Image from ucreative.com

CMP784

DEEP LEARNING

Lecture #06 – Understanding and Visualizing Convolutional Neural Networks

HACETTEPE UNIVERSITY COMPUTER VISION LAB

Erkut Erdem // Hacettepe University // Fall 2022

Previously on CMP784

- convolution layer
- design guidelines for CNNs
- CNN architectures
- transfer learning
- semantic segmentation networks
- object detection networks


detail from the visualization of ResNet-18 // Graphcore

Lecture Overview

- more on transfer learning
- visualizing neuron activations
- visualizing class activations
- pre-images
- adversarial examples
- adversarial training

Disclaimer: Much of the material and slides for this lecture were borrowed from
—Andrea Vedaldi's tutorial on Understanding Visual Representations
—Wojciech Samek's talk on Towards explainable Deep Learning
—Efstratios Gavves and Max Willing's UvA deep learning class
—Fei-Fei Li, Justin Johnson and Serana Yeung's CS231n class
—Ian Goodfellow's talk on Adversarial Examples and Adversarial Training
—Justin Johnson's EECS 498/598 class

Image Representations

$\mathbf{x}$

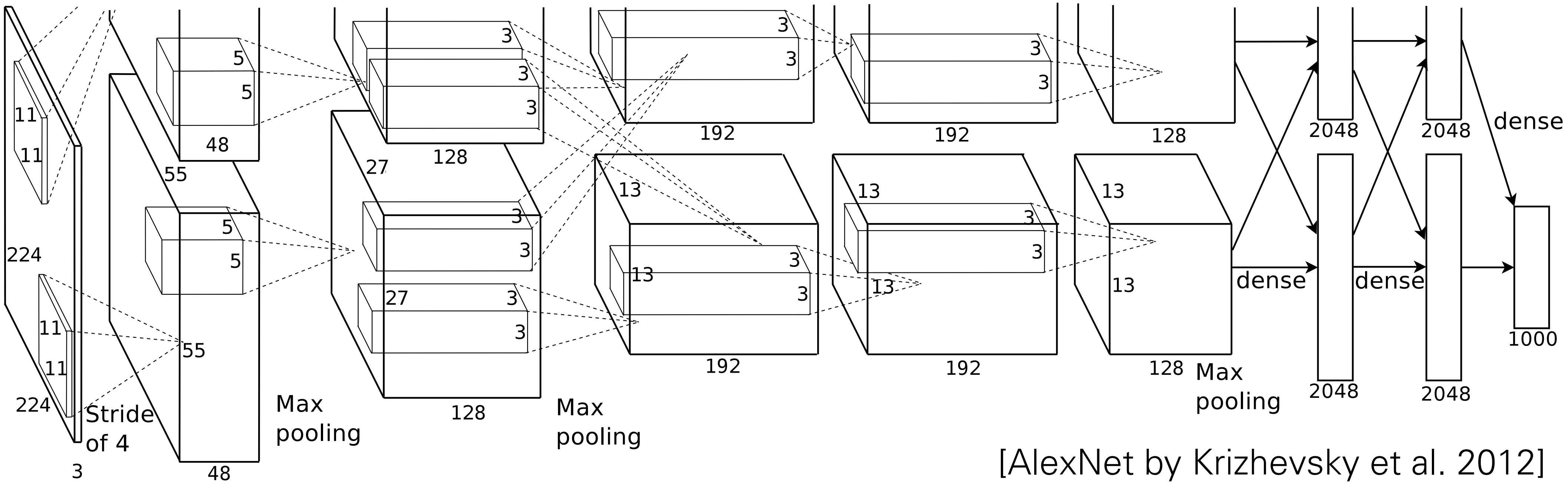
encoder Φ

representation

$\Phi(\mathbf{x}) \in \mathbb{R}^d$

- An **encoder** maps the data into a **vectorial representation**
- Facilitate labelling of images, text, sound, videos, ...

Modern Convolutional Nets



[AlexNet by Krizhevsky et al. 2012]

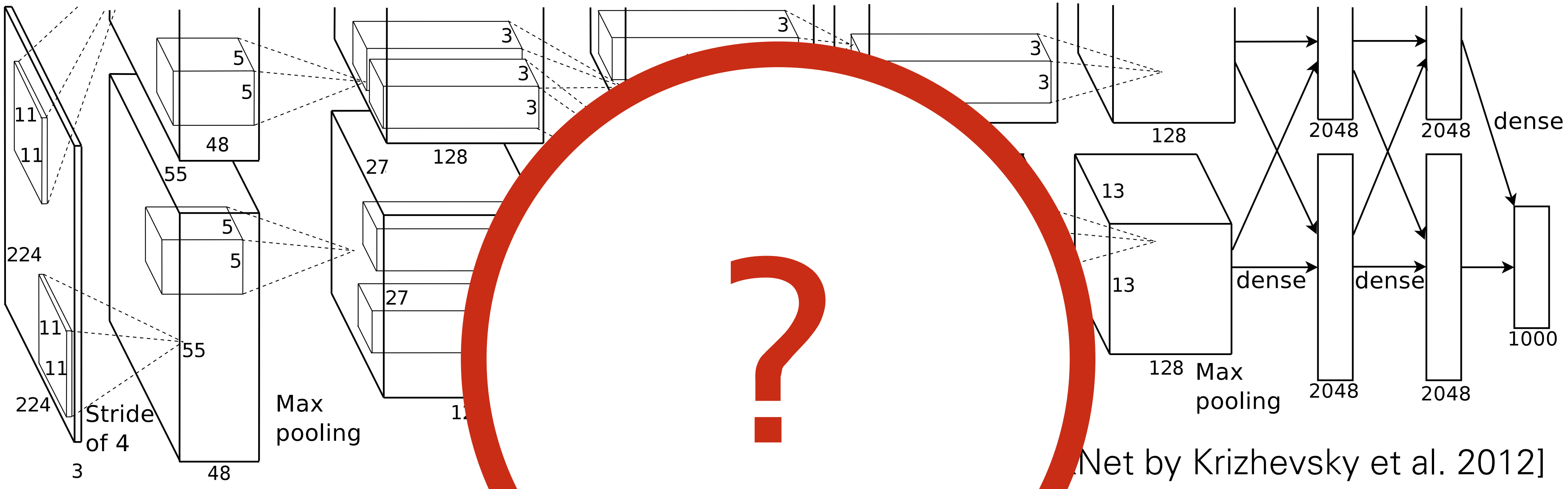
Excellent **performance** in most image understanding tasks

Learn a sequence of **general-purpose representations**

Millions of parameters learned from data

The "**meaning**" of the representation is unclear

Modern Convolutional Nets



Net by Krizhevsky et al. 2012]

Excellent **performance** in most
understanding tasks

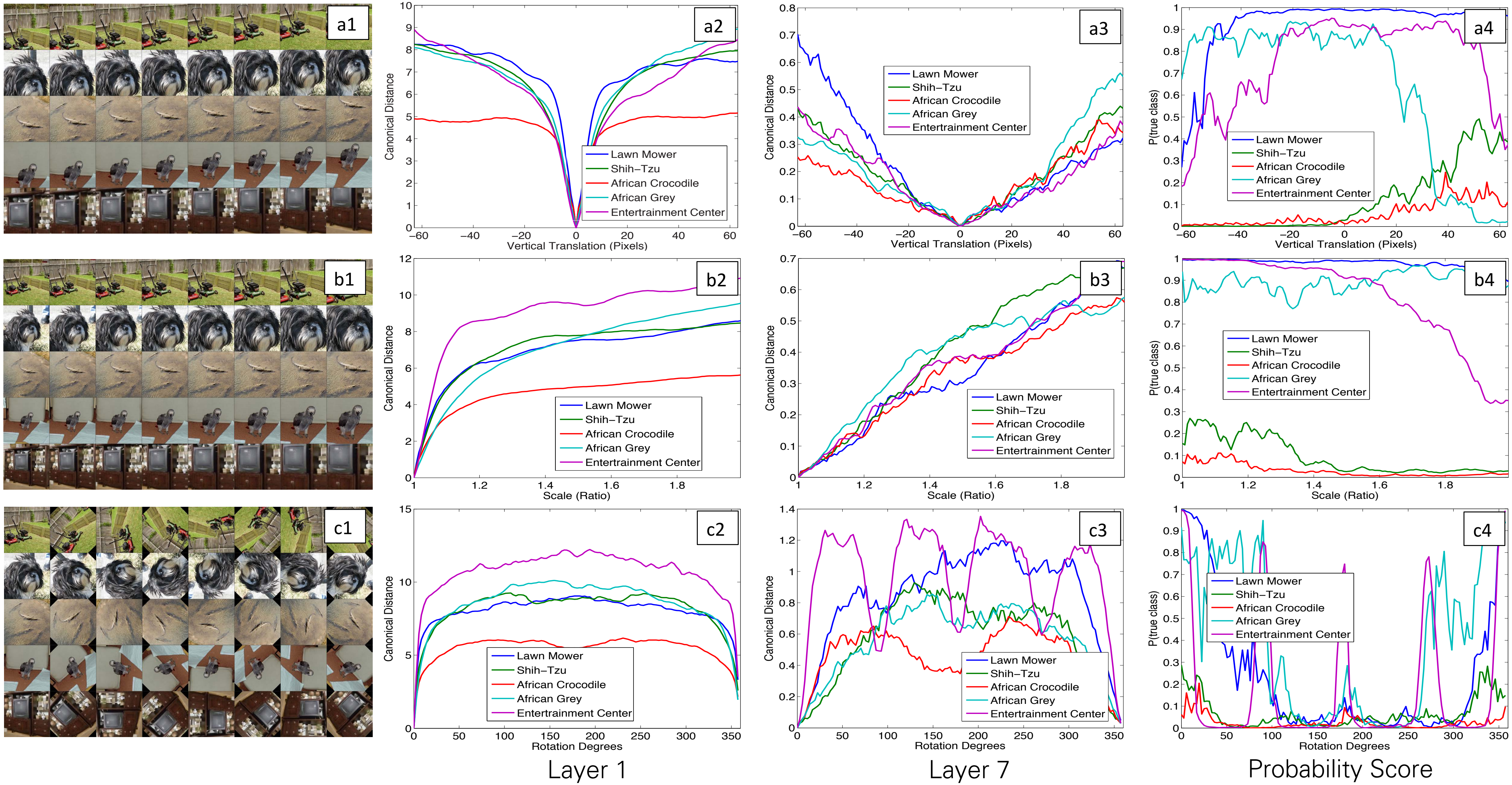
Learn a sequence of **general-purpose representations**

parameters learned from data

meaning” of the representation is
unclear

Transfer Learning with Deep Networks

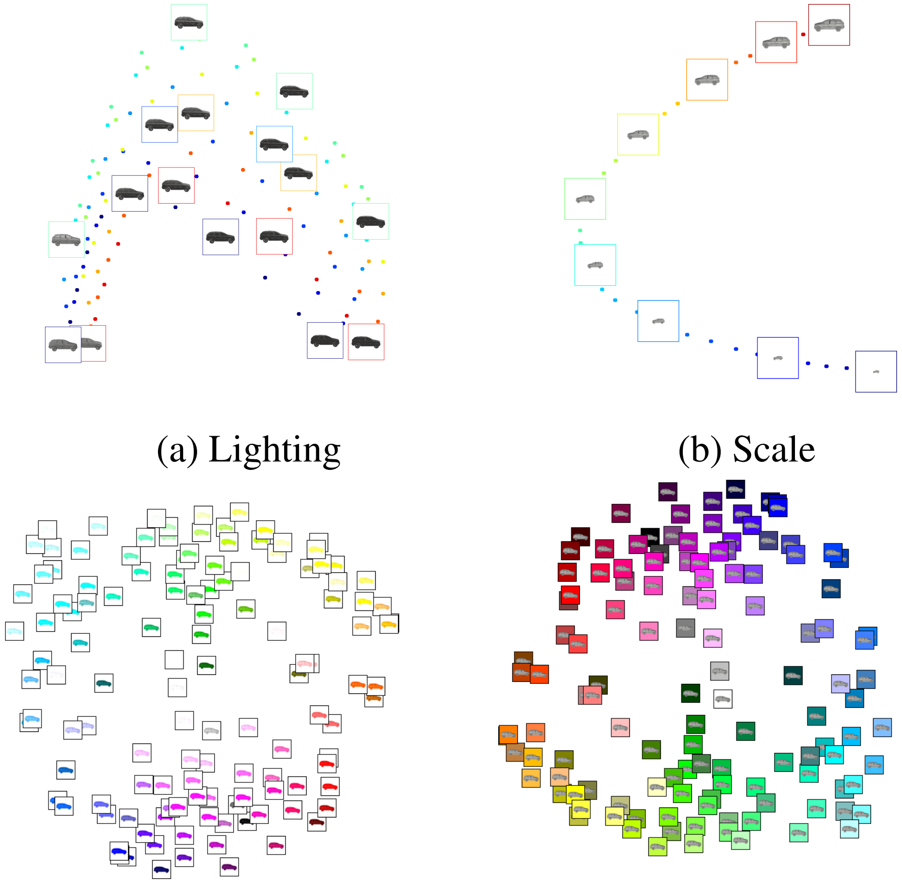
Invariance and Covariance



Matthew D. Zeiler, Rob Fergus. **Visualizing and Understanding Convolutional Networks**. arXiv 2013.

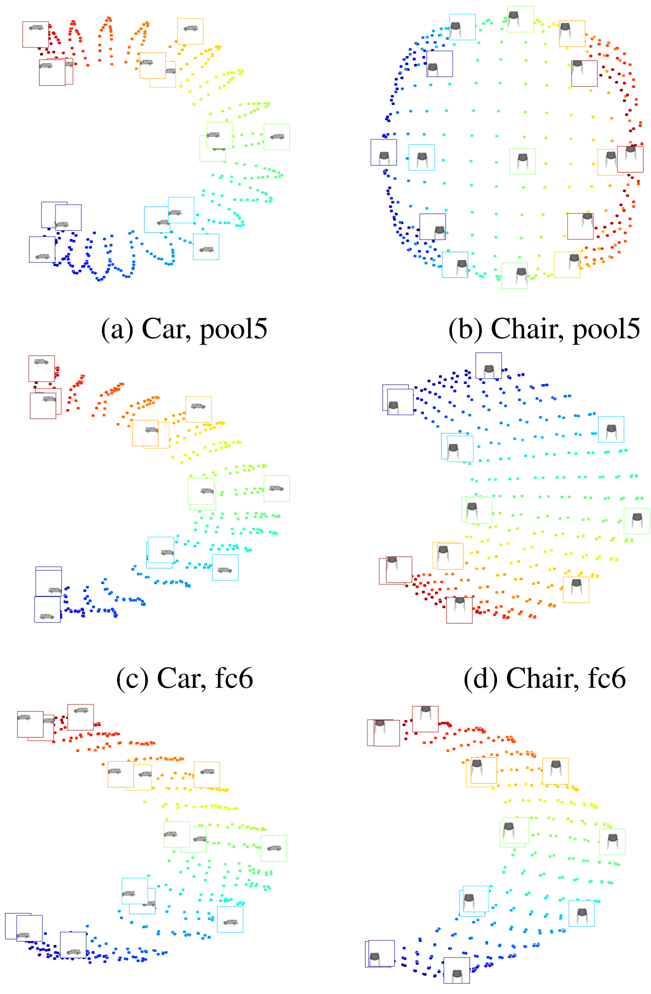
Filter Invariance and Equivariance

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



(a) Lighting (b) Scale

(c) Object color (d) Background color



(a) Car, pool5 (b) Chair, pool5

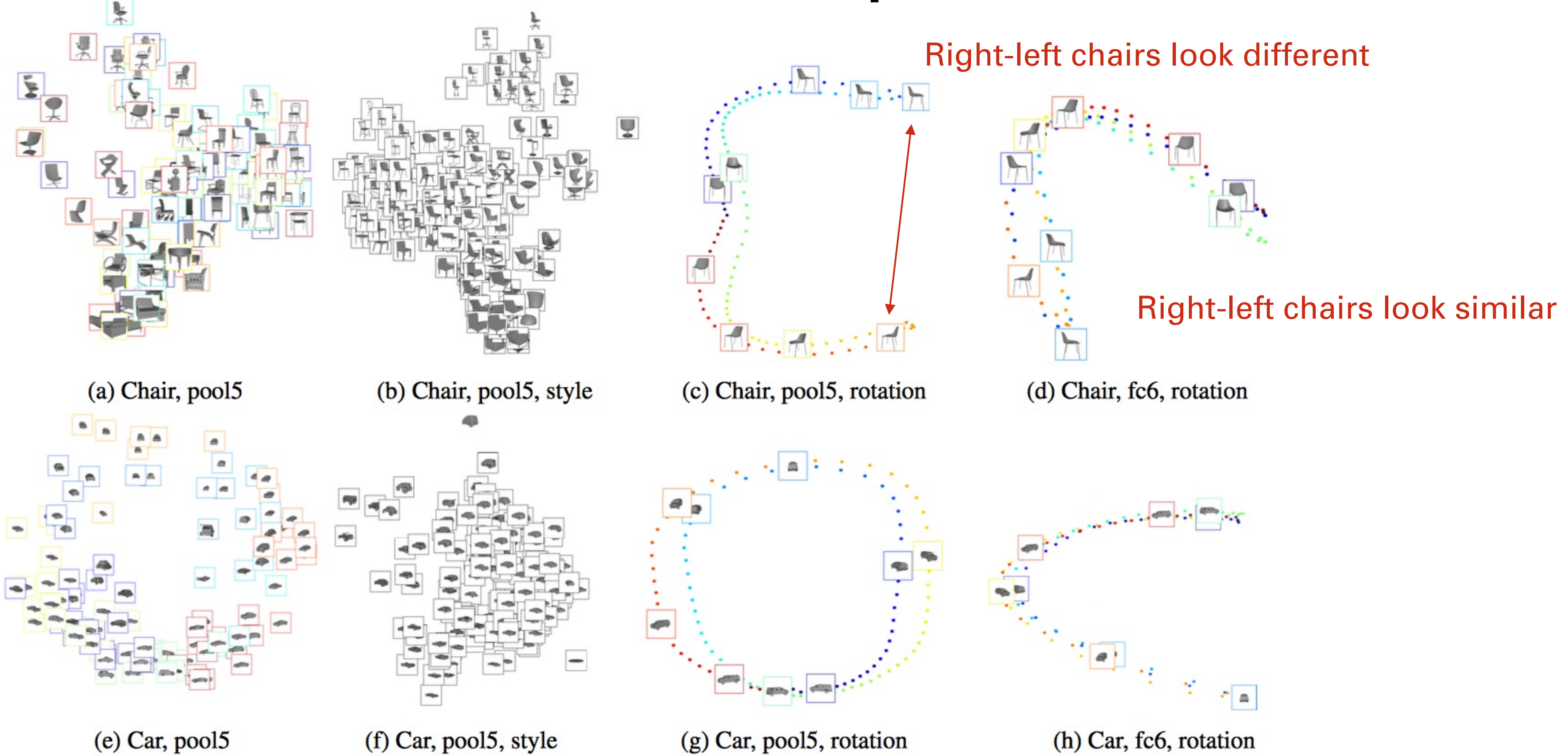
(c) Car, fc6 (d) Chair, fc6

(e) Car, fc7 (f) Chair, fc7

		pool5	fc6	fc7
Viewpoint	Places	26.8 % 8.5	21.4 % 7.0	17.8 % 5.9
	AlexNet	26.4 % 8.3	19.4 % 7.2	15.6 % 6.0
	VGG	21.2 % 10.0	16.4 % 7.7	12.3 % 6.2
Style	Places	26.8 % 136.3	39.1 % 105.5	49.4 % 54.6
	AlexNet	28.2 % 121.1	40.3 % 125.5	49.4 % 96.7
	VGG	26.4 % 181.9	44.3 % 136.3	56.2 % 94.2
Δ^L	Places	46.8 %	39.5 %	32.9 %
	AlexNet	45.0 %	40.3 %	35.0 %
	VGG	52.4 %	39.3 %	31.5 %

Mathieu Aubry and Bryan C. Russell. **Understanding deep features with computer-generated imagery**. ICCV 2015.

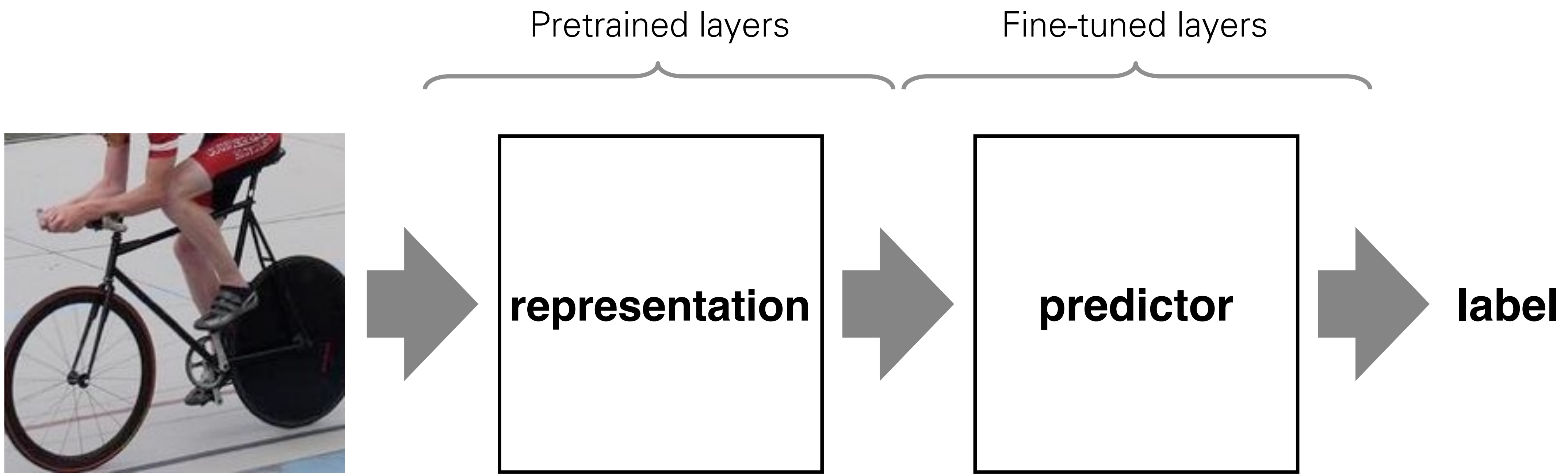
Filter Invariance and Equivariance



Mathieu Aubry and Bryan C. Russell. **Understanding deep features with computer-generated imagery.** ICCV 2015.

Pre-training and Transfer Learning

[Evaluations in A. S. Razavian, 2014, Chatfield et al., 2014]



CNN as universal representations

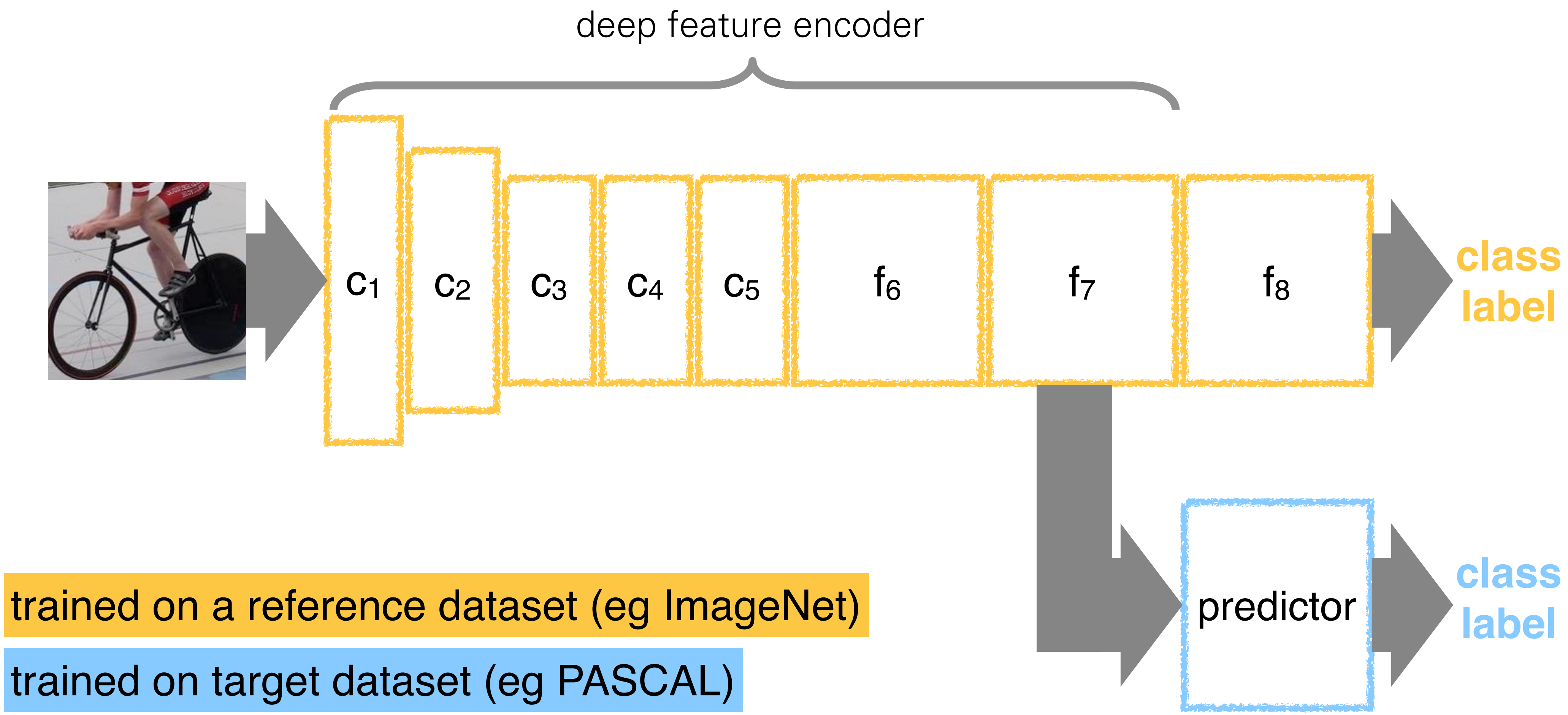
- First several layers in most CNNs are generic
- They can be reused when training data is comparatively scarce.

Application

- Pre-train on ImageNet classification 1M images
- Cut at some deep conv or FC layer to get features

Transfer Learning

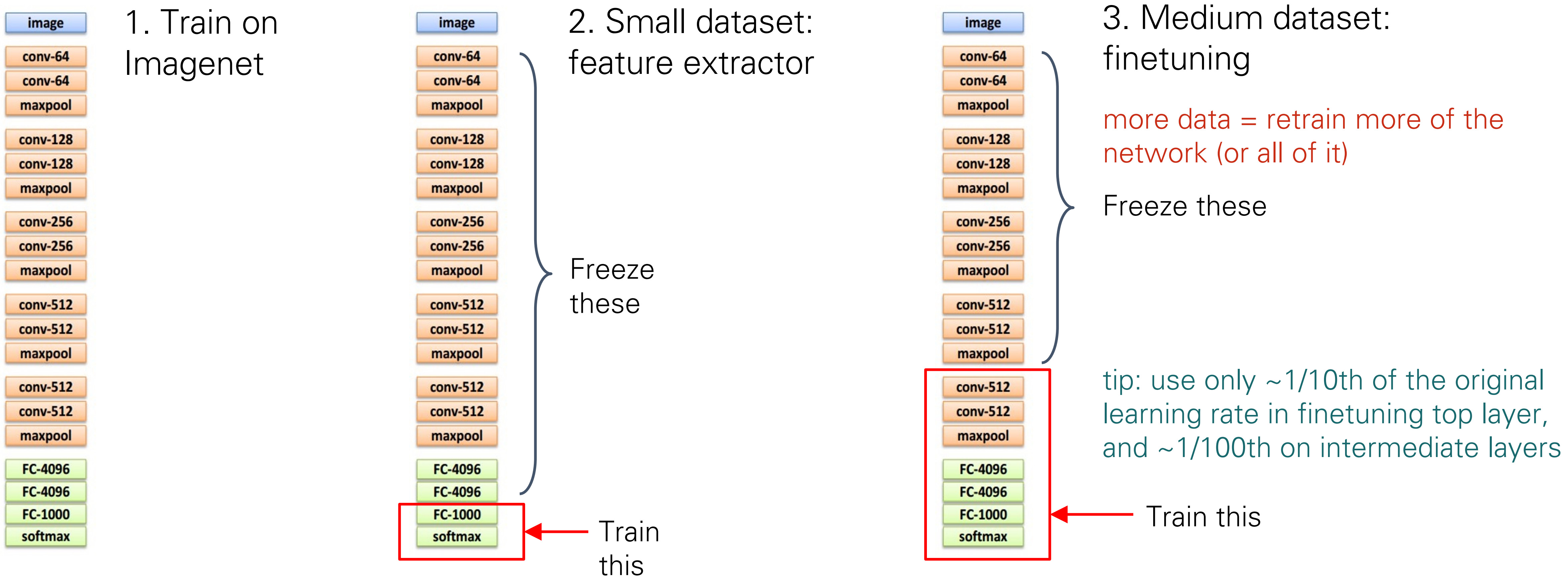
Deep representations are generic



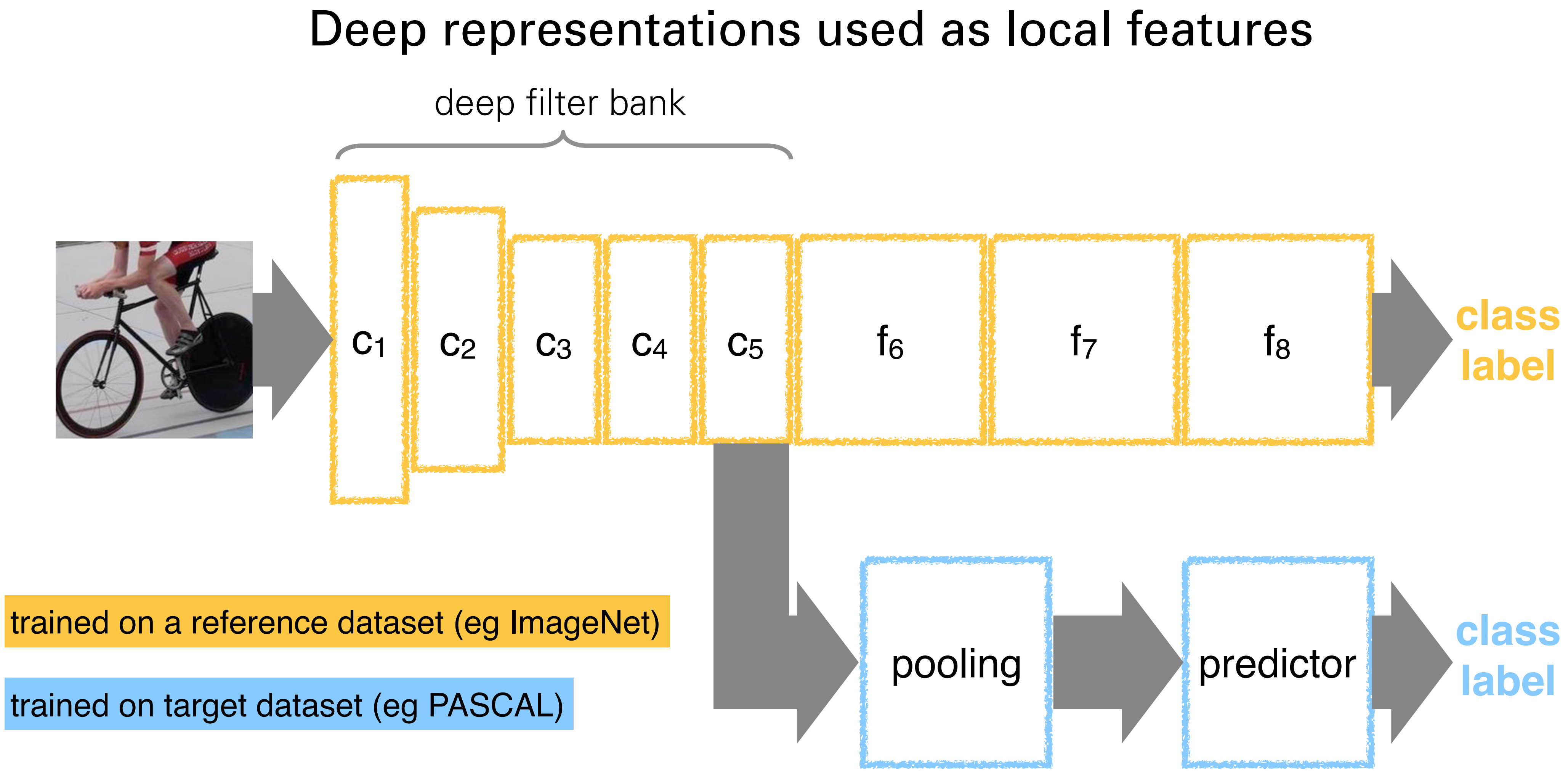
- A general purpose deep encoder is obtained by chopping off the last layers of a CNN trained on a large dataset.

Transfer Learning with CNNs

- Keep layers 1-7 of our ImageNet-trained model fixed
- Train a new softmax classifier on top using the training images of the new dataset.



CNNs as Filter Banks

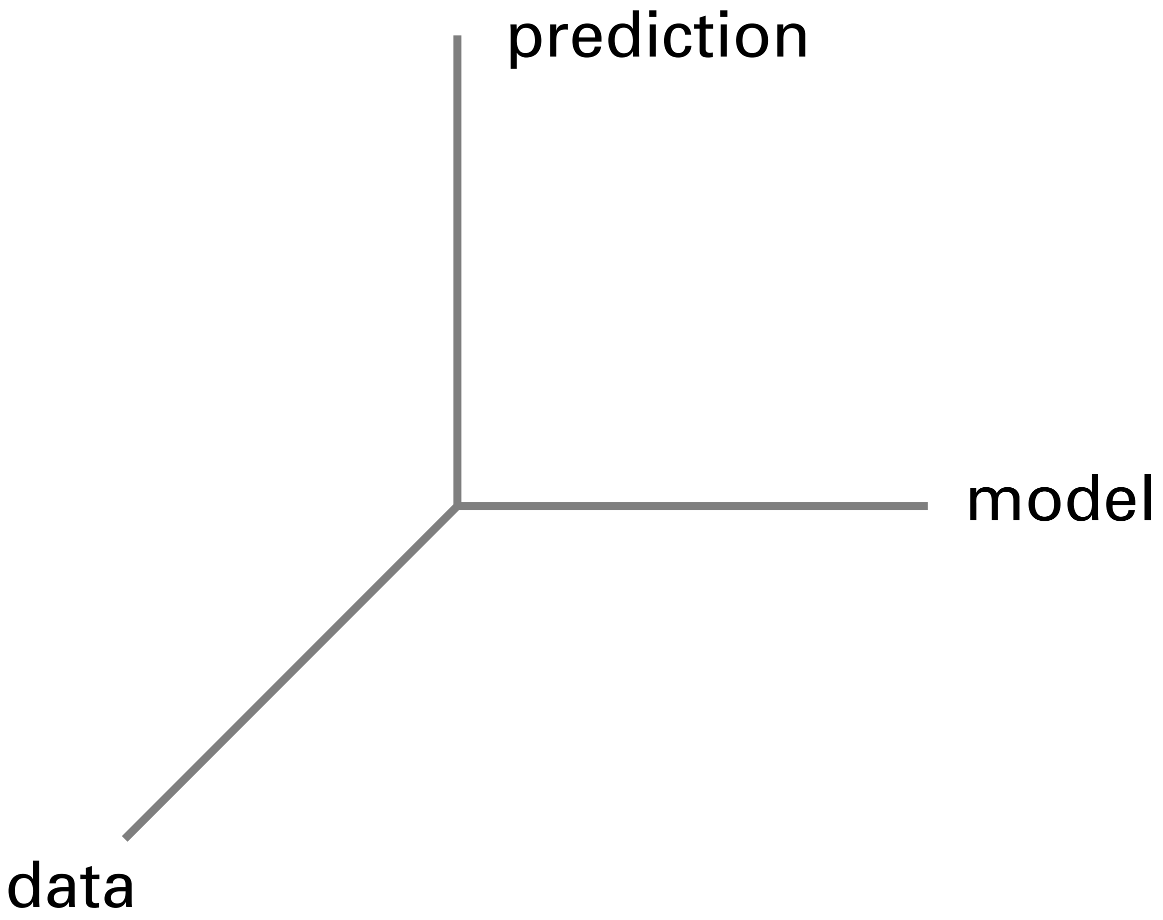


- In R-CNN and similar models, the most important shared component are the convolutional features.

Interpretability

Dimensions of Interpretation

Different dimensions
of “interpretability”

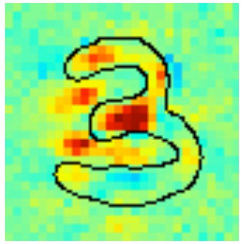


Dimensions of Interpretation

Different dimensions of “interpretability”

prediction

“Explain why a certain pattern x has been classified in a certain way f(x).”



model

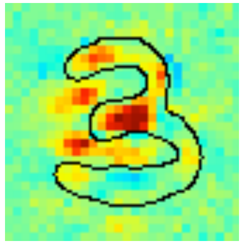
data

Dimensions of Interpretation

Different dimensions of "interpretability"

prediction

"Explain why a certain pattern x has been classified in a certain way f(x)."



model

"What would a pattern belonging to a certain category typically look like according to the model."

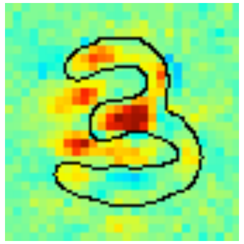
data

Dimensions of Interpretation

Different dimensions
of “interpretability”

prediction

“Explain why a certain pattern x has been classified in a certain way f(x).”



model

“What would a pattern belonging to a certain category typically look like according to the model.”

data

“Which dimensions of the data are most relevant for the task.”

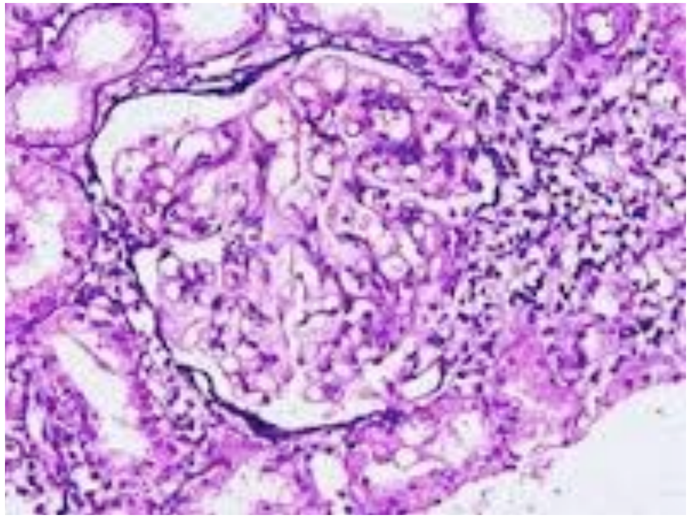
Why Interpretability?

1) Verify that classifier works as expected

Wrong decisions can be costly and dangerous

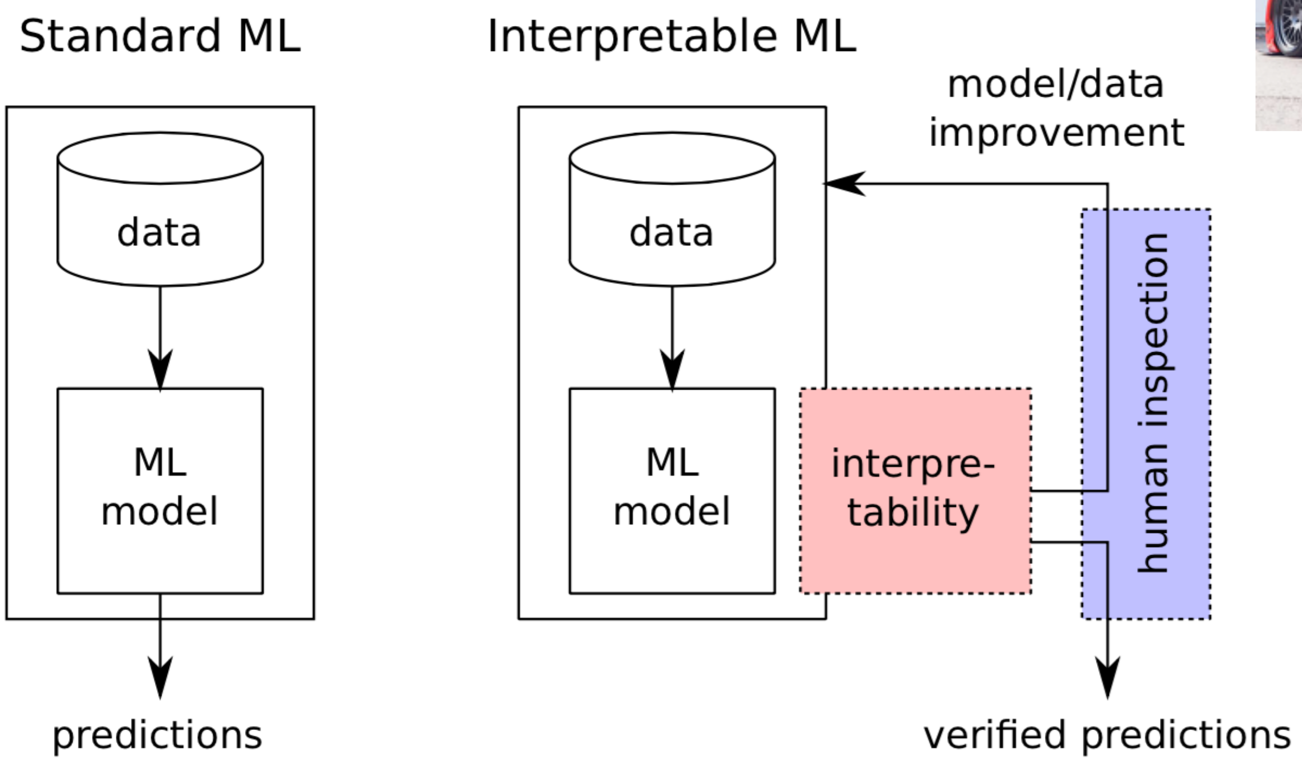
"Autonomous car crashes, because it wrongly recognizes ..."

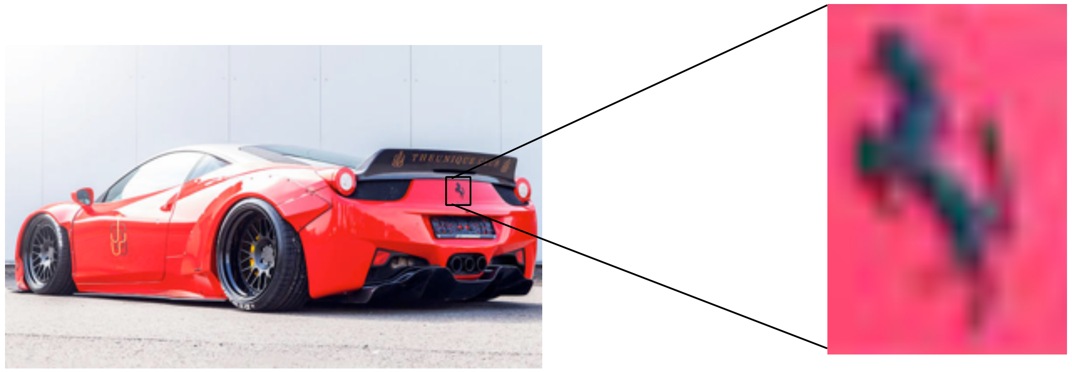
"AI medical diagnosis system misclassifies patient's disease ..."



Why Interpretability?

2) Improve classifier





Generalization error

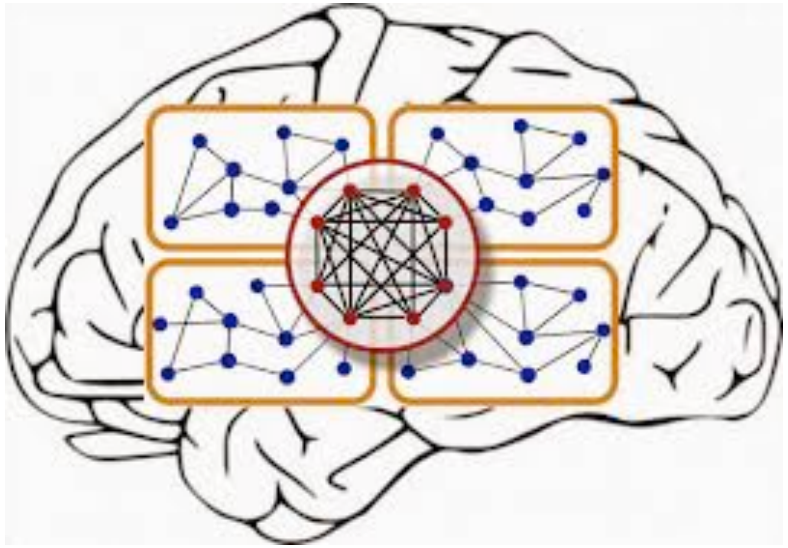
Generalization error + human experience

Why Interpretability?

