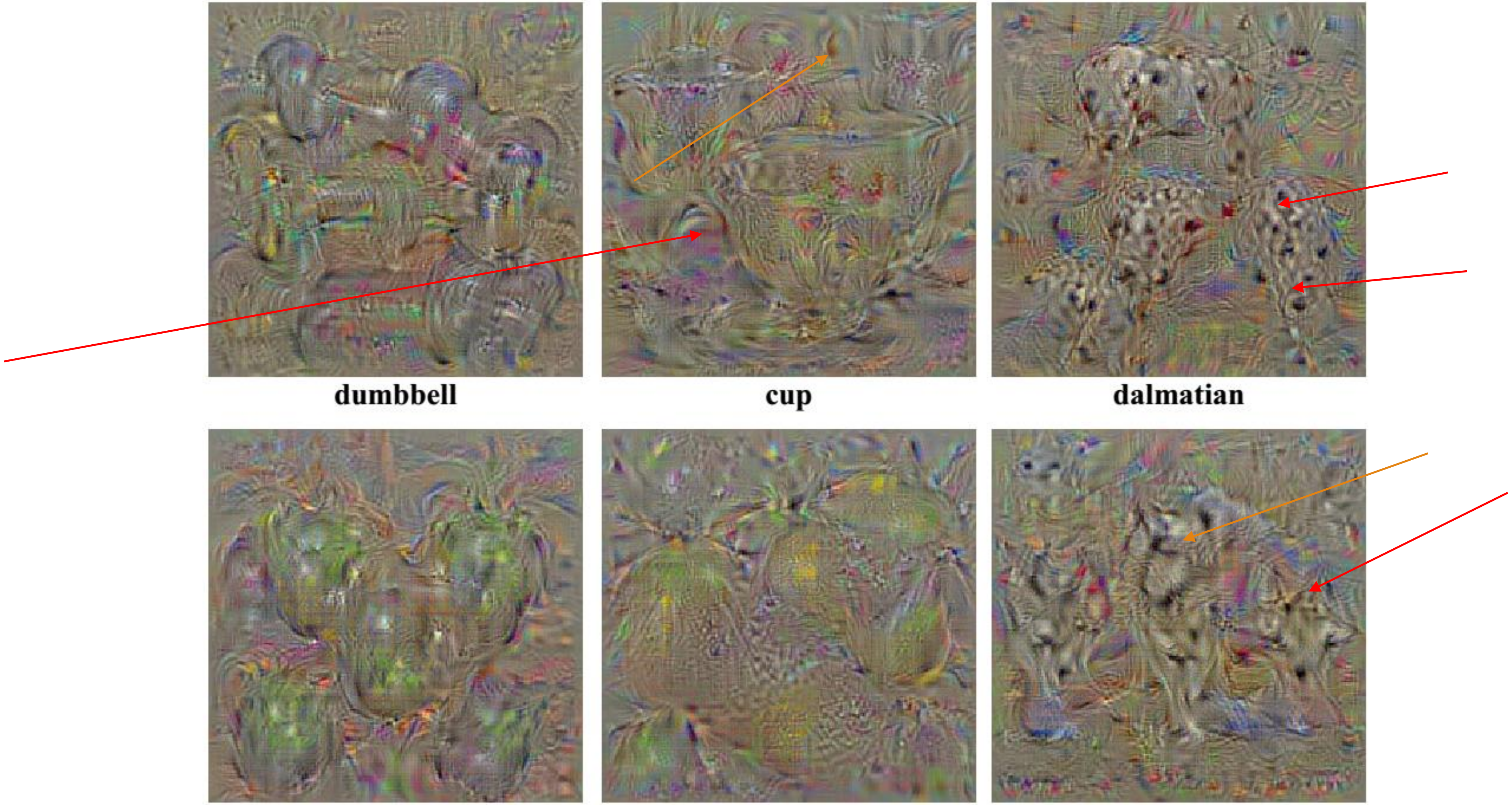
- What is the image with the highest “dogness score”

$$\arg\max_{I} S_c(I;\theta) - \lambda|\mathrm{I}|^2$$

  - The parameters θ are fixed during the training
  - Optimization is done with respect to the image I
  - Initialized with the “average training image”

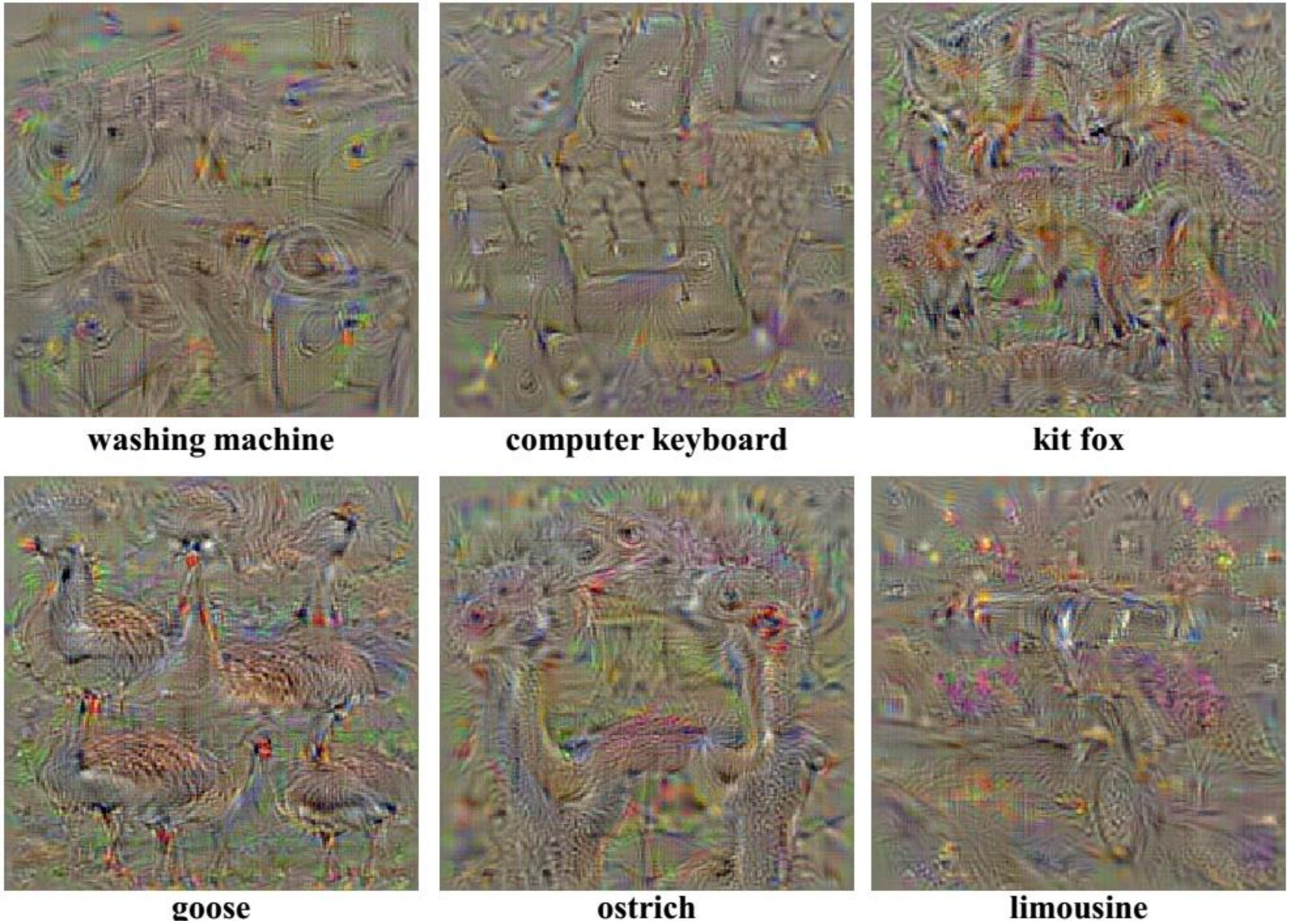
Maximum-scoring class images

Maximum-scoring class images



dumbbell, cup, dalmatian, bell pepper, lemon, husky

Maximum-scoring class images



Class-specific image saliency

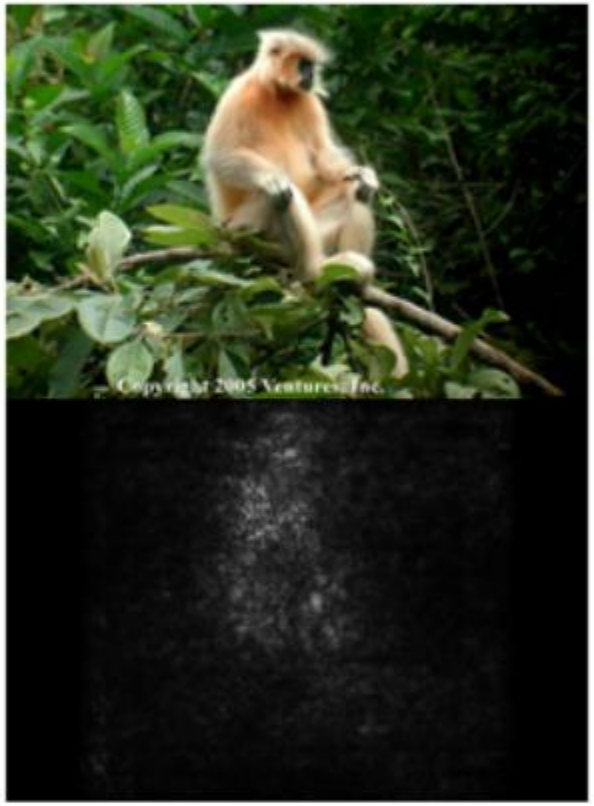
- Given the “monkey” class, what are the most “monkey-ish” parts in my image?
- Approximate S_c around an initial point I_0 with the first order Taylor expansion

  $S_c(I)|_{I_0} \approx w^T I + b$, where $w = \frac{\partial S_c}{\partial I}|_{I_0}$ from backpropagation