3) Learn from the learning machine

"It's not a human move. I've never seen a human play this move." (Fan Hui)

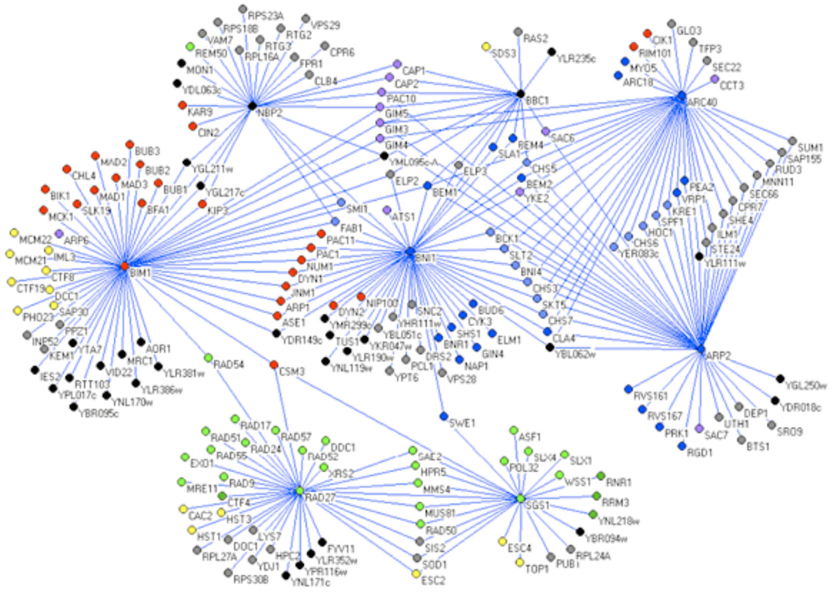
Old promise:
"Learn about the human brain."

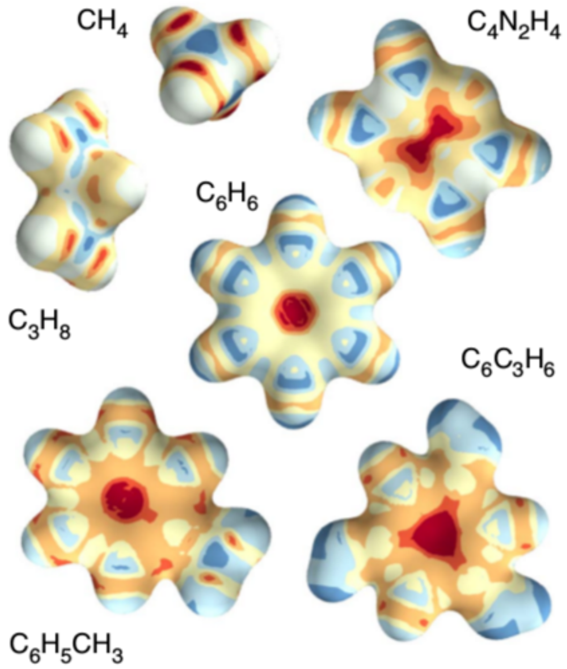


Why Interpretability?

4) Interpretability in the sciences

Learn about the physical / biological / chemical mechanisms.
(e.g. find genes linked to cancer, identify binding sites ...)





Why Interpretability?

5) Compliance to legislation

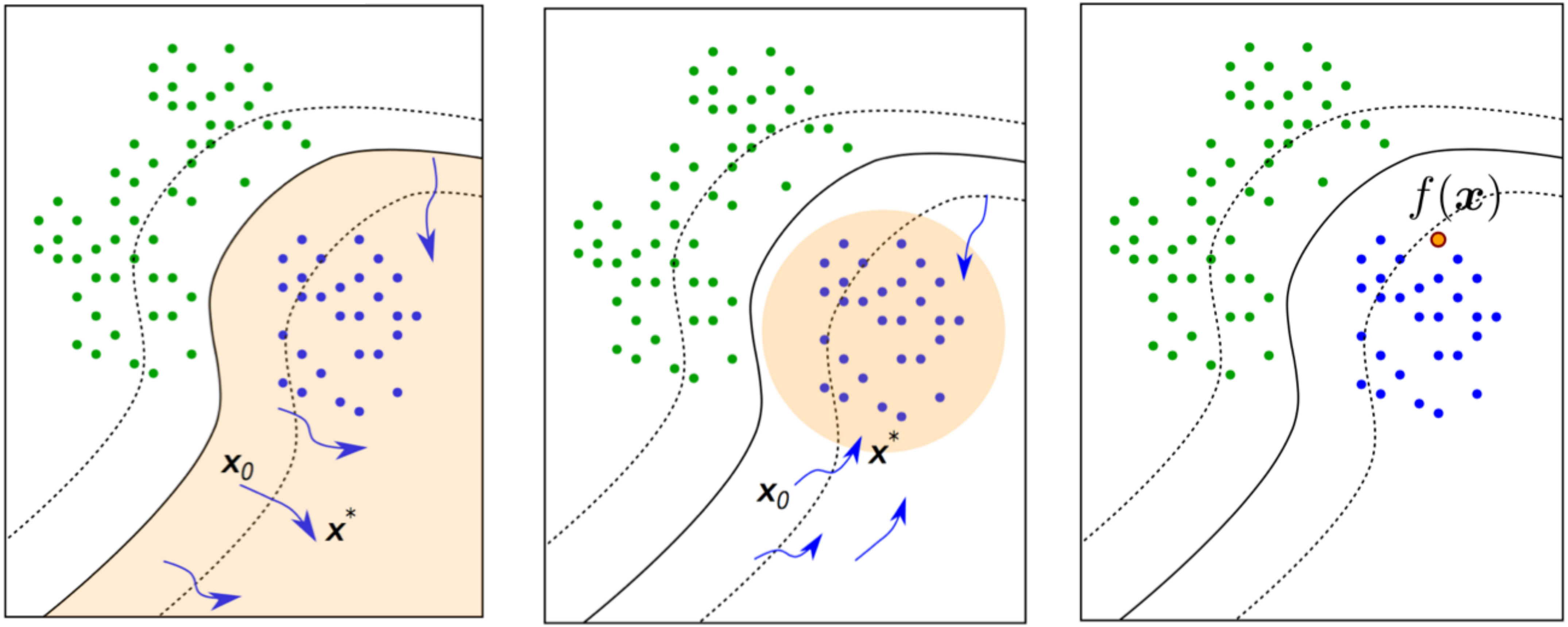
European Union's new General Data Protection Regulation → "right to explanation"

Retain human decision in order to assign responsibility.

"With interpretability we can ensure that ML models work in compliance to proposed legislation."

Dimensions of Interpretation

model analysis → **decision analysis**



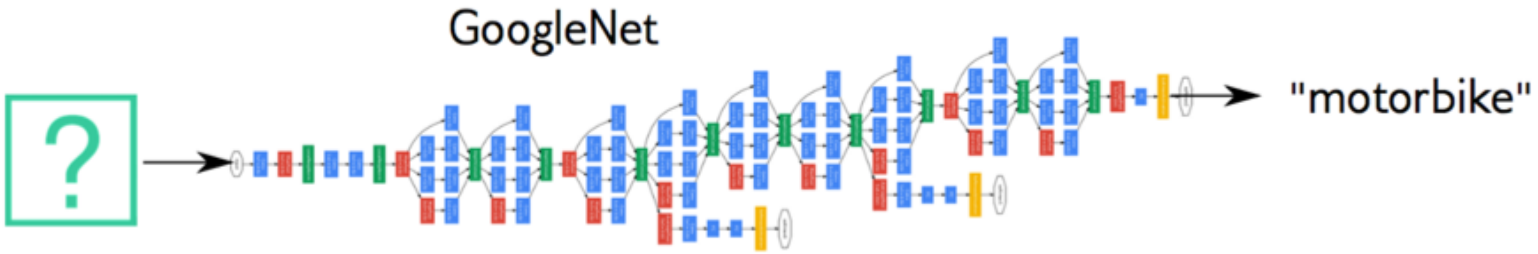
Find the input pattern that maximizes class probability.

Find the most likely input pattern for a given class.

Explain individual prediction.

Dimensions of Interpretation

- **Finding a prototype:**



Question: How does a “motorbike“ typically look like

- **Individual explanation:**

Question: Why is <u>this</u> example classified as motorbike?

Some Approaches

- Visualize patches that maximally activate neurons
- Visualize the weights
- Visualize the representation space (e.g. with t-SNE)
- Occlusion experiments
- Human experiment comparisons
- Deconv approaches (single backward pass)
- Optimization over image approaches (optimization)

Visualize patches that maximally activate neurons

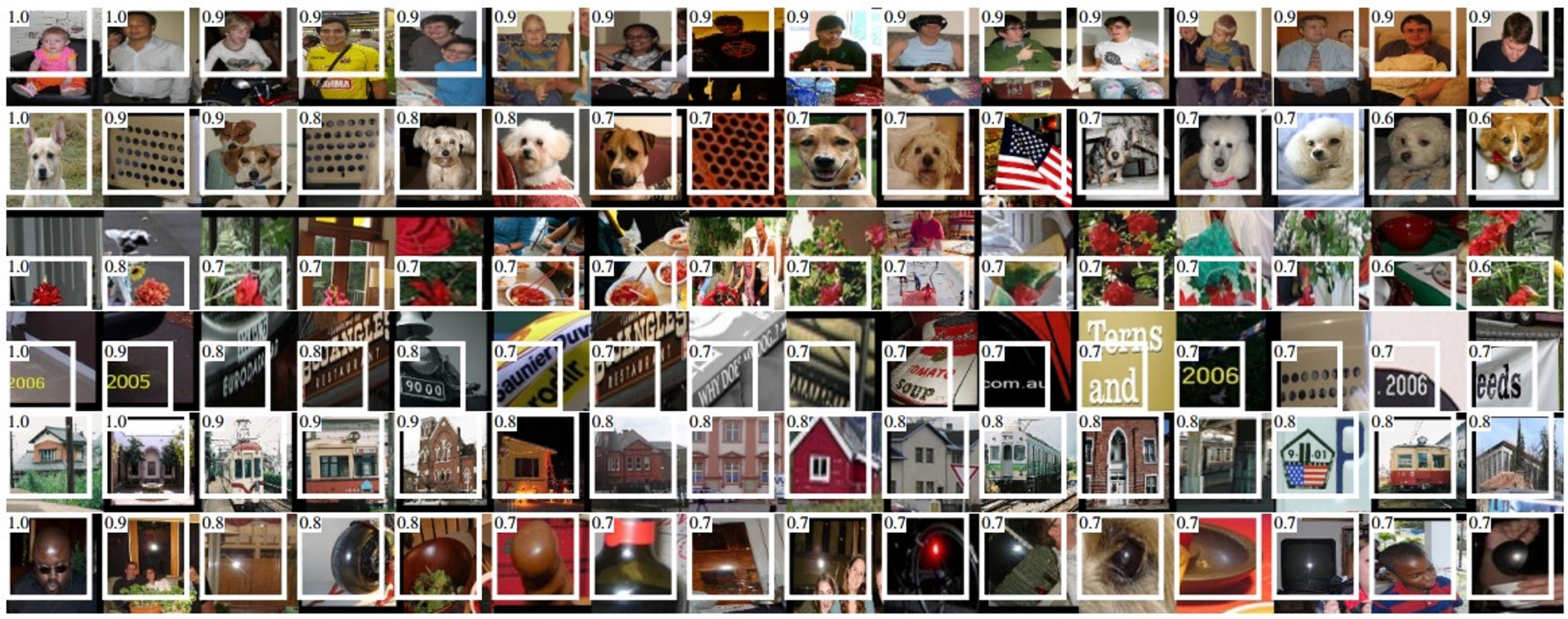
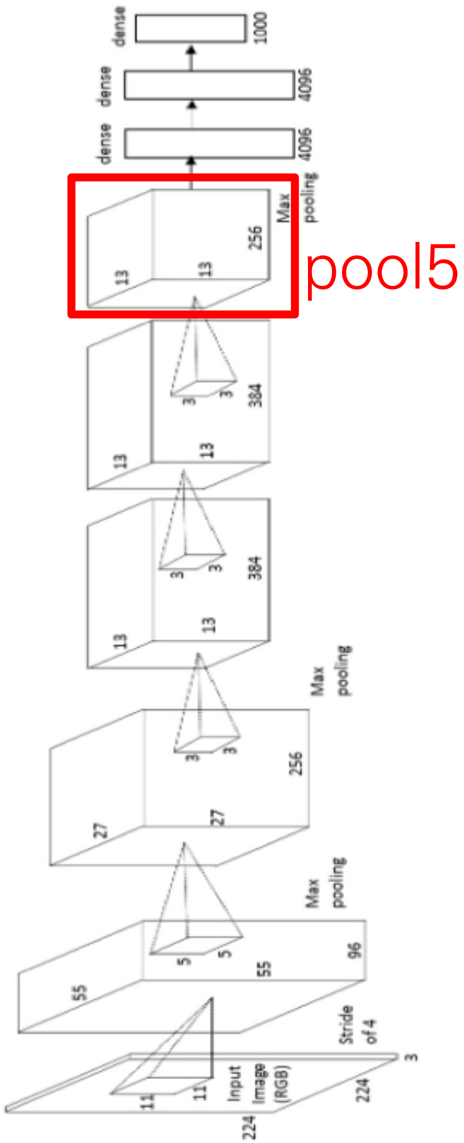


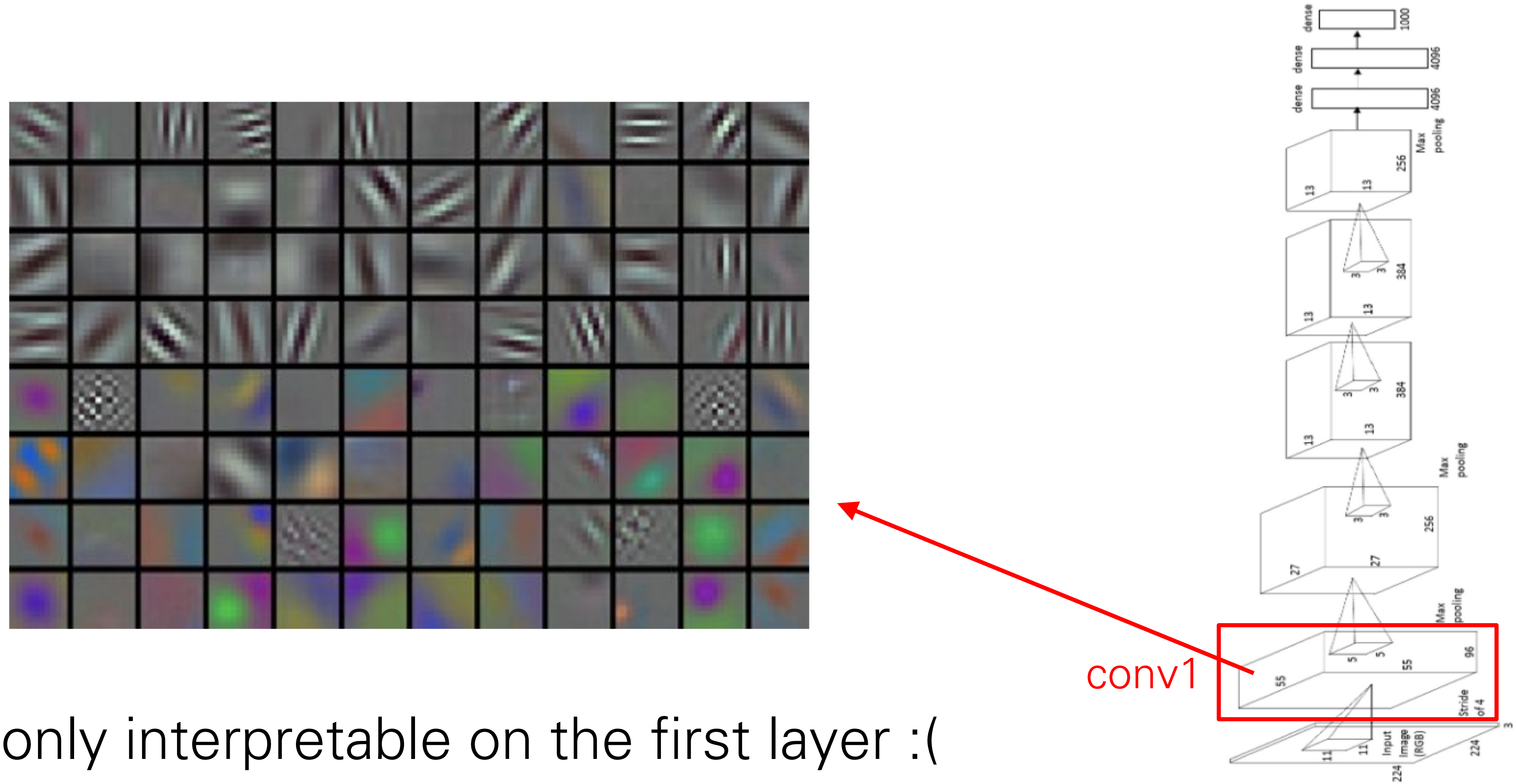
Figure 4: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

Rich feature hierarchies for accurate object detection and semantic segmentation [Girshick, Donahue, Darrell, Malik]

one-stream AlexNet

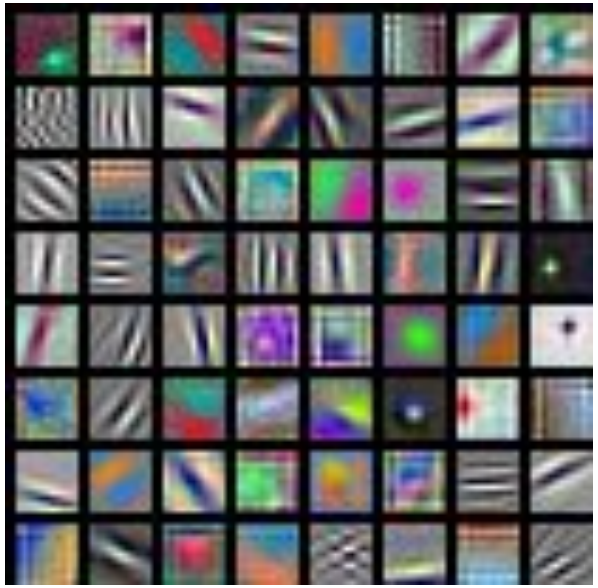


First Layer: Visualize Filters



only interpretable on the first layer :(

First Layer: Visualize Filters

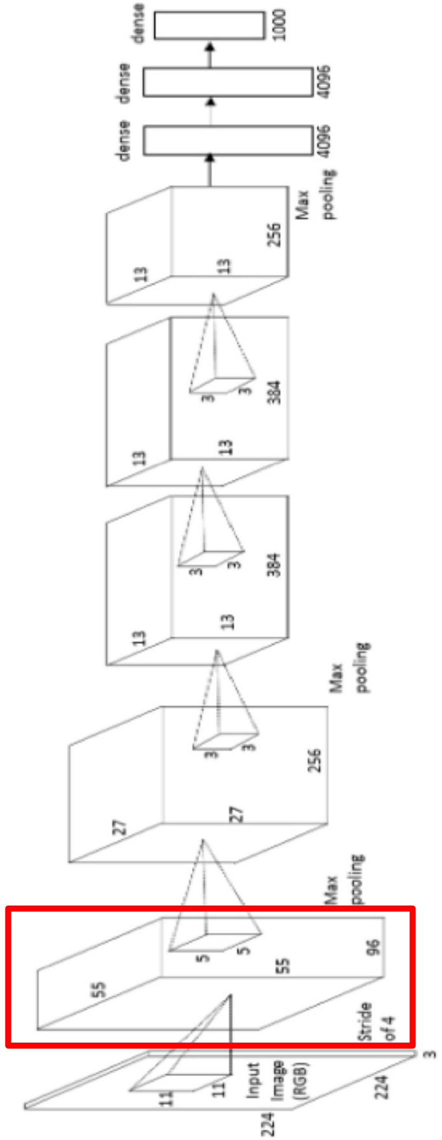


AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7



Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

Higher Layers: Visualize Filters

layer 1 weights

layer 2 weights

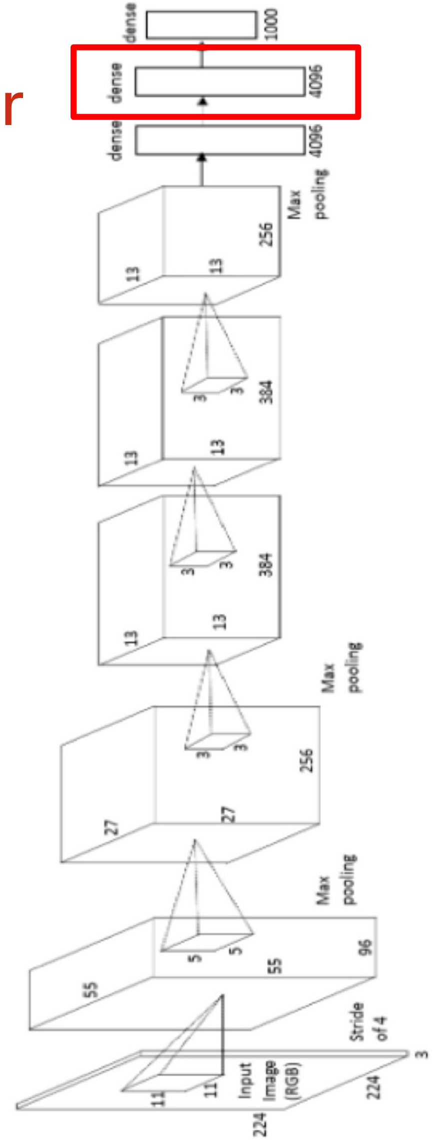
layer 3 weights

you can still do it for higher layers, it's just not that interesting

(these are taken from ConvNetJS CIFAR-10 demo)

Last Layer

fc7 layer

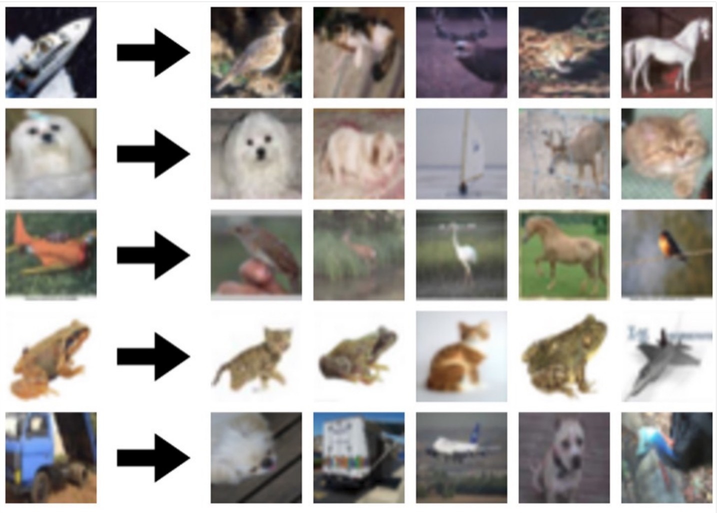


4096-dimensional feature vector for an image
(layer immediately before the classifier)

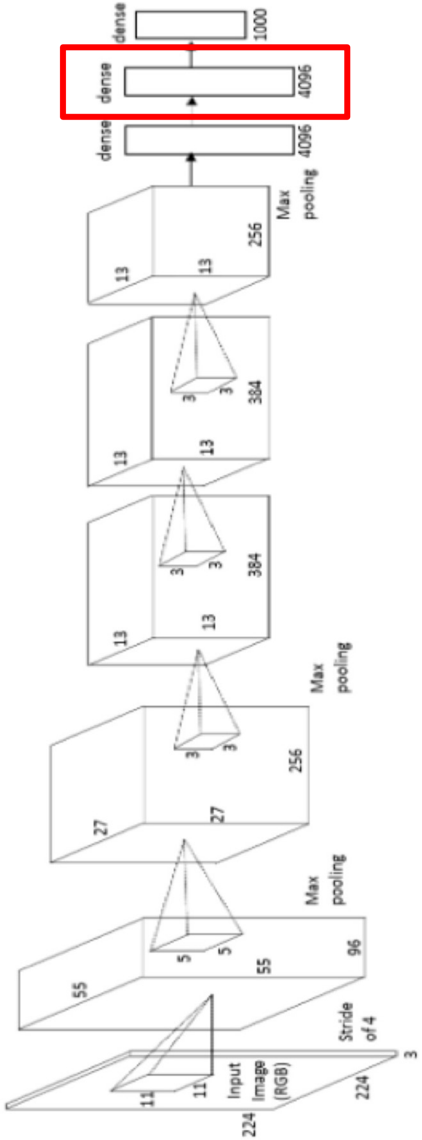
Run the network on many images, collect the feature vectors

Last Layer: Nearest Neighbors

Recall: Nearest neighbors in pixel space



Test image L2 Nearest neighbors in feature space



Last Layer: Dimensionality Reduction

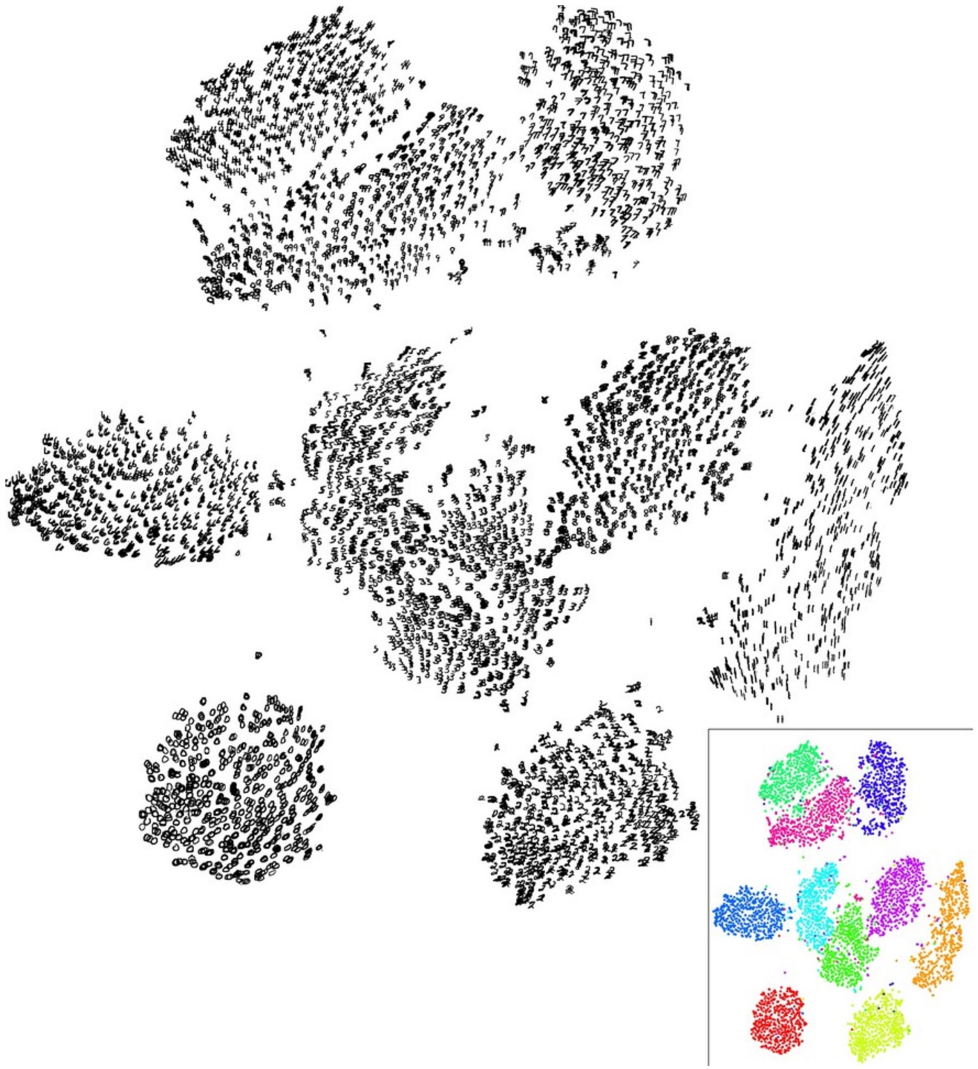
t-SNE visualization

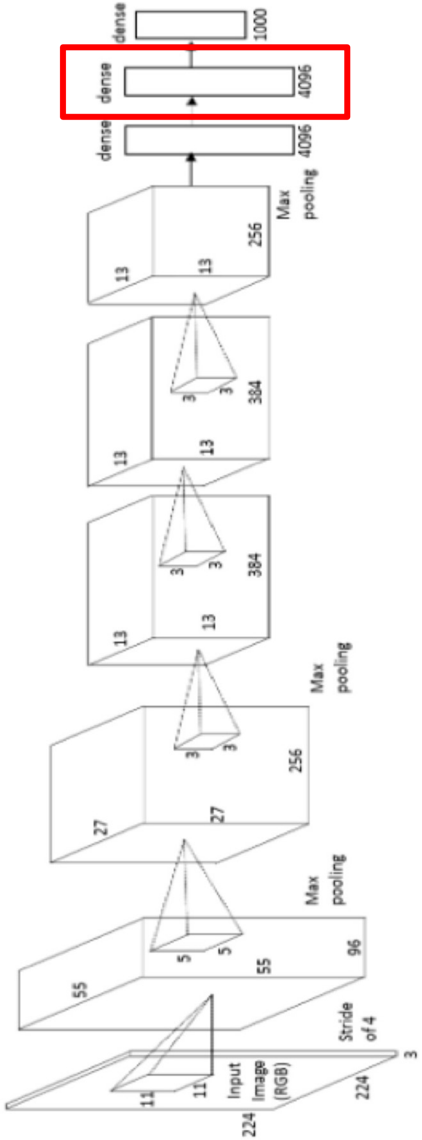
[van der Maaten & Hinton]

- Embed high-dimensional points so that locally, pairwise distances are conserved

- i.e. similar things end up in similar places. dissimilar things end up wherever

- **Right**: Example embedding of MNIST digits (0-9) in 2D

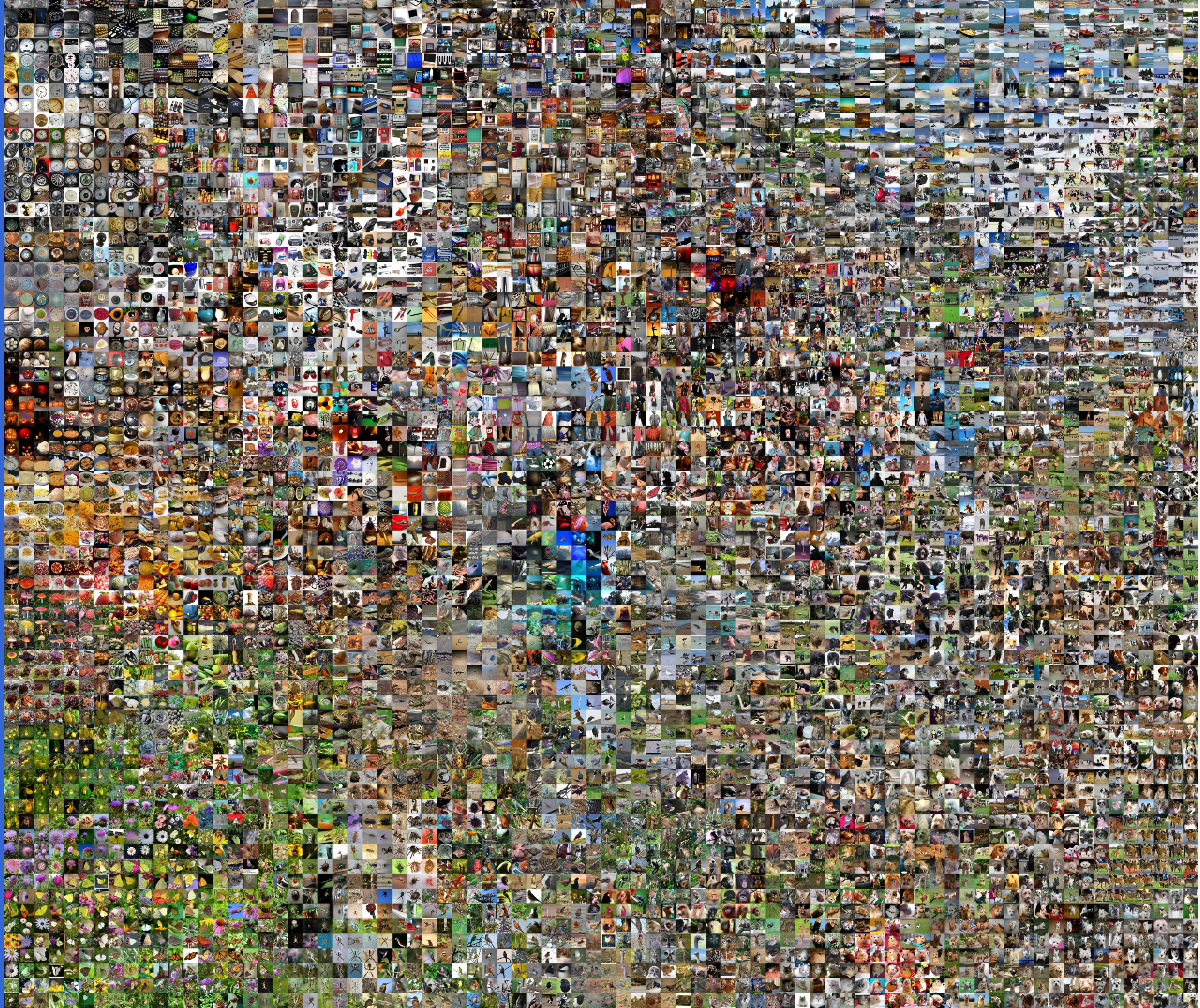




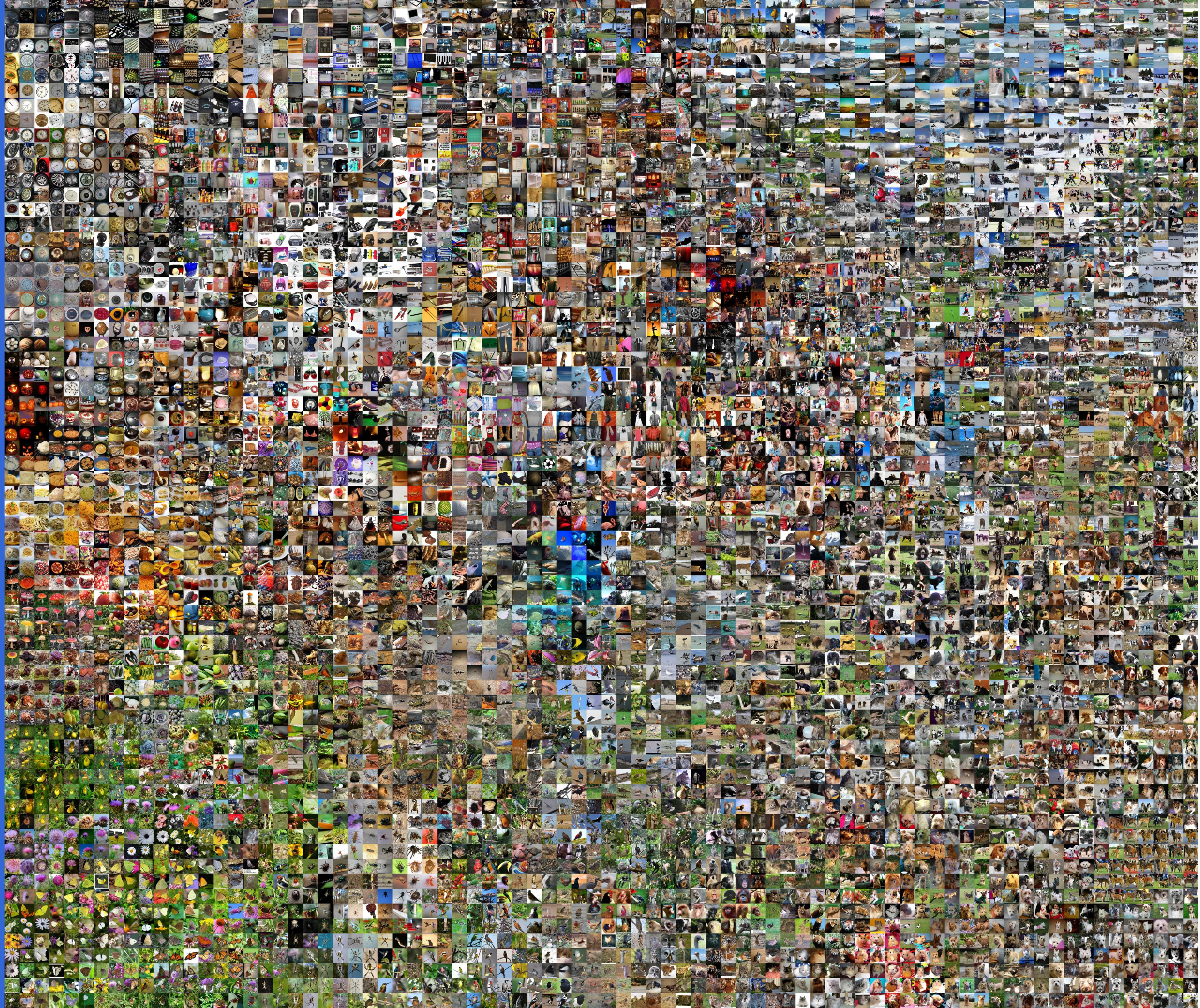
t-SNE visualization:

- two images are placed nearby if their CNN codes are close. See more:

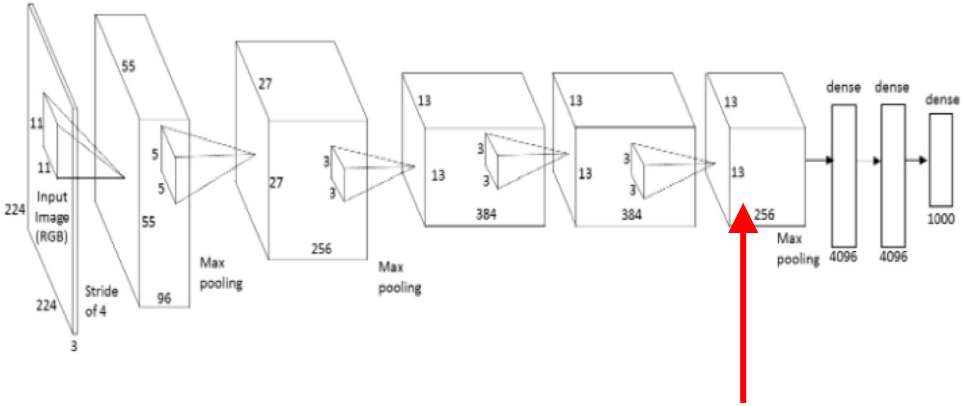
http://cs.stanford.edu/people/karpathy/cnnembed/

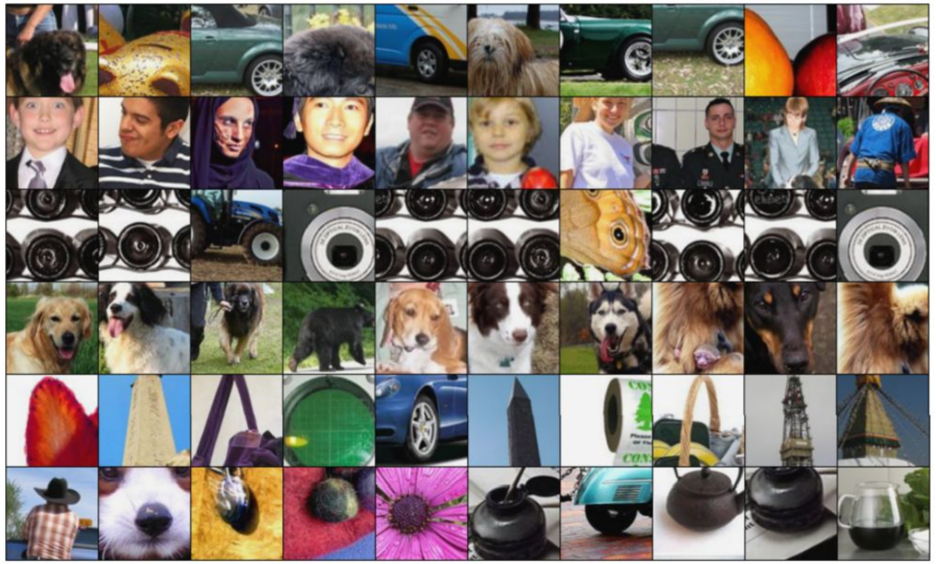


t-SNE visualization:



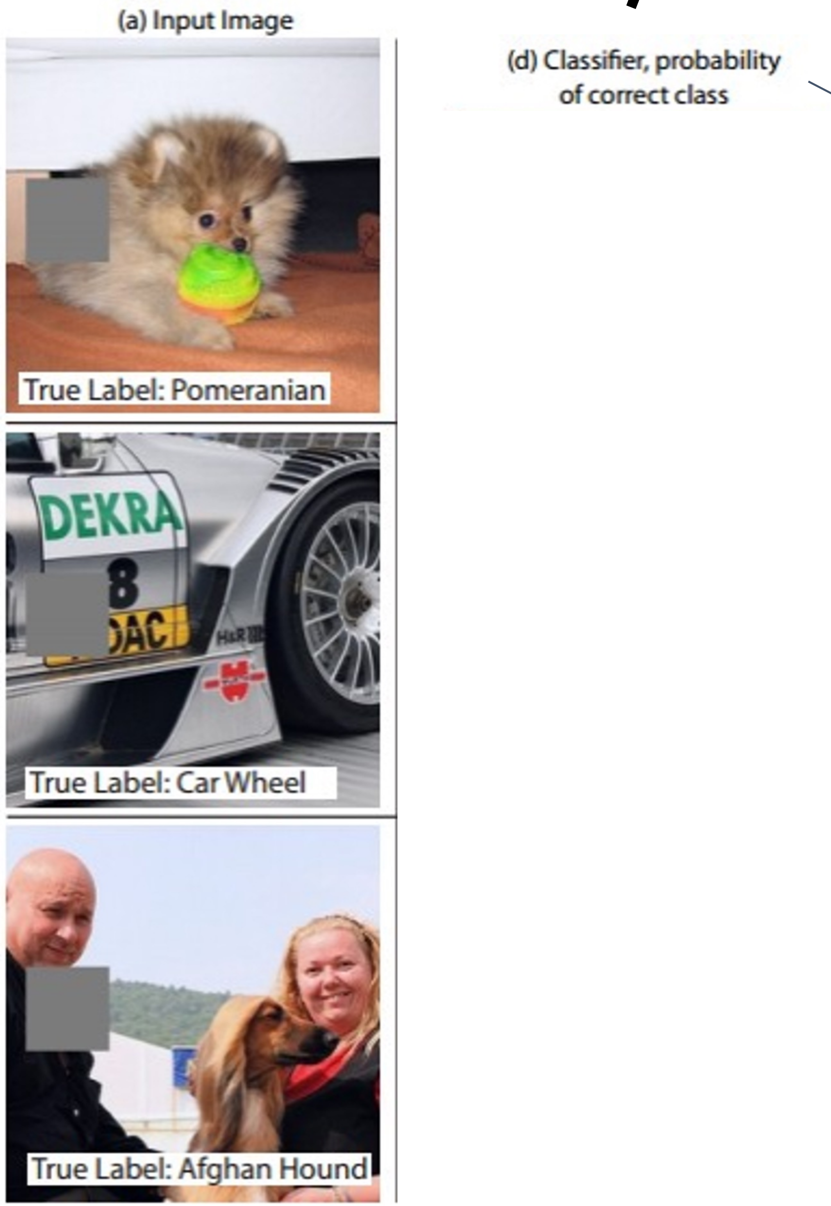
Visualize patches that maximally activate neurons





- Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128
- Run many images through the network, record values of chosen channel
- Visualize image patches that correspond to maximal activations

Which Pixels Matter? Saliency via Occlusion

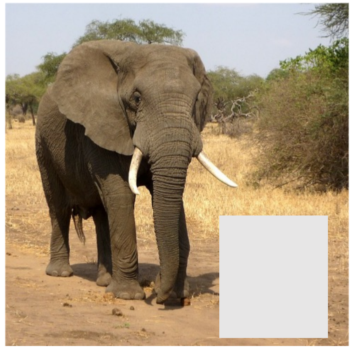


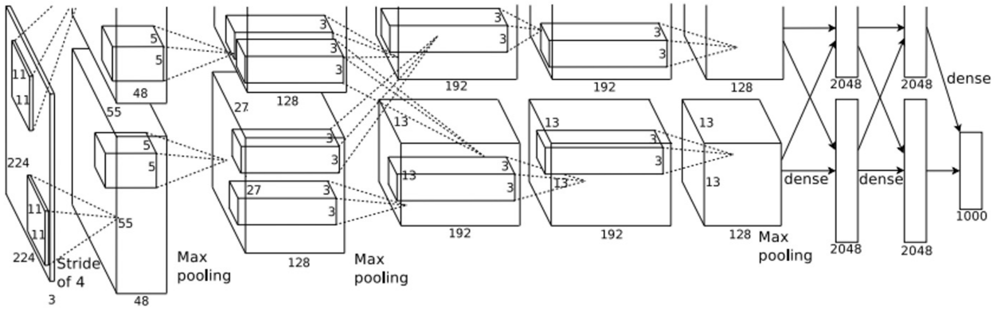
(as a function of the position of the square of zeros in the original image)

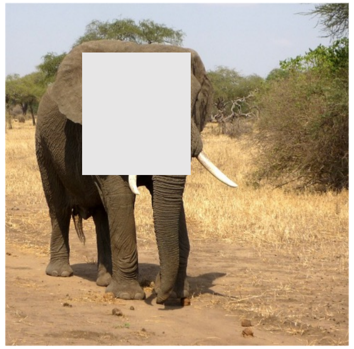
Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

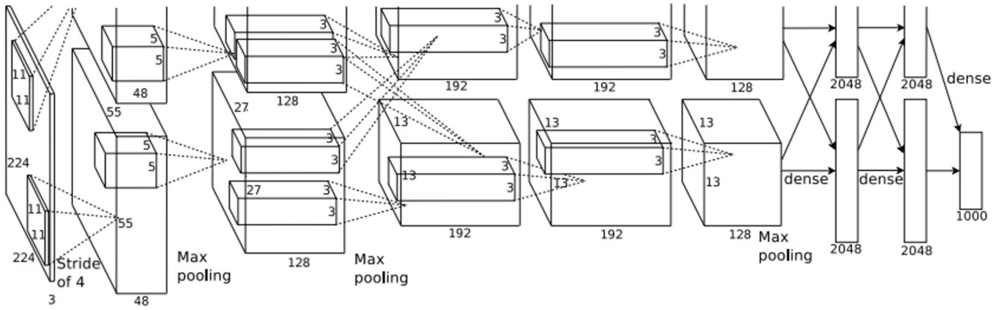
Which Pixels Matter? Saliency via Occlusion