  - Solution is locally optimal

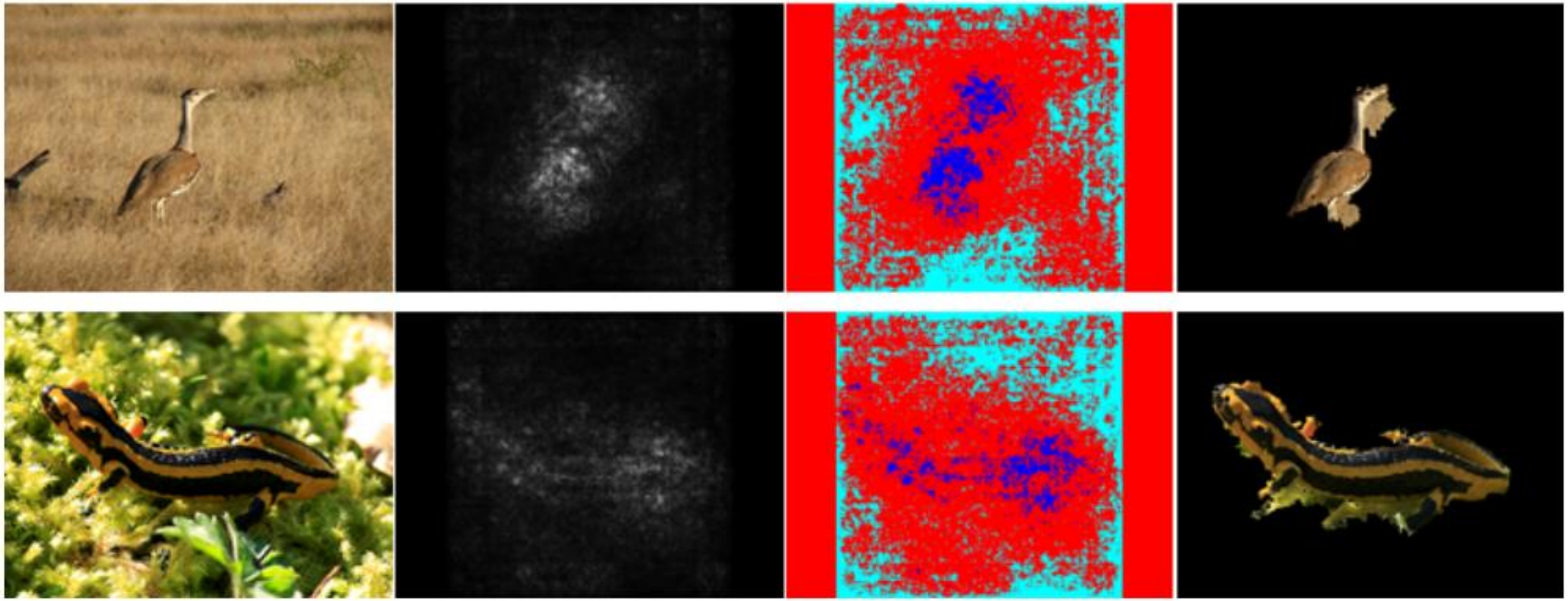
Examples

Examples

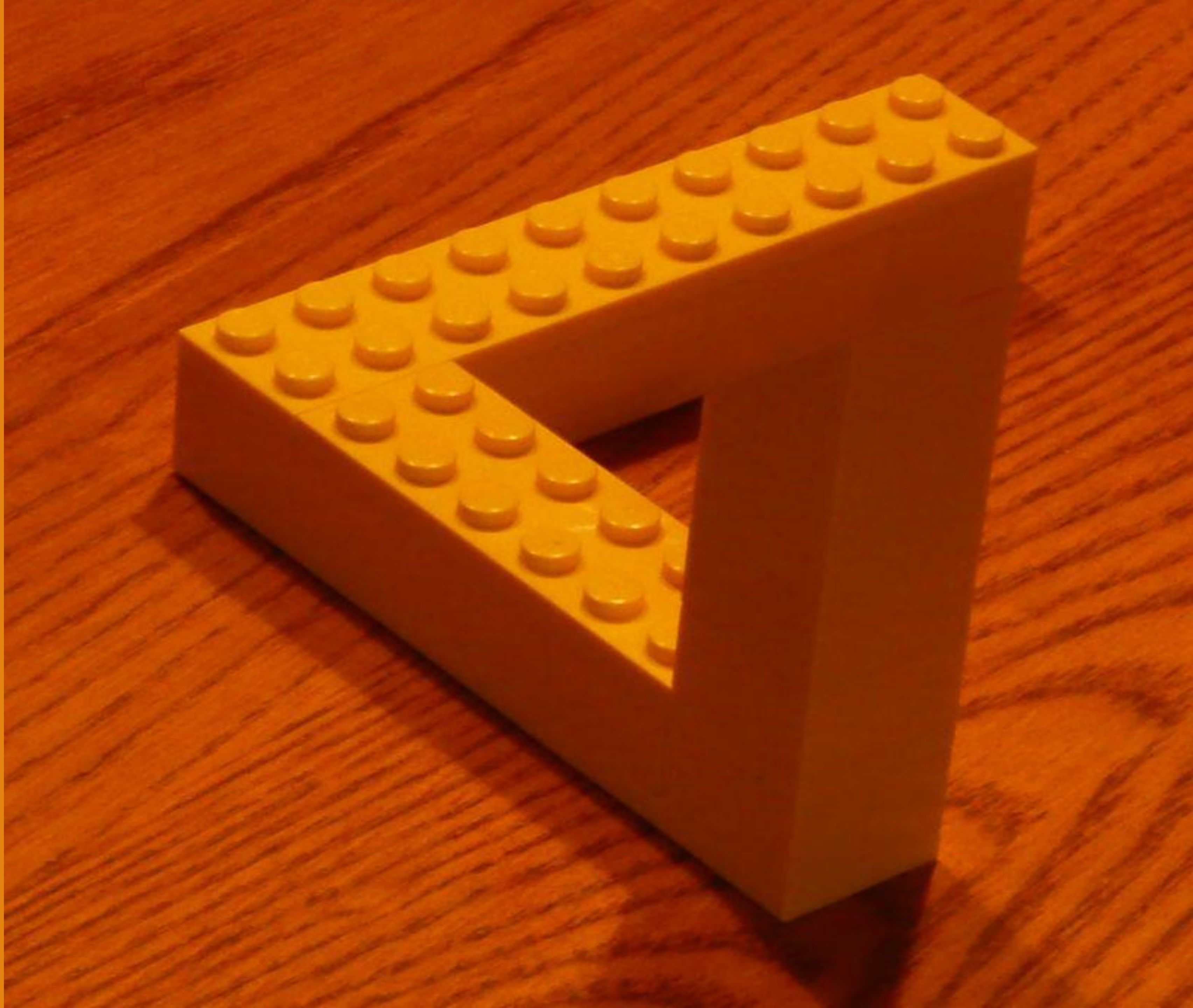


Object localization using class saliency

GrabCut



Fooling a Convnet



Fooling images

- What if we follow a similar procedure but with a different goal
- Generate “visually random” images
  - Images that make a lot of sense to a Convnet but no sense at all to us
- Or, assume we make very small changes to a picture (invisible to the naked eye)
  - Is a convnet always invariant to these changes?
  - Or could it be fooled?

Smoothness assumption

- We assume our classifiers enjoy **local generalization**
- Assume an image containing a cat lies at coordinates x
- Assume also a small change r around x, such that $x + r < \varepsilon$
  - ε is a very small constant
- Is the smoothness assumption reasonable?
- Or can we "break" it by some adversarial examp
  - E.g. if we correctly classify x as *"Argan goat"*, with the right r make the convnet see a *"BMW i6"*
  - The $x + r$ would be adversarial examples

Generating adversarial examples

- If $f \colon \mathcal{R}^m \to \{1 \ \ldots, k\}$ our goal can be mathematically described as

$$\min ||r||^2$$
$$s.t. \quad f(x+r) = l, x + r \in [0,1]^m$$

- The goal is to optimize the distortion r so that the predicted label l is different than the original label

Adversarial images

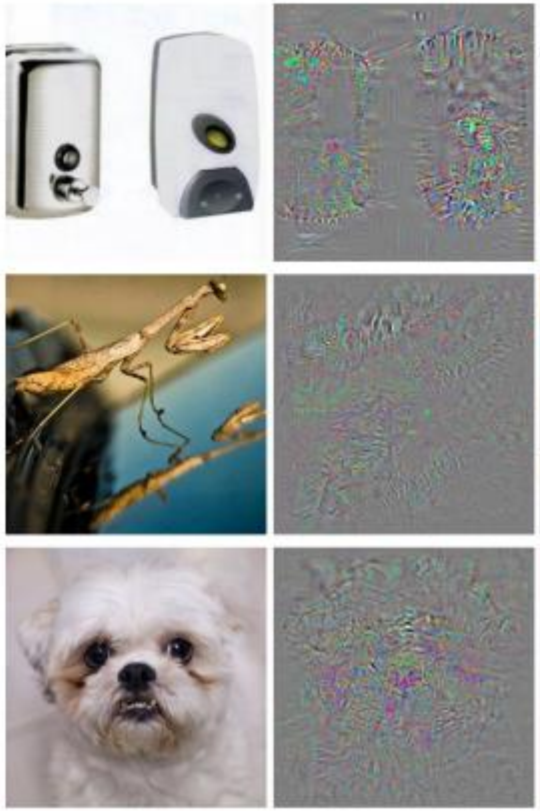
Image

Image

Adversarial images

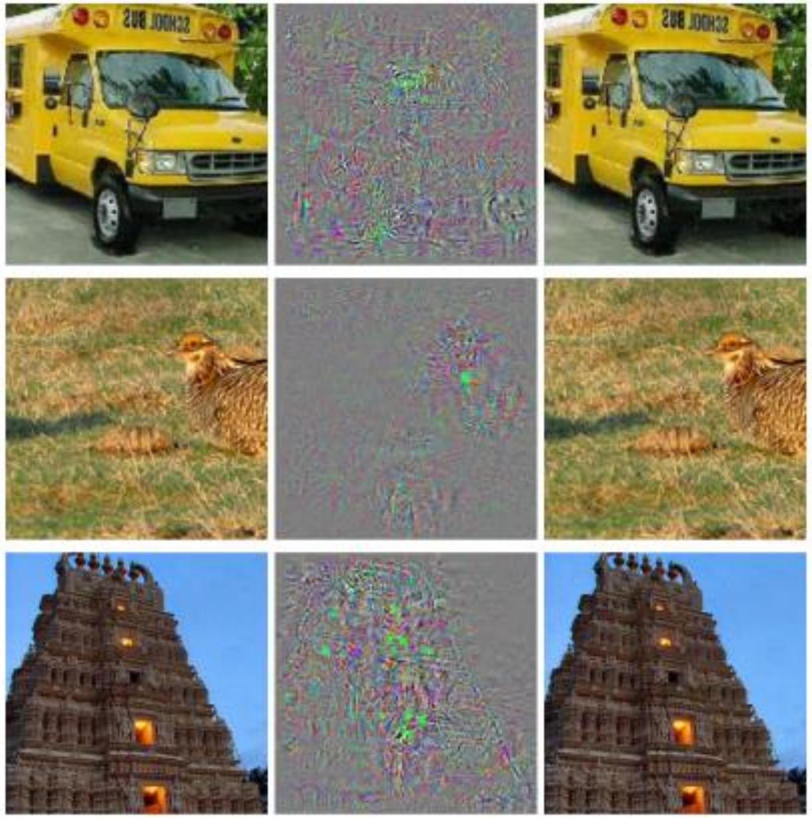
Image+Noise

Image+Noise

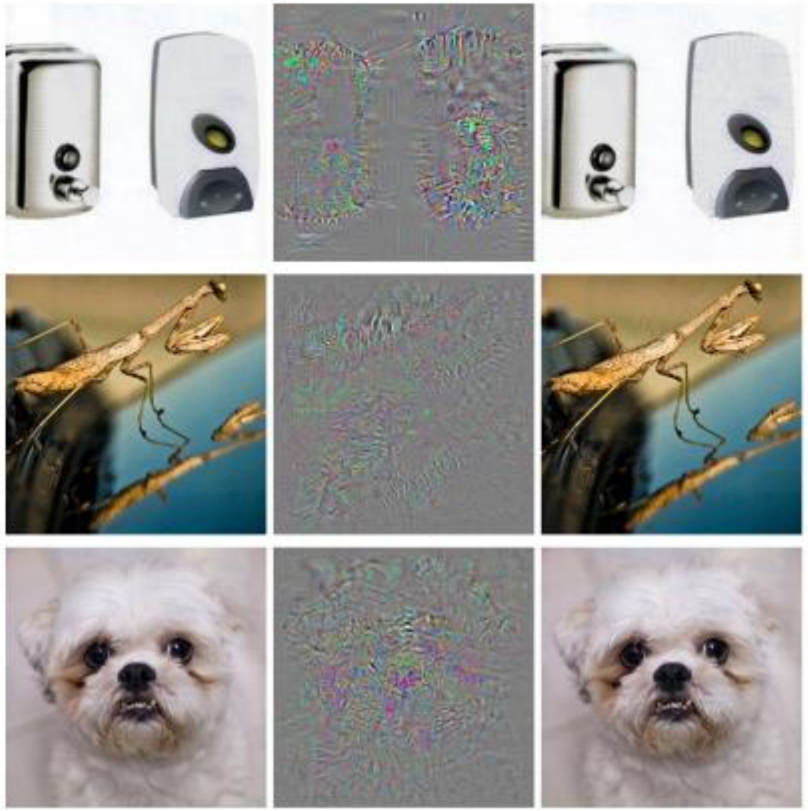


Adversarial images

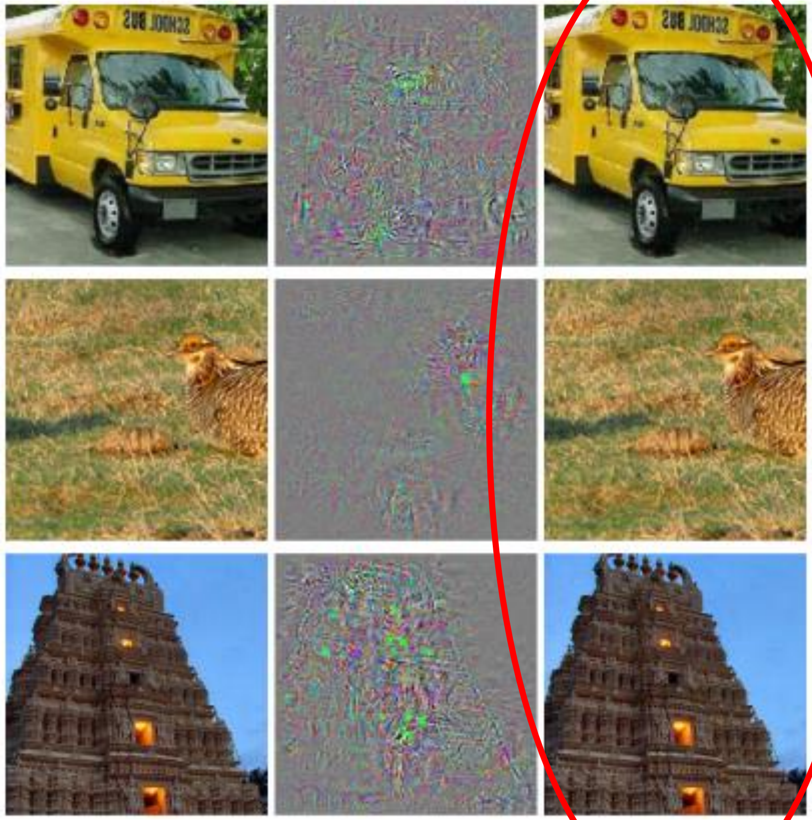
Image+Noise=Image'

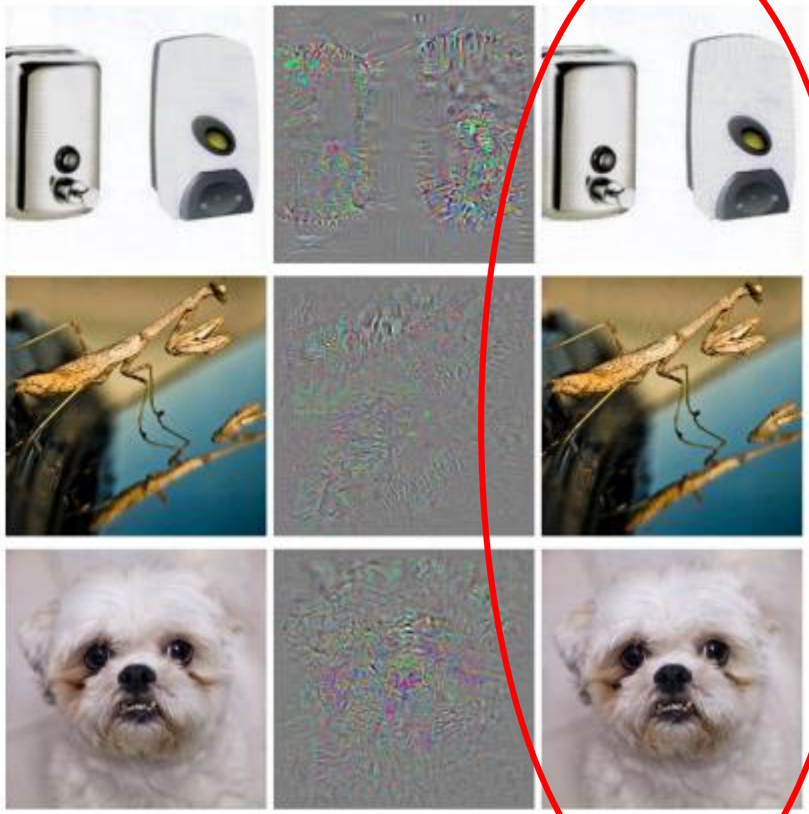


Image+Noise=Image'