Mask part of the image before feeding to CNN,
check how much predicted probabilities change





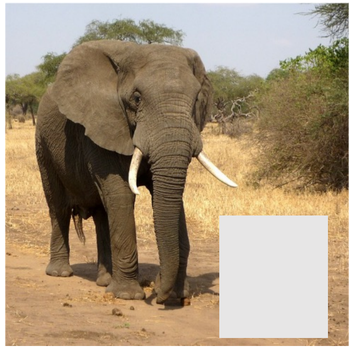


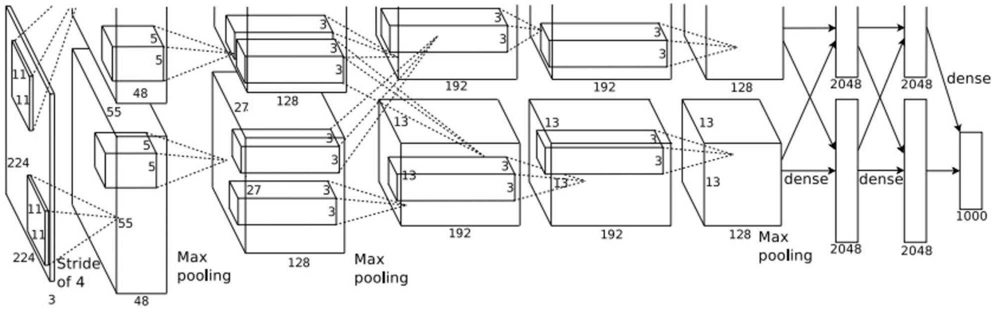


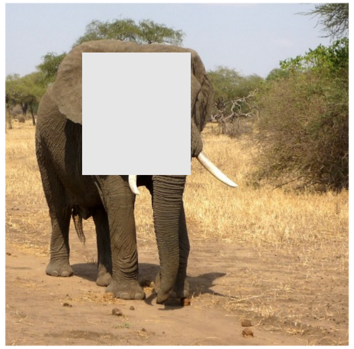
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

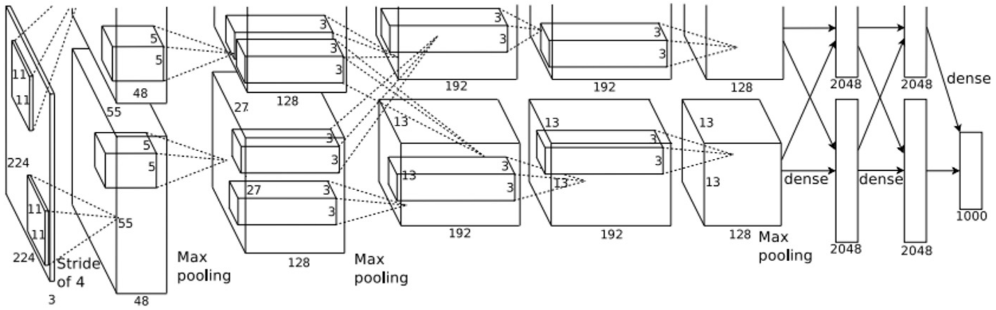
Which Pixels Matter? Saliency via Occlusion

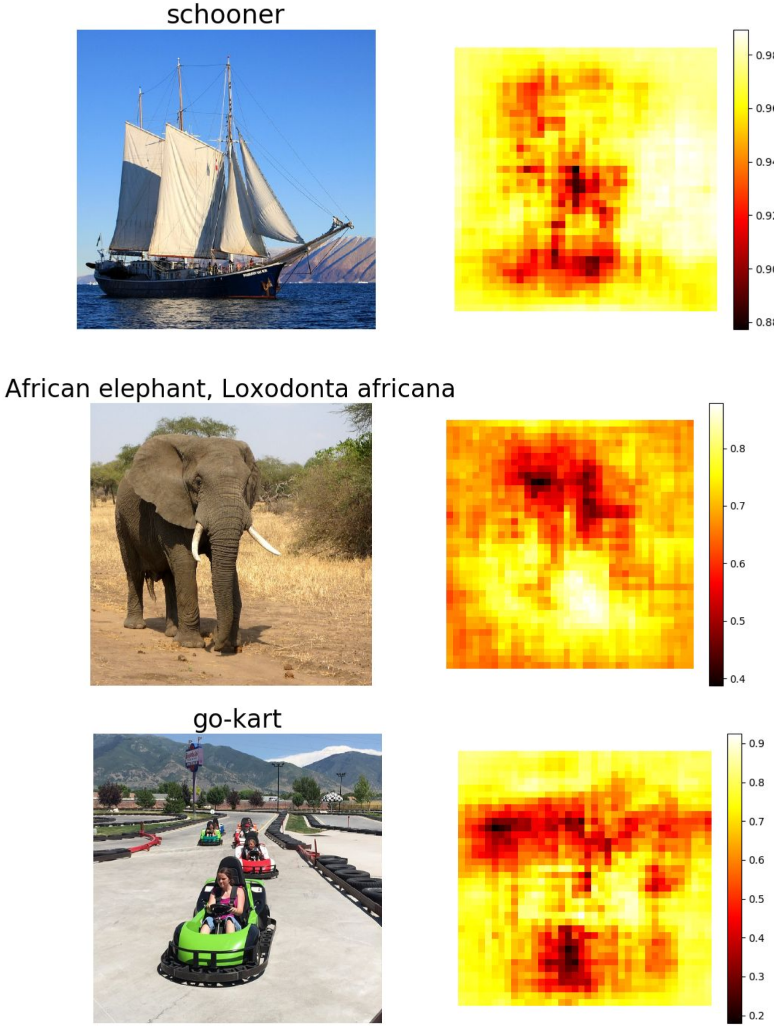
Mask part of the image before feeding to CNN, check how much predicted probabilities change







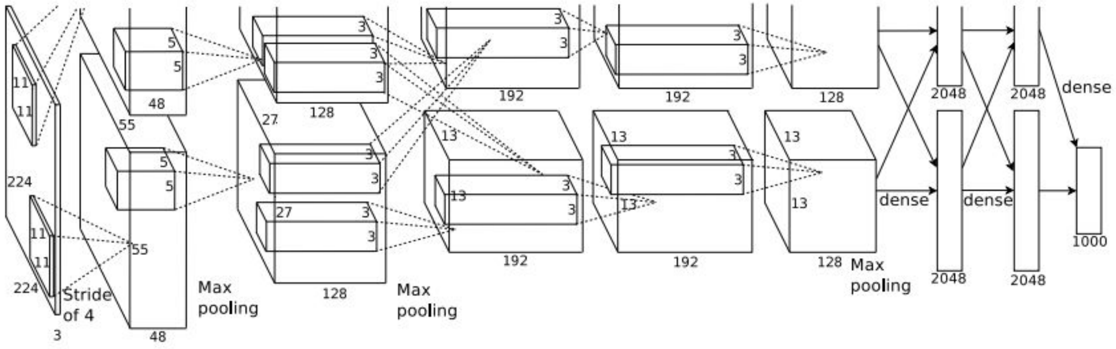




Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Which pixels matter? Saliency via Backprop

Forward pass: Compute probabilities

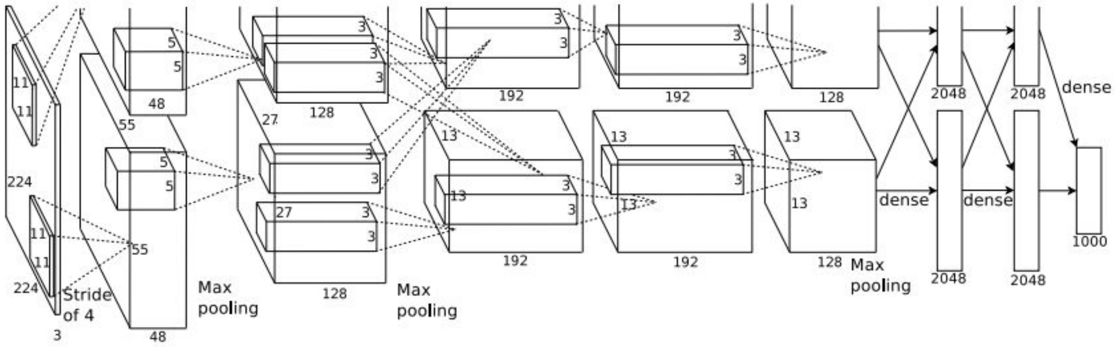


Dog

K. Simonyan, A. Vedaldi and A. Zisserman. Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps. ICLR Workshop 2014

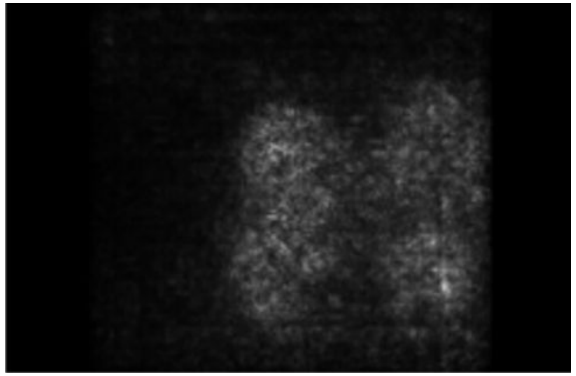
Which pixels matter? Saliency via Backprop

Forward pass: Compute probabilities



Dog

Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels



K. Simonyan, A. Vedaldi and A. Zisserman. Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps. ICLR Workshop 2014

Which pixels matter? Saliency via Backprop

- Given the "monkey" class, what are the most "monkey-ish" parts in my image?

- Approximate S_c around an initial point $\boldsymbol{I}_0$ with the first order Taylor expansion

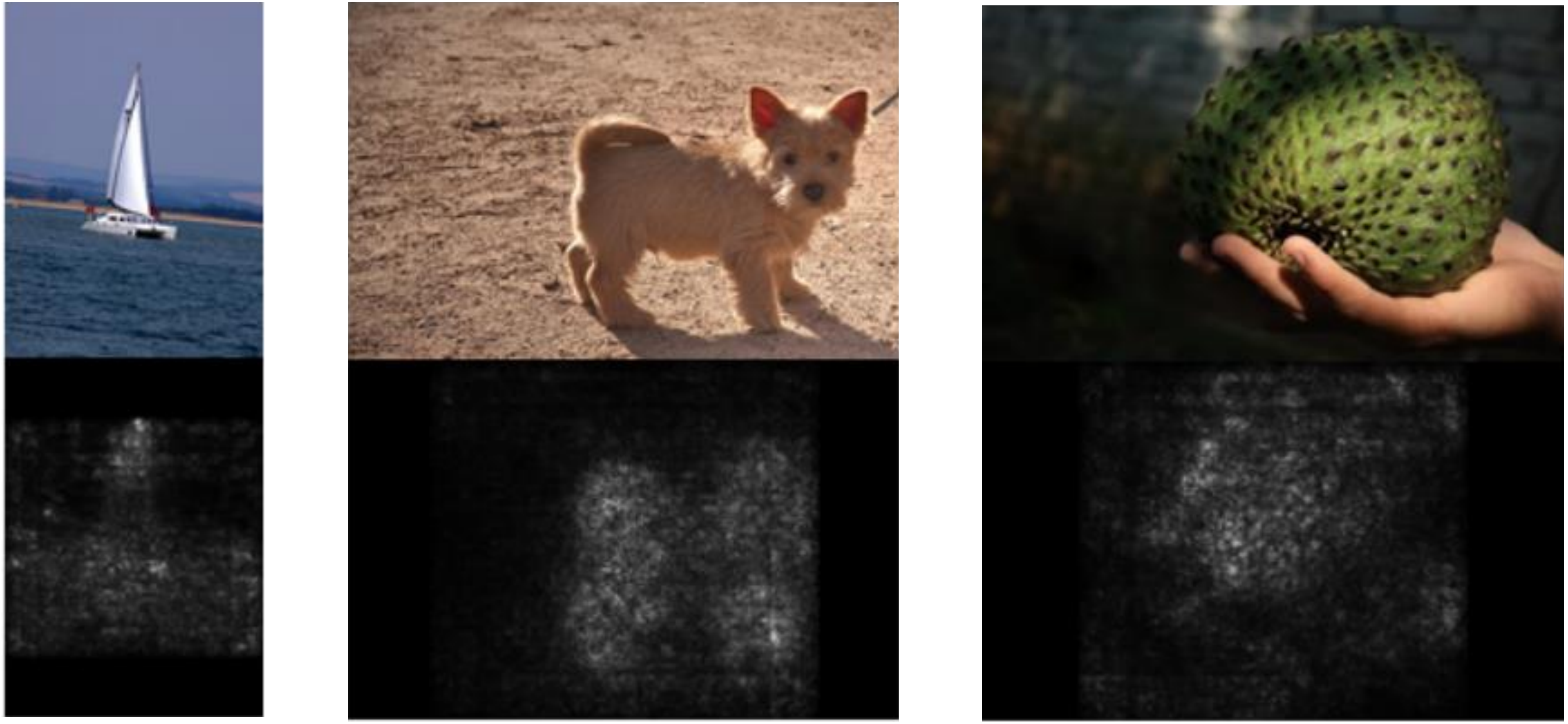
$$S_c(I)|_{I_0} \approx w^T I + b \text{ , where } w = \frac{\partial S_c}{\partial I}|_{I_0}$$

  from backpropagation

  – Solution is locally optimal

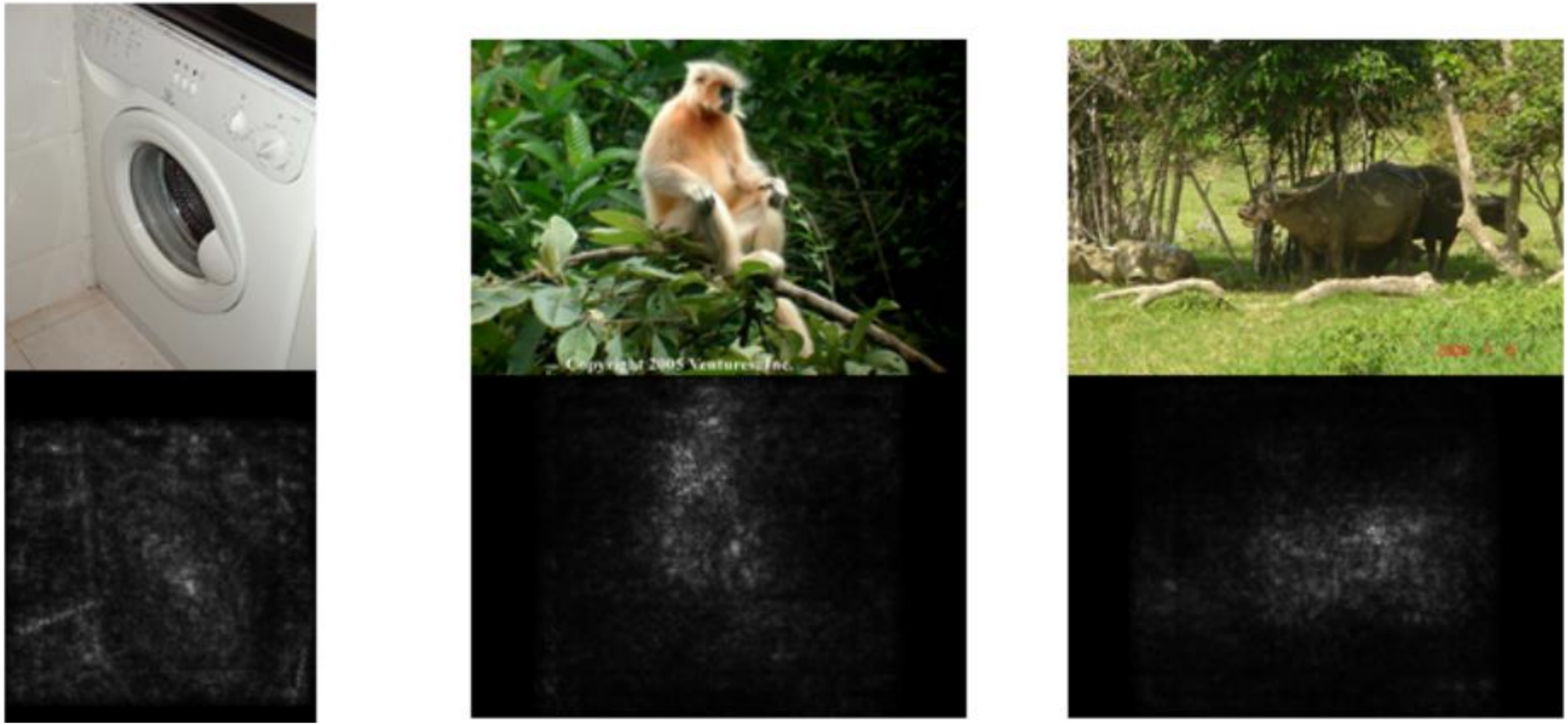
K. Simonyan, A. Vedaldi and A. Zisserman. Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps. ICLR Workshop 2014

Which pixels matter? Saliency via Backprop



K. Simonyan, A. Vedaldi and A. Zisserman. Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps. ICLR Workshop 2014

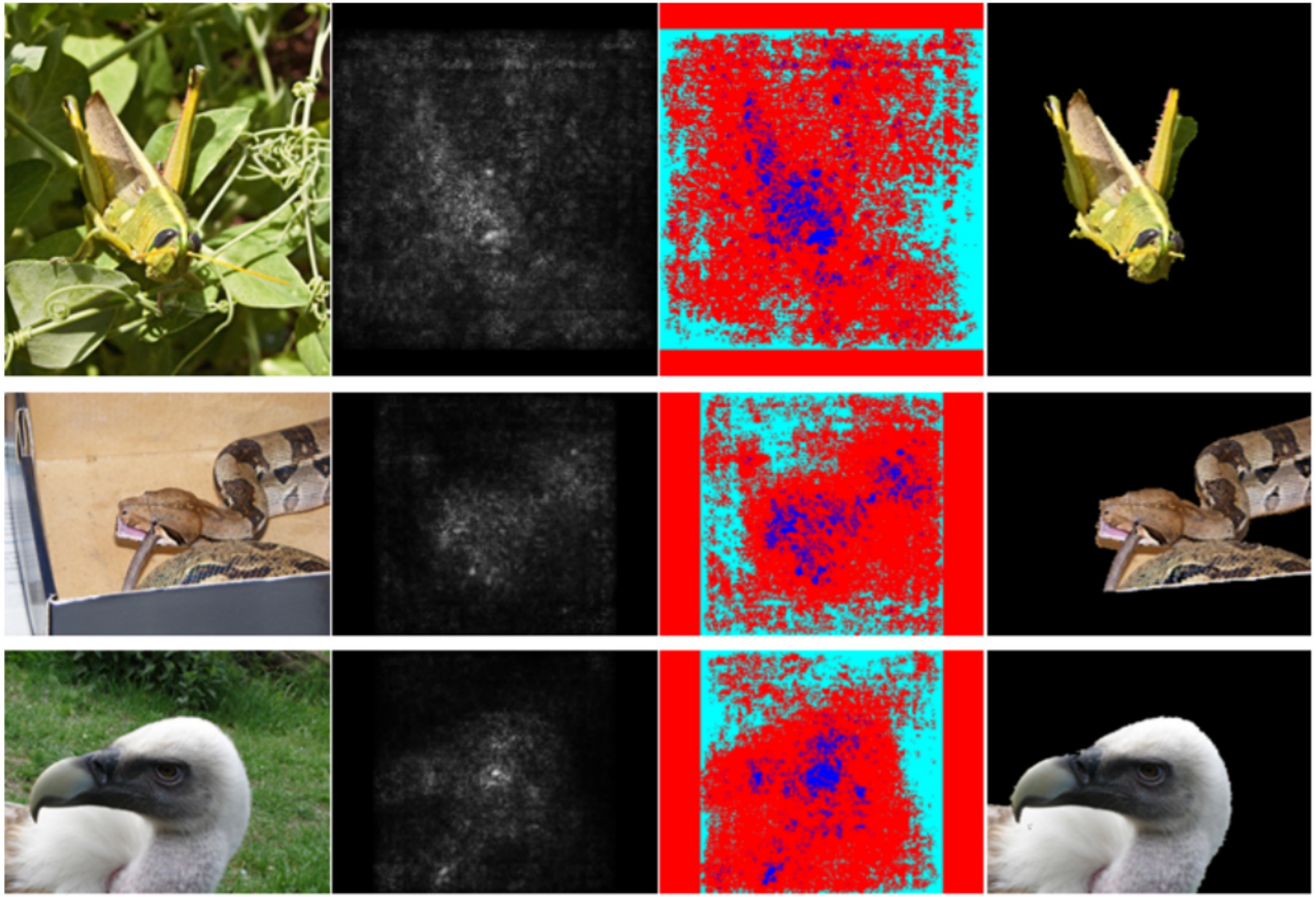
Which pixels matter? Saliency via Backprop



K. Simonyan, A. Vedaldi and A. Zisserman. Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps. ICLR Workshop 2014

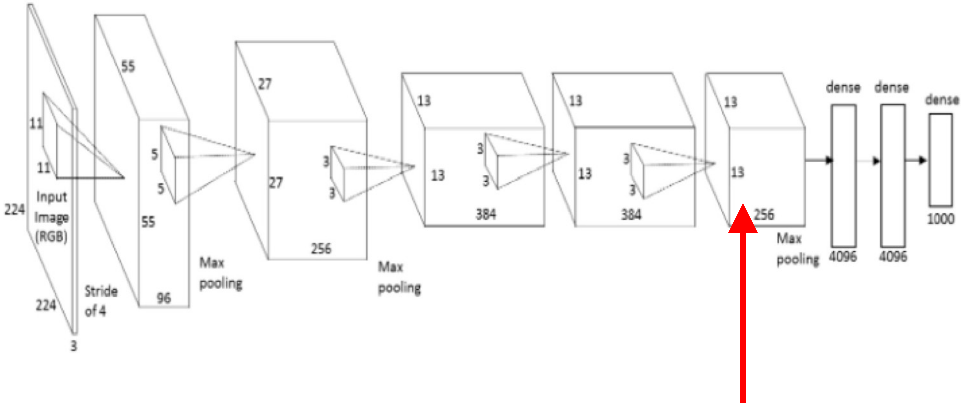
Saliency Maps: Segmentation without Supervision

Use GrabCut
on saliency map



K. Simonyan, A. Vedaldi and A. Zisserman. Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps. ICLR Workshop 2014

Intermediate Features via (guided) backprop

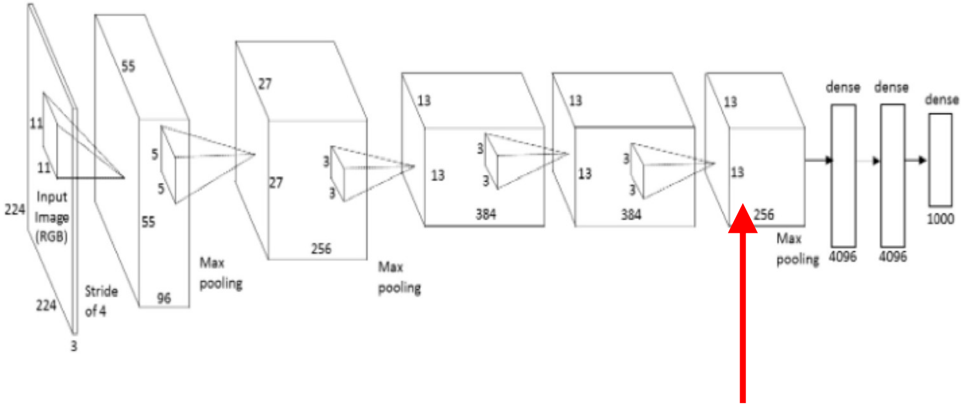


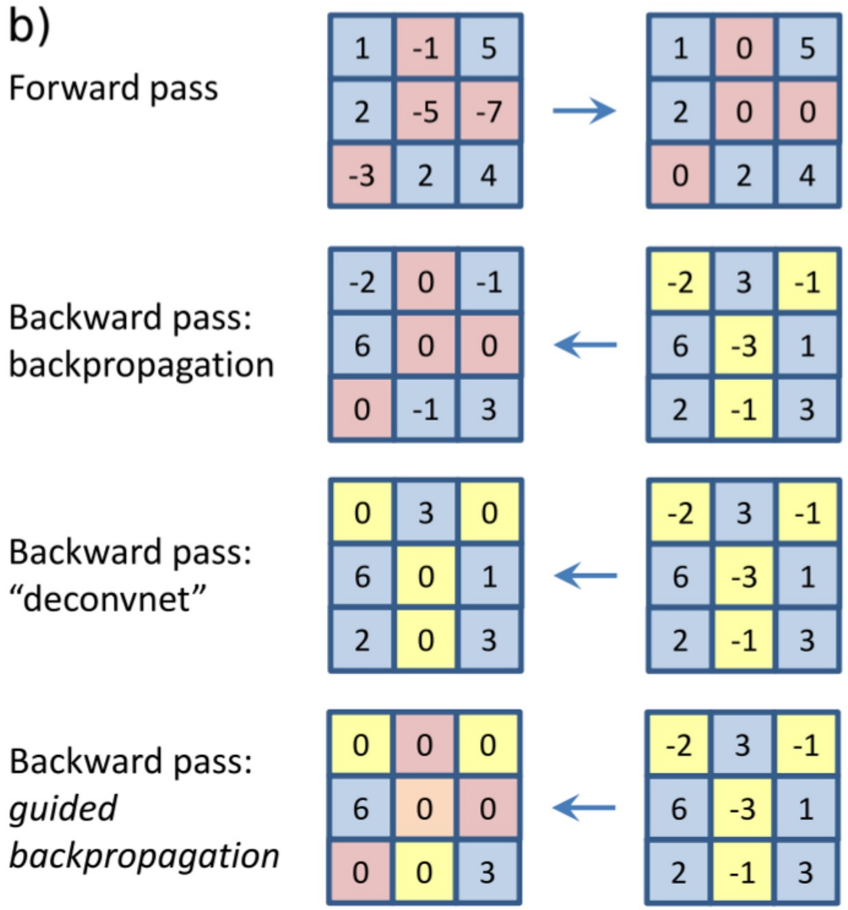
Pick a single intermediate neuron, e.g. one value in 128 x 13 x 13 conv5 feature map

Compute gradient of neuron value with respect to image pixels

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

Intermediate Features via (guided) backprop





Pick a single intermediate neuron, e.g. one value in 128 x 13 x 13 conv5 feature map

Compute gradient of neuron value with respect to image pixels

Images come out nicer if you only backprop positive gradients through each ReLU (guided backprop)

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

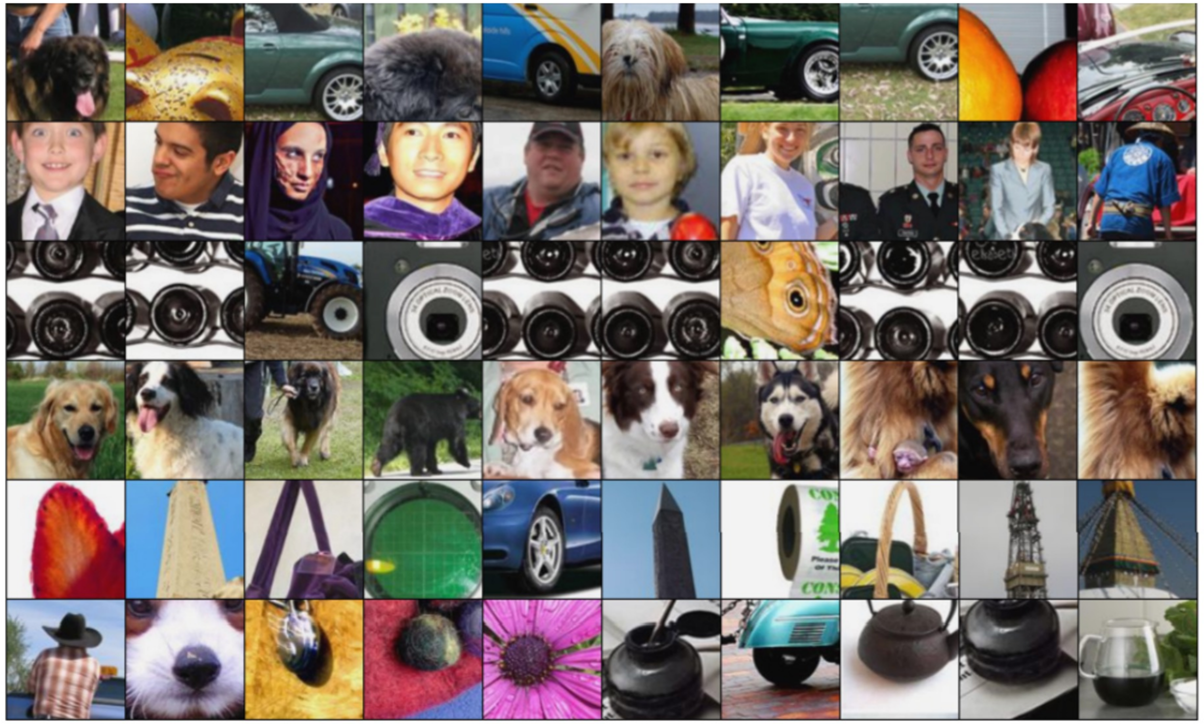
Intermediate Features via (guided) backprop

Maximally activating patches
(Each row is a different neuron)

Guided Backprop

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

Intermediate Features via (guided) backprop

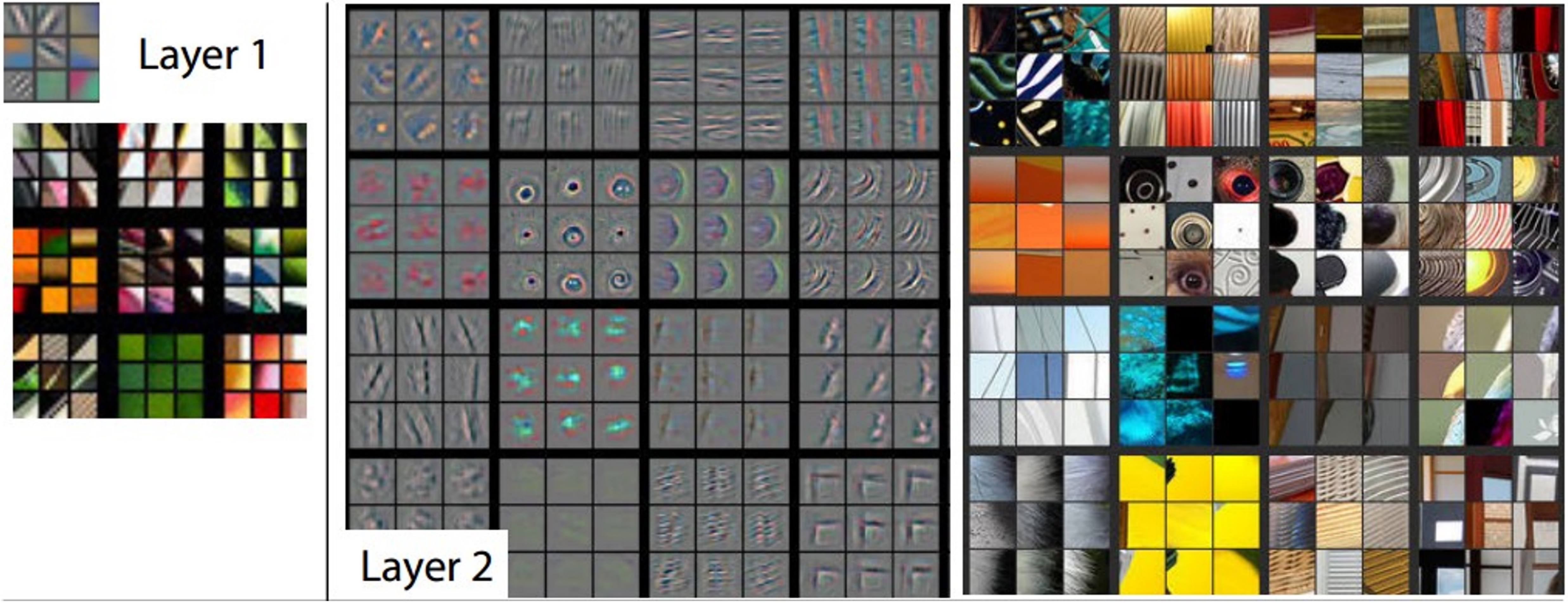


Maximally activating patches
(Each row is a different neuron)

Guided Backprop

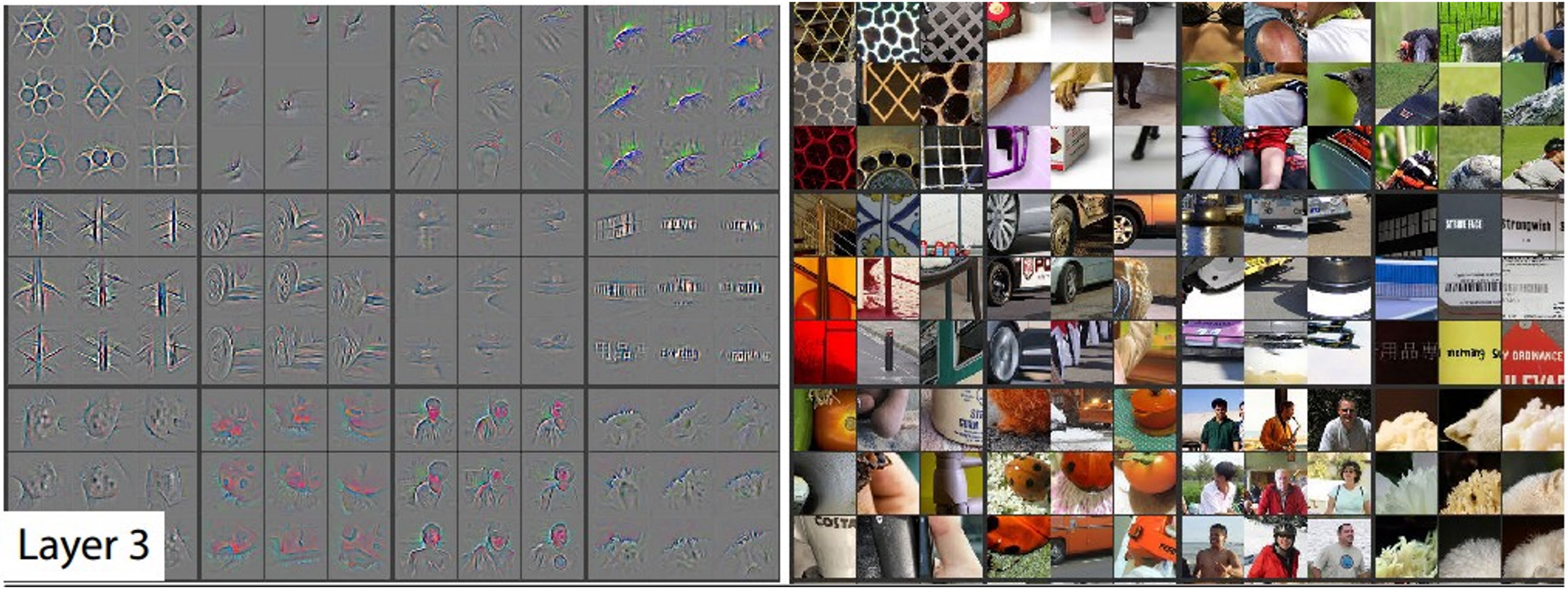
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualizing arbitrary neurons along the way to the top...



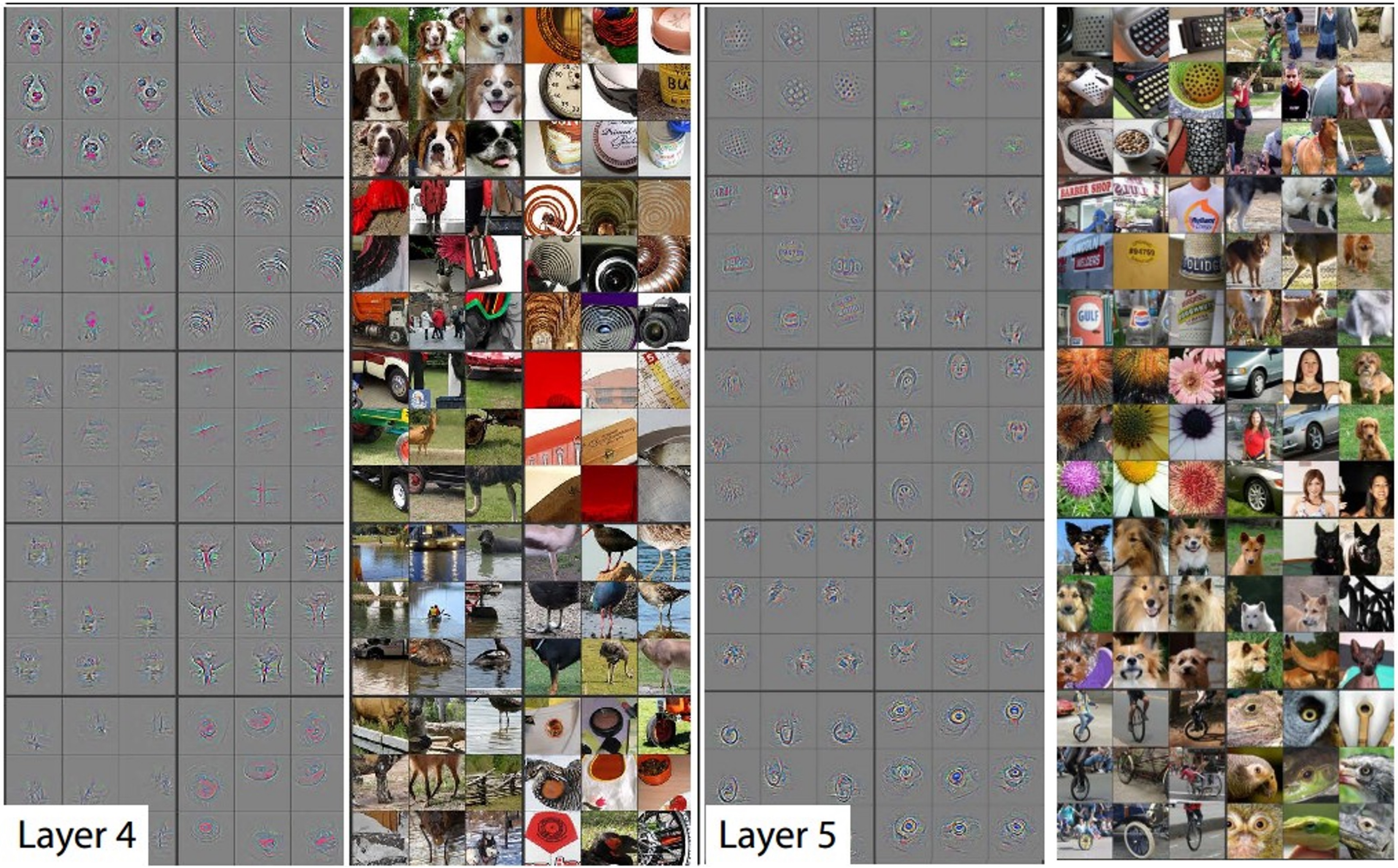
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualizing arbitrary neurons along the way to the top...



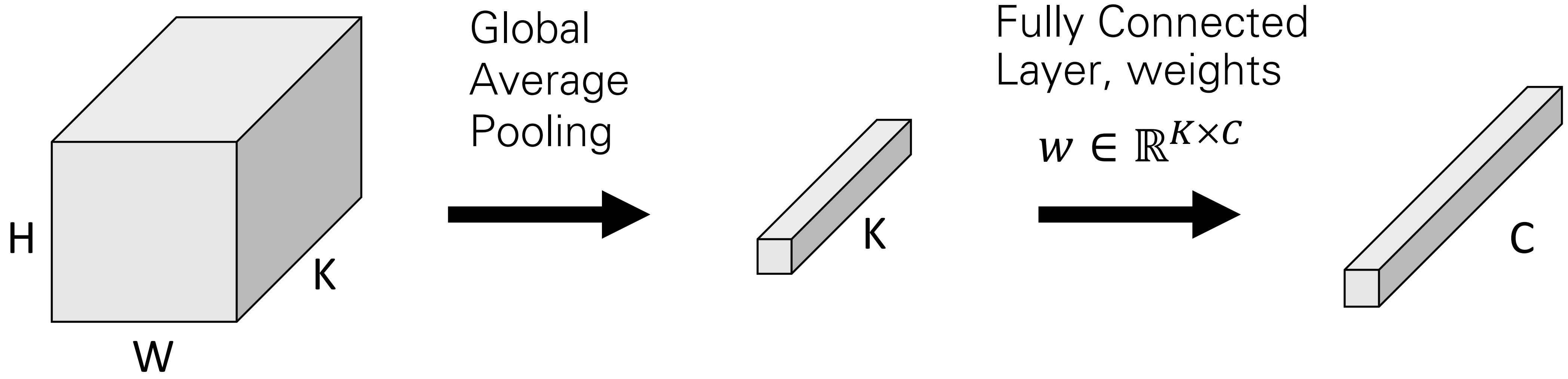
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualizing arbitrary neurons along the way to the top...