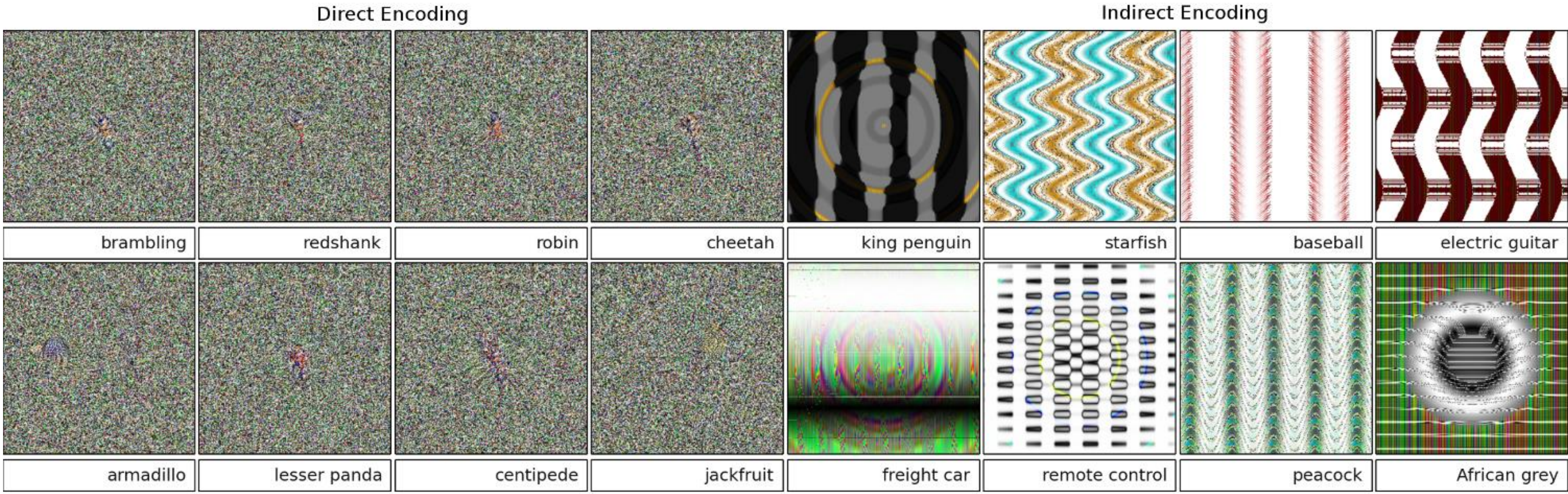
Adversarial images



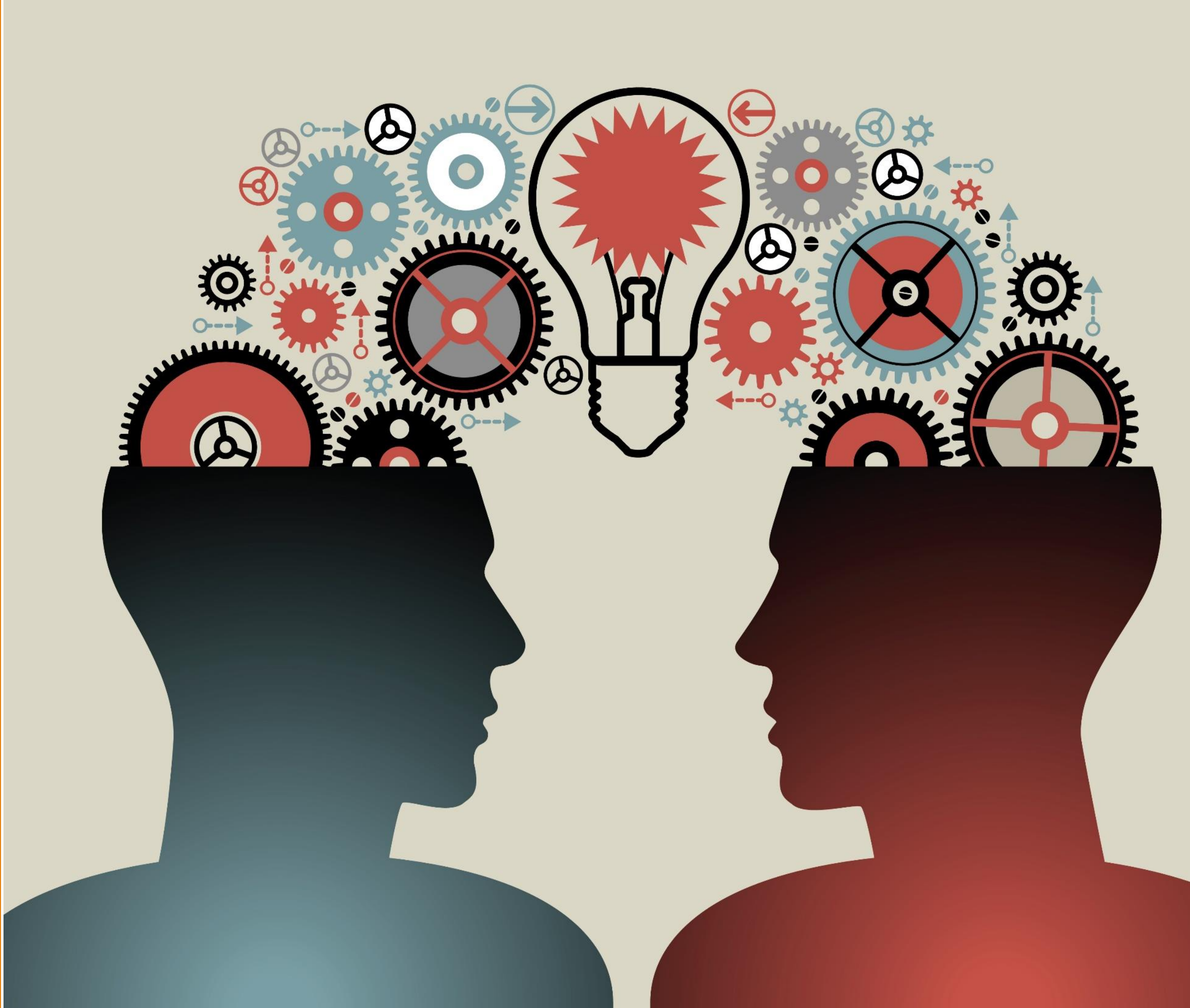


Image'=Predicted as Ostrich, Struthiocamelus

More adversarial images



Knowledge transfer



CNNs and dataset size

- A CNN can have millions of parameters
- What about the dataset size?
- Could we still train a CNN without overfitting problems?