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

Class Activation Mapping (CAM)



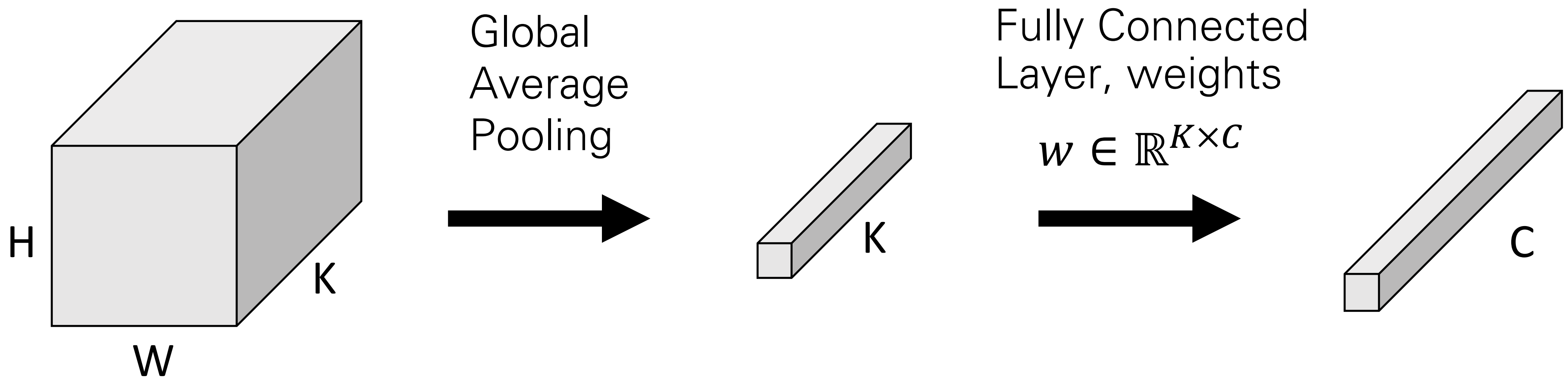
Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

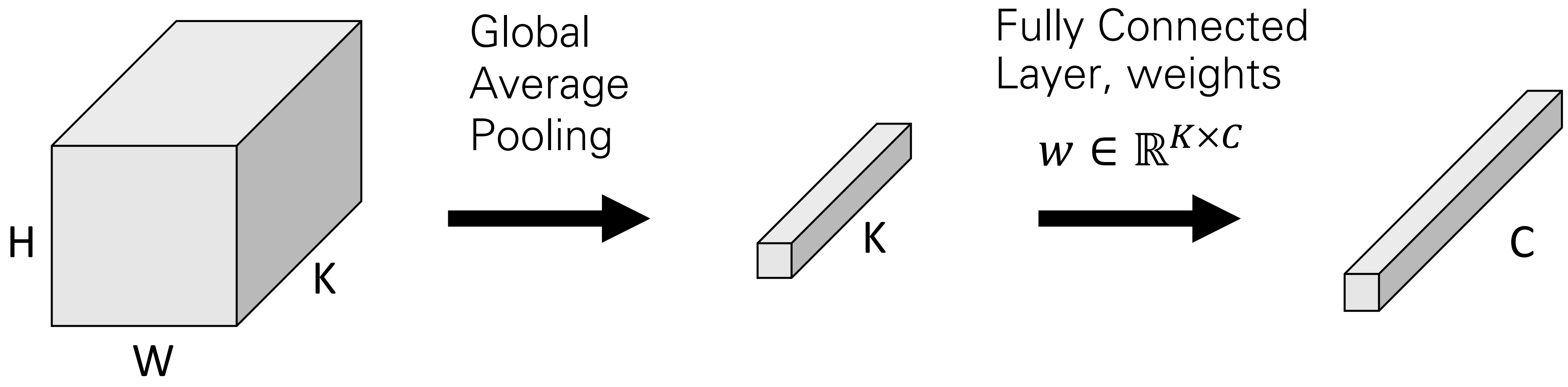
Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k}$$

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

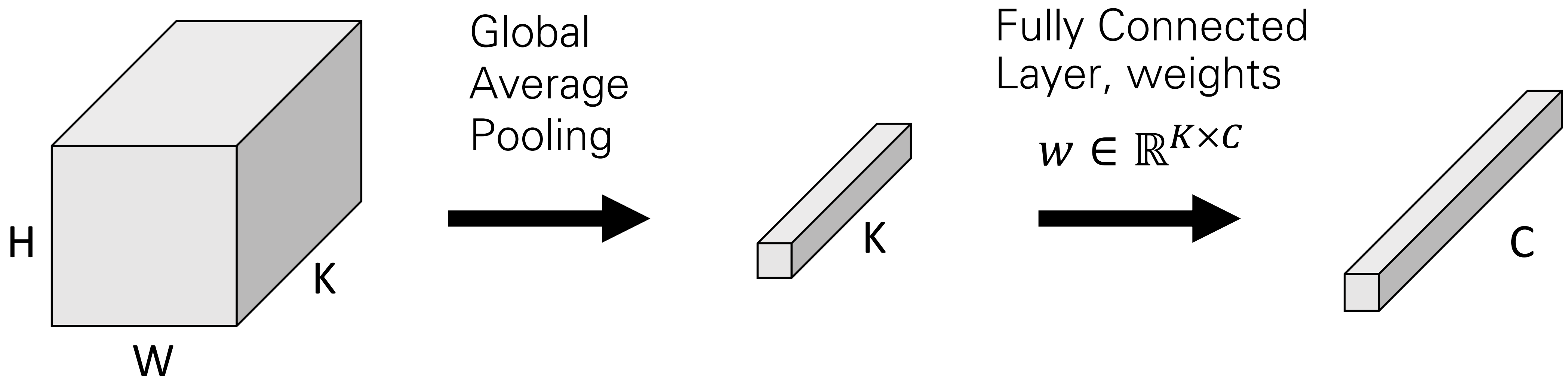
Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \quad S_c = \sum_k w_{k,c} F_k$$

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

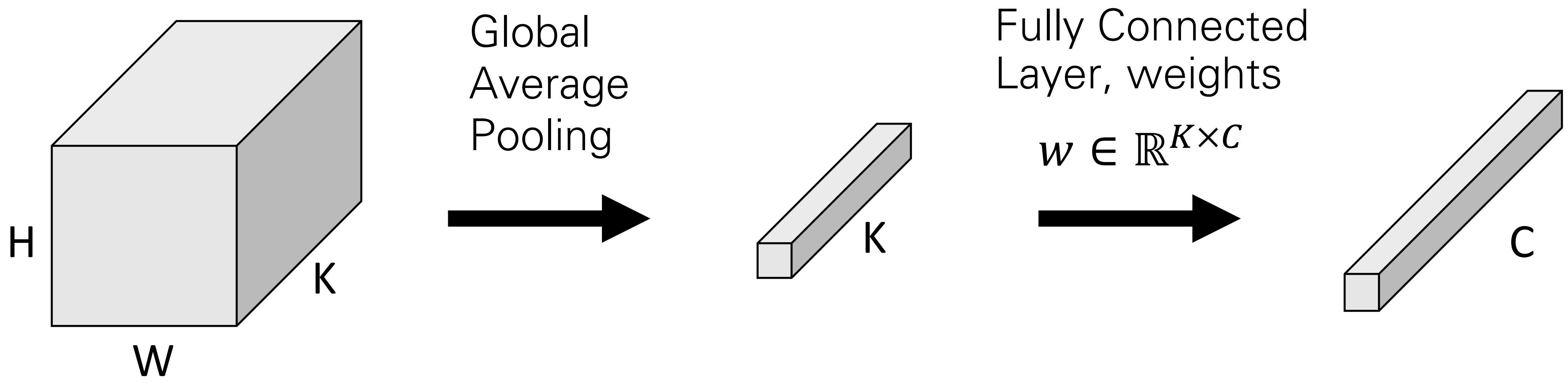
Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \qquad S_c = \sum_{k} w_{k,c} F_k = \frac{1}{HW} \sum_{k} w_{k,c} \sum_{h,w} f_{h,w,k}$$

Zhou et al, “Learning Deep Features for Discriminative Localization”, CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Pooled features:
$F \in \mathbb{R}^{K}$

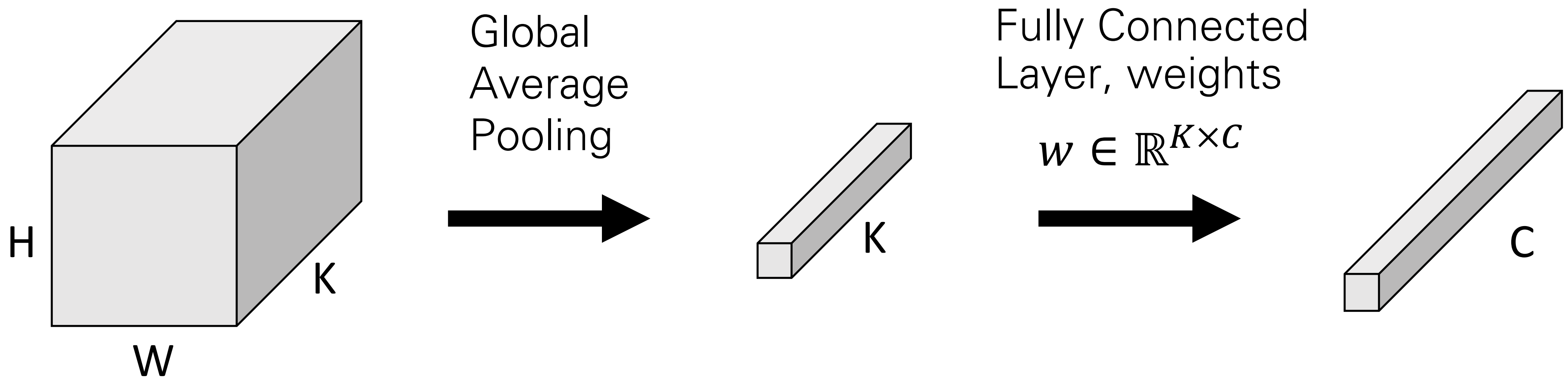
Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \qquad S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$

$$= \frac{1}{HW} \sum_{h,w} \sum_k w_{k,c} f_{h,w,k}$$

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Pooled features:
$F \in \mathbb{R}^{K}$

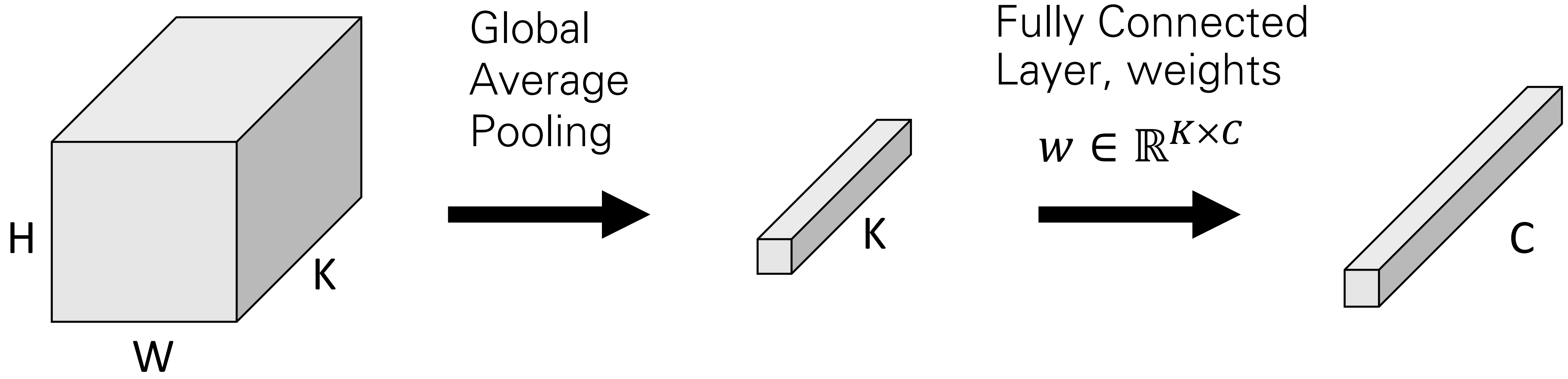
Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \qquad S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$

$$= \frac{1}{HW} \sum_{h,w} \sum_k \boldsymbol{w_{k,c} f_{h,w,k}}$$

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k}$$

$$S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$
$$= \frac{1}{HW} \sum_{h,w} \sum_k \boldsymbol{w}_{k,c} \boldsymbol{f}_{h,w,k}$$

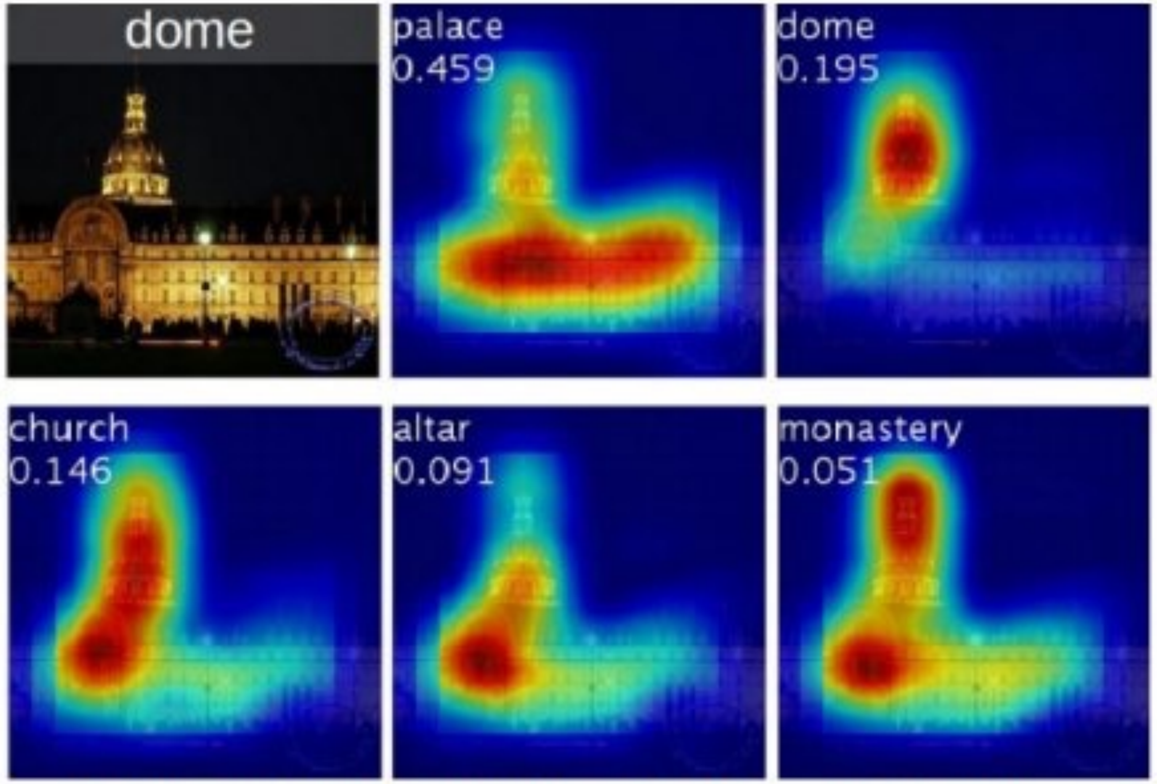
Class Activation Maps:

$$\boldsymbol{M} \in \mathbb{R}^{C,H,W}$$

$$\boldsymbol{M}_{c,h,w} = \sum_k \boldsymbol{w}_{k,c} \boldsymbol{f}_{h,w,k}$$

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



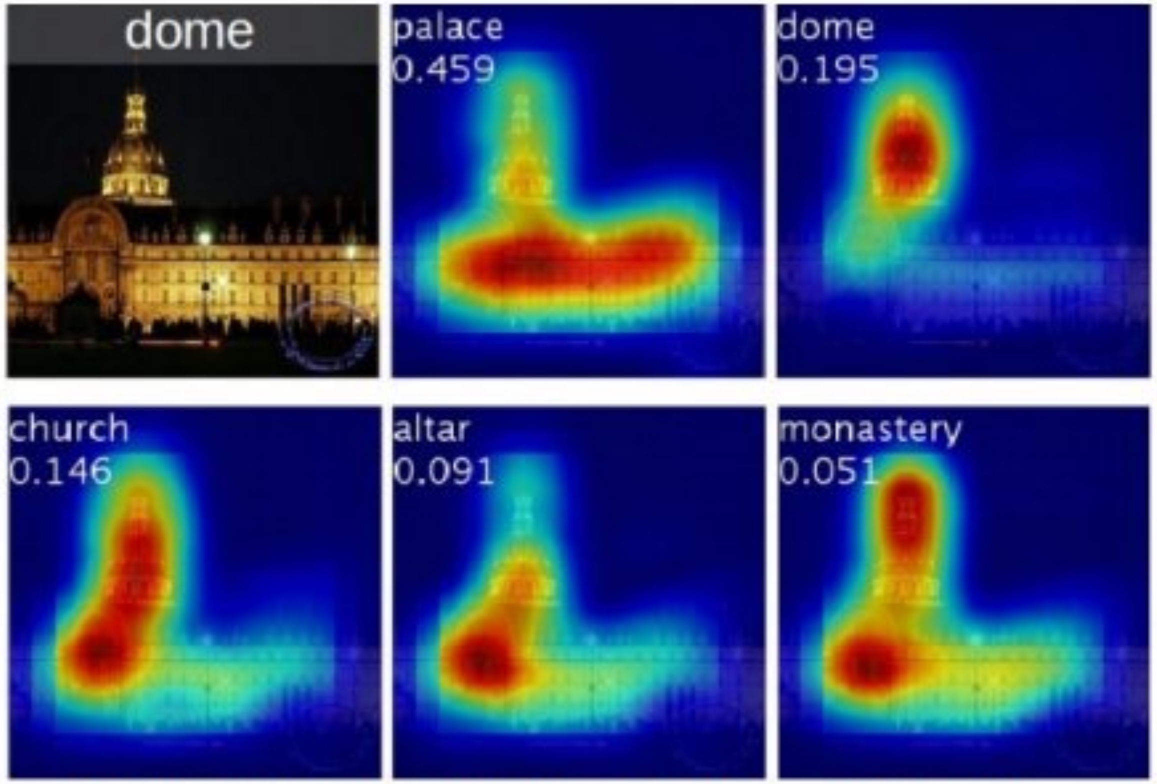
Class activation maps of top 5 predictions

Class activation maps for one object class

Zhou et al, “Learning Deep Features for Discriminative Localization”, CVPR 2016

Class Activation Mapping (CAM)

Problem: Can only apply to last conv layer



Class activation maps of top 5 predictions

Class activation maps for one object class

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A:

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A:

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

3. Global Average Pool the gradients to get weights $\alpha \in \mathbb{R}^{K}$:

$$\alpha_k = \frac{1}{HW} \sum_{h,w} \frac{\partial S_c}{\partial A_{h,w,k}}$$

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A:

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

3. Global Average Pool the gradients to get weights $\alpha \in \mathbb{R}^{K}$:

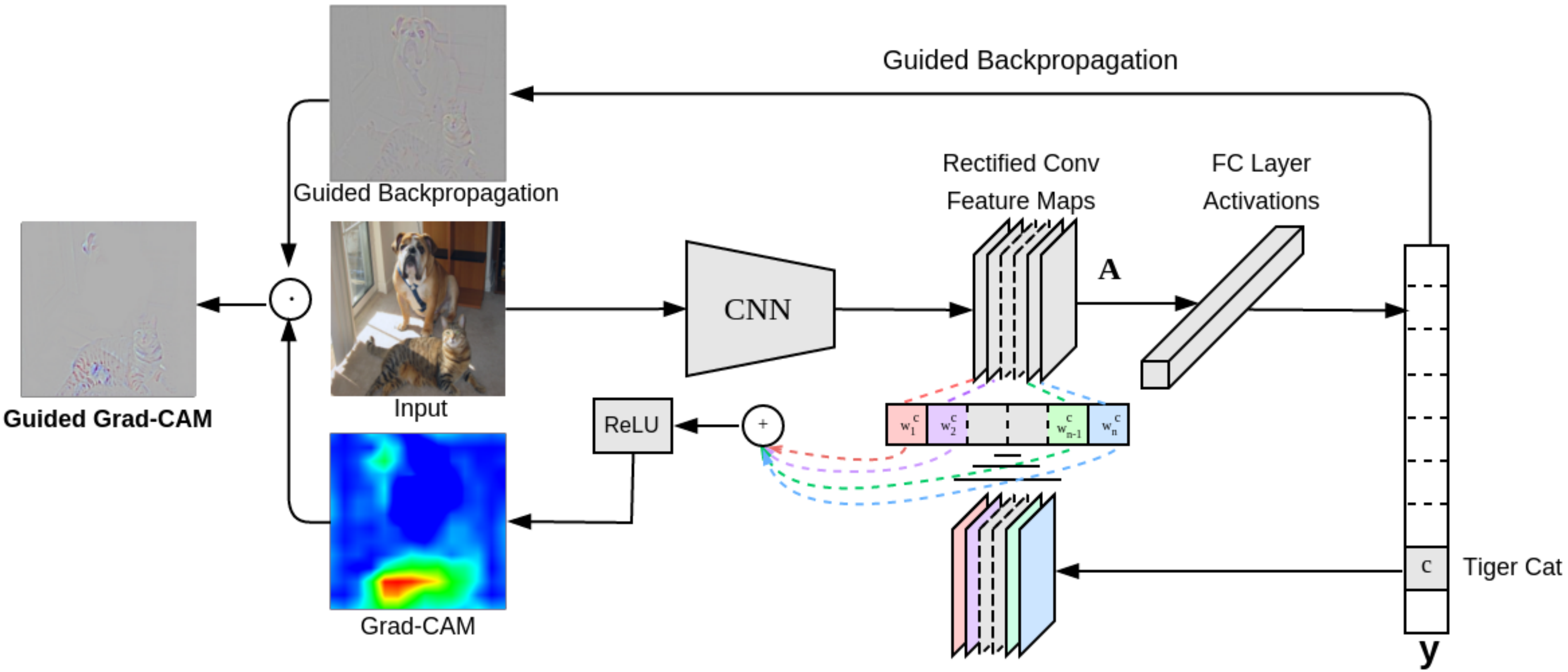
$$\alpha_k = \frac{1}{HW} \sum_{h,w} \frac{\partial S_c}{\partial A_{h,w,k}}$$

4. Compute activation map $M^c \in \mathbb{R}^{H,W}$:

$$M^c_{h,w} = ReLU\left(\sum_k \alpha_k A_{h,w,k}\right)$$

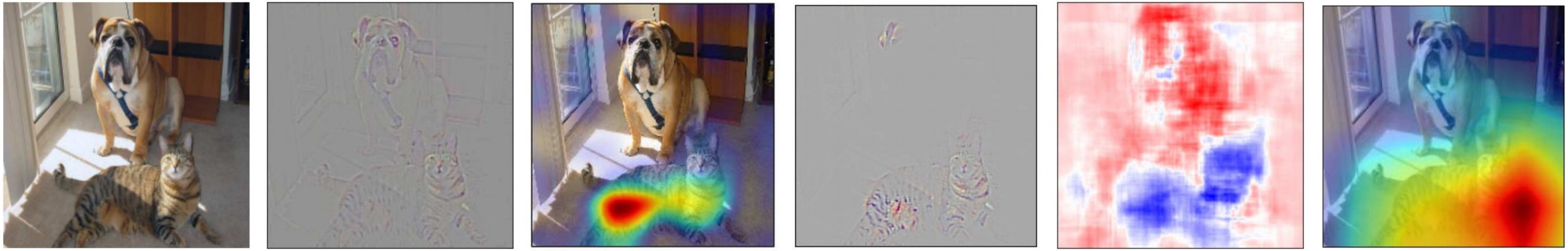
Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)



Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)



(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat' (d) Guided Grad-CAM 'Cat' (e) Occlusion map for 'Cat' (f) ResNet Grad-CAM 'Cat'

(g) Original Image (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog' (j) Guided Grad-CAM 'Dog' (k) Occlusion map for 'Dog' (l) ResNet Grad-CAM 'Dog'

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)

Can also be applied beyond classification models, e.g. image captioning

A group of people flying kites on a beach

A man is sitting at a table with a pizza

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Visualizing CNN Features: Gradient Ascent

(Guided) backprop:
Find the part of an image that a neuron responds to

Gradient ascent:
Generate a synthetic image that maximally activates a neuron

$$I^* = \arg\max_I \boxed{f(I)} + \boxed{R(I)}$$

Neuron value

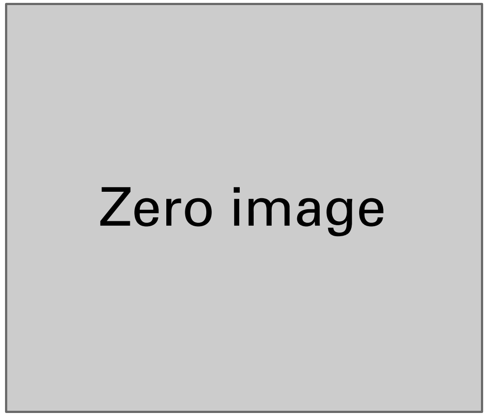
Natural image regularizer

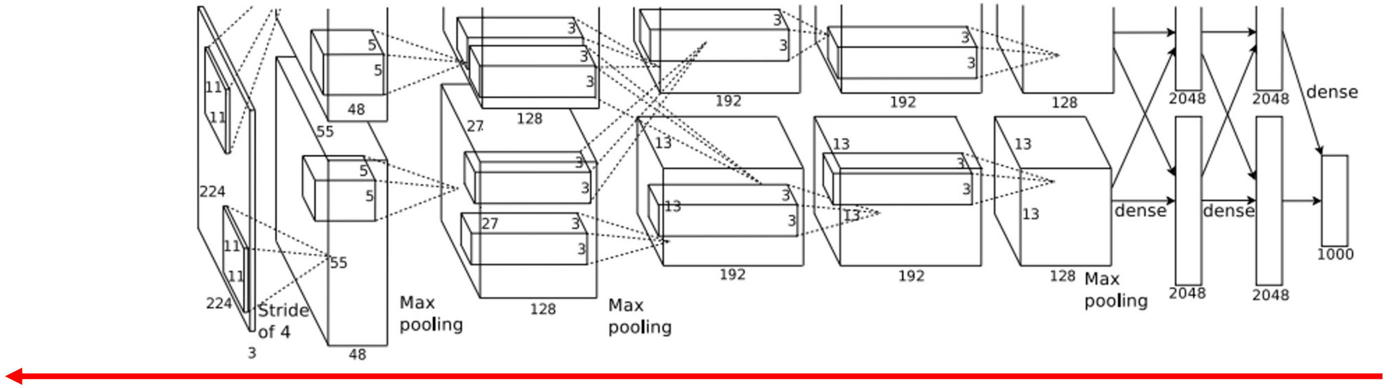
Visualizing CNN Features: Gradient Ascent

1. Initialize image to zeros

$$\arg\max_{I} \boxed{S_c(I)} - \lambda \|I\|_2^2$$

Score for class c (before Softmax)





Repeat:

2. Forward image to compute current scores
3. Backprop to get gradient of neuron value with respect to image pixels
4. Make a small update to the image

Visualizing CNN Features: Gradient Ascent

$$\arg\max_I S_c(I) - \boxed{\lambda \|I\|_2^2}$$

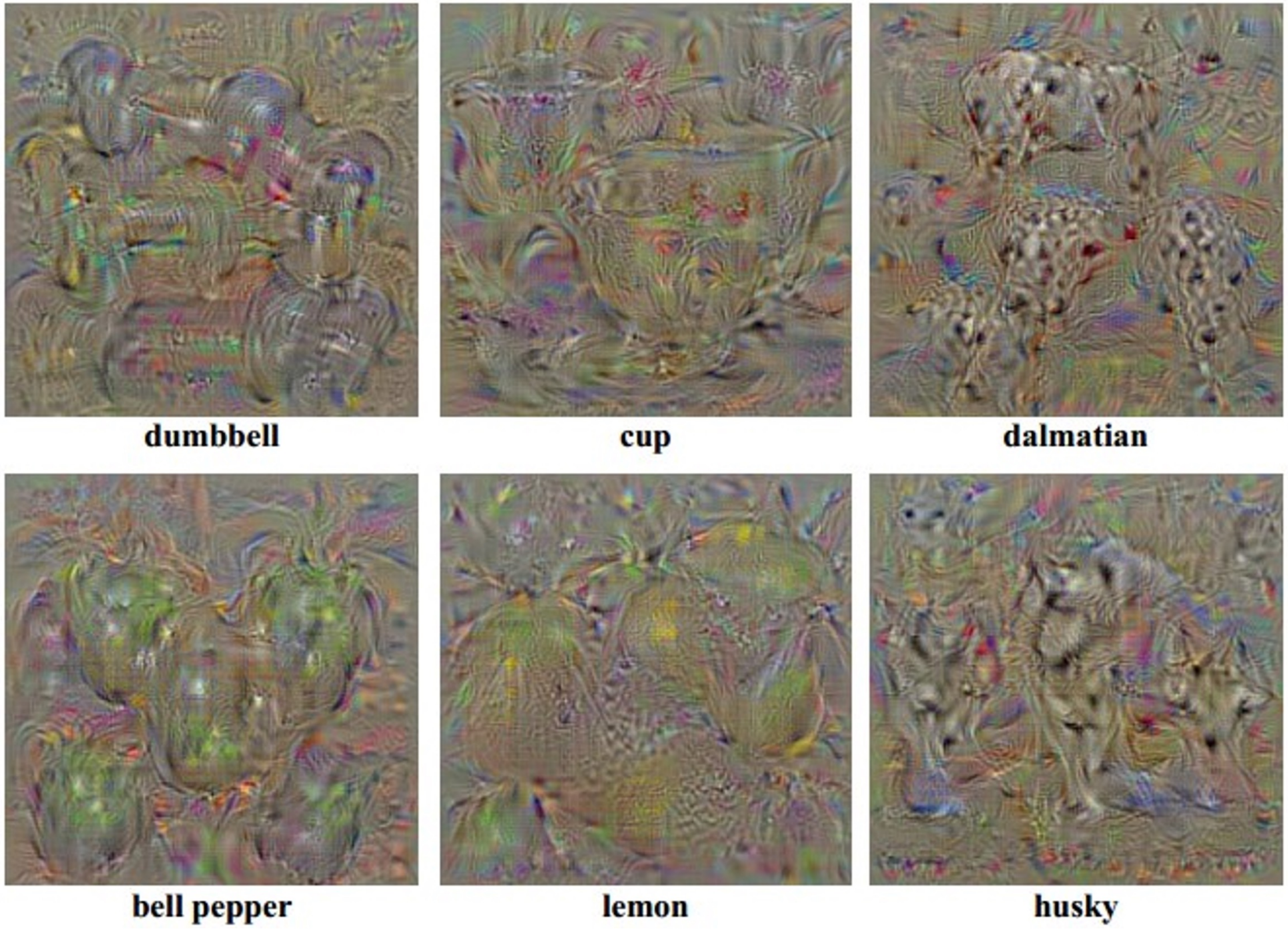
Simple regularizer: Penalize
L2 norm of generated image

K. Simonyan, A. Vedaldi and A. Zisserman. Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps. ICLR Workshop 2014

Visualizing CNN Features: Gradient Ascent

$$\arg\max_I S_c(I) - \boxed{\lambda \| I \|_2^2}$$

Simple regularizer: Penalize L2 norm of generated image

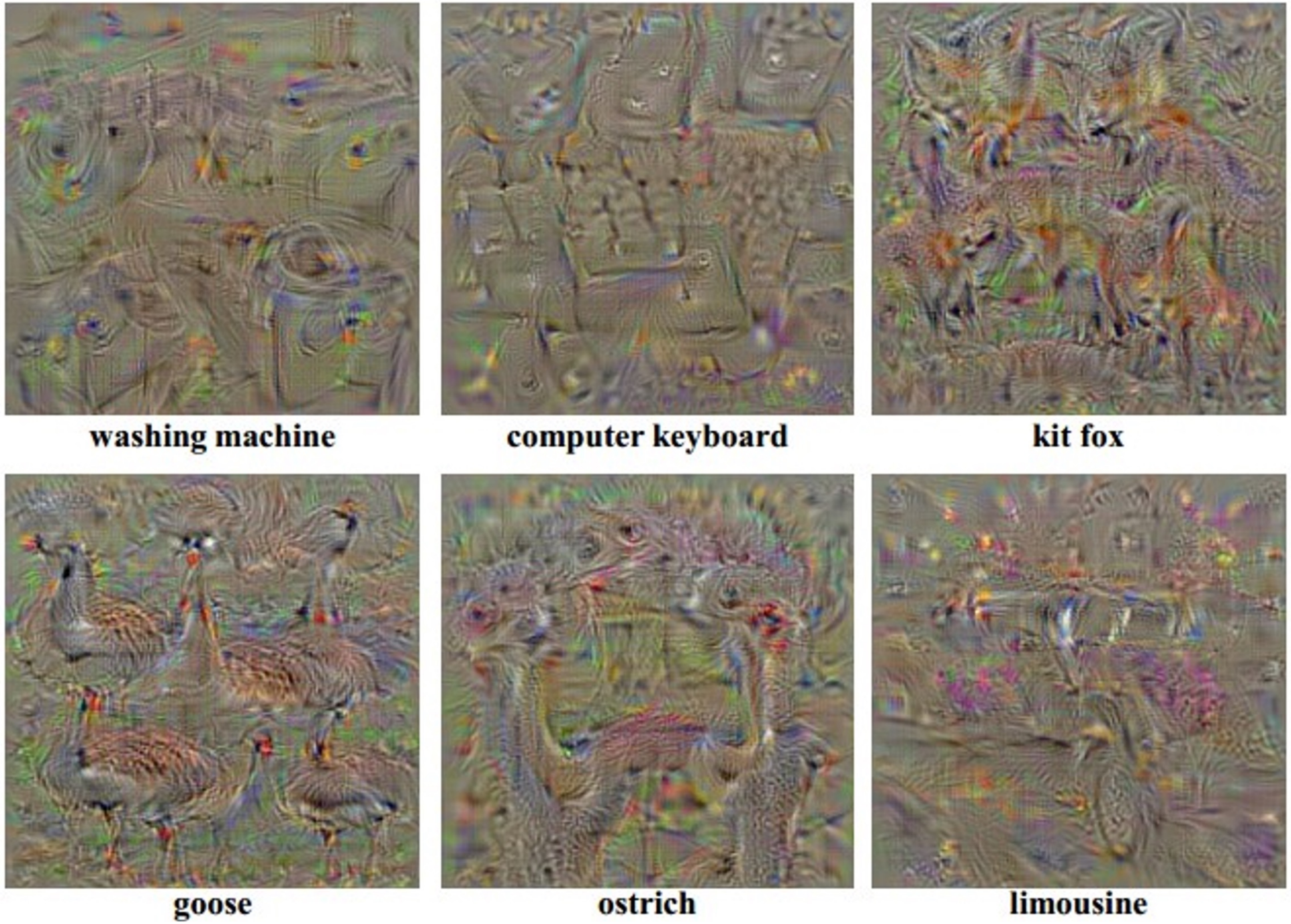


K. Simonyan, A. Vedaldi and A. Zisserman. Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps. ICLR Workshop 2014

Visualizing CNN Features: Gradient Ascent

$$\arg\max_I S_c(I) - \boxed{\lambda \|I\|_2^2}$$

Simple regularizer: Penalize L2 norm of generated image



K. Simonyan, A. Vedaldi and A. Zisserman. Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps. ICLR Workshop 2014

Visualizing CNN Features: Gradient Ascent

$$\arg\max_I S_c(I) - \boxed{\lambda \|I\|_2^2}$$

Better regularizer: Penalize L2 norm of image; also during optimization periodically

1. Gaussian blur image
2. Clip pixels with small values to 0
3. Clip pixels with small gradients to 0

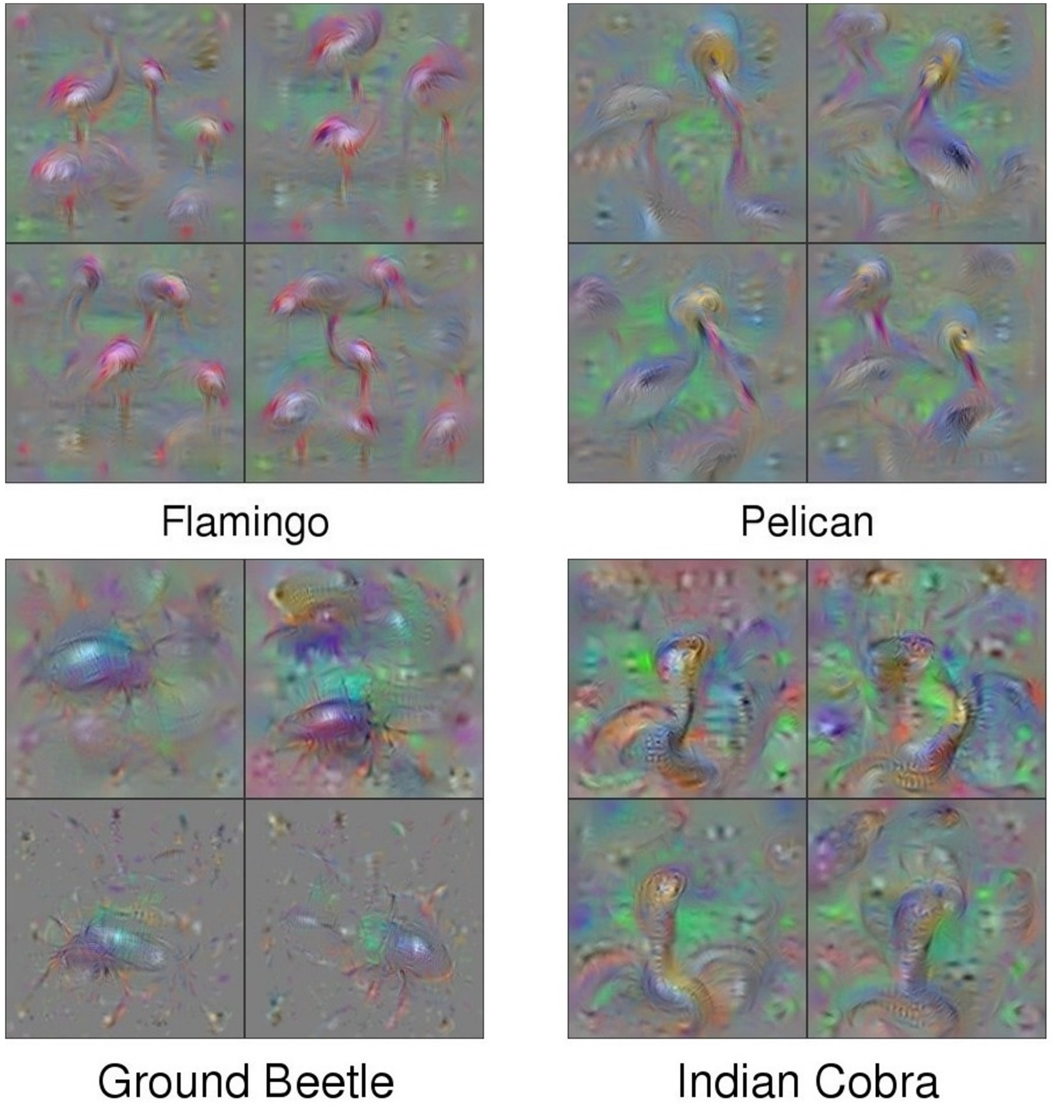
Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Visualizing CNN Features: Gradient Ascent

$$\arg\max_I S_c(I) - \boxed{\lambda \|I\|_2^2}$$

Better regularizer: Penalize L2 norm of image; also during optimization periodically

1. Gaussian blur image
2. Clip pixels with small values to 0
3. Clip pixels with small gradients to 0



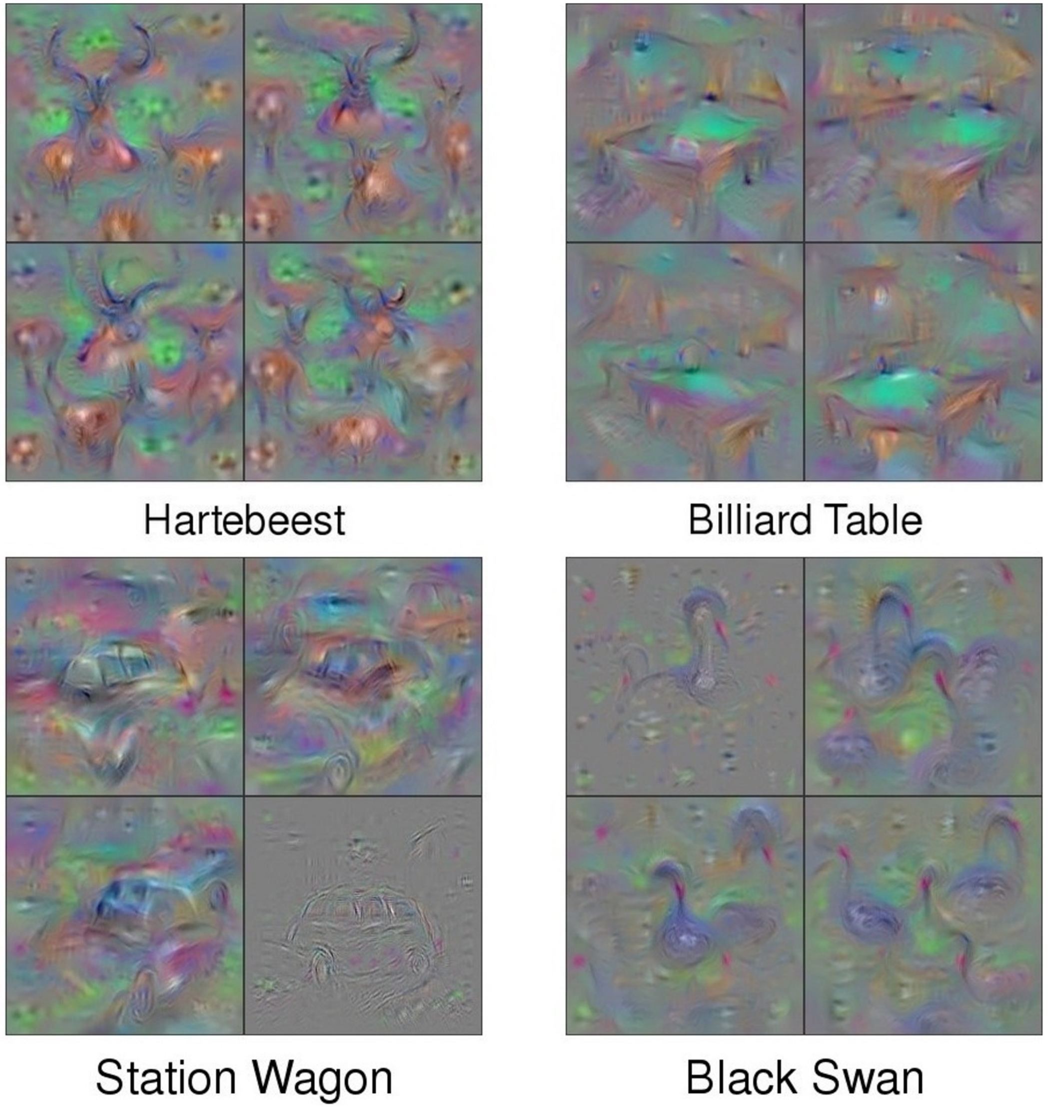
Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Visualizing CNN Features: Gradient Ascent

$$\arg\max_{I} S_c(I) - \boxed{\lambda \|I\|_2^2}$$

Better regularizer: Penalize L2 norm of image; also during optimization periodically

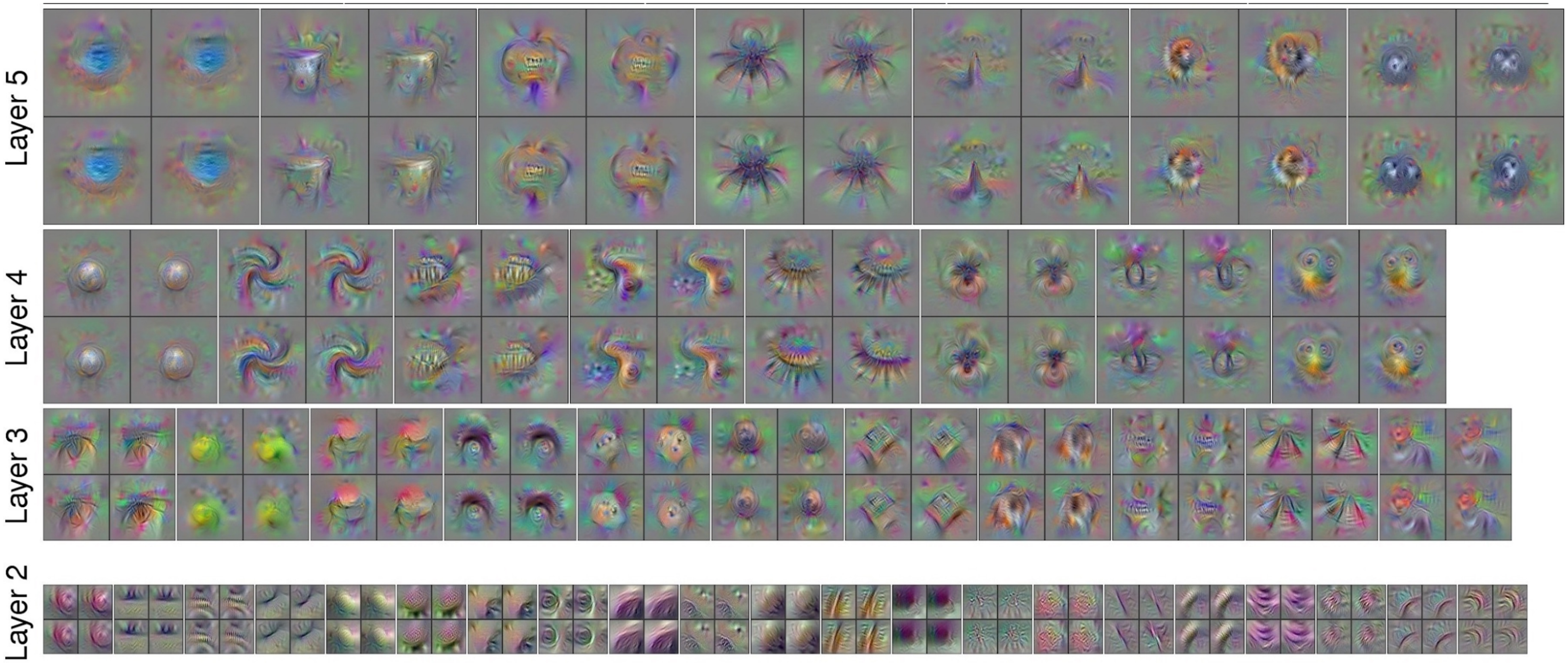
1. Gaussian blur image
2. Clip pixels with small values to 0
3. Clip pixels with small gradients to 0



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Visualizing CNN Features: Gradient Ascent

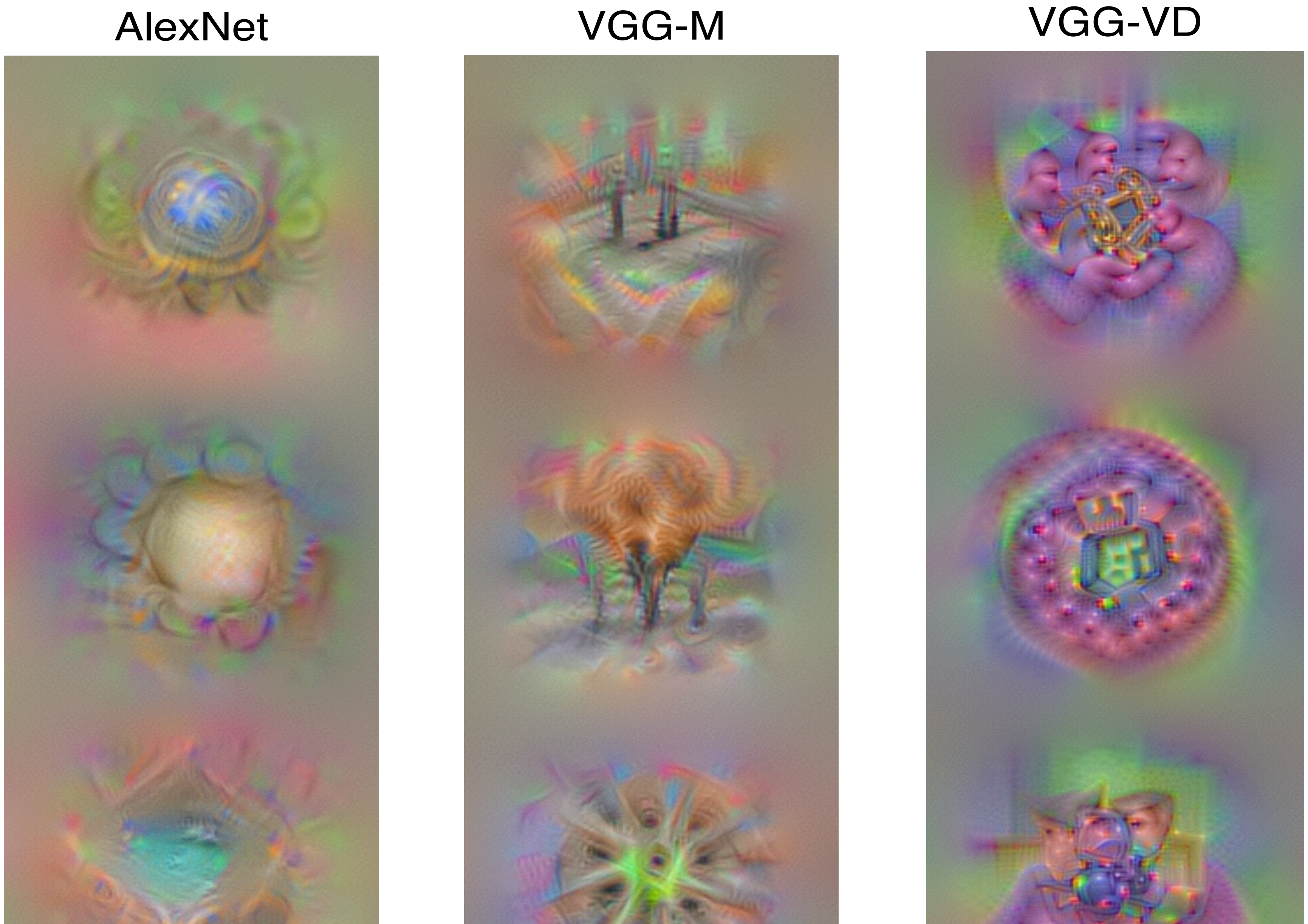
Use the same approach to visualize intermediate features