Transfer learning

- Assume two datasets, T and S
- Dataset S is
  - fully annotated, plenty of images
  - We can build a model h_S
- Dataset T is
  - Not as much annotated, or much fewer images
  - The annotations of T do not need to overlap with S
- We can use the model h_S to learn a better h_T
- This is called transfer learning

Imagenet: 1million

h_A

My dataset: 1,000

h_B

Convnets are good in transfer learning

- Even if our dataset T is not large, we can train a CNN for it
- Pre-train a network on the dataset S
- Then, there are two solutions
  - Fine-tuning
  - CNN as feature extractor

Solution I: Fine-tune h_T using h_S as initialization

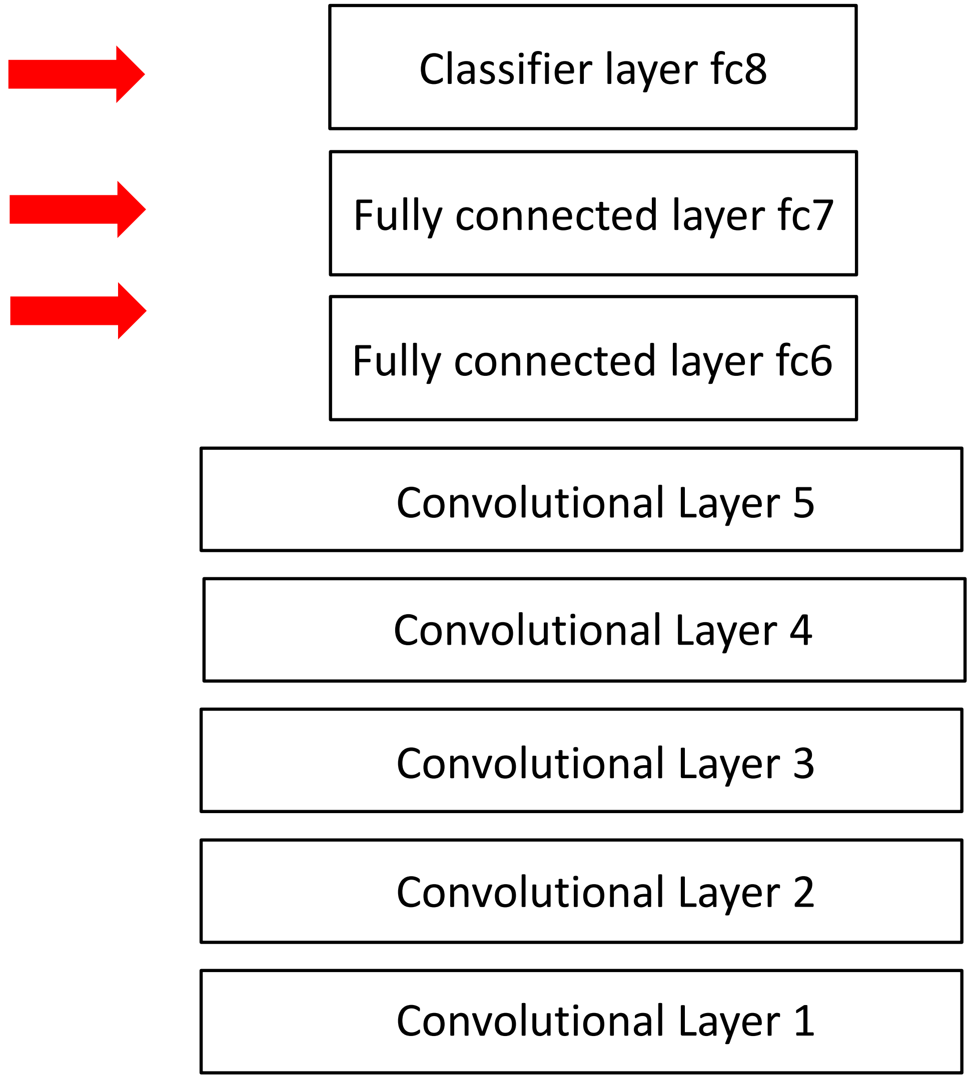
- Assume the parameters of S are already a good start near our final local optimum
- Use them as the initial parameters for our new CNN for the target dataset

$$\theta_{\mathrm{T},\,l}^{(t=0)} = \theta_{\mathrm{S},\,l} \text{ for some layers } l = 1, 2, \ldots$$

- This is a good solution when the dataset T is relatively big
  - E.g. for Imagenet S with approximately 1 million images
  - For a dataset T with more than a few thousand images should be ok
- What layers to initialize and how?

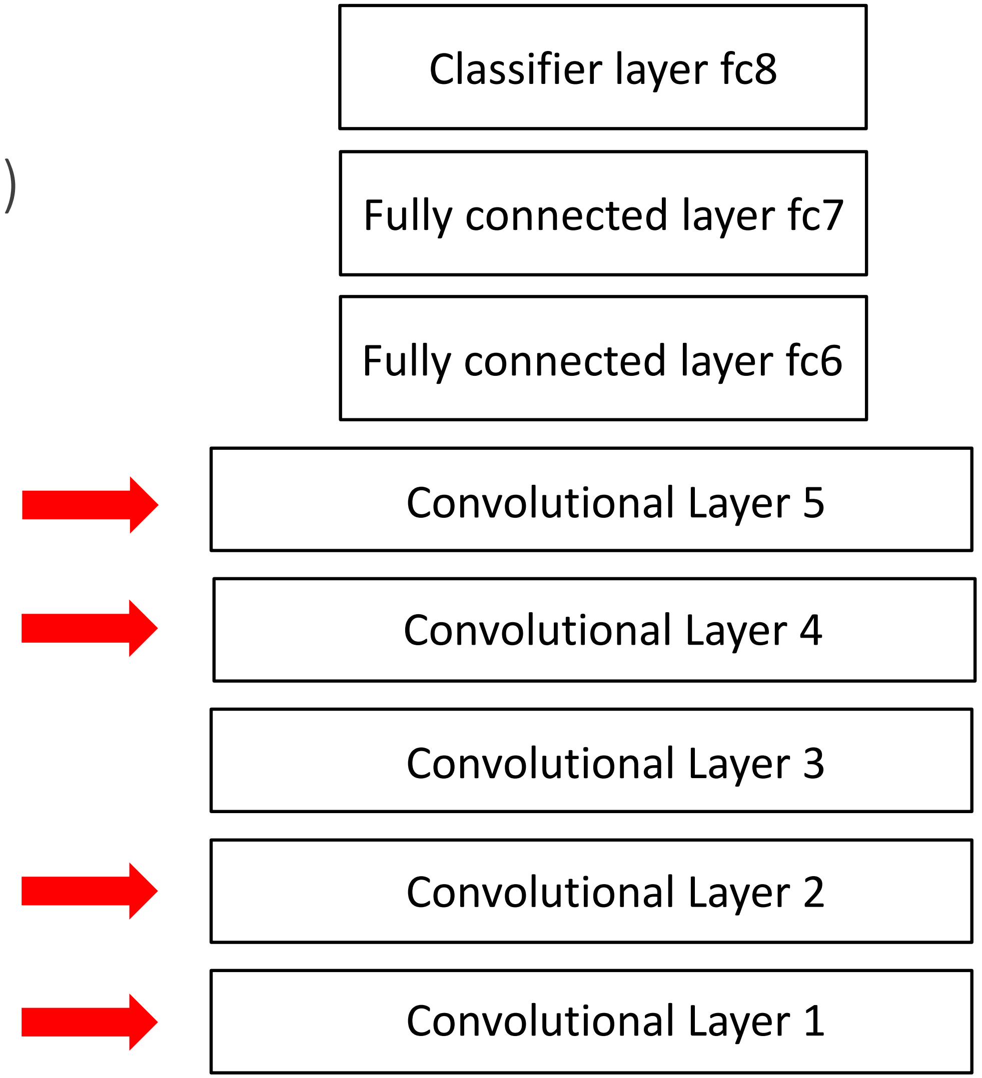
Initializing h_T with h_S

- Classifier layer to loss
  - The loss layer essentially is the “classifier”
  - Same labels → keep the weights from h_S
  - Different labels → delete the layer and start over
  - When too few data, fine-tune only this layer
- Fully connected layers
  - Very important for fine-tuning
  - Sometimes you need to completely delete the last before the classification layer if datasets are very different
  - Capture more semantic, “specific” information
  - Always try first when fine-tuning
  - If you have more data, fine-tune also these layers



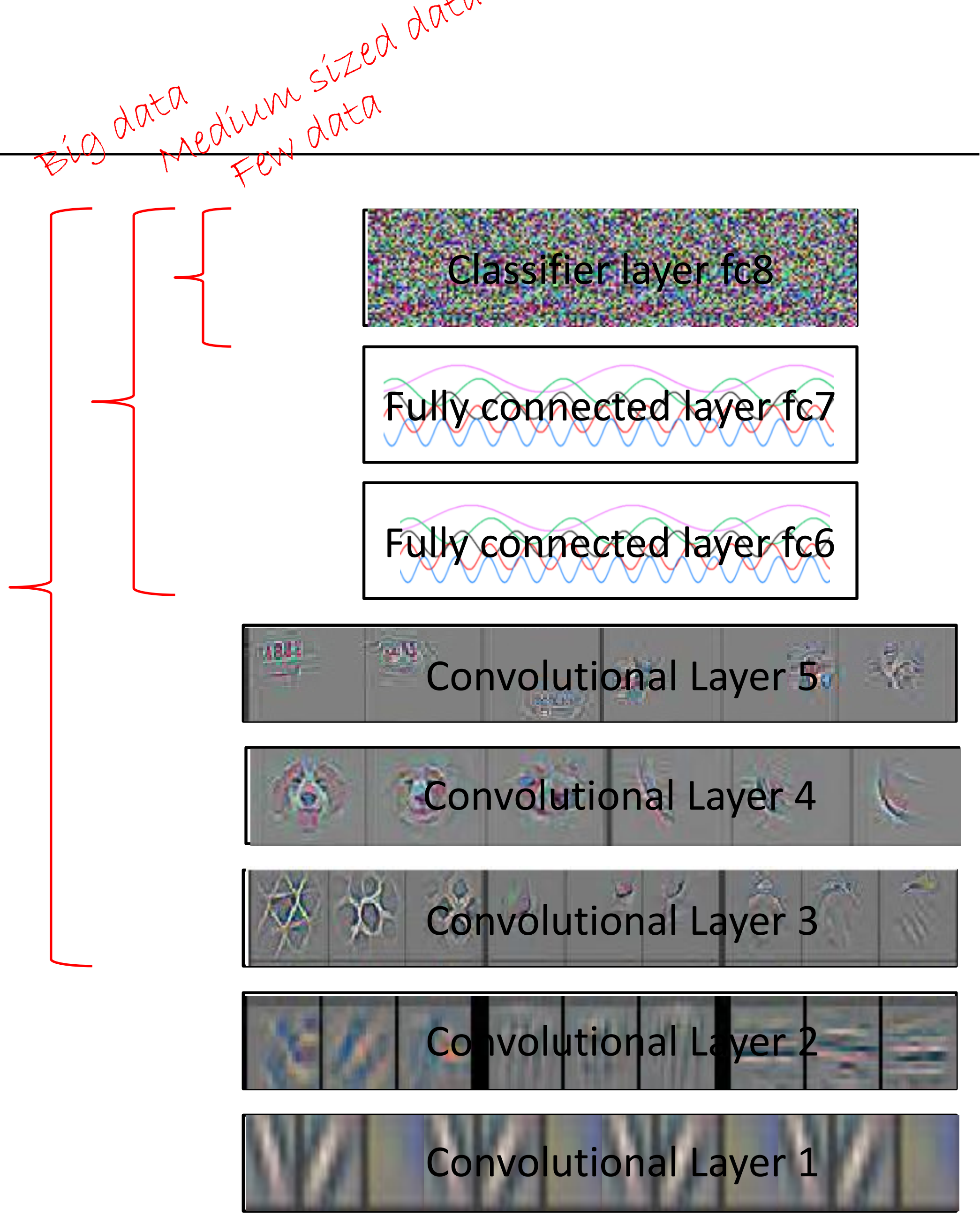
Initializing h_T with h_S

- Upper convolutional layers (conv4, conv5)
  - Mid-level spatial features (face, wheel detectors ...)
  - Can be different from dataset to dataset
  - Capture more generic information
  - Fine-tuning pays off
  - Fine-tune if dataset is big enough
- Lower convolutional layers (conv1, conv2)
  - They capture low level information
  - This information does not change usually
  - Probably, no need to fine-tune but no harm trying



How to fine-tune?

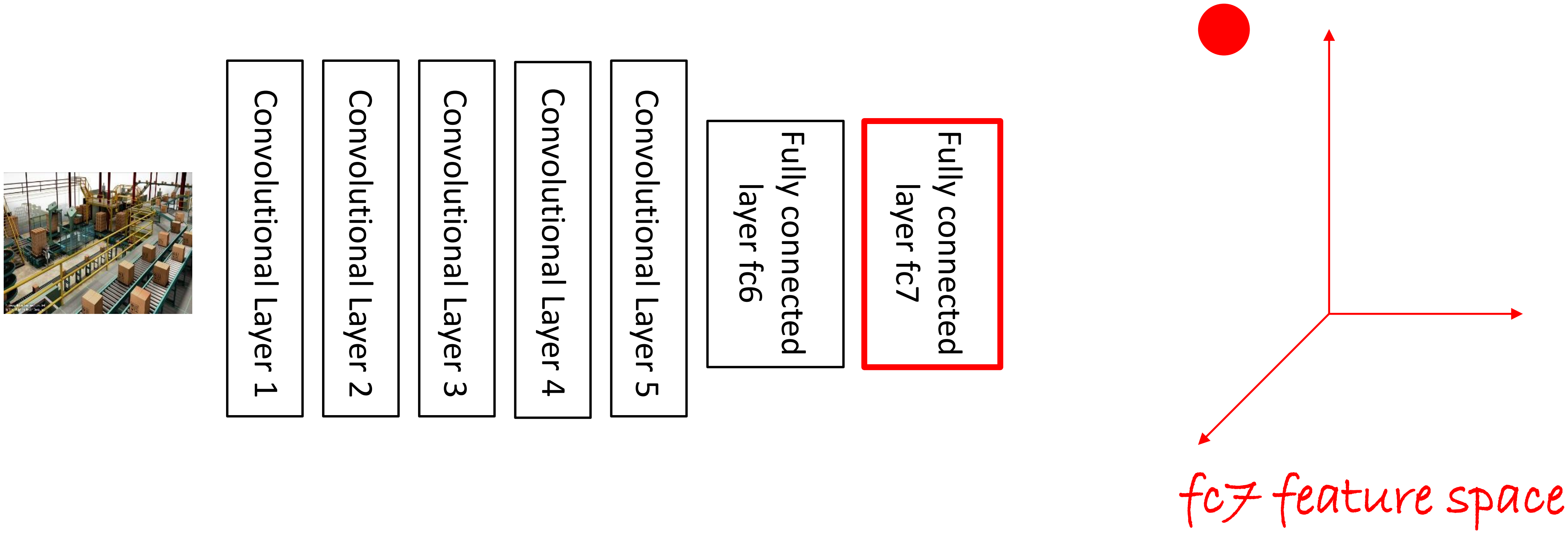
- For layers initialized from h_S use a mild learning rate
  - Remember: your network is already close to a near optimum
  - If too aggressive, learning might diverge
  - A learning rate of 0.001 is a good starting choice (assuming 0.01 was the original learning rate)
- For completely new layers (e.g. loss) use aggressive learning rate
  - If too small, the training will converge very slowly
  - Remember: the rest of the network is near a solution, this layer is very far from one
  - A learning rate of 0.01 is a good starting choice
- If datasets are very similar, fine-tune only fully connected layers
- If datasets are different and you have enough data, fine-tune all layers