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Network Comparison

"conv5" features



Deep Visualization Toolbox

yosinski.com/deepvis

#deepvis

Jason Yosinski

Jeff Clune

Anh Nguyen

Thomas Fuchs

Hod Lipson

Cornell University

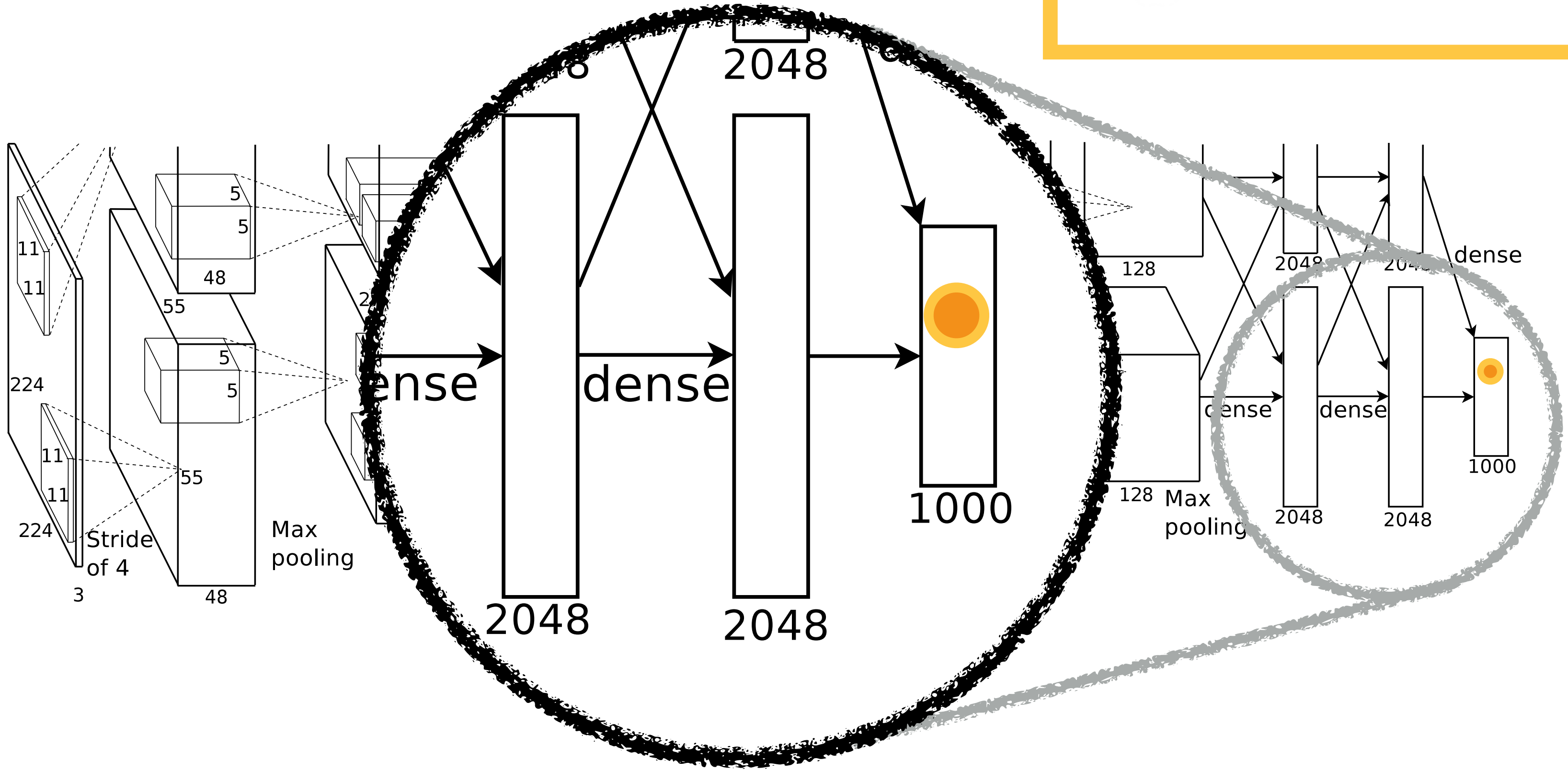
University of Wyoming

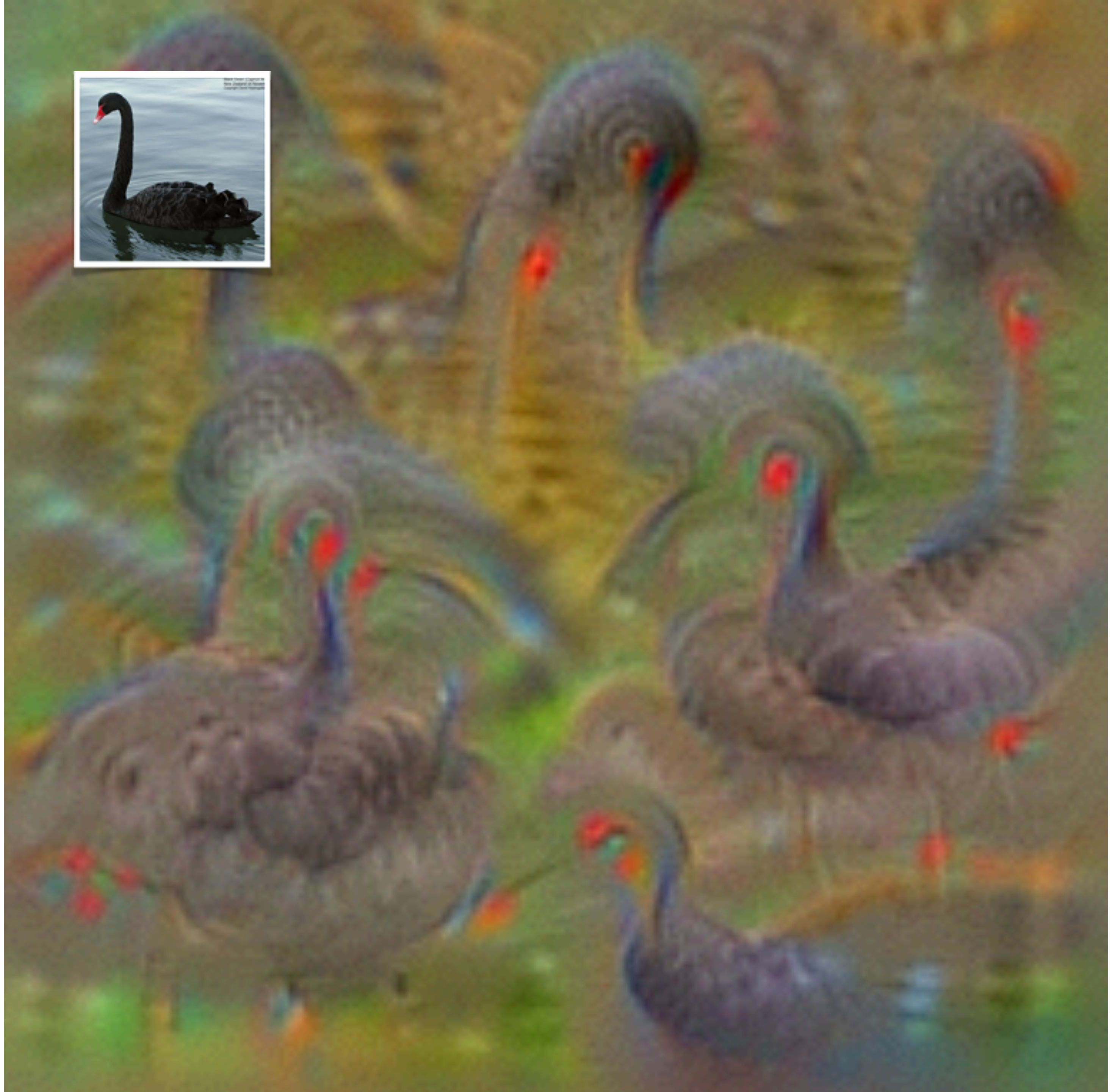
Jet Propulsion Laboratory
California Institute of Technology

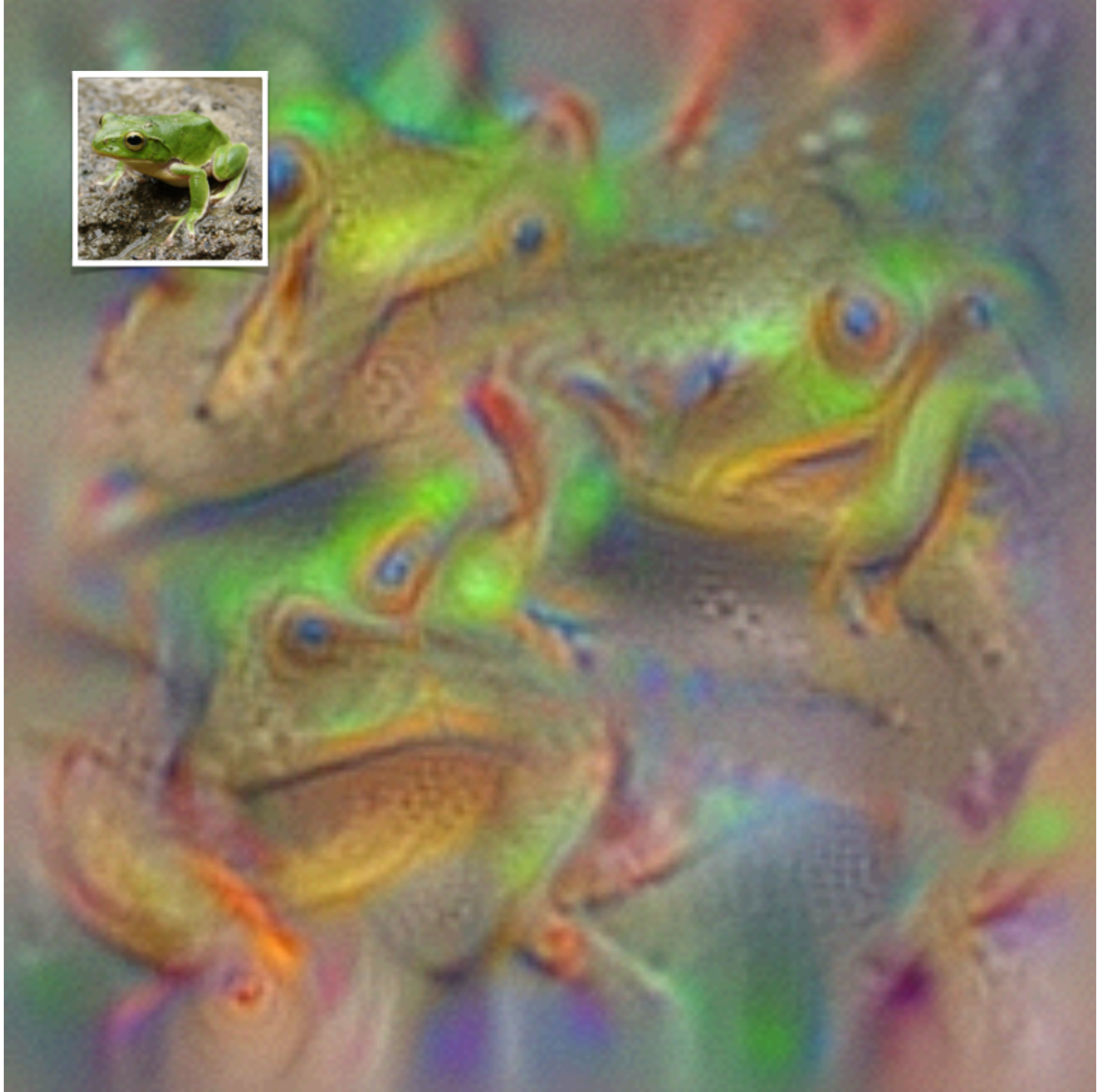
Activation Maximization

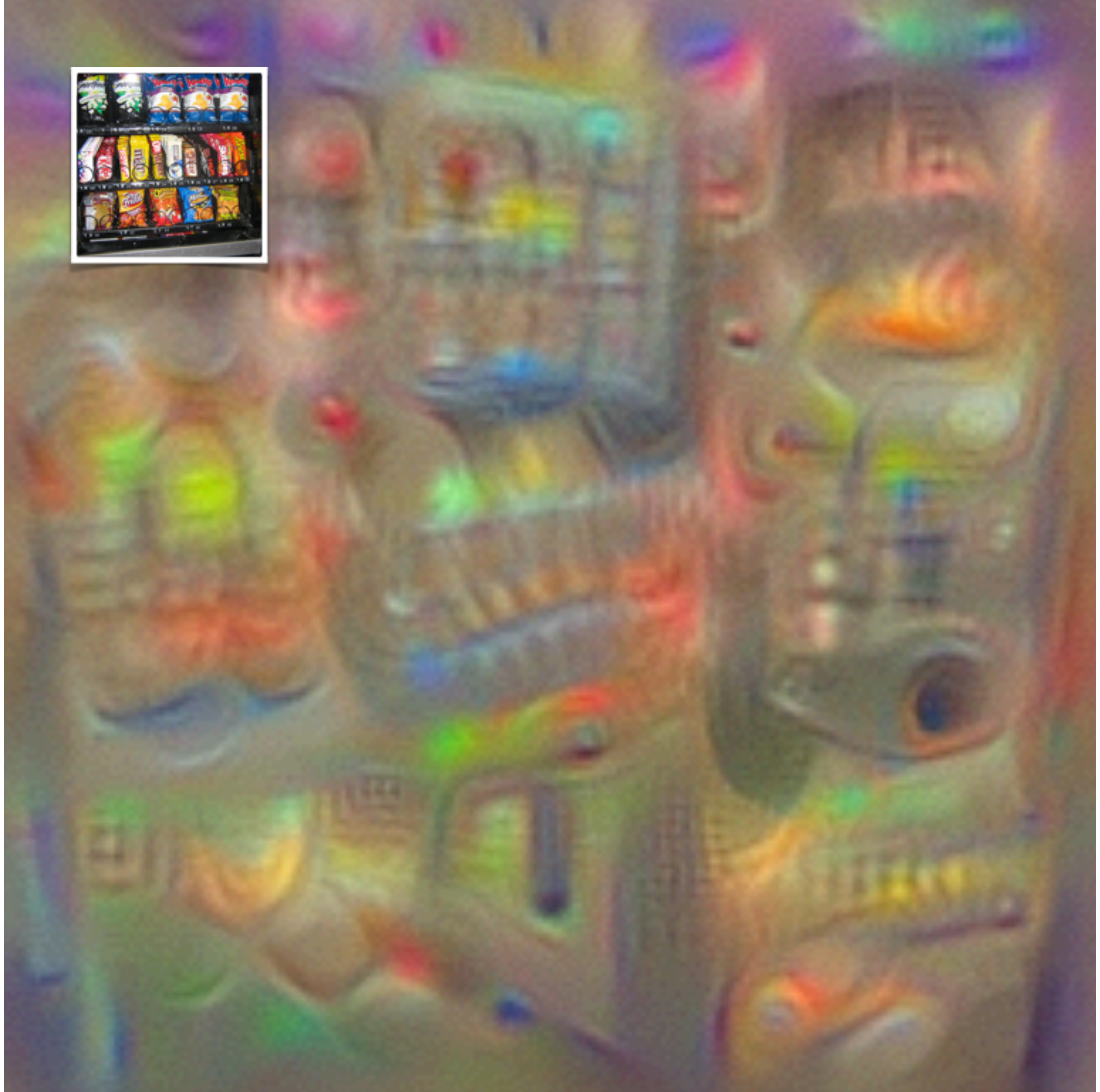
- Look for an image that maximally activates a **specific feature component**

$$\min_{\mathbf{x}} -\langle \mathbf{e}_k, \Phi(\mathbf{x}) \rangle + R_{TV}(\mathbf{x}) + R_{\alpha}(\mathbf{x})$$









Visualizing CNN Features: Gradient Ascent

Adding "multi-faceted" visualization gives even nicer results:
(Plus more careful regularization, center-bias)

Reconstructions of multiple feature types (facets) recognized by the same "grocery store" neuron

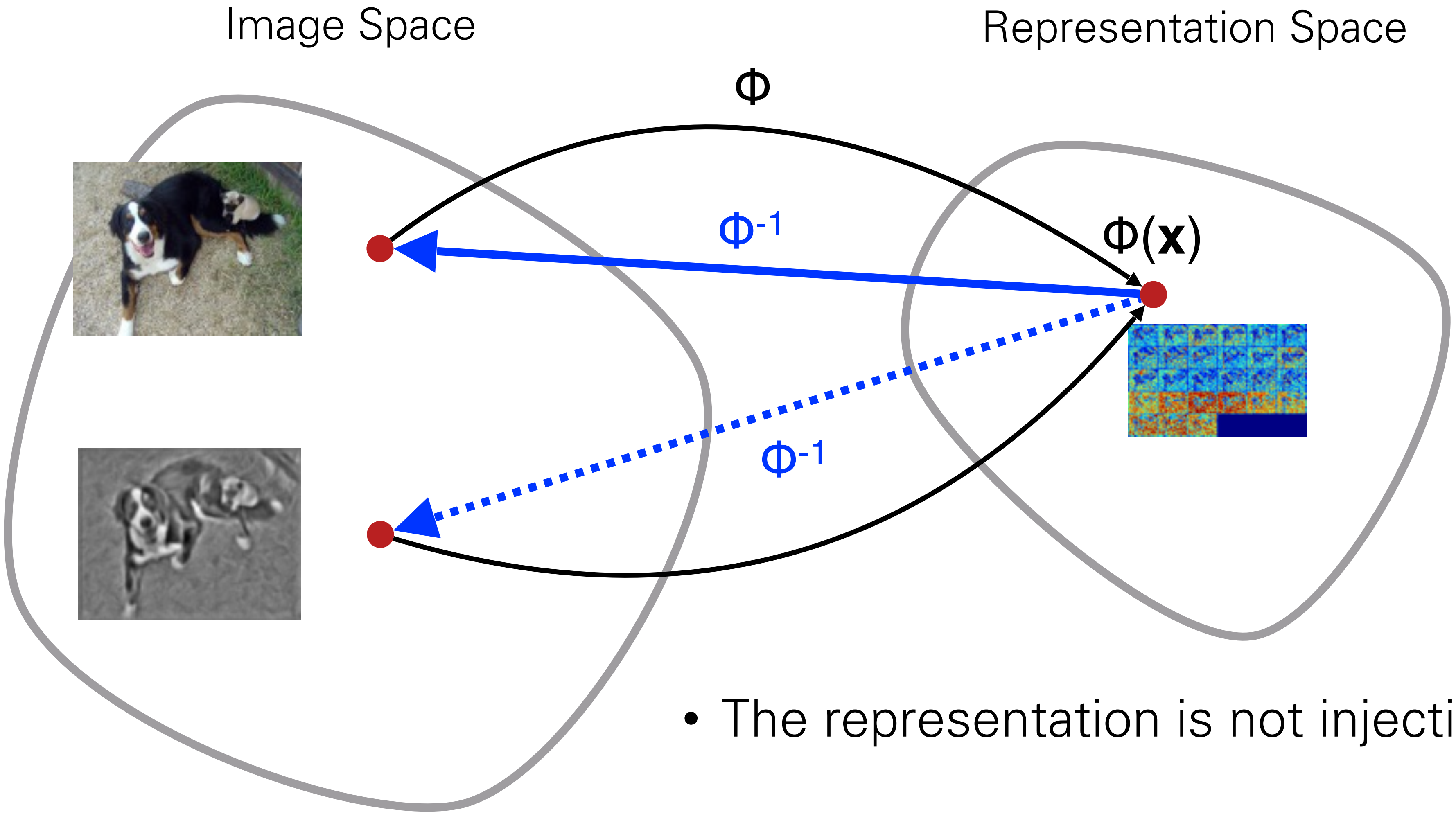
Corresponding example training set images recognized by the same neuron as in the "grocery store" class

Nguyen et al, "Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks", ICML Visualization for Deep Learning Workshop 2016.

Visualizing CNN Features: Gradient Ascent

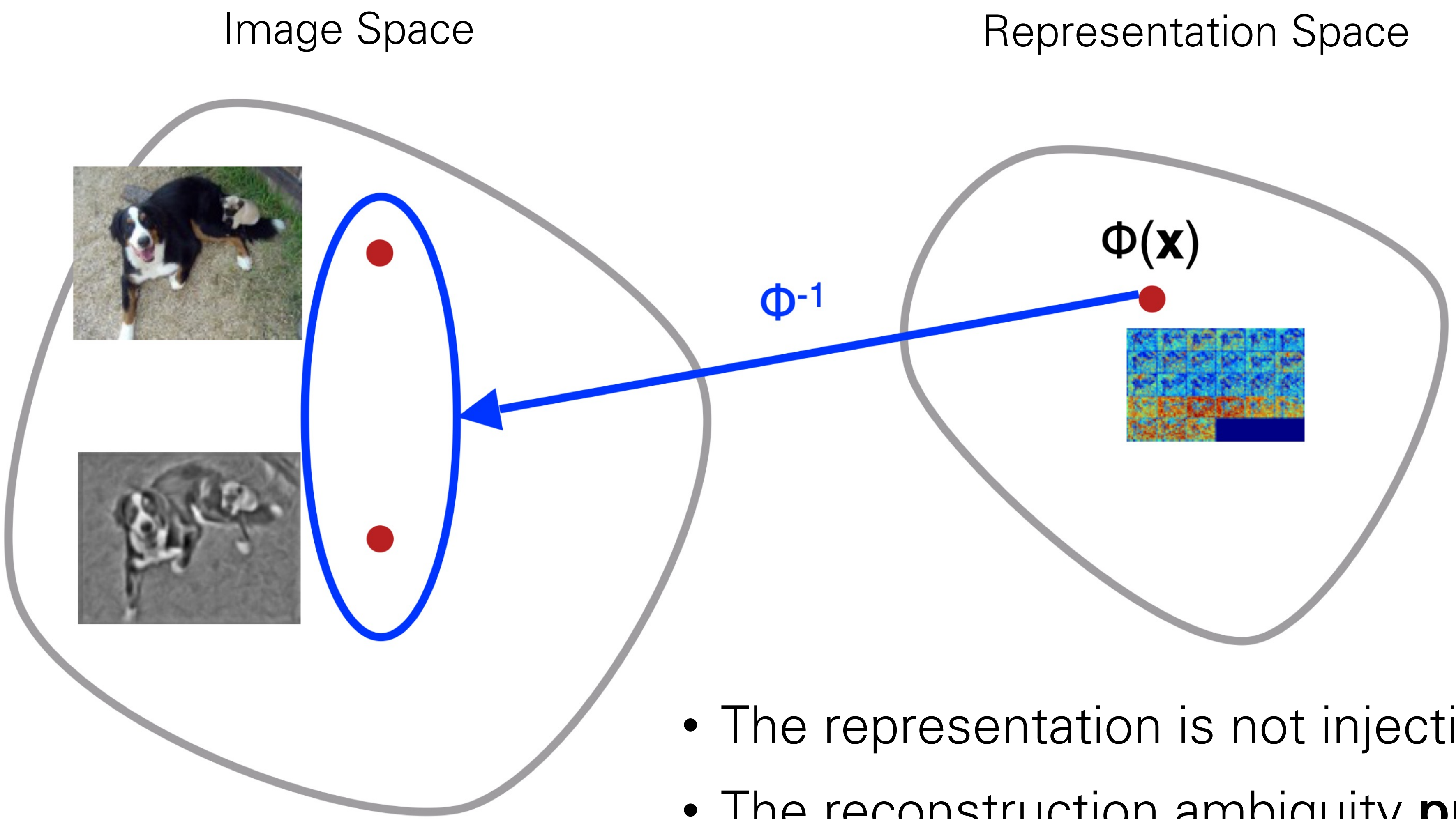
Nguyen et al, "Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks", ICML Visualization for Deep Learning Workshop 2016.

Understanding the Model: Pre-Images



- The representation is not injective

Understanding the Model: Pre-Images



- The representation is not injective
- The reconstruction ambiguity **provides useful information about the representation**

Finding a Pre-Image

A simple yet general and effective method

$$\min_{\mathbf{x}} \|\Phi(\mathbf{x}) - \Phi_0\|_2^2$$

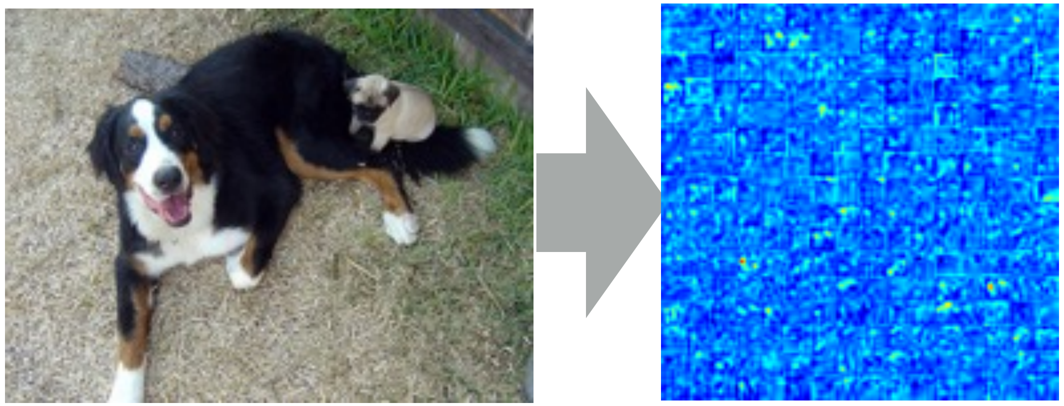


Image Representation

Finding a Pre-Image

A simple yet general and effective method

$$\min_{\mathbf{x}} \|\Phi(\mathbf{x}) - \Phi_0\|_2^2$$

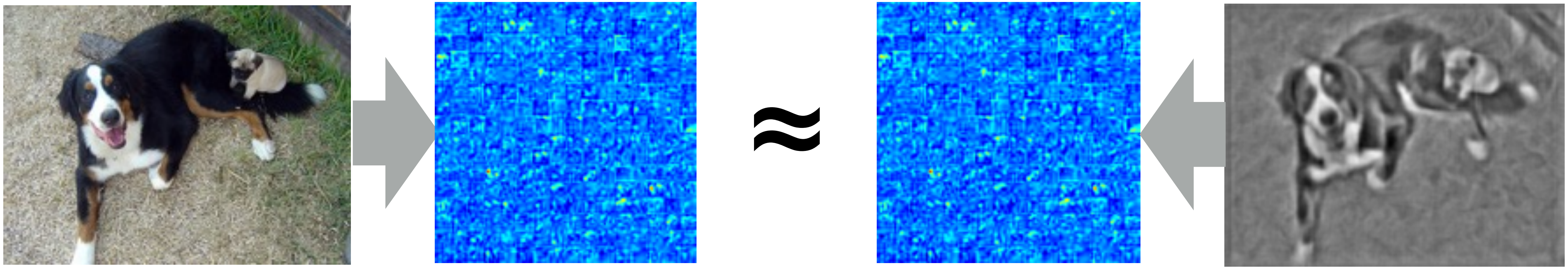


Image Representation Reconstruction Pre-Image

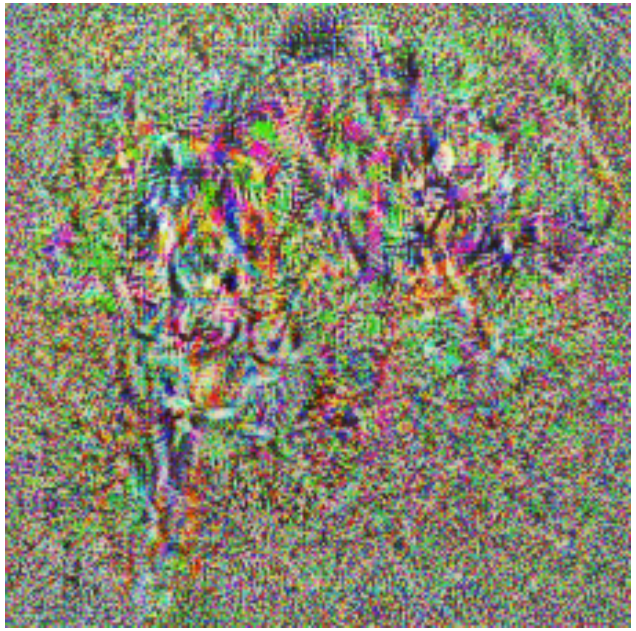
- Start from **random noise**
- Optimize using stochastic **gradient descent**

Finding a Pre-Image

A simple yet general and effective method

$$\min_{\mathbf{x}} \|\Phi(\mathbf{x}) - \Phi_0\|_2^2$$

No prior

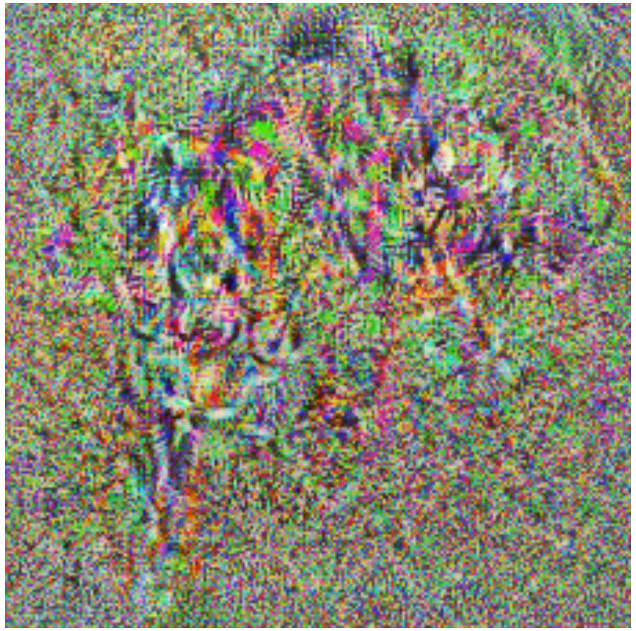


Finding a Pre-Image

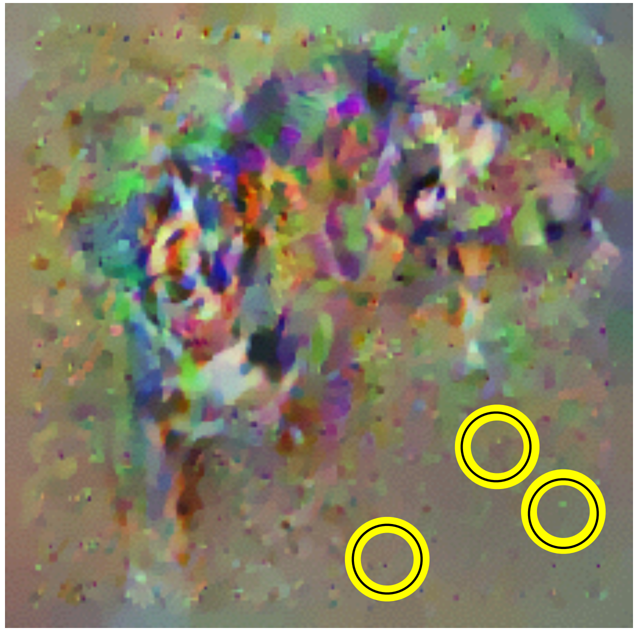
A simple yet general and effective method

$$\min_{\mathbf{x}} \|\Phi(\mathbf{x}) - \Phi_0\|_2^2 + R_{TV}(\mathbf{x})$$

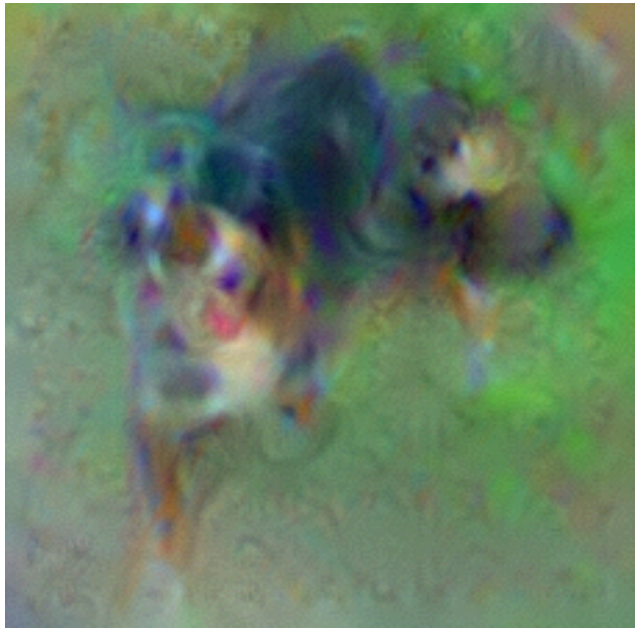
No prior



TV-norm β = 1

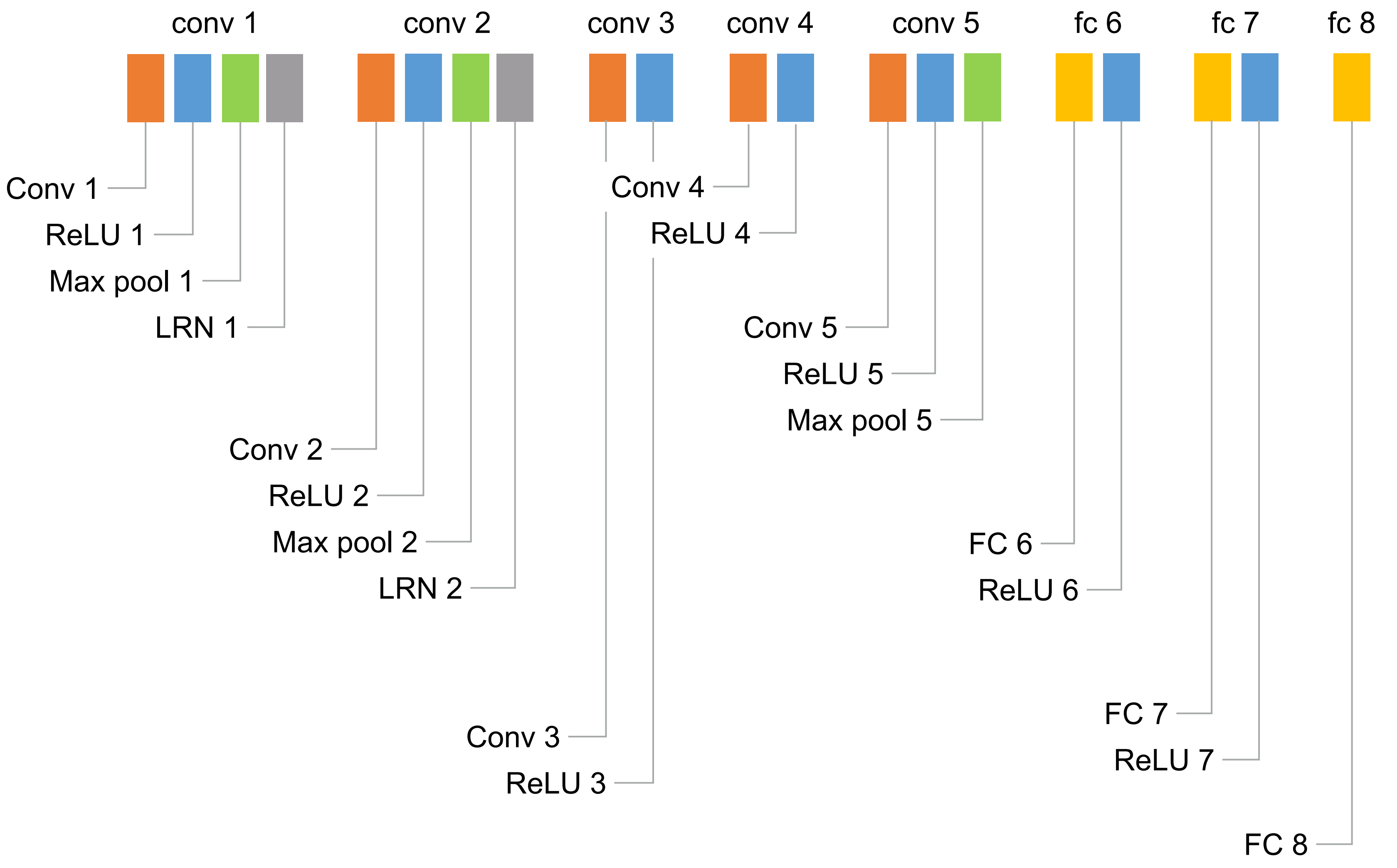


TV-norm β = 2

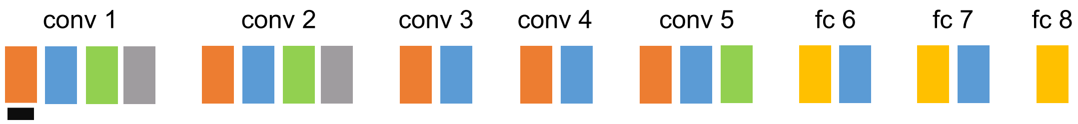


Inverting a Deep CNN

AlexNet [Krizhevsky et al. 2012]