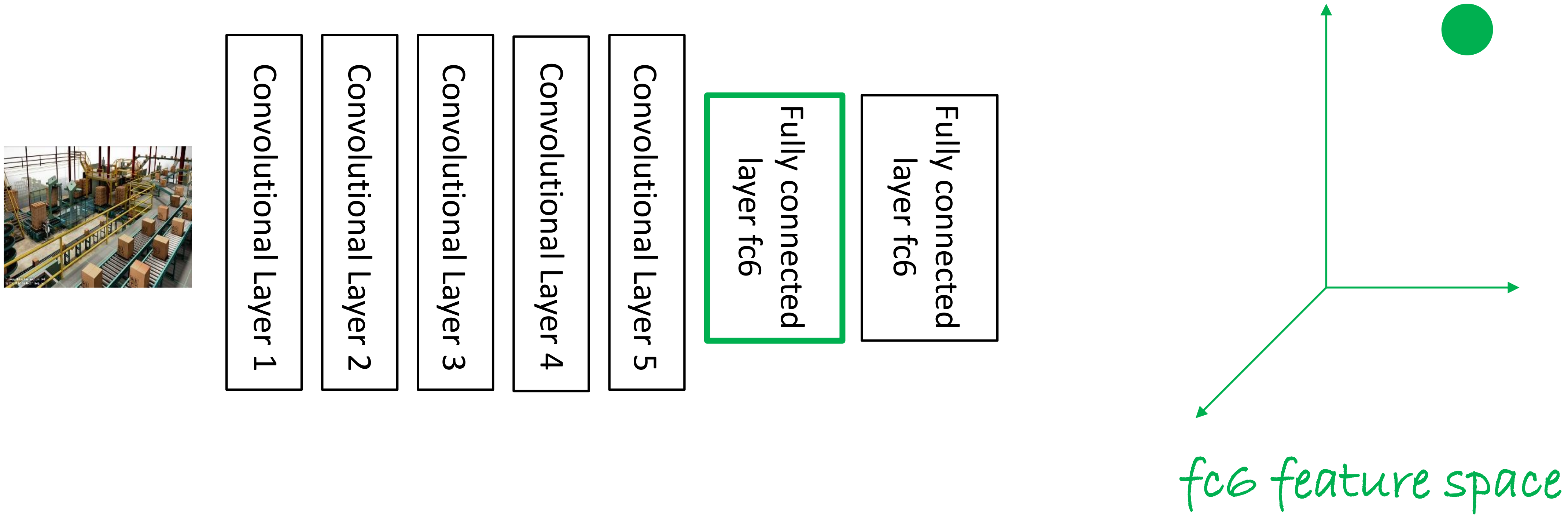
Solution II: Use h_s as a feature extractor for h_T

- Essentially similar to a case of solution I
  - but train only the loss layer
- Essentially use the network as a pretrained feature extractor
- This is a good solution if the dataset T is small
  - Any fine-tuning of layer might cause overfitting
- Or when we don't have the resources to train a deep net
- Or when we don't care for the best possible accuracy

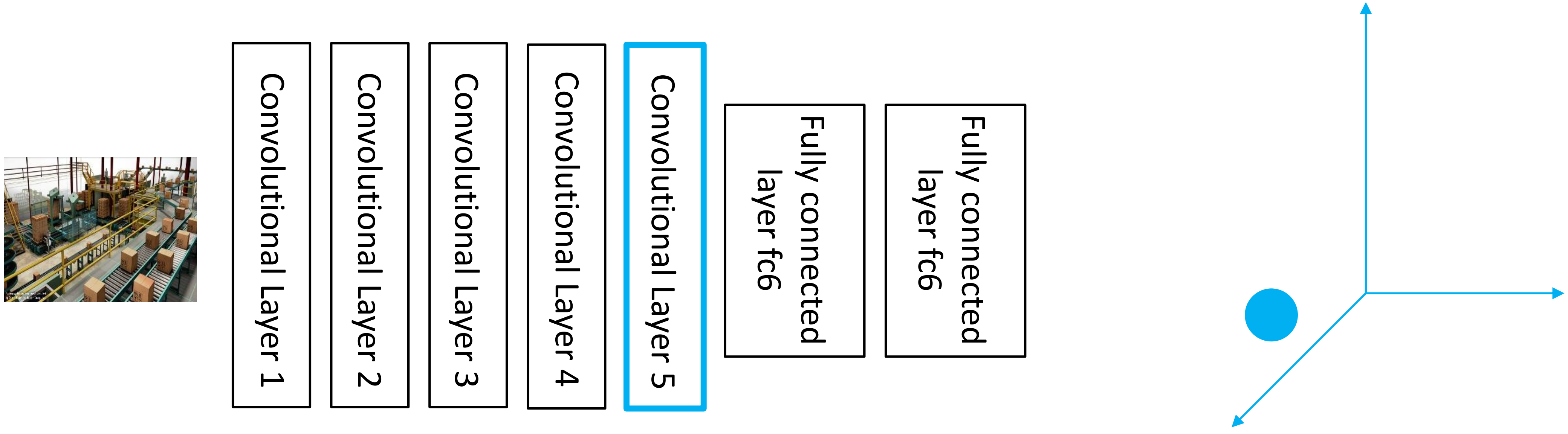
Deep features from different layers



Deep features from different layers



Deep features from different layers



Conv5 feature space

Which layer?

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)	$\mathbf{85.4 \pm 0.4}$	$\mathbf{72.6 \pm 0.1}$

Lower layer features capture more basic information (texture, etc). Good for image-to-image comparisons, image retrieval

Higher layer features are capture more semantic information. Good for higher level classification

Visualizing and Understanding Convolutional Networks, Zeiler and Fergus, ECCV 2014

Summary

- What do convolutions look like?
- Build on the visual intuition behind Convnets
- Deep Learning Feature maps
- Transfer Learning

Reading material & references

- http://www.deeplearningbook.org/
  - Part II: Chapter 11

[Aubry2015] Aubry, Russell. *Understanding deep features with computer-generated imagery*, ICCV, 2015
[Nguyen2015] Nguyen, Yosinksi, Clune. *Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images*, CVPR, 2015
[Simonyan2014] Simonyan, Vedaldi, Zisserman . *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*, CVPR 2014
[Zeiler2014] Zeiler, Fergus, *Visualizing and Understanding Convolutional Networks*, ECCV, 2014

Next lecture

- Convolutional Neural Networks for Object Detection and Segmentation
- Convolutional Neural Networks and Structured Prediction