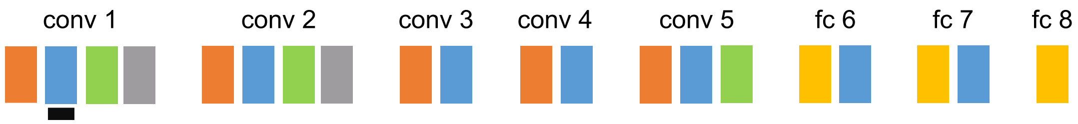


Inverting a Deep CNN



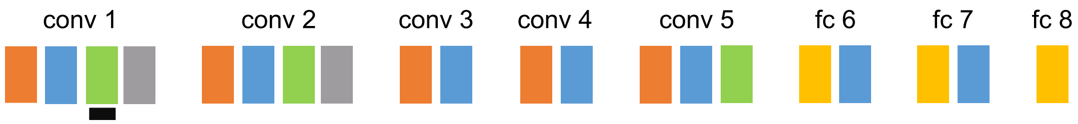
Original Image

Inverting a Deep CNN



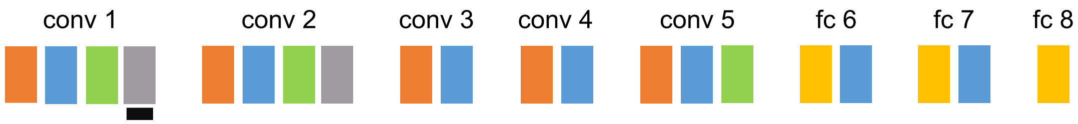
Original
Image

Inverting a Deep CNN

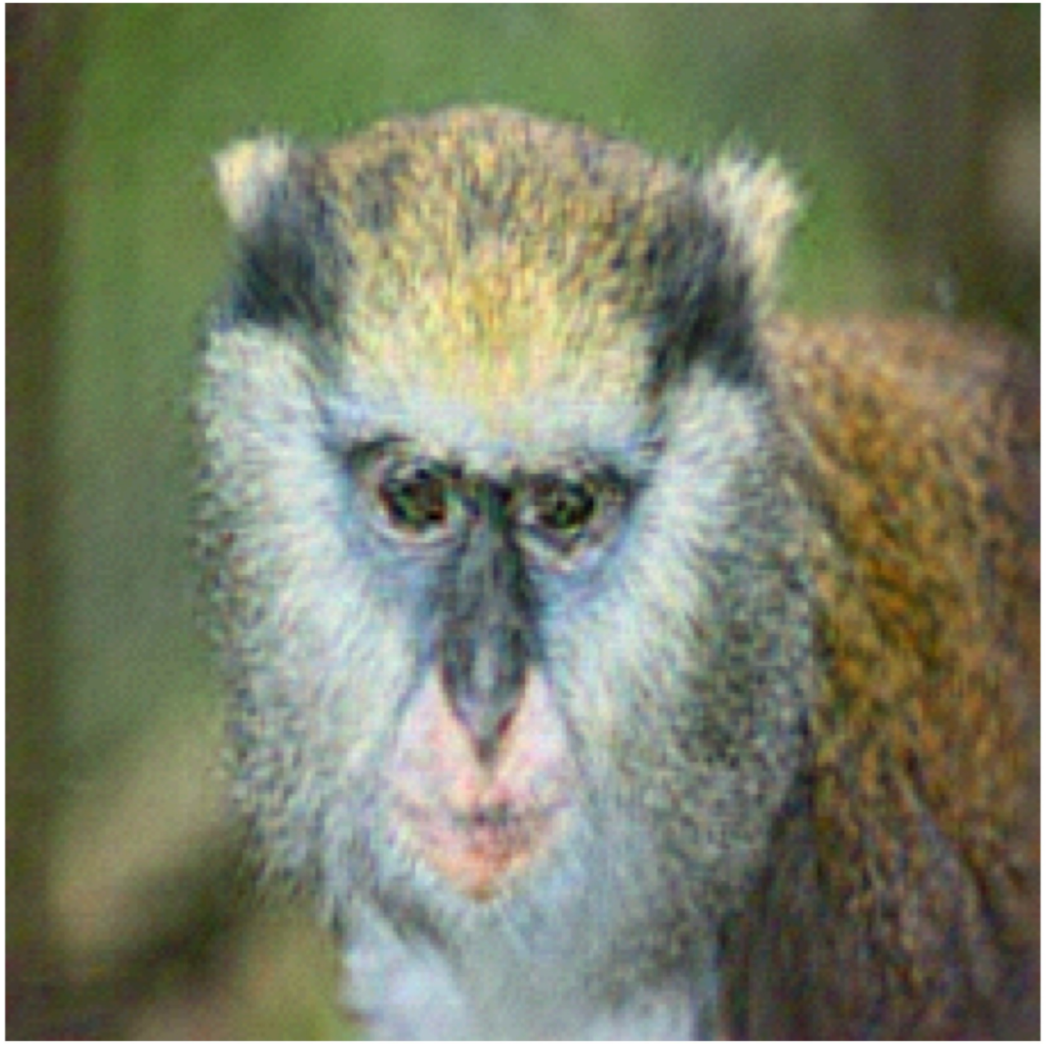


Original Image

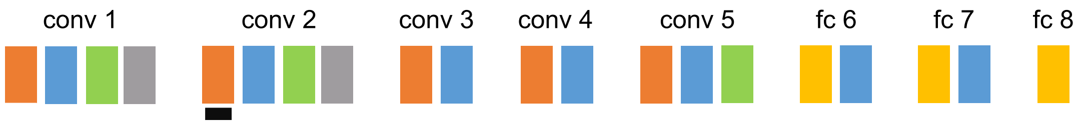
Inverting a Deep CNN



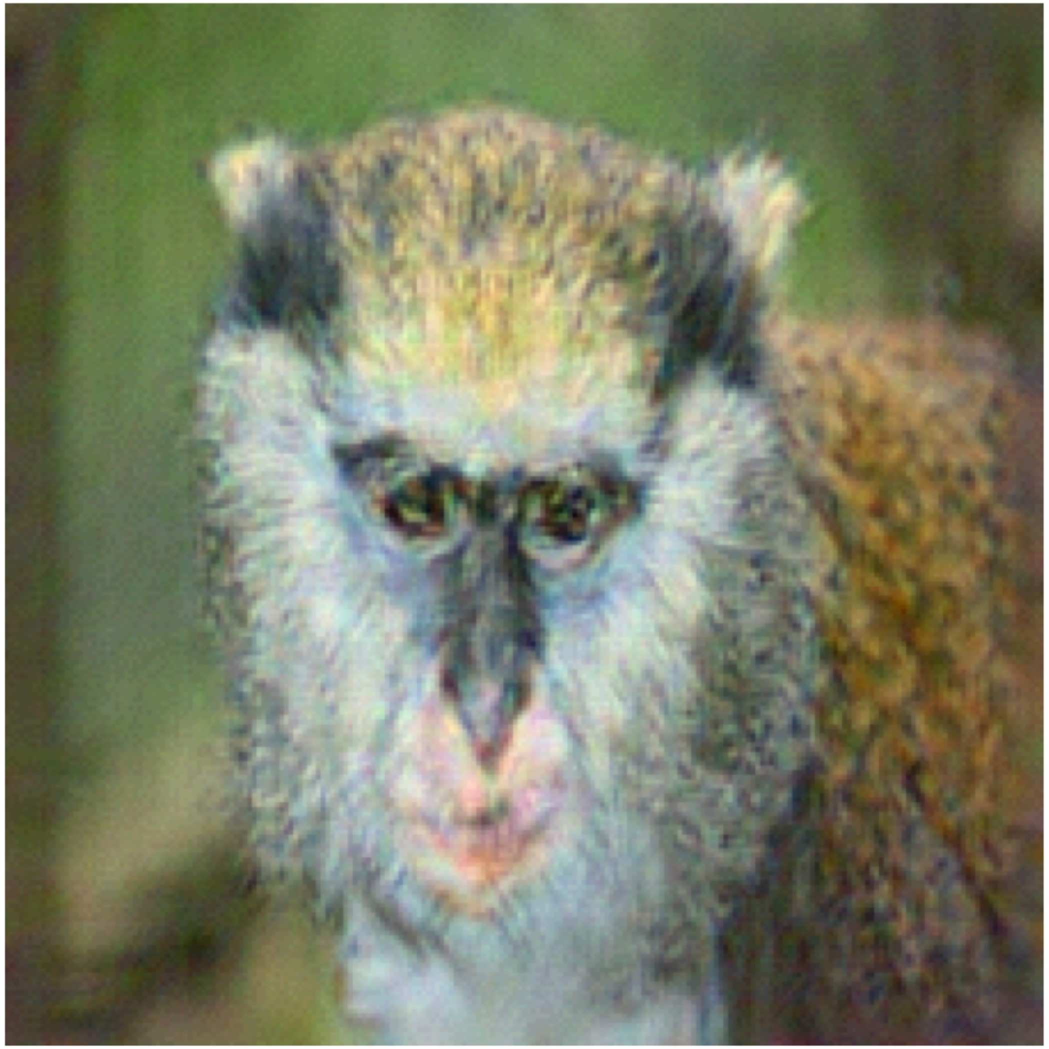
Original Image



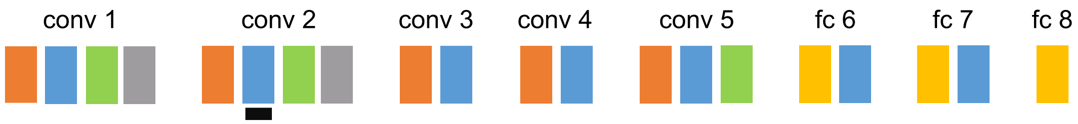
Inverting a Deep CNN



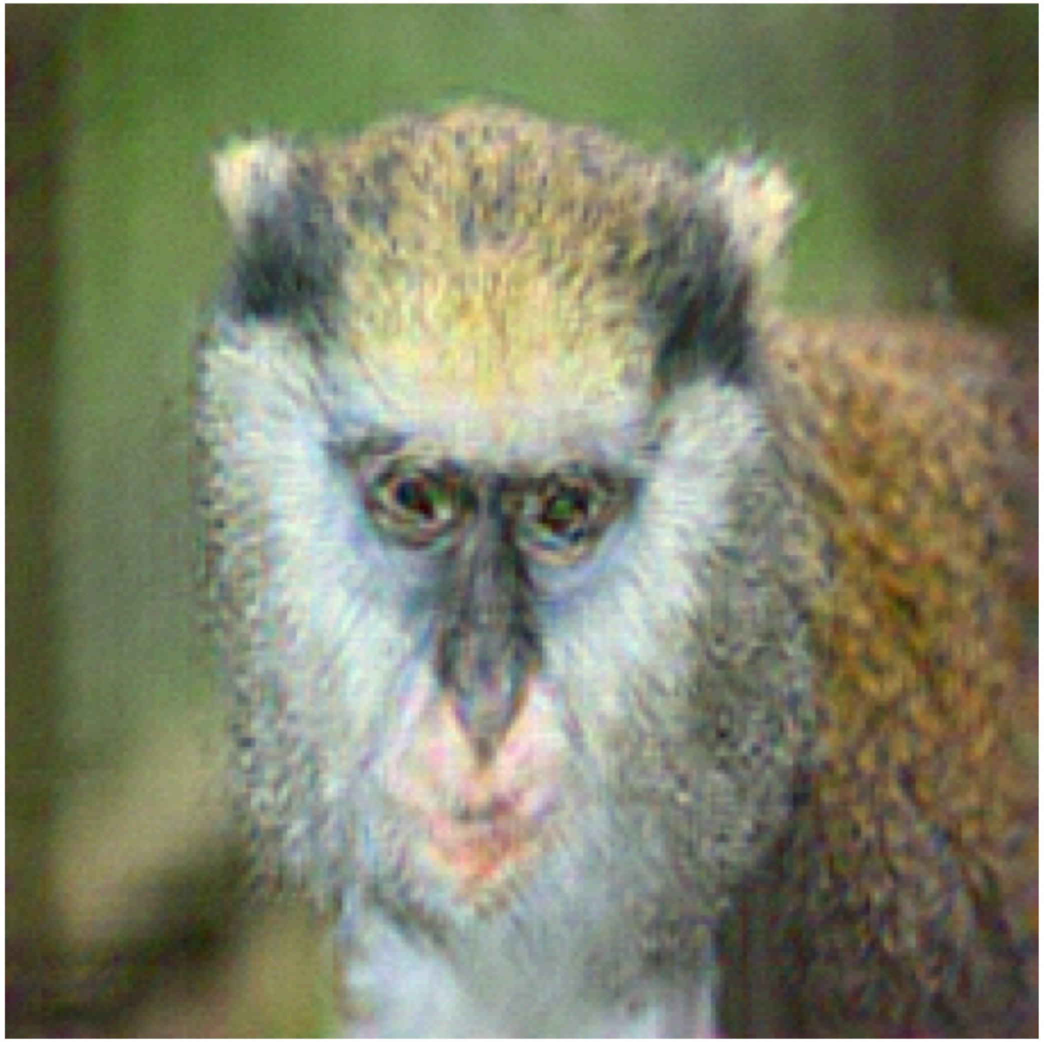
Original
Image



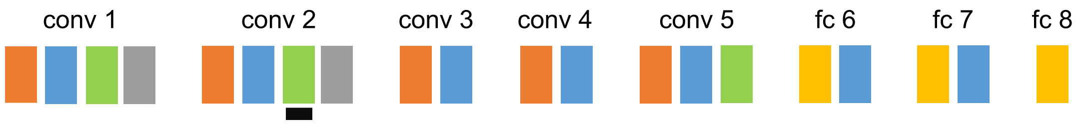
Inverting a Deep CNN



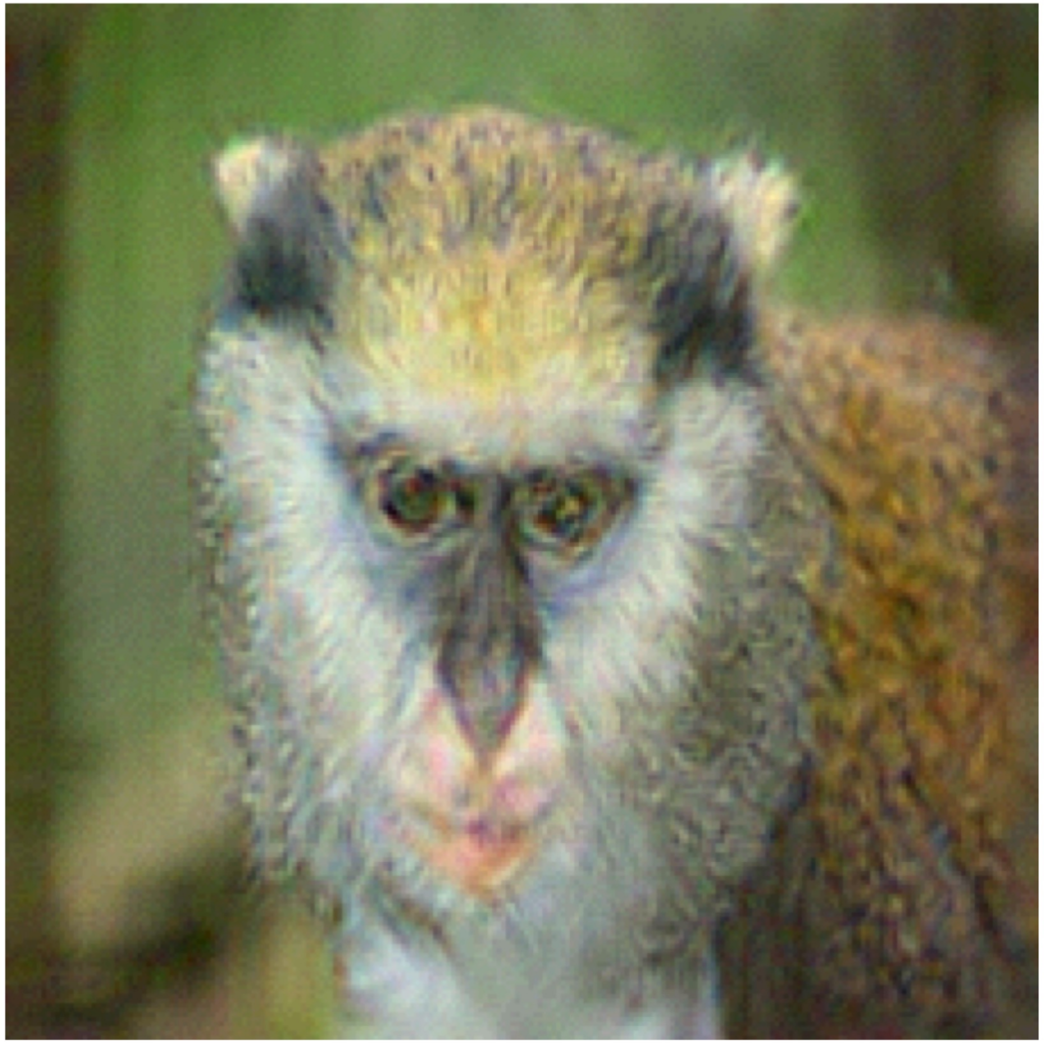
Original Image



Inverting a Deep CNN

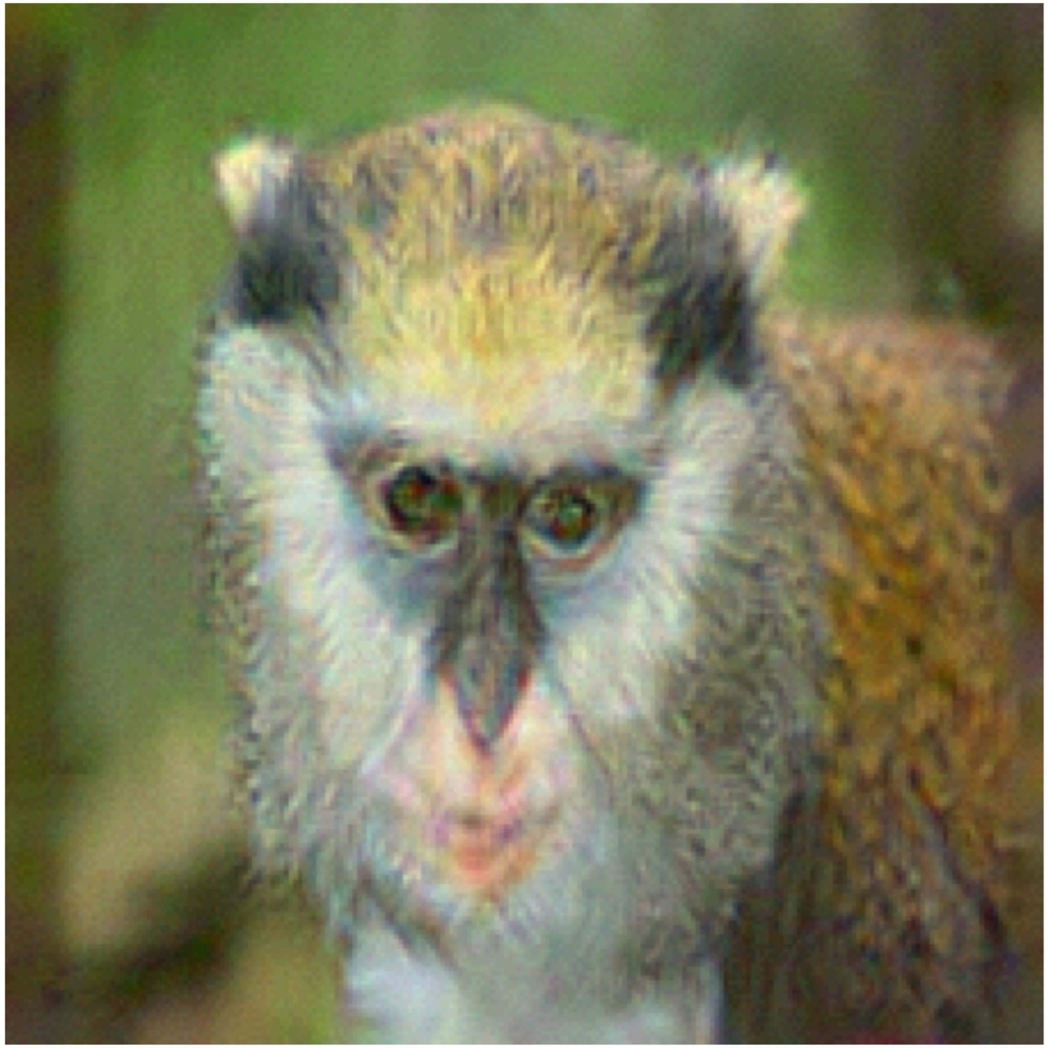


Original Image

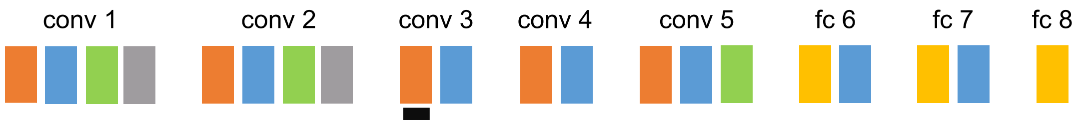


Inverting a Deep CNN

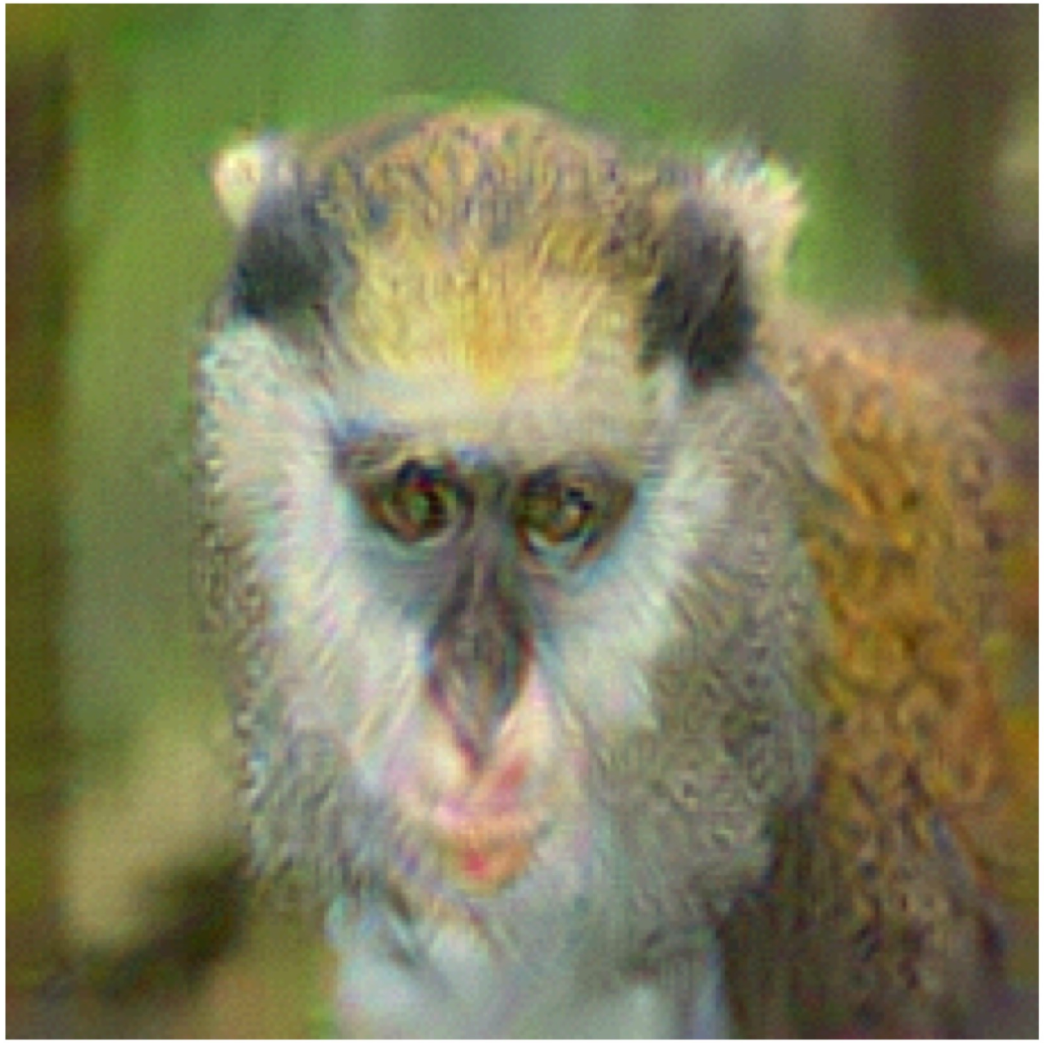
Original
Image



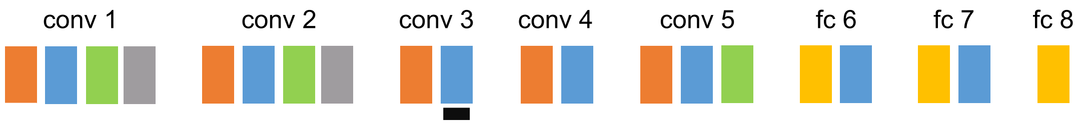
Inverting a Deep CNN



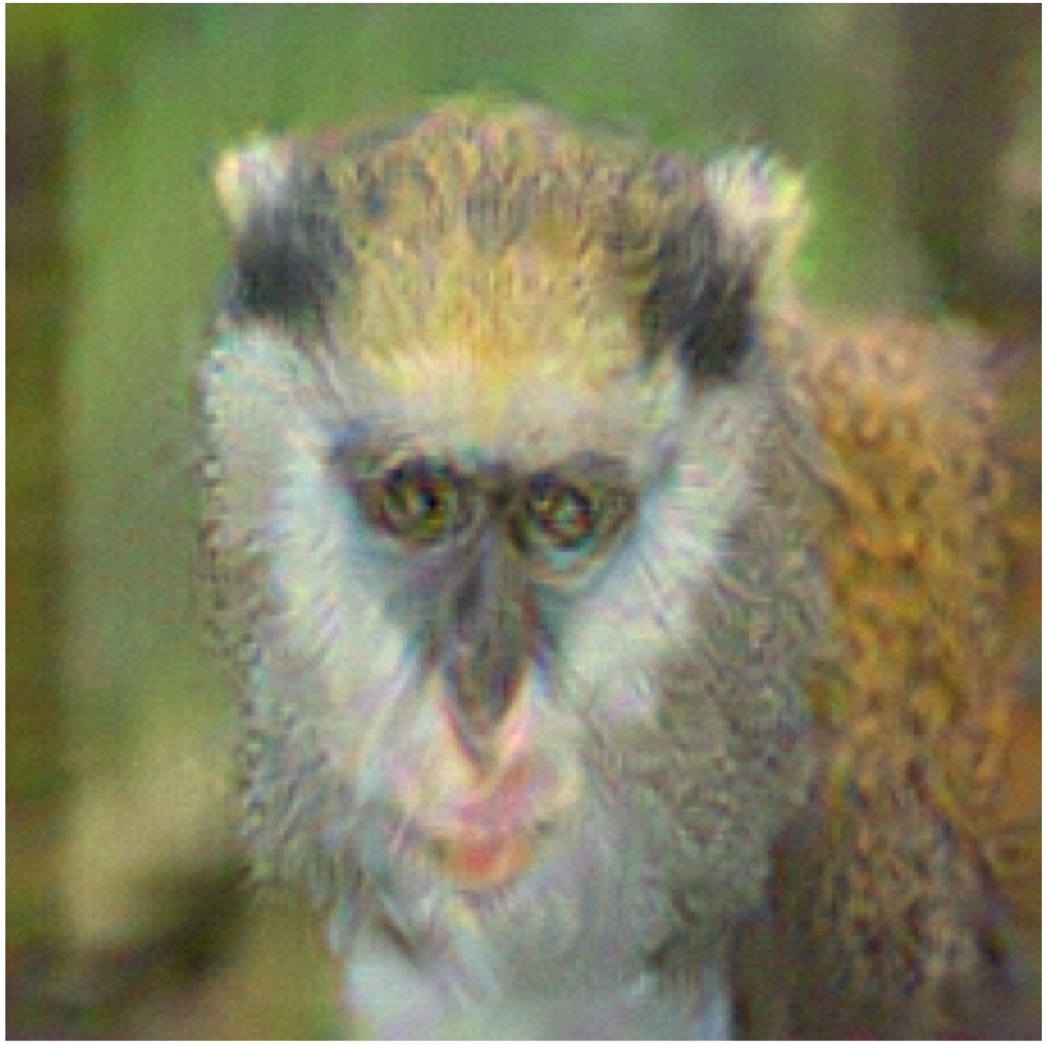
Original
Image



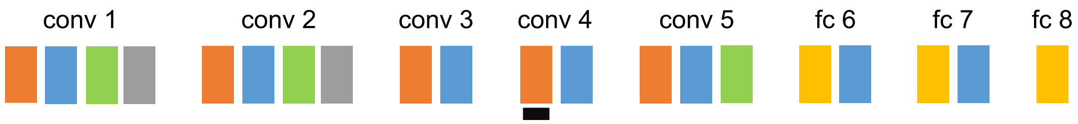
Inverting a Deep CNN



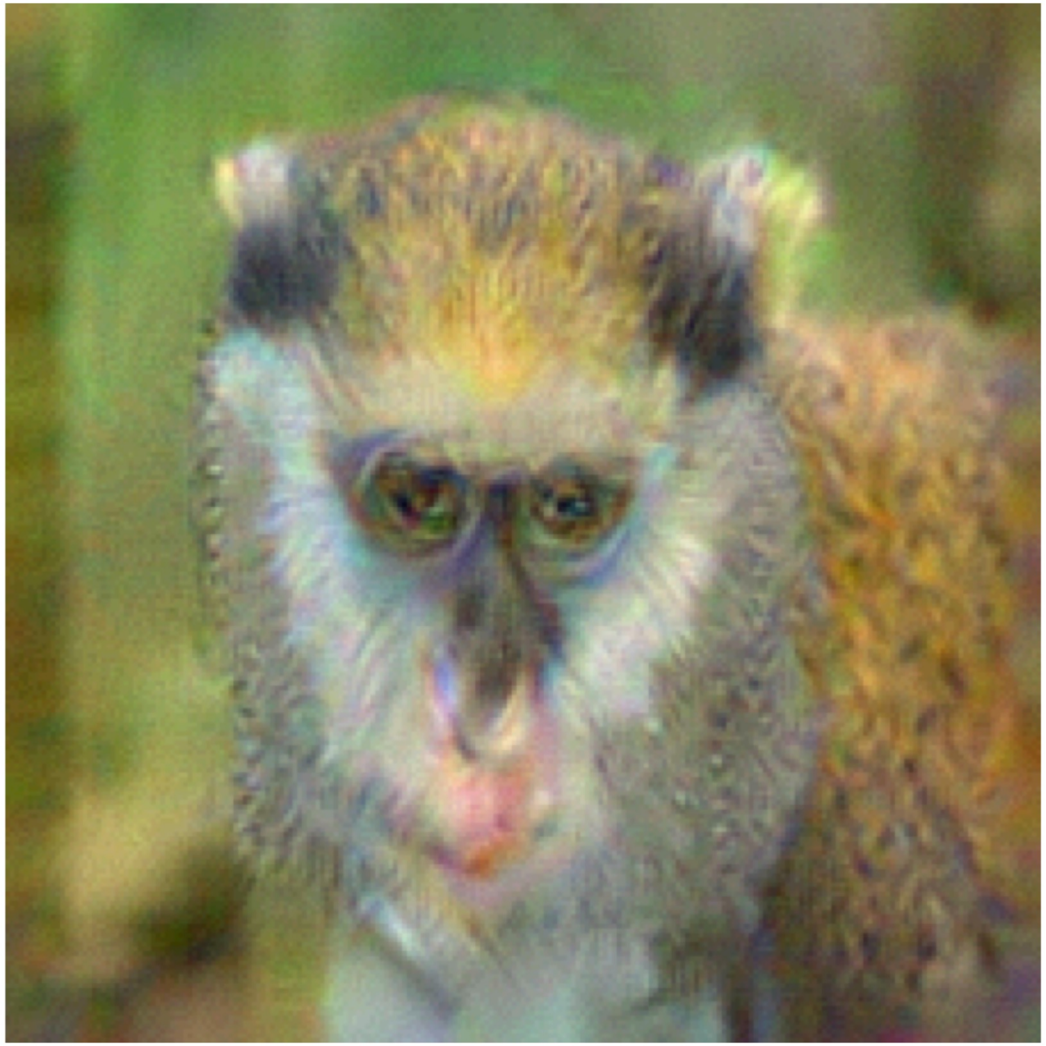
Original Image



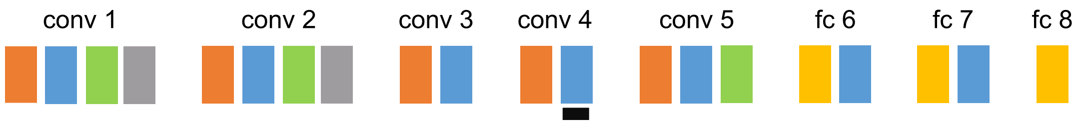
Inverting a Deep CNN



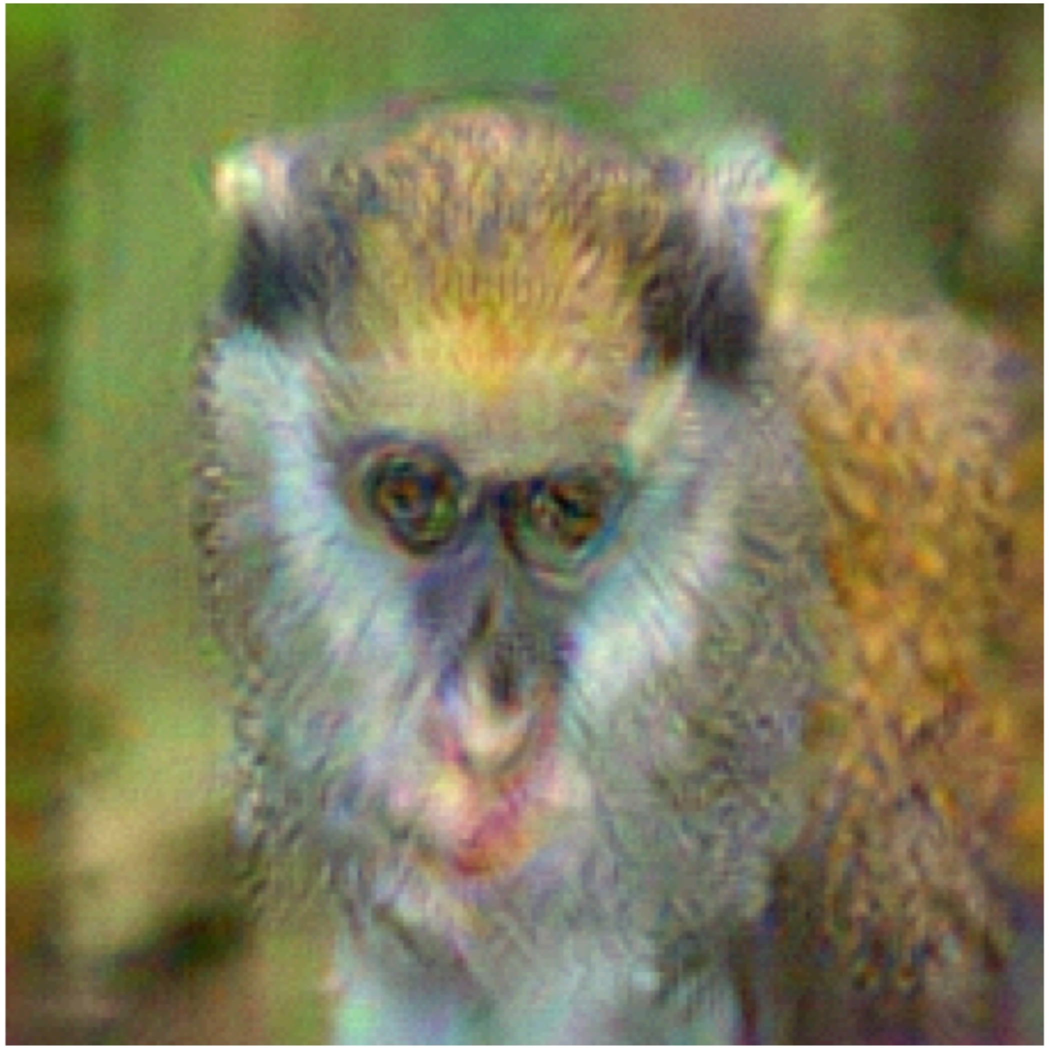
Original
Image



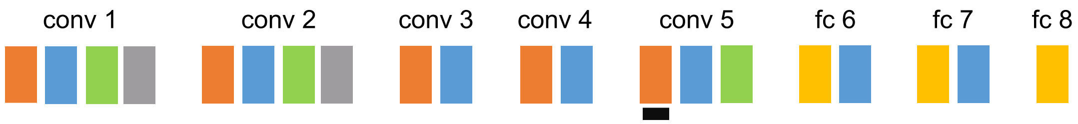
Inverting a Deep CNN



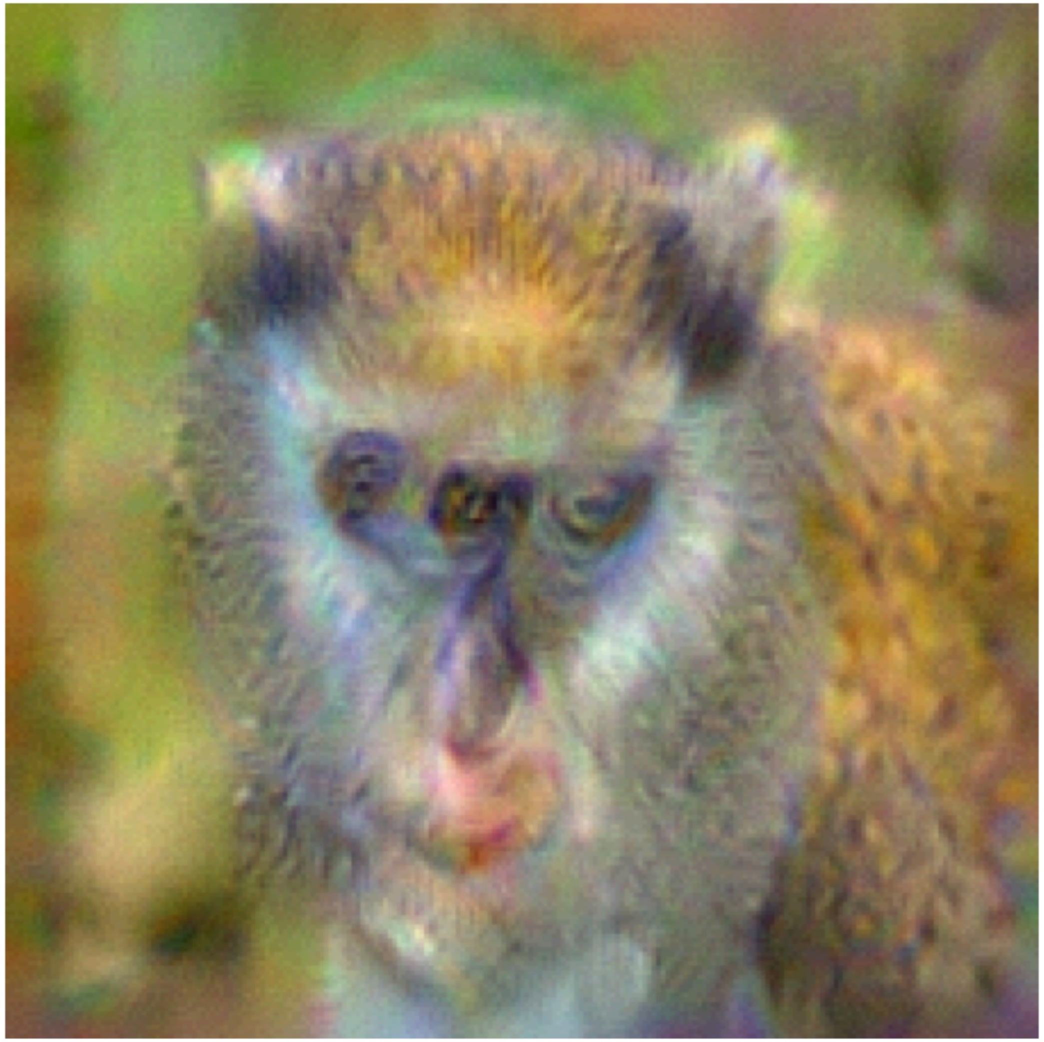
Original
Image



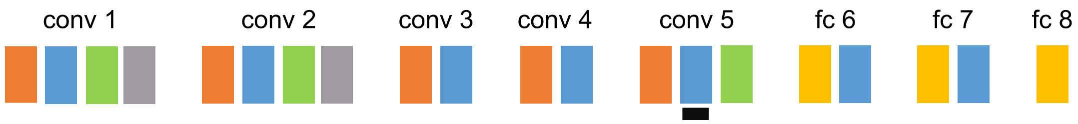
Inverting a Deep CNN



Original
Image

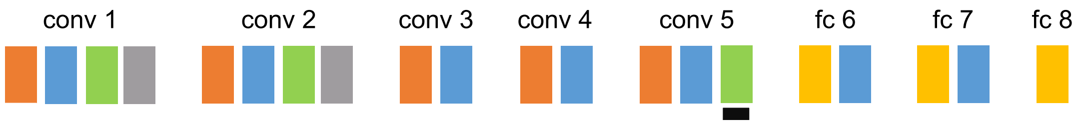


Inverting a Deep CNN



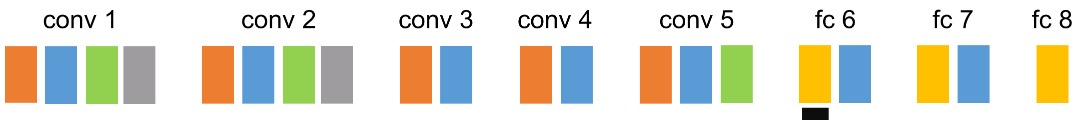
Original
Image

Inverting a Deep CNN

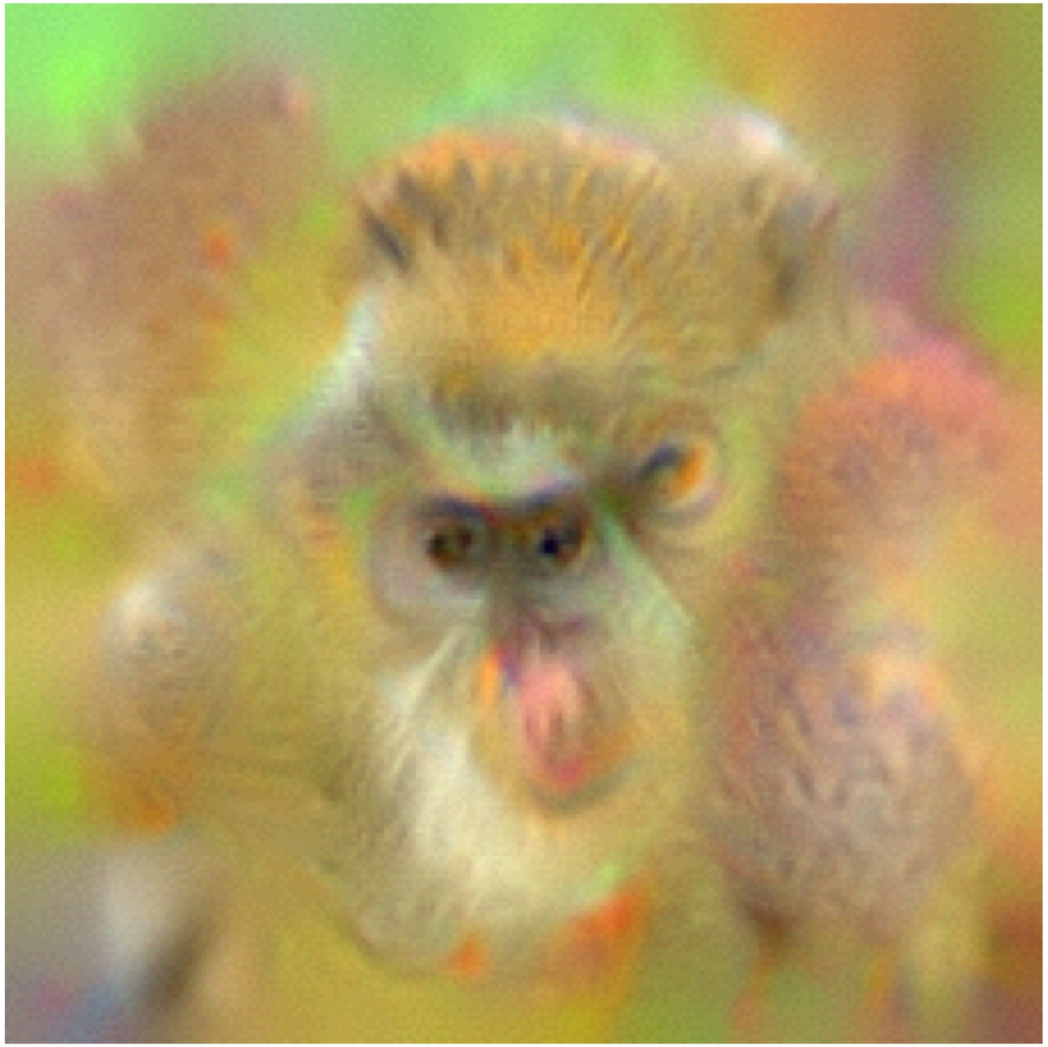


Original Image

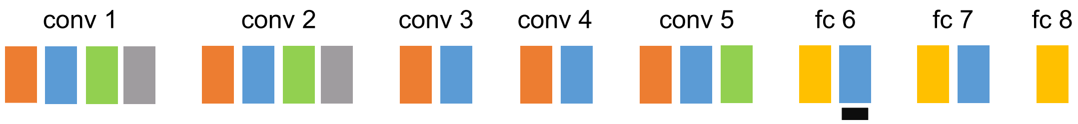
Inverting a Deep CNN



Original Image

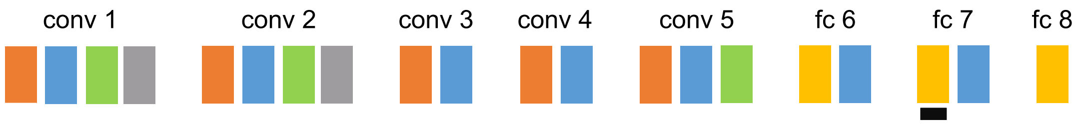


Inverting a Deep CNN

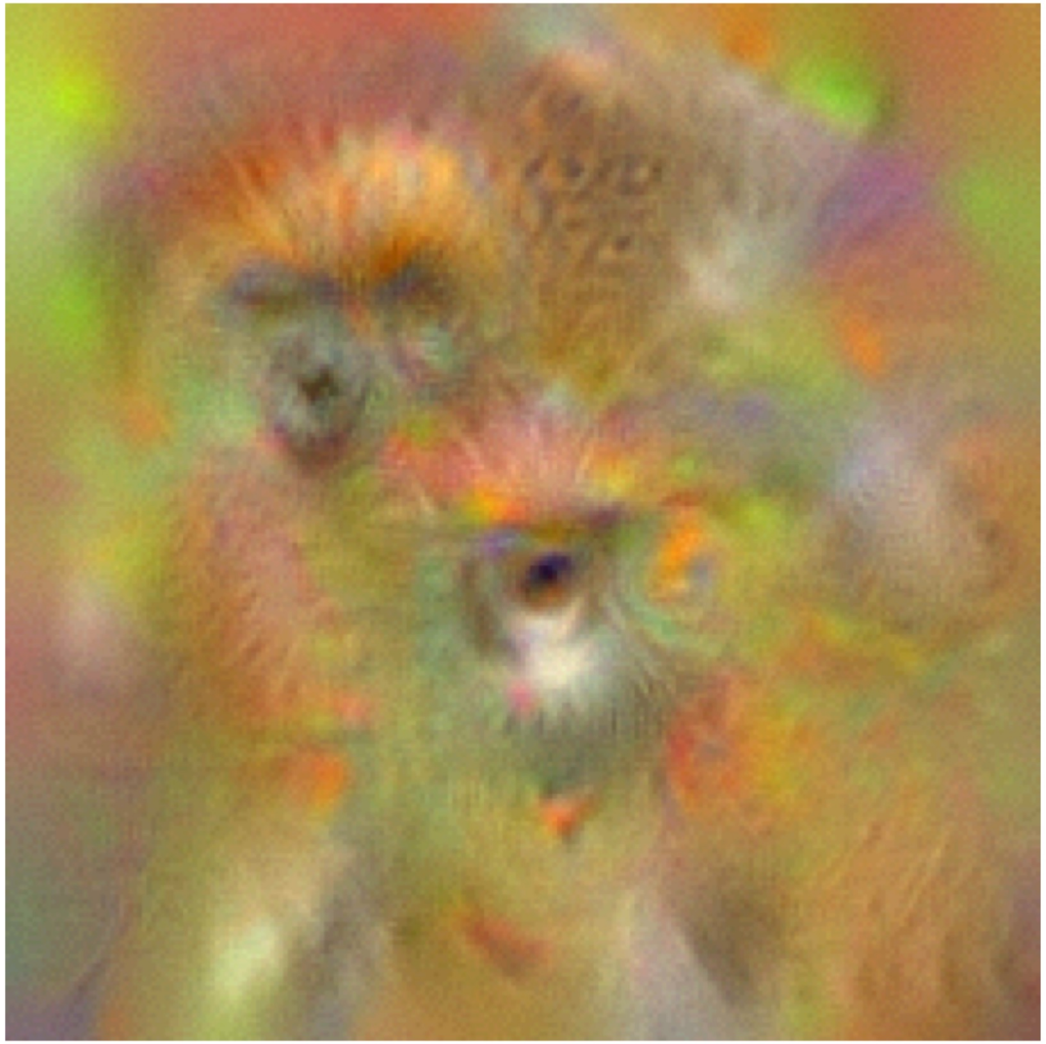


Original Image

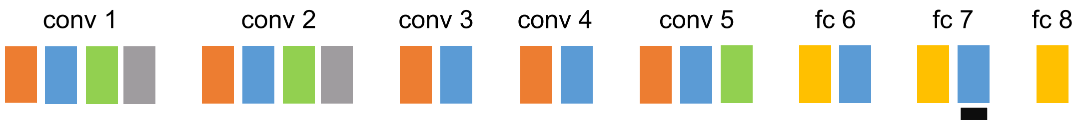
Inverting a Deep CNN



Original Image

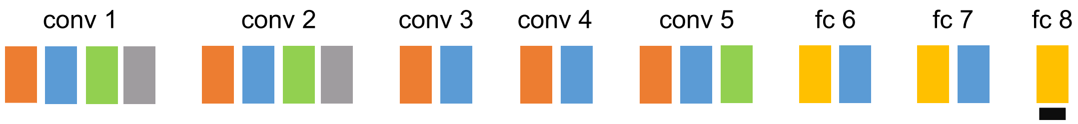


Inverting a Deep CNN



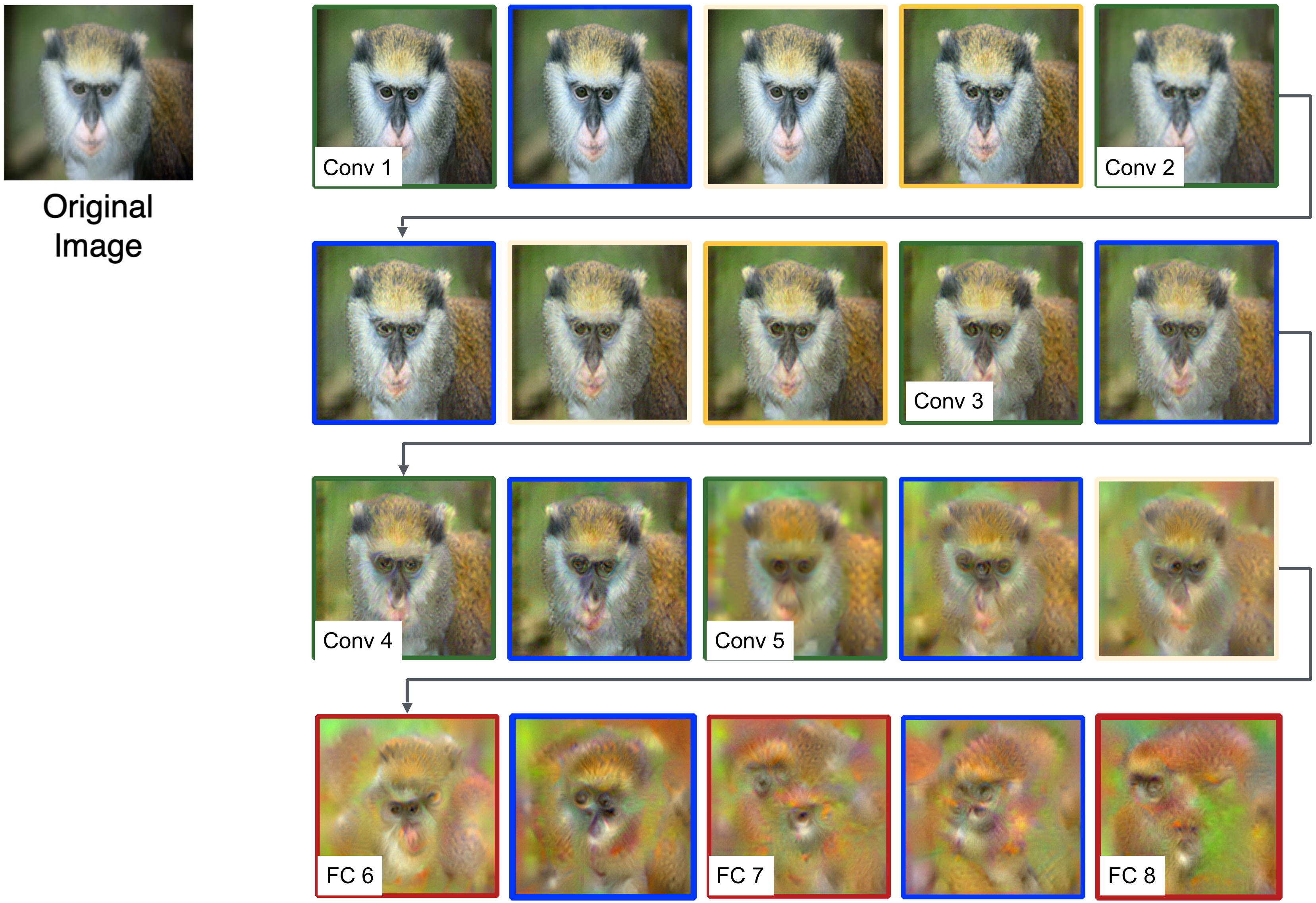
Original Image

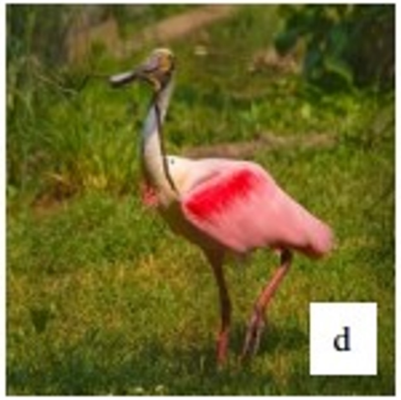
Inverting a Deep CNN



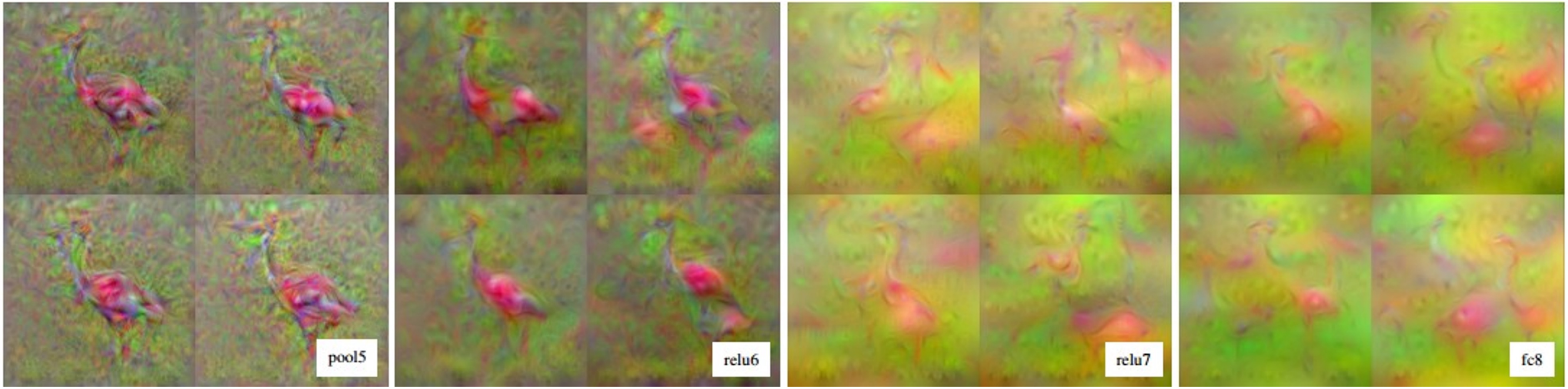
Original Image

Inverting a Deep CNN





Multiple reconstructions. Images in quadrants all "look" the same to the CNN (same code)



Inverting Visual Representations with Convolutional Networks [Dosovitskiy and Brox2016]

Minimize mean squared error:

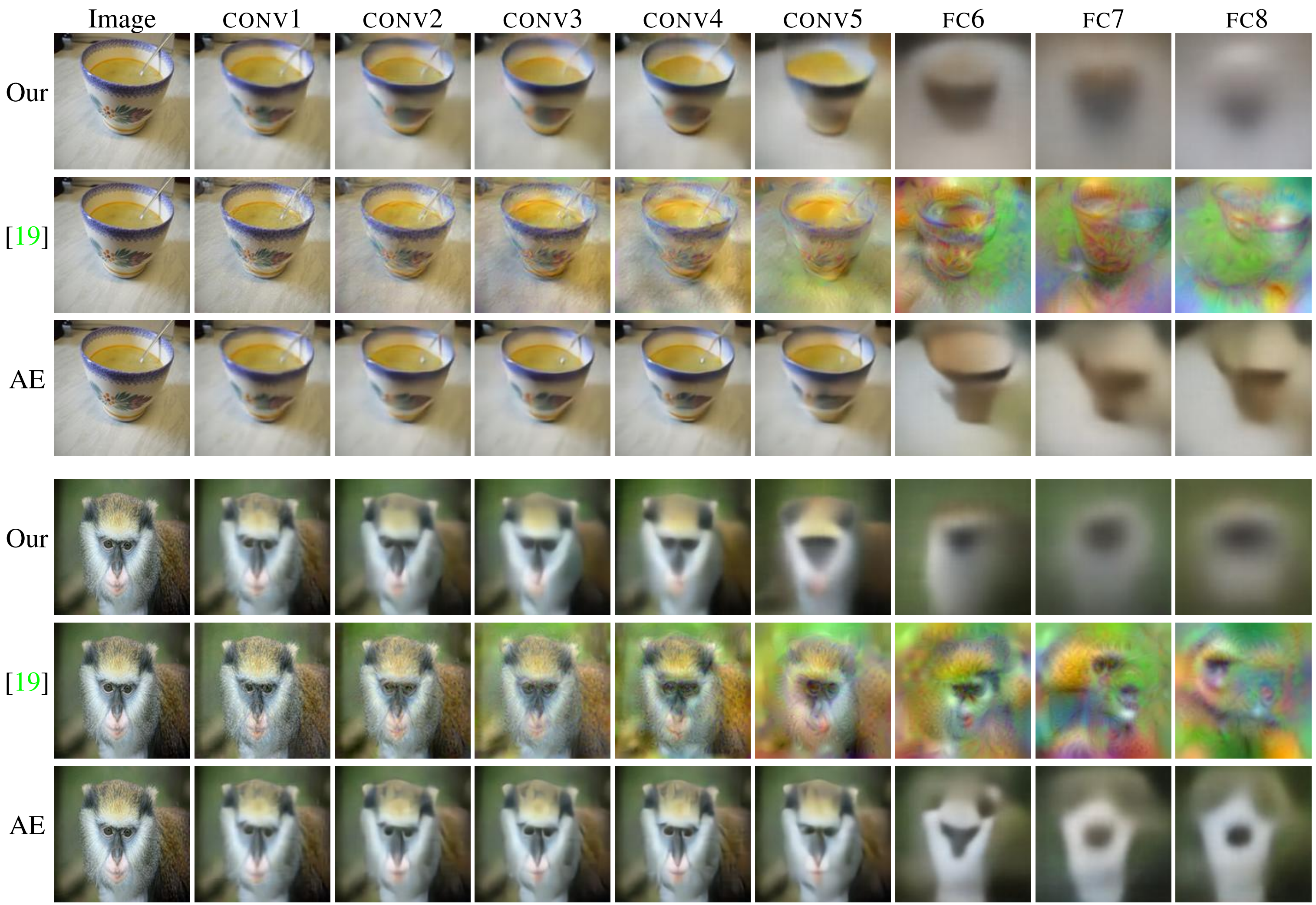
$$\mathbb{E}_{\mathbf{x},\boldsymbol{\phi}}\,||\mathbf{x} - f(\boldsymbol{\phi})||^2$$

Pre-image as the conditional expectation:

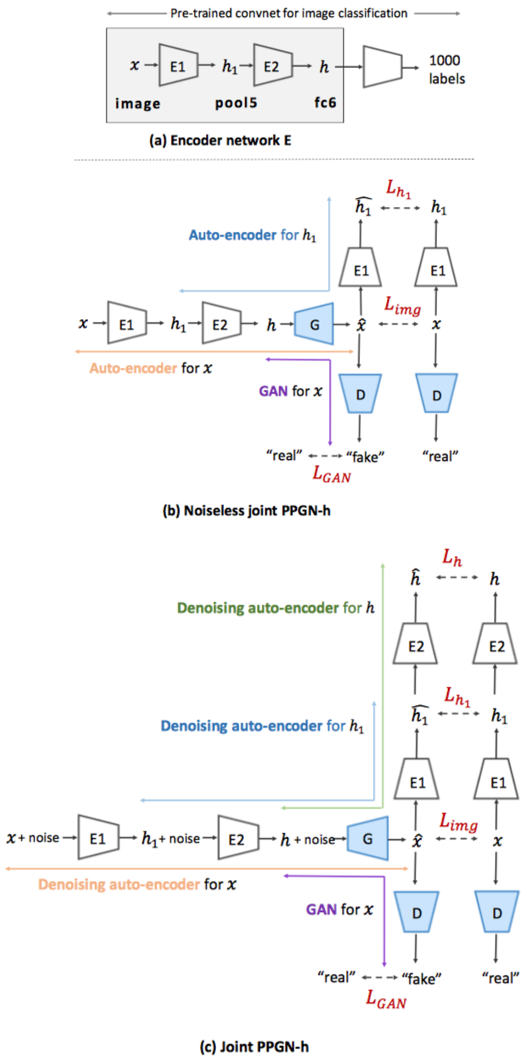
$$\hat{f}(\boldsymbol{\phi}_0) = \mathbb{E}_{\mathbf{x}}\left[\mathbf{x}\,|\,\boldsymbol{\phi} = \boldsymbol{\phi}_0\right],$$

Given a training set of images and their features, learn weights of an deconvolutional network:

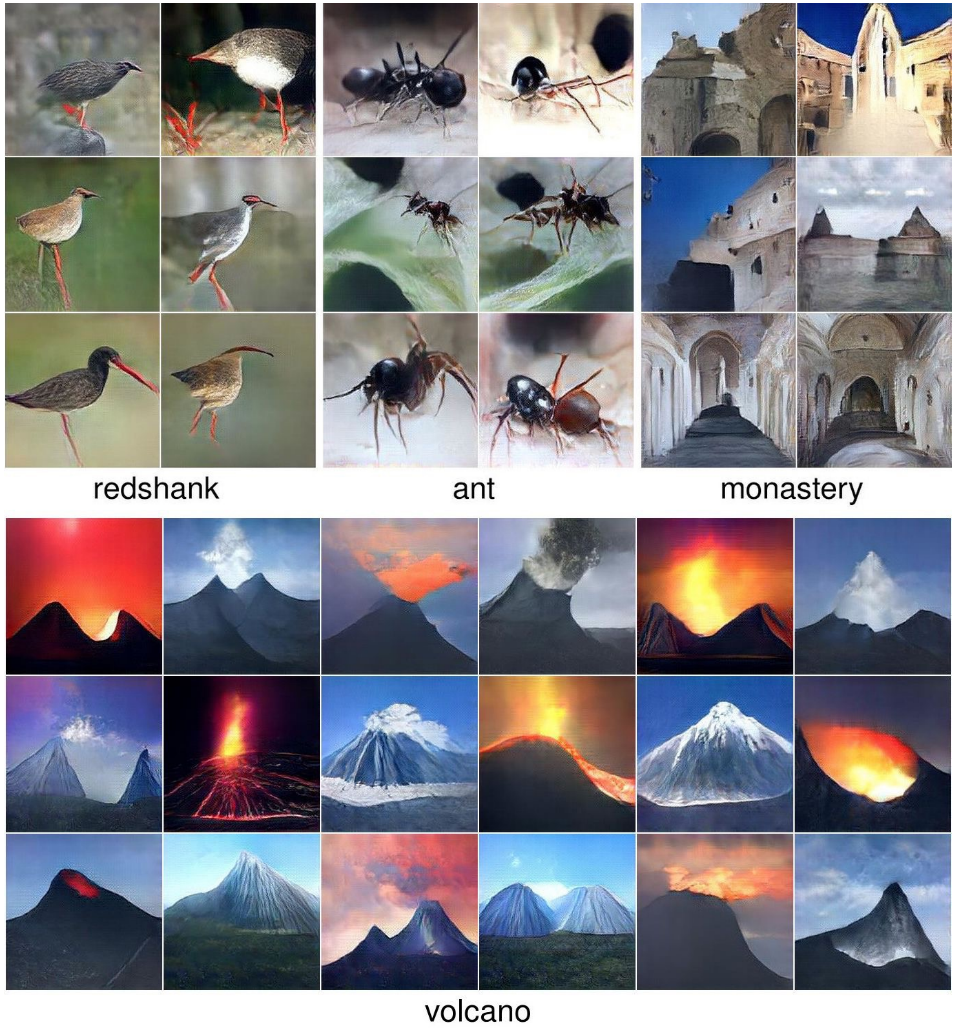
$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_i ||\mathbf{x}_i - f(\boldsymbol{\phi}_i, \mathbf{w})||_2^2.$$



Visualizing CNN Features: Gradient Ascent



Employs auto-encoder and generative adversarial network components



Nguyen et al, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," NIPS 2016

Visualizing CNN Features: Gradient Ascent

Nguyen et al, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," NIPS 2016

Caricaturization

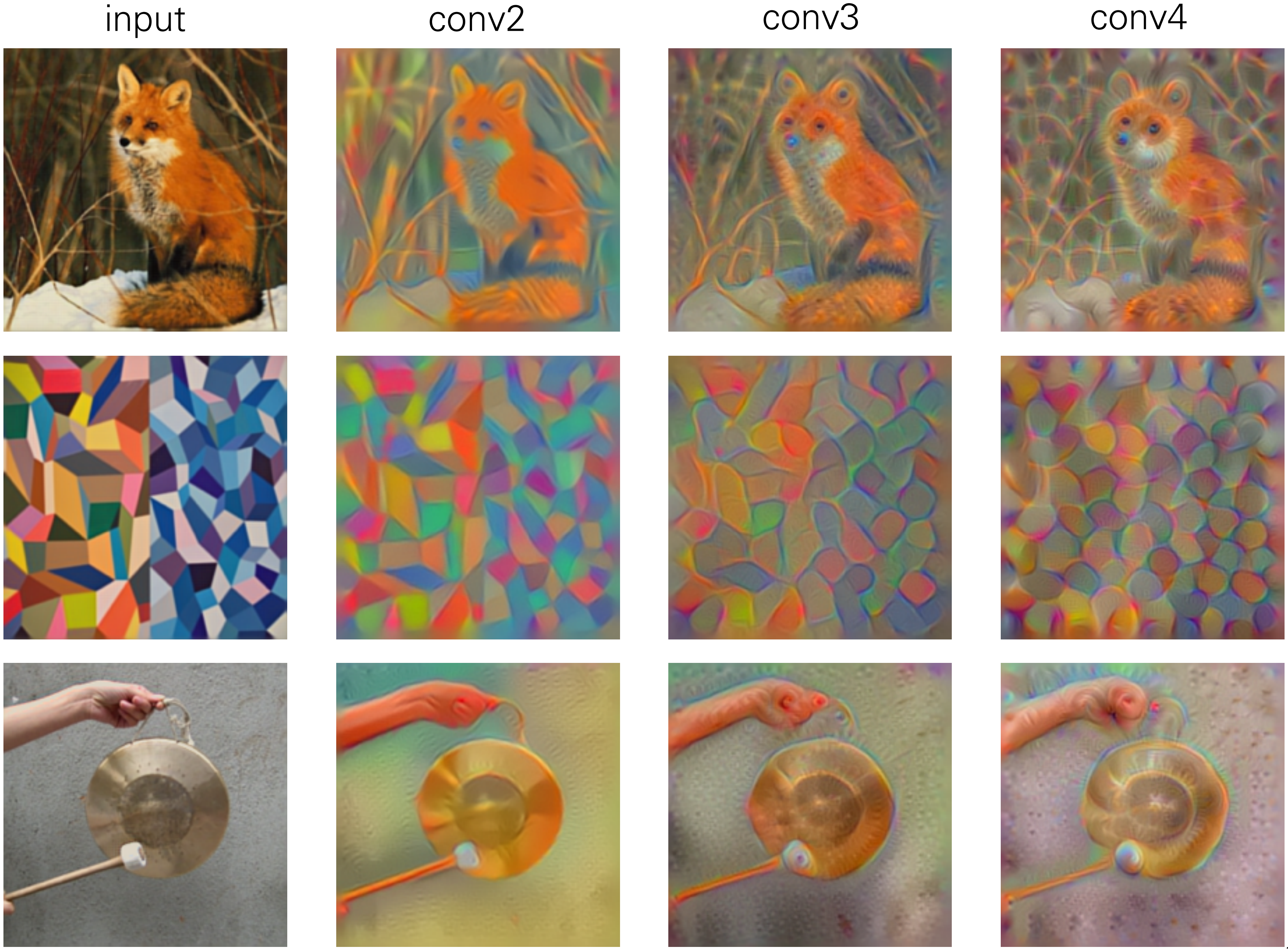
[Google Inceptionism 2015, Mahendran et al. 2015]

- Emphasize patterns that are detected by a certain representation

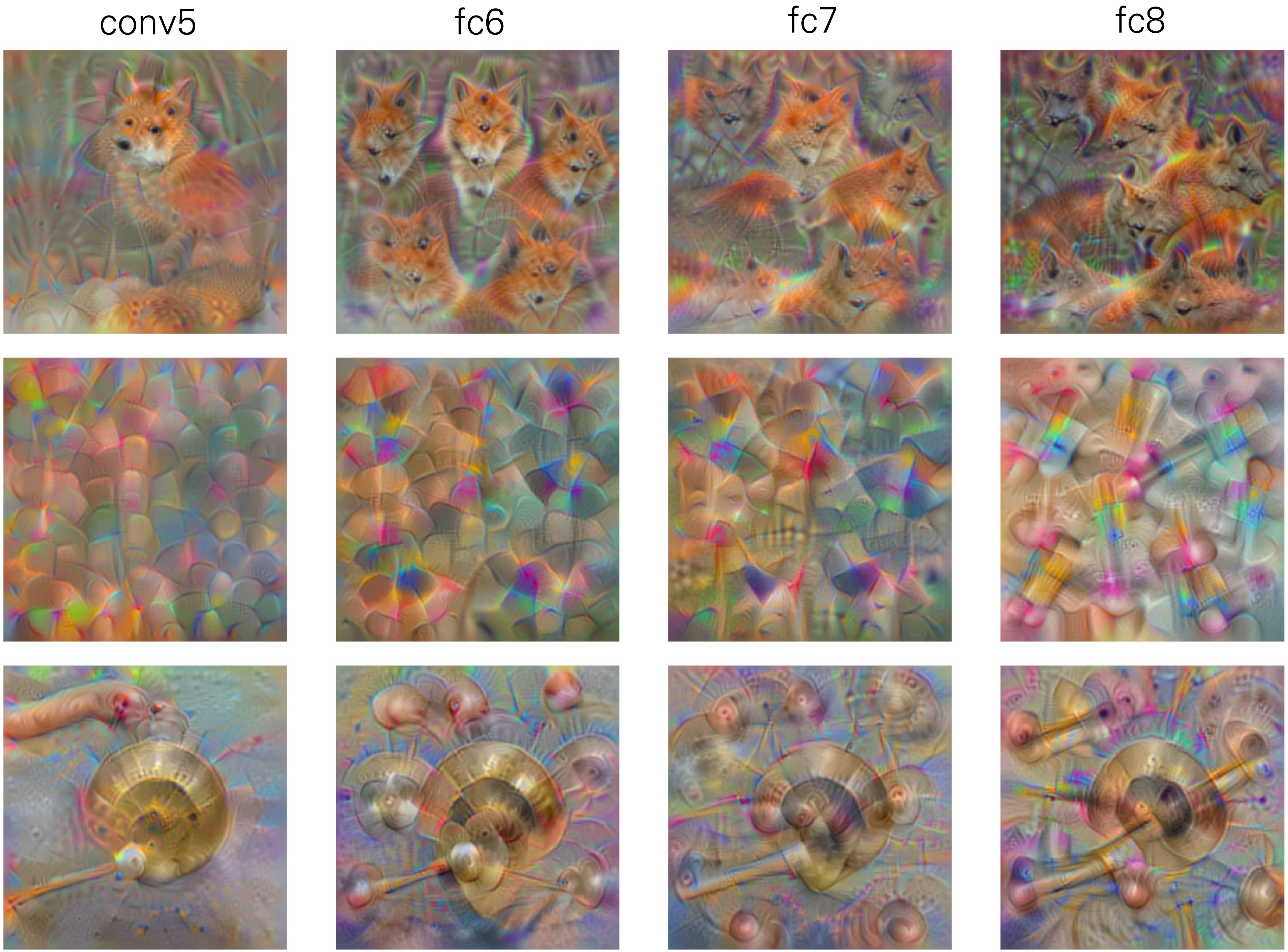
$$\boxed{\min_{\mathbf{x}} -\langle \Phi(\mathbf{x}_0), \Phi(\mathbf{x})\rangle + R_{TV}(\mathbf{x}) + R_\alpha(\mathbf{x})}$$

- Key differences:
  - The starting point **is** the image $\mathbf{x}_0$
  - particular configurations of features are emphasized, not individual features

Results (VGG-M)



Results (VGG-M)



Interlude: Neural Art

- Surprisingly, the filters learned by discriminative neural networks capture well the "style" of an image.

  This can be used to transfer the style of an image (e.g. a painting) to any other.

Optimization based

- L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. In Proc. NIPS, 2015.

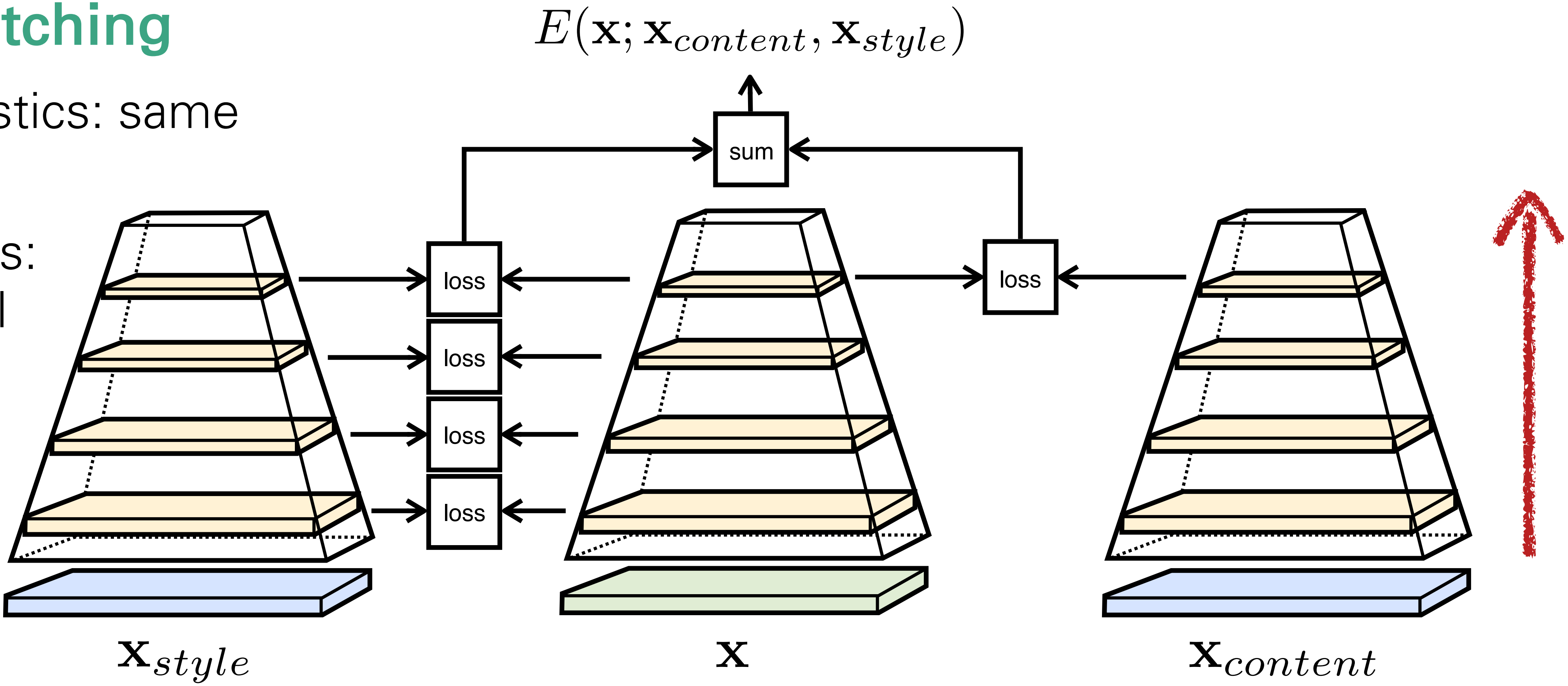
Feed-forward neural network equivalents

- D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. Proc. ICML, 2016.
- J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In Proc. ECCV, 2016.

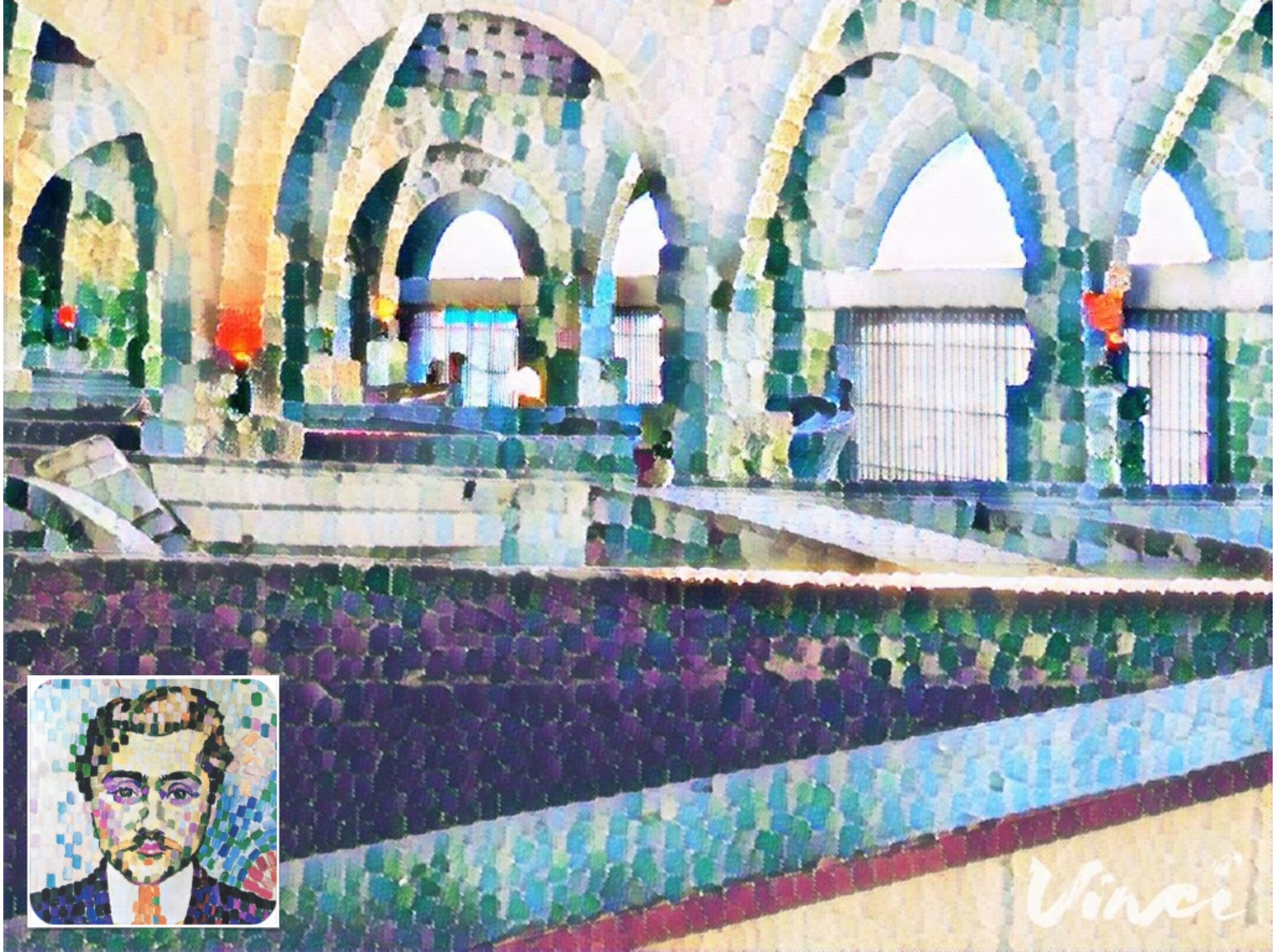
Generation by Moment Matching

Moment matching

- Content statistics: same as inversion
- Style statistics: cross-channel correlations



$$\mathbf{x}^* = \arg\min_{\mathbf{x}} E(\mathbf{x}; \mathbf{x}_{content}, \mathbf{x}_{style})$$


Vinci


Vinci

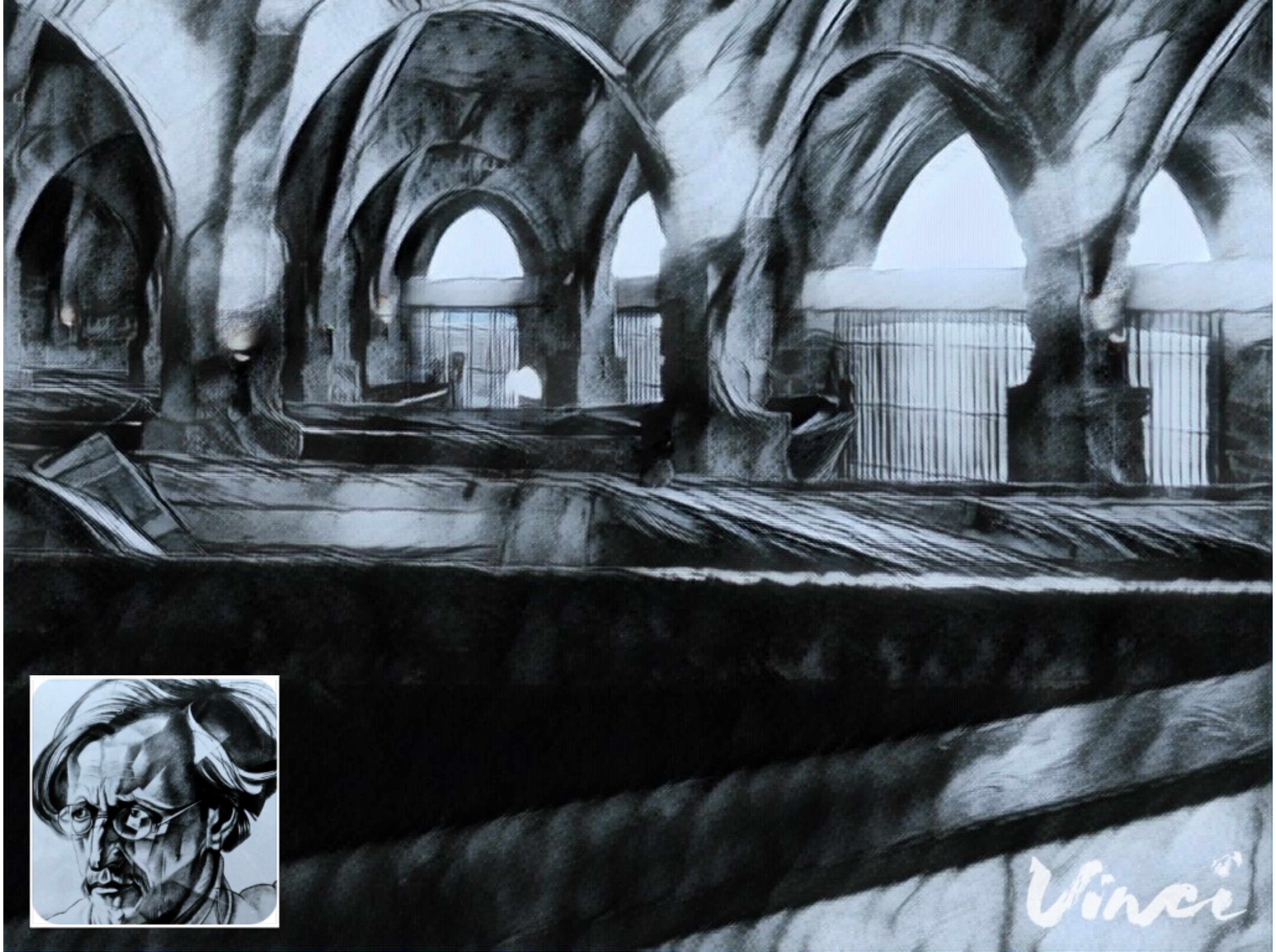

Vinci


Vinci


Vinci


Vinci


Vinci


Vinci


Vinci


Vinci

Artistic style transfer for videos

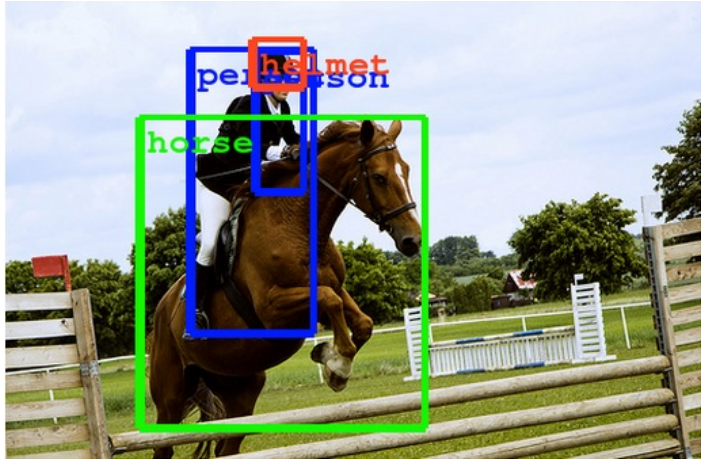
Manuel Ruder
Alexey Dosovitskiy
Thomas Brox

University of Freiburg
Chair of Pattern Recognition and Image Processing

https://www.youtube.com/watch?v=Khuj4ASldmU

Fooling Deep Networks

Since 2013, deep neural networks have matched human performance at...



(Szegedy et al, 2014)

...recognizing objects and faces....

(Taigmen et al, 2013)

(Goodfellow et al, 2013)

...solving CAPTCHAS and reading addresses...

(Goodfellow et al, 2013)

and other tasks...

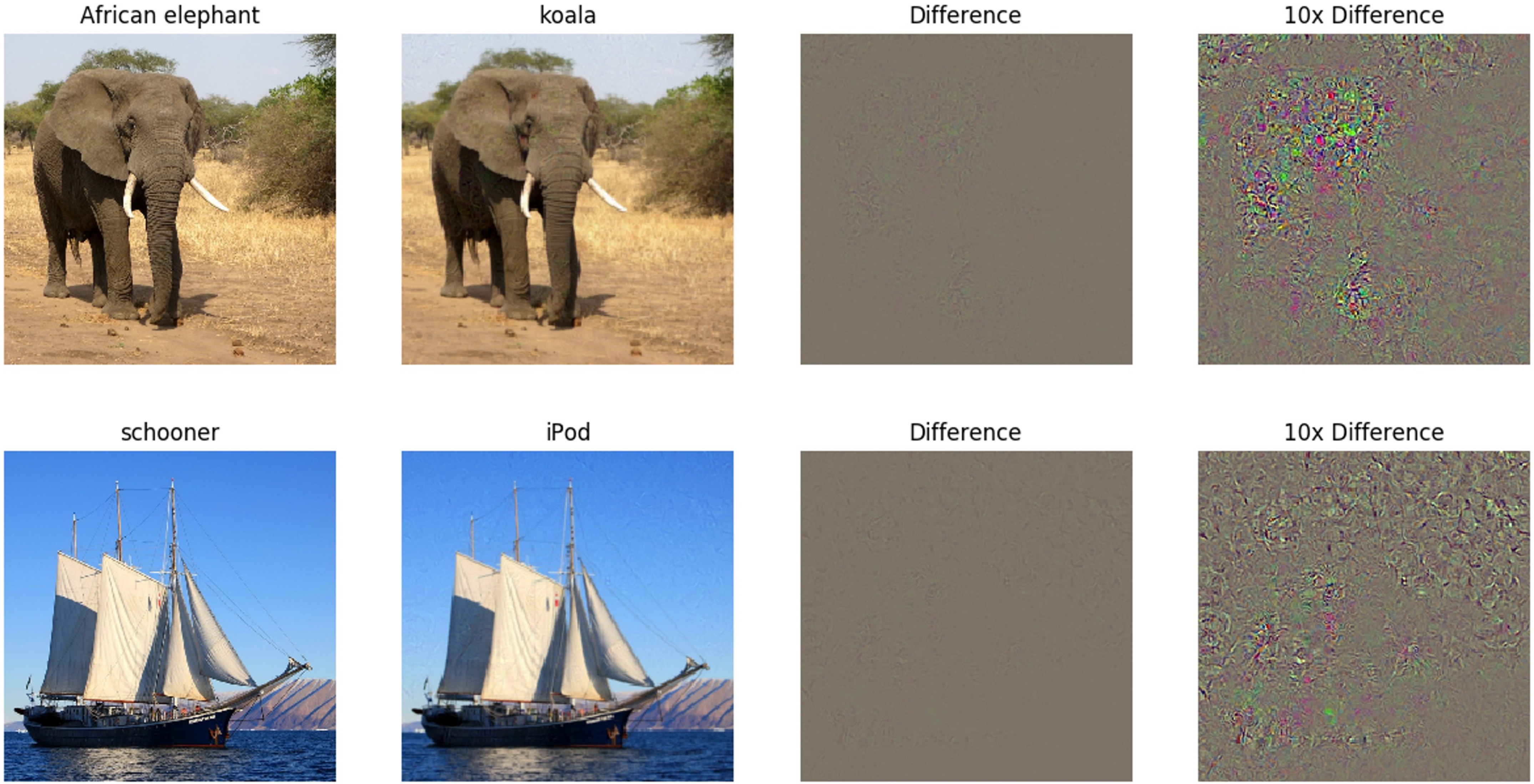
Fooling images

- 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?

Adversarial Examples

1. Start from an arbitrary image
2. Pick an arbitrary category
3. Modify the image (via gradient ascent) to maximize the class score
4. Stop when the network is fooled

Adversarial Examples



Adversarial Attacks and Defense

Adversarial Attack: Method for generating adversarial examples for a network

Adversarial Defense: Change to network architecture, training, etc that make it harder to attack

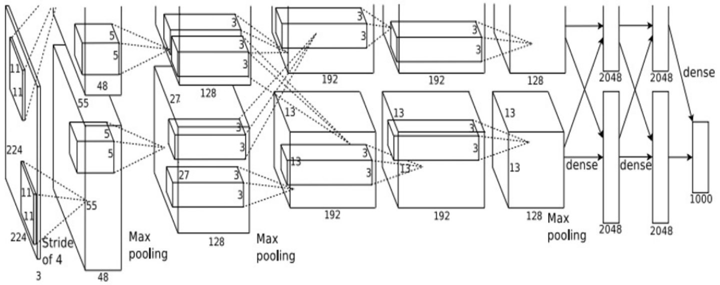
Adversarial Attacks and Defense

Adversarial Attack: Method for generating adversarial examples for a network — Easy

Adversarial Defense: Change to network architecture, training, etc that make it harder to attack — Hard

Adversarial Attacks

White-box attack: We have access to the network architecture and weights. Can get outputs, gradients for arbitrary input images.

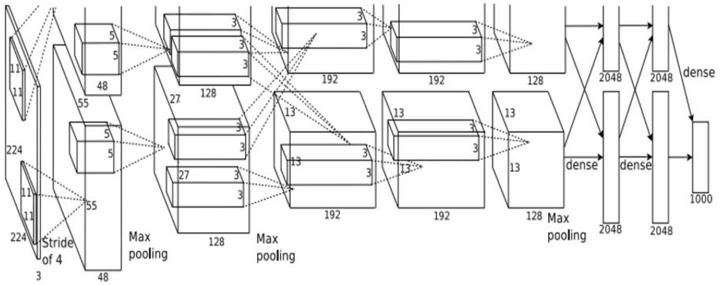


P(elephant) = 0.9
P(cat) = 0.05
...

Papernot et al, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples", 2016

Adversarial Attacks

White-box attack: We have access to the network architecture and weights. Can get outputs, gradients for arbitrary input images.



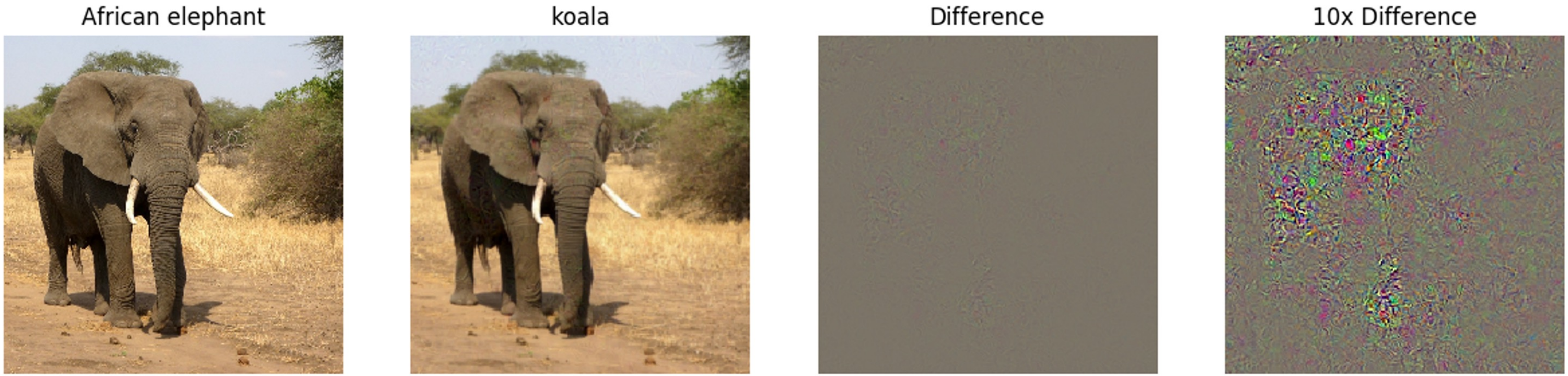
P(elephant) = 0.9
P(cat) = 0.05
…

Black-box attack: We don't know network architecture or weights; can only get network predictions for arbitrary input images

P(elephant) = 0.9
P(cat) = 0.05
…

Papernot et al, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples", 2016

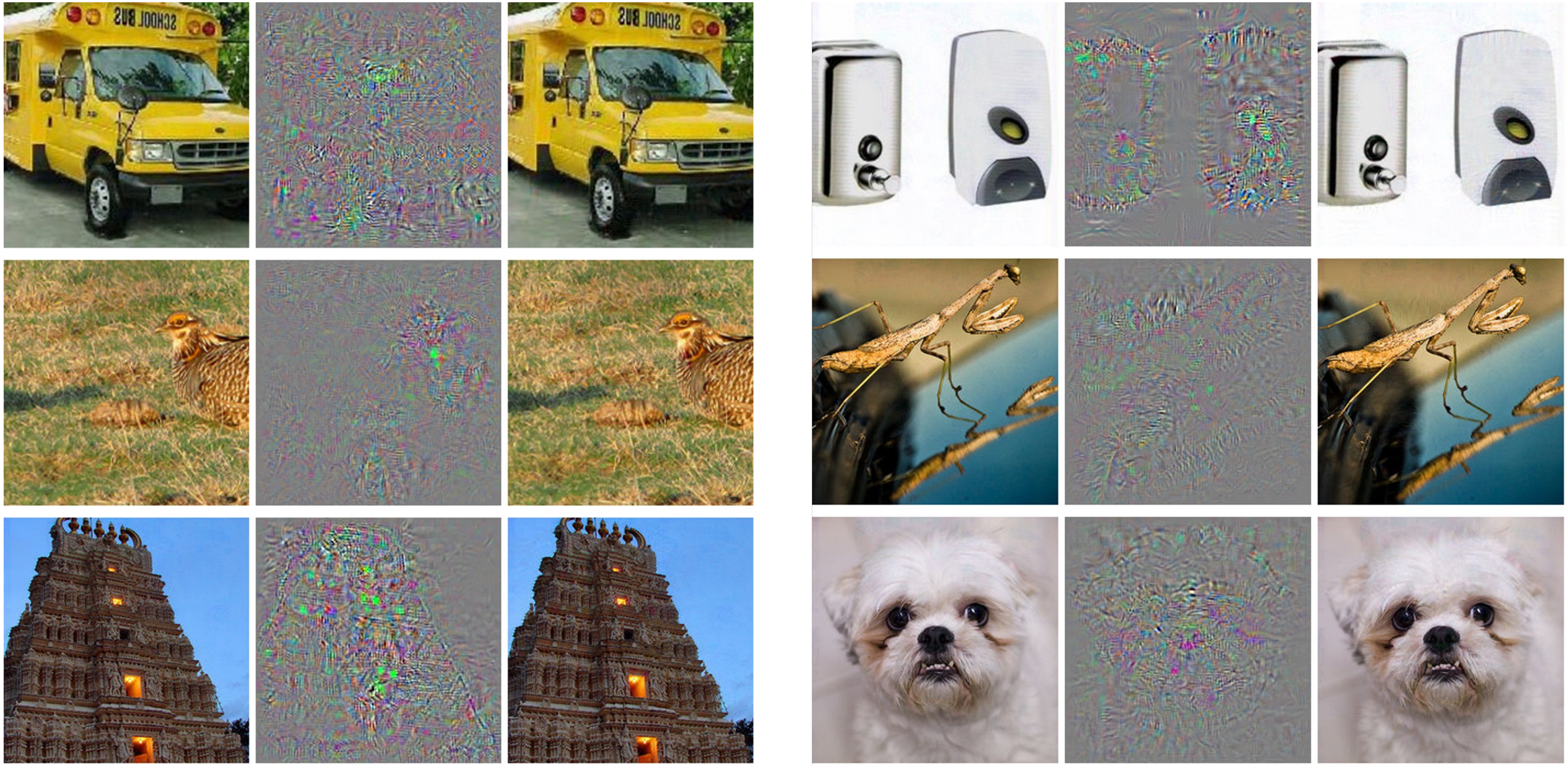
Adversarial Examples



Huge area of research!

Security concern for networks deployed in the wild

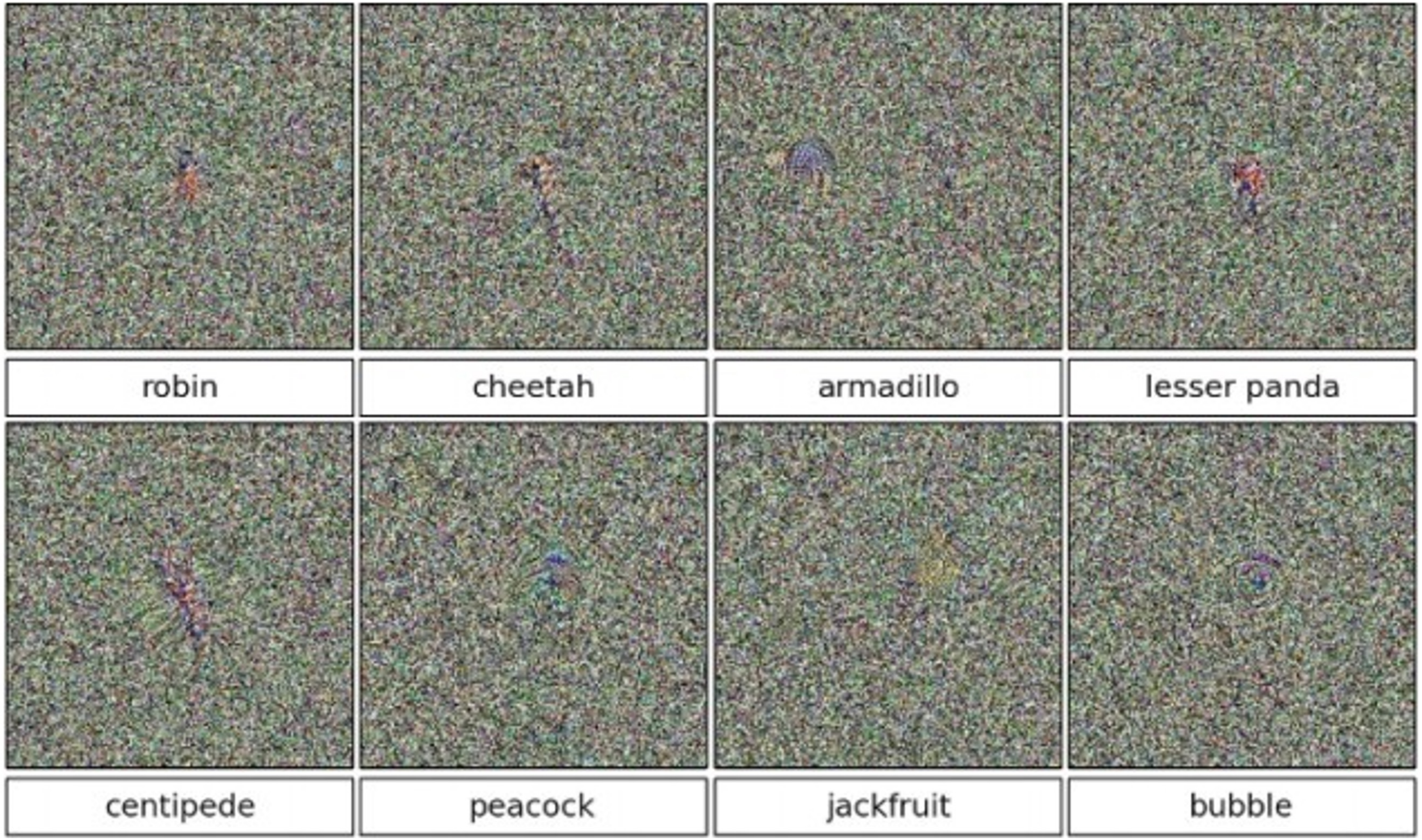
Intriguing properties of neural networks
[Szegedy et al., 2013]



correct +distort ostrich correct +distort ostrich

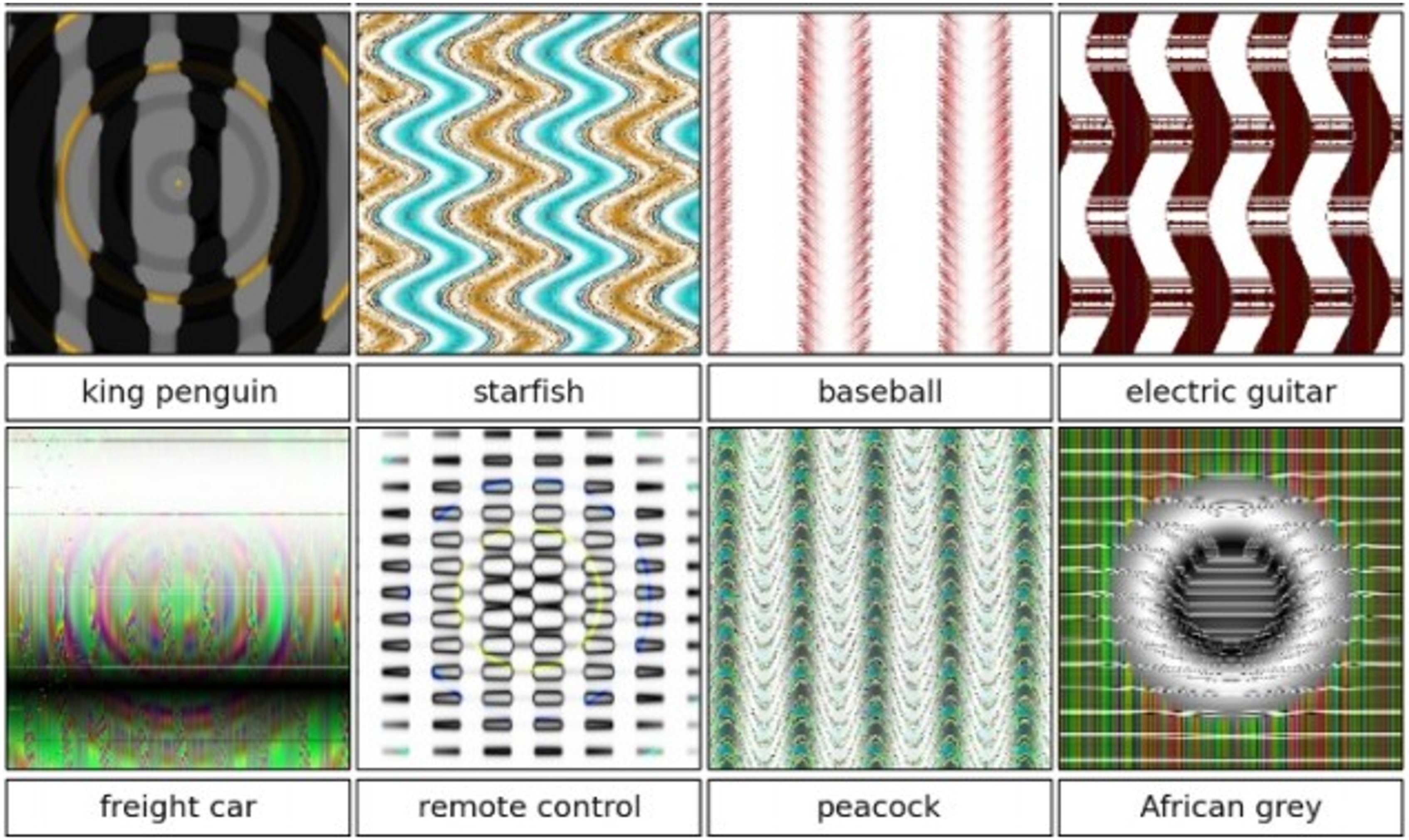
Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images
[Nguyen, Yosinski, Clune, 2014]

>99.6% confidences



Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images
[Nguyen, Yosinski, Clune, 2014]

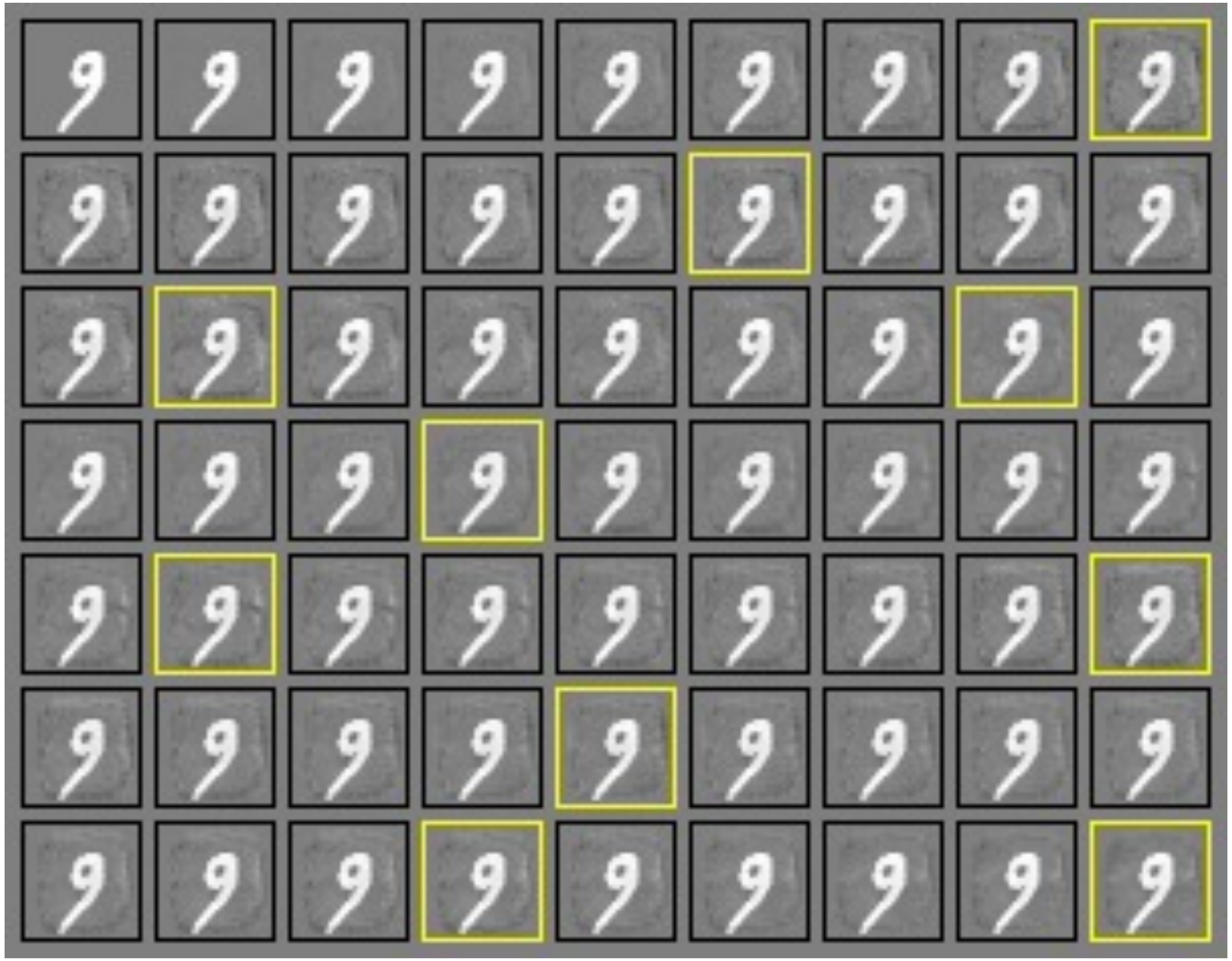
>99.6% confidences



Not just for neural nets

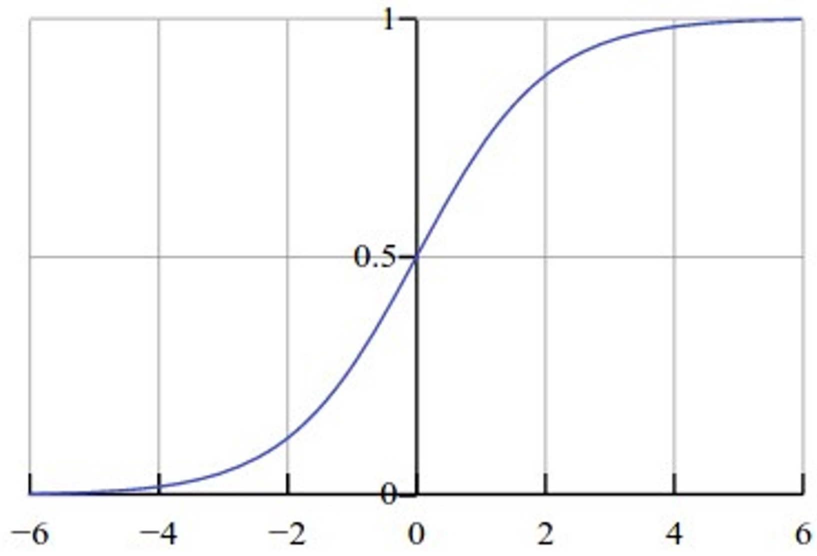
- Linear models
  - Logistic regression
  - Softmax regression
  - SVMs
- Decision trees
- Nearest neighbors

Attacking a Linear Model



- Softmax regression
- Turning “9” into other digits
- Yellow boxes denote misclassifications

Lets fool a binary linear classifier: (logistic regression)



$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x+b)}} = \sigma(w^T x + b)$$

Since the probabilities of class 1 and 0 sum to one, the probability for class 0 is $P(y=0 \mid x; w, b) = 1 - P(y=1 \mid x; w, b)$. Hence, an example is classified as a positive example (y = 1) if $\sigma(w^T x + b) > 0.5$, or equivalently if the score $w^T x + b > 0$.

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1	← input example
w	-1	-1	1	-1	1	-1	1	1	-1	1	← weights

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1	← input example
w	-1	-1	1	-1	1	-1	1	1	-1	1	← weights

class 1 score = dot product:
= -2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3
=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474
i.e. the classifier is **95%** certain that this is class 0 example.

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1	← input example
w	-1	-1	1	-1	1	-1	1	1	-1	1	← weights
adversarial **x**	?	?	?	?	?	?	?	?	?	?	

class 1 score = dot product:
= -2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3
=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474
i.e. the classifier is **95%** certain that this is class 0 example.

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1
adversarial **x**	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5

class 1 score before:

-2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

-1.5+1.5+3.5+2.5+2.5-1.5+1.5-3.5-4.5+1.5 = 2

=> probability of class 1 is now 1/(1+e^(-(2))) = 0.88

i.e. we improved the class 1 probability from 5% to 88%

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1
adversarial **x**	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5

class 1 score before:
-2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3
=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

-1.5+1.5+3.5+2.5+2.5-1.5+1.5-3.5-4.5+1.5 = 2
=> probability of class 1 is now 1/(1+e^(-(2))) = 0.88
i.e. we improved the class 1 probability from 5% to 88%

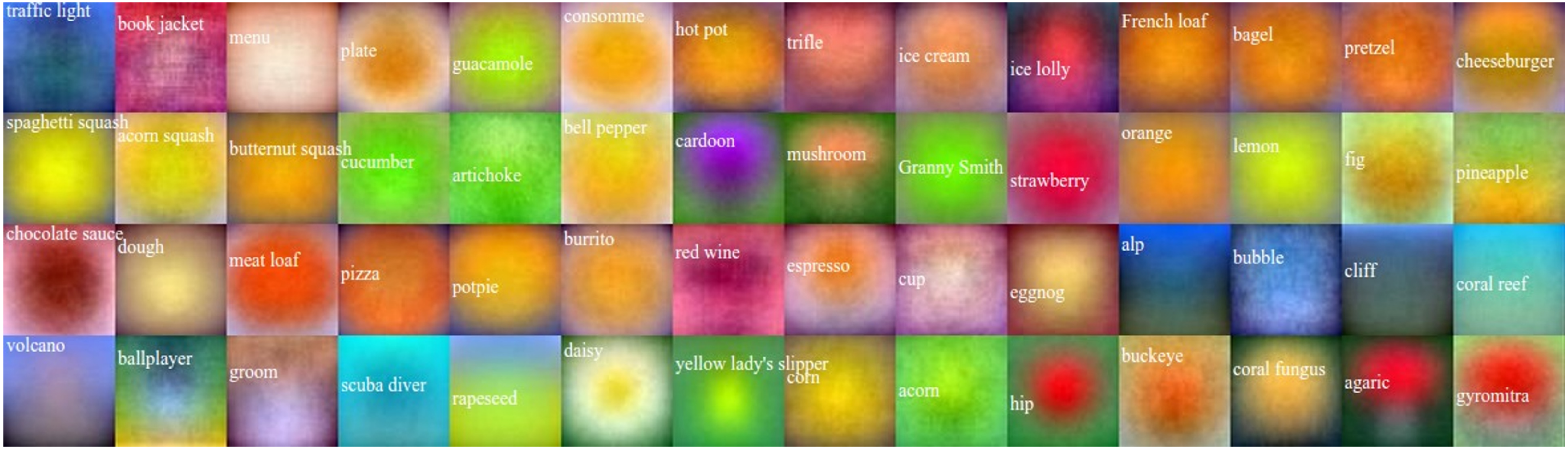
This was only with 10 input dimensions. A 224x224 input image has 150,528.

(It's significantly easier with more numbers, need smaller nudge for each)

Blog post: Breaking Linear Classifiers on ImageNet

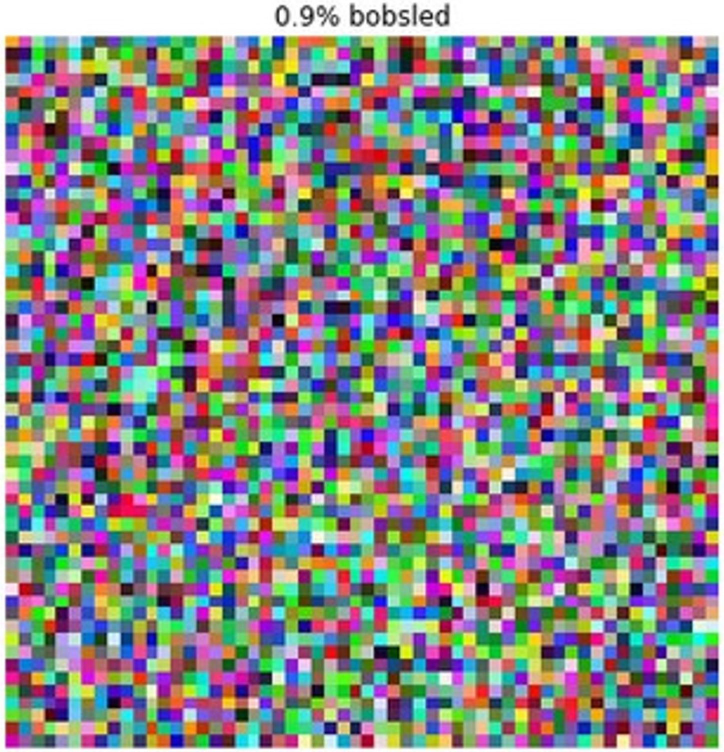
Recall CIFAR-10 linear classifiers:

ImageNet classifiers:

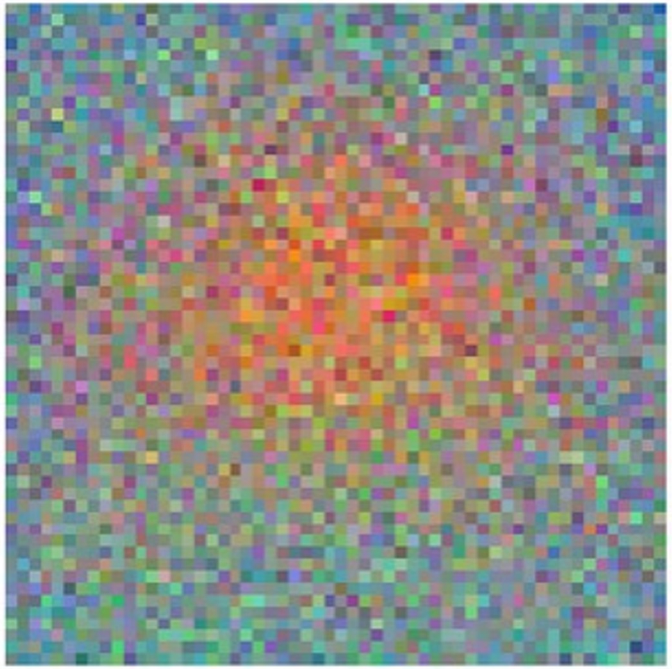


http://karpathy.github.io/2015/03/30/breaking-convnets/

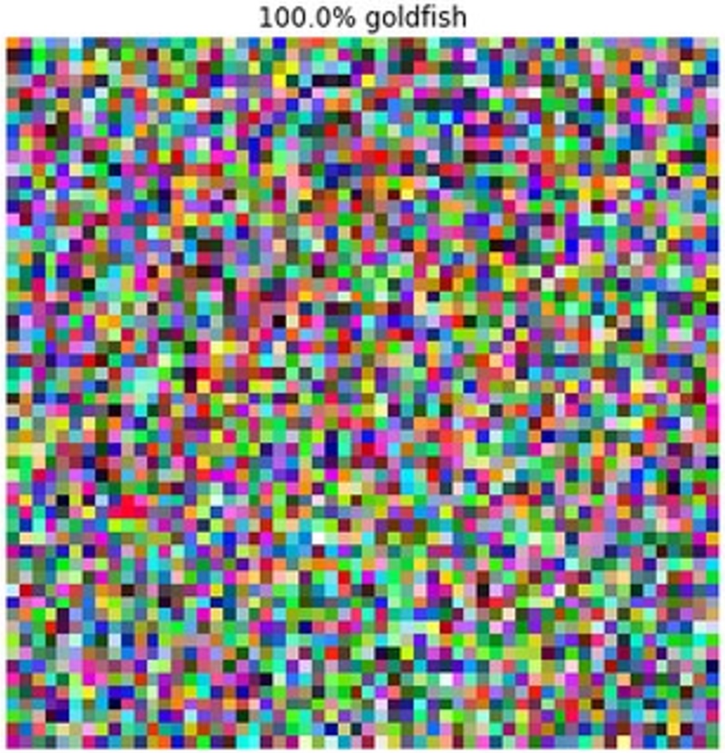
mix in a tiny bit of
Goldfish classifier weights



+

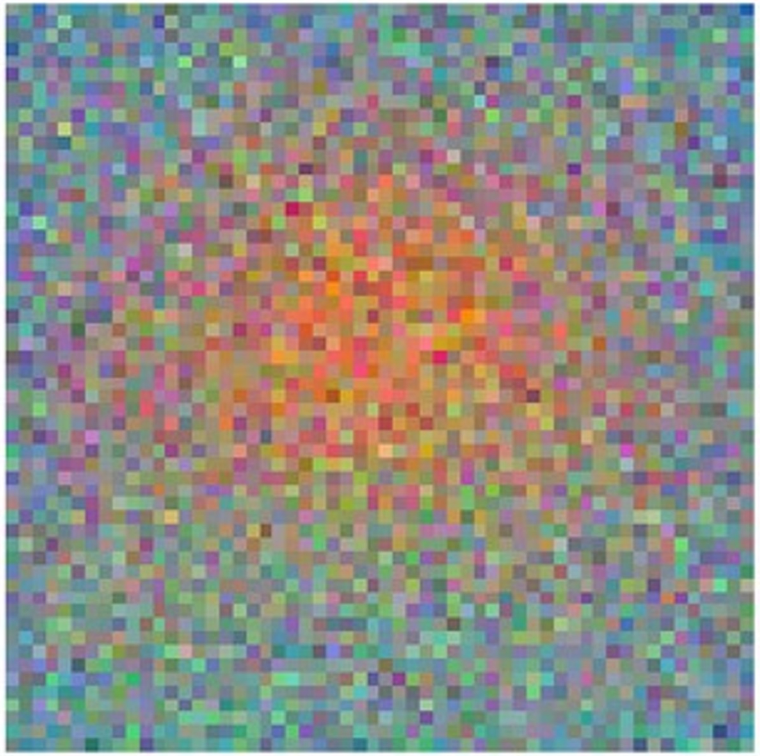


=

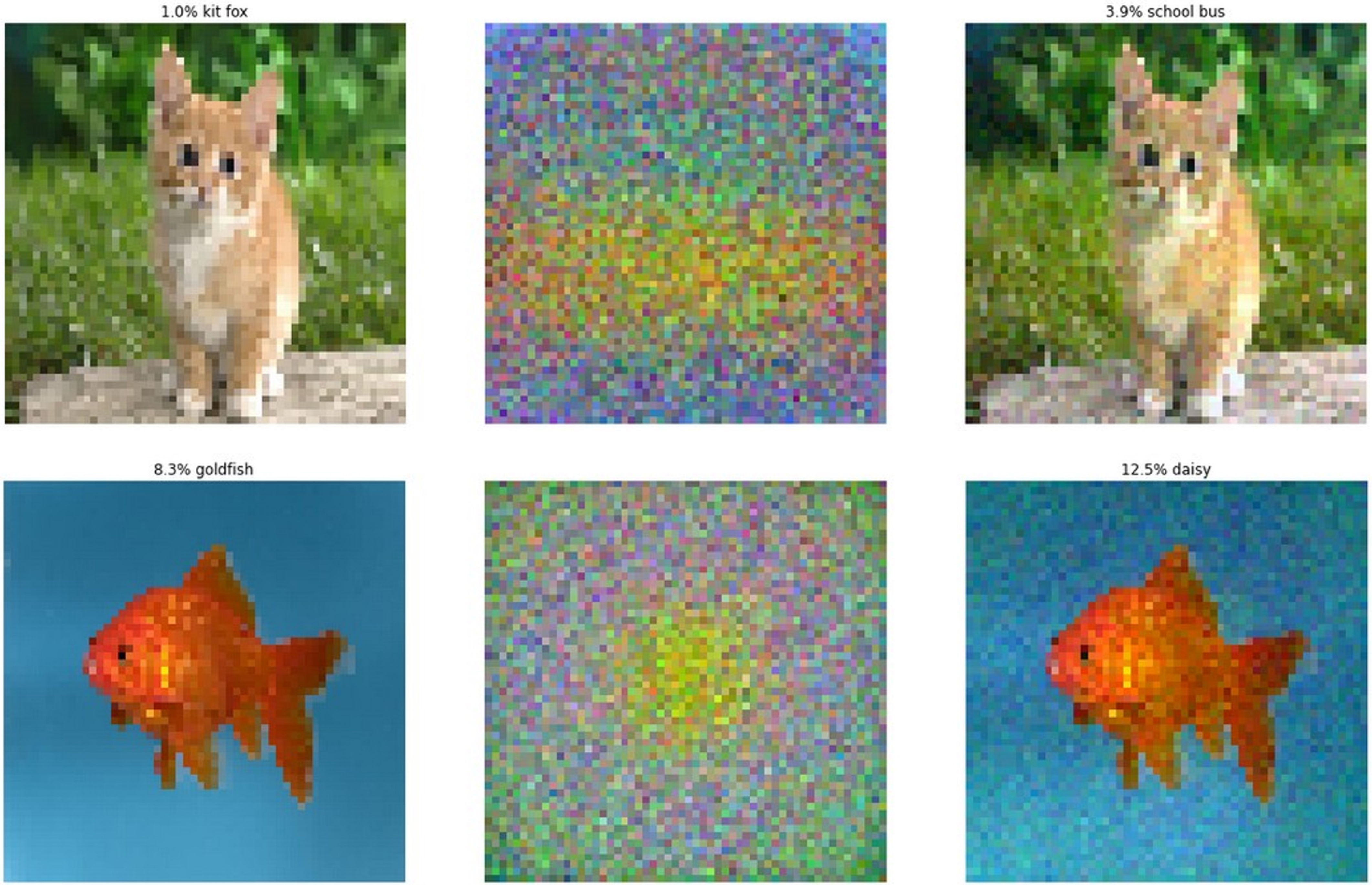


100% Goldfish

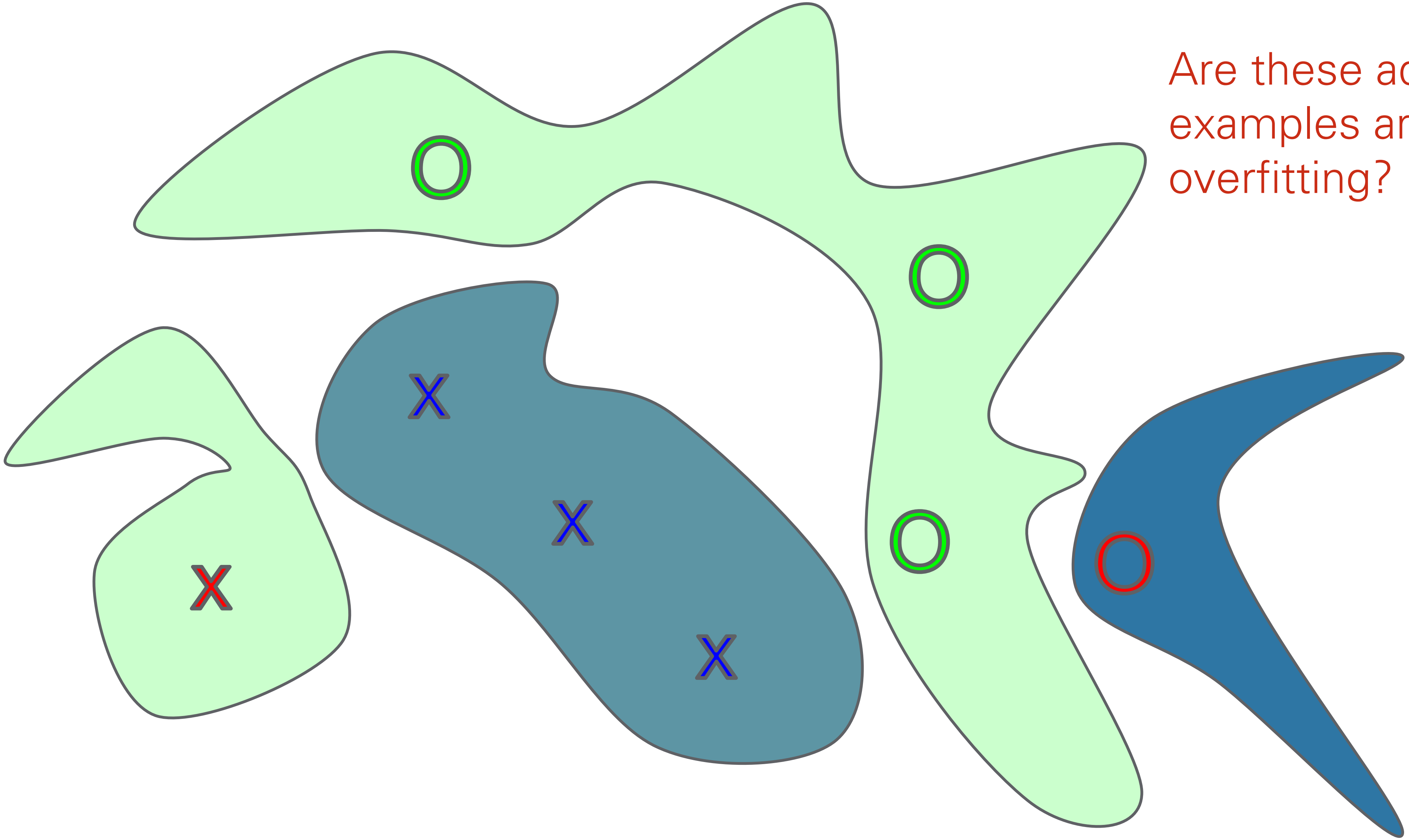
1.0% kit fox



8.0% goldfish

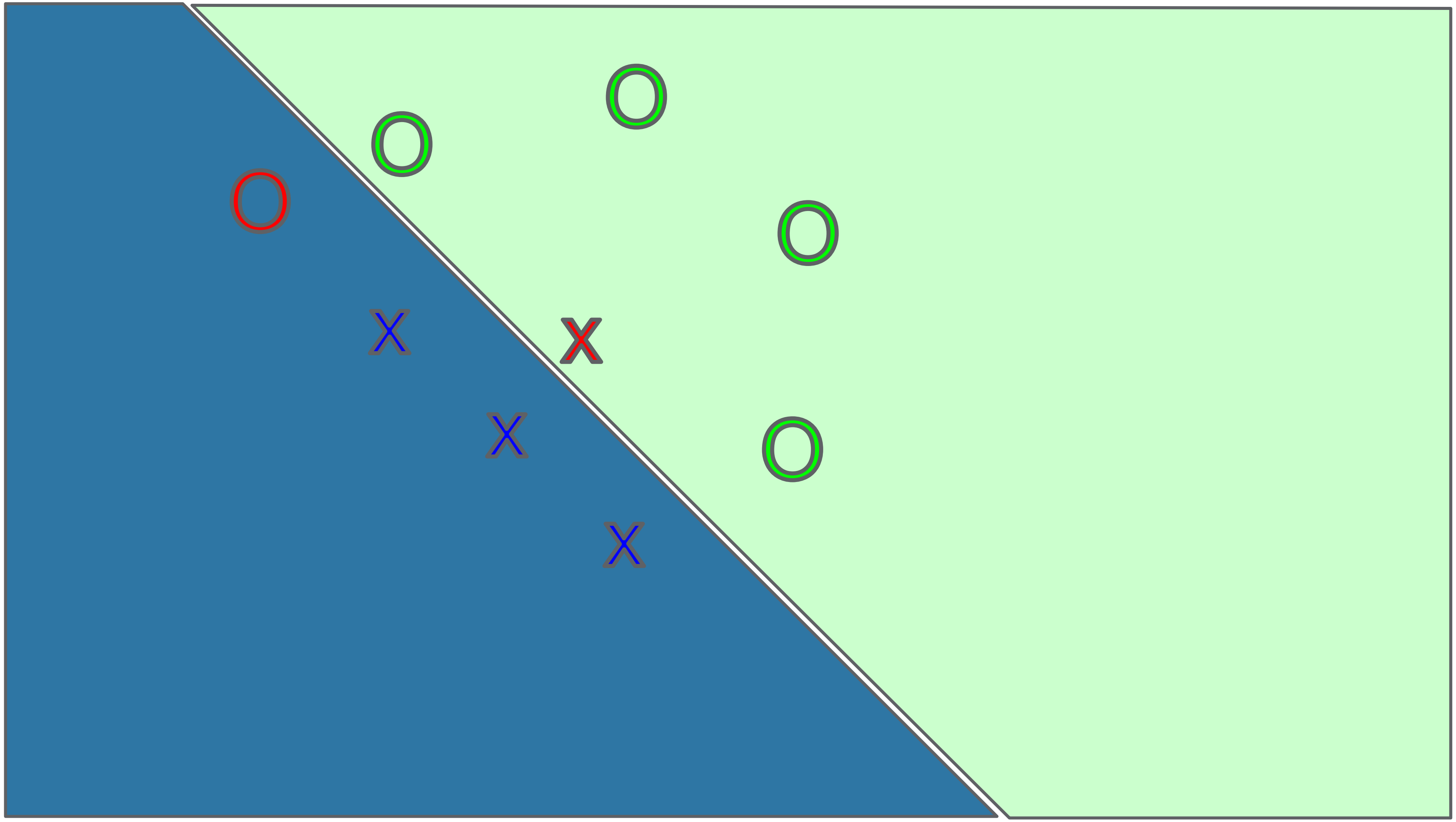

1.0% kit fox
3.9% school bus
8.3% goldfish
12.5% daisy

Adversarial Examples from Overfitting



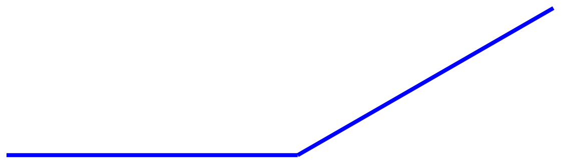
Are these adversarial examples are related to overfitting?

Adversarial Examples from Excessive Linearity

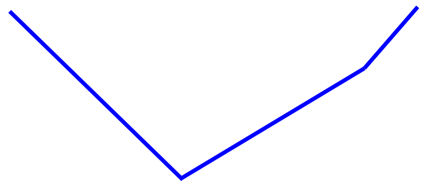


Modern deep nets are very piecewise linear

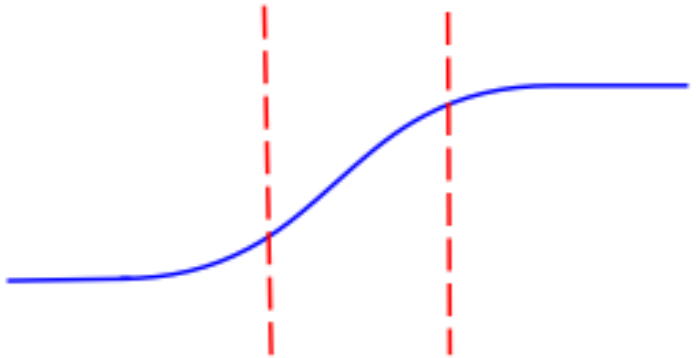
Rectified linear unit



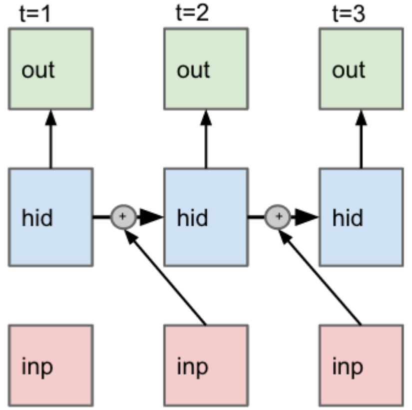
Maxout



Carefully tuned sigmoid



LSTM



The Fast Gradient Sign Method

$$J(\tilde{\boldsymbol{x}}, \boldsymbol{\theta}) \approx J(\boldsymbol{x}, \boldsymbol{\theta}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x}).$$

Maximize

$$J(\boldsymbol{x}, \boldsymbol{\theta}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x})$$

subject to

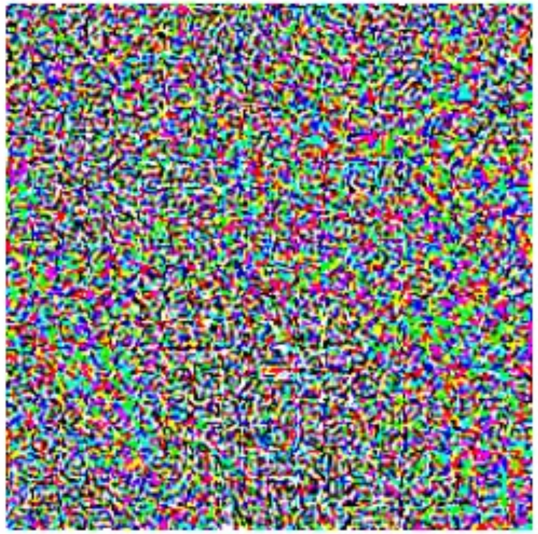
$$||\tilde{\boldsymbol{x}} - \boldsymbol{x}||_\infty \leq \epsilon$$

$$\Rightarrow \tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \mathrm{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{x})\right).$$

Adversarial examples in the physical world - Kurakin, et al - 2016
Explaining and Harnessing Adversarial Examples - Goodfellow, et al - 2014

Adversarial Examples

$+ .007 \times$



$=$

"panda"
57.7% confidence

"nematode"
8.2% confidence

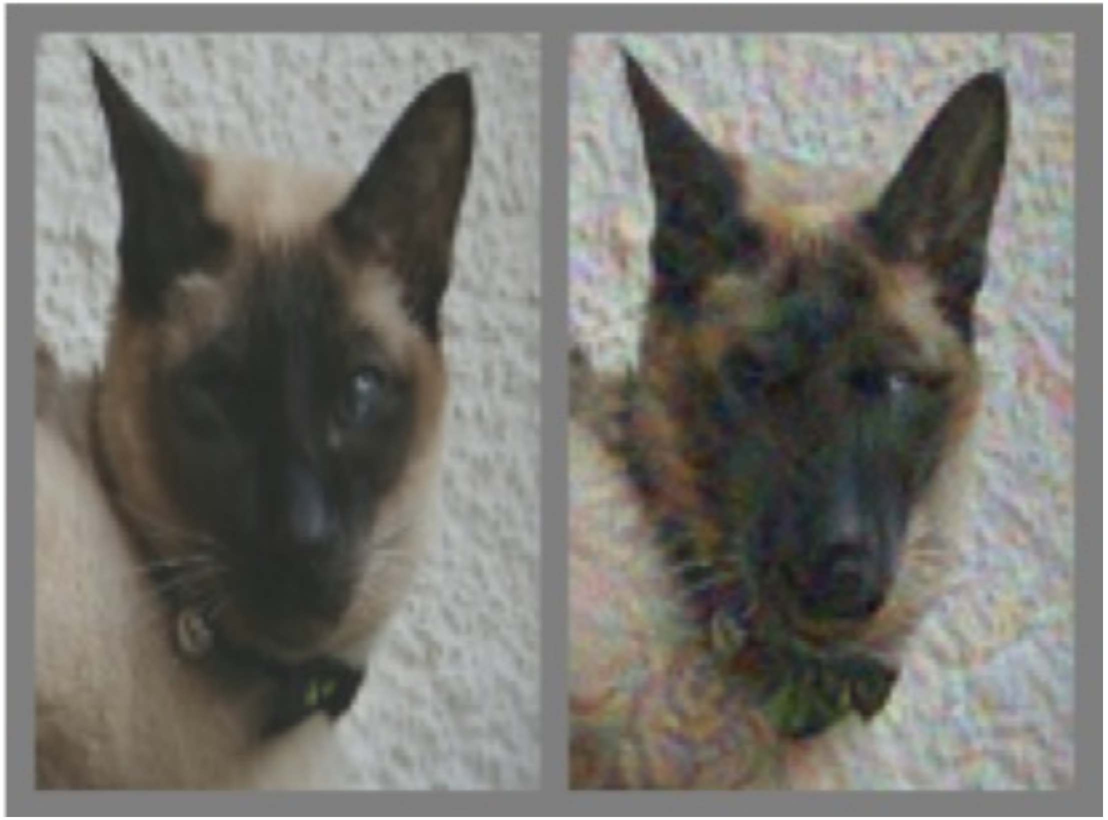
"gibbon"
99.3 % confidence

$$X^{adv} = X + \epsilon\, \mathrm{sign}\big(\nabla_X J(X, y_{true})\big)$$

Score of label y_{true}, given input image X

Adversarial examples in the physical world - Kurakin, et al - 2016
Explaining and Harnessing Adversarial Examples - Goodfellow, et al - 2014

Adversarial Examples that Fool both Human and Computer Vision



Left: An image of a cat

Right: The same image after it has been adversarially perturbed to look like a dog

(Elsayed et al., 2018)

Practical Attacks

- Fool real classifiers trained by remotely hosted API (MetaMind, Amazon, Google)

- Fool malware detector networks

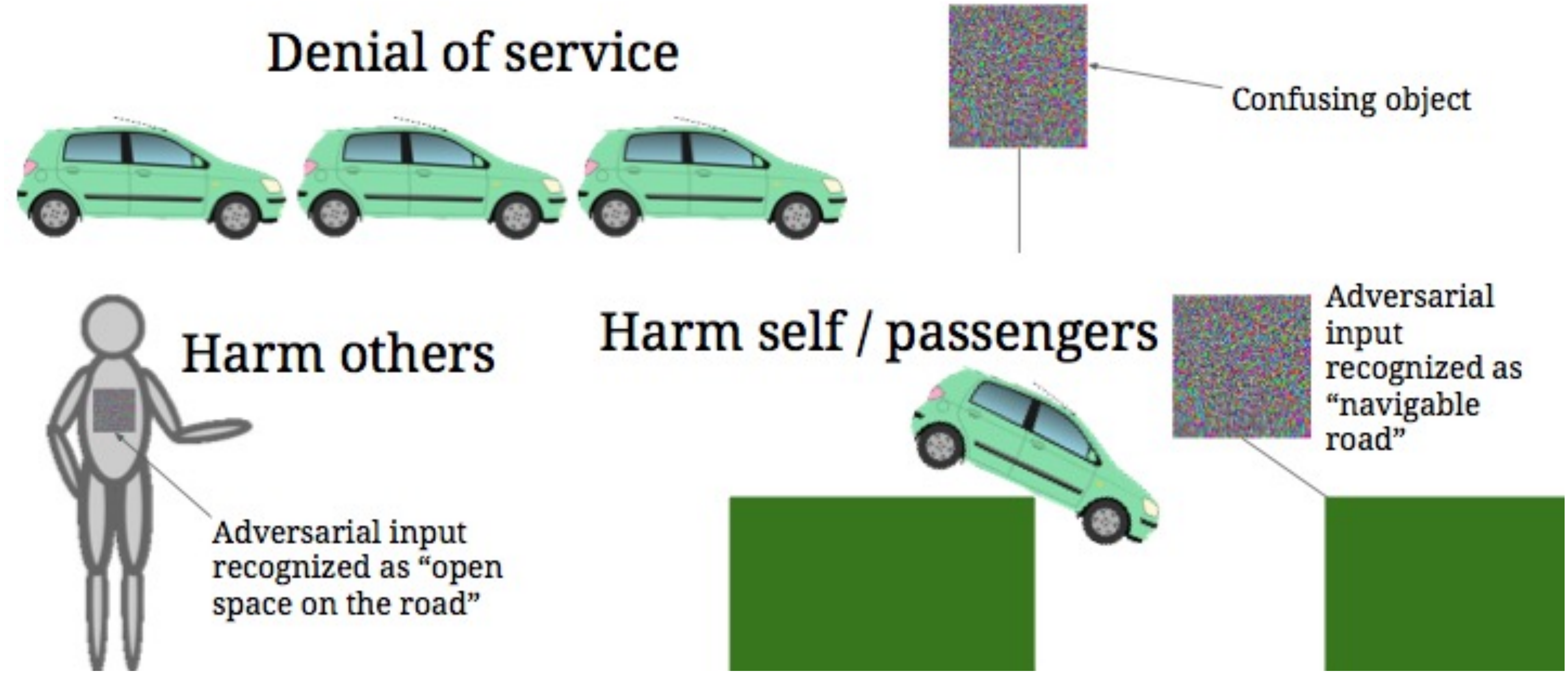
- Display adversarial examples in the physical world and fool machine learning systems that perceive them through a camera

Adversarial Examples in the Physical World

(a) Printout (b) Photo of printout (c) Cropped image

Adversarial examples in the physical world - Kurakin, et al - 2016

Hypothetical Attacks on Autonomous Vehicles



Physical Adversarial Examples

- Physical adversarial examples against the YOLO detector
- Adversarial examples take the form of sticker perturbations that are apply to a real STOP sign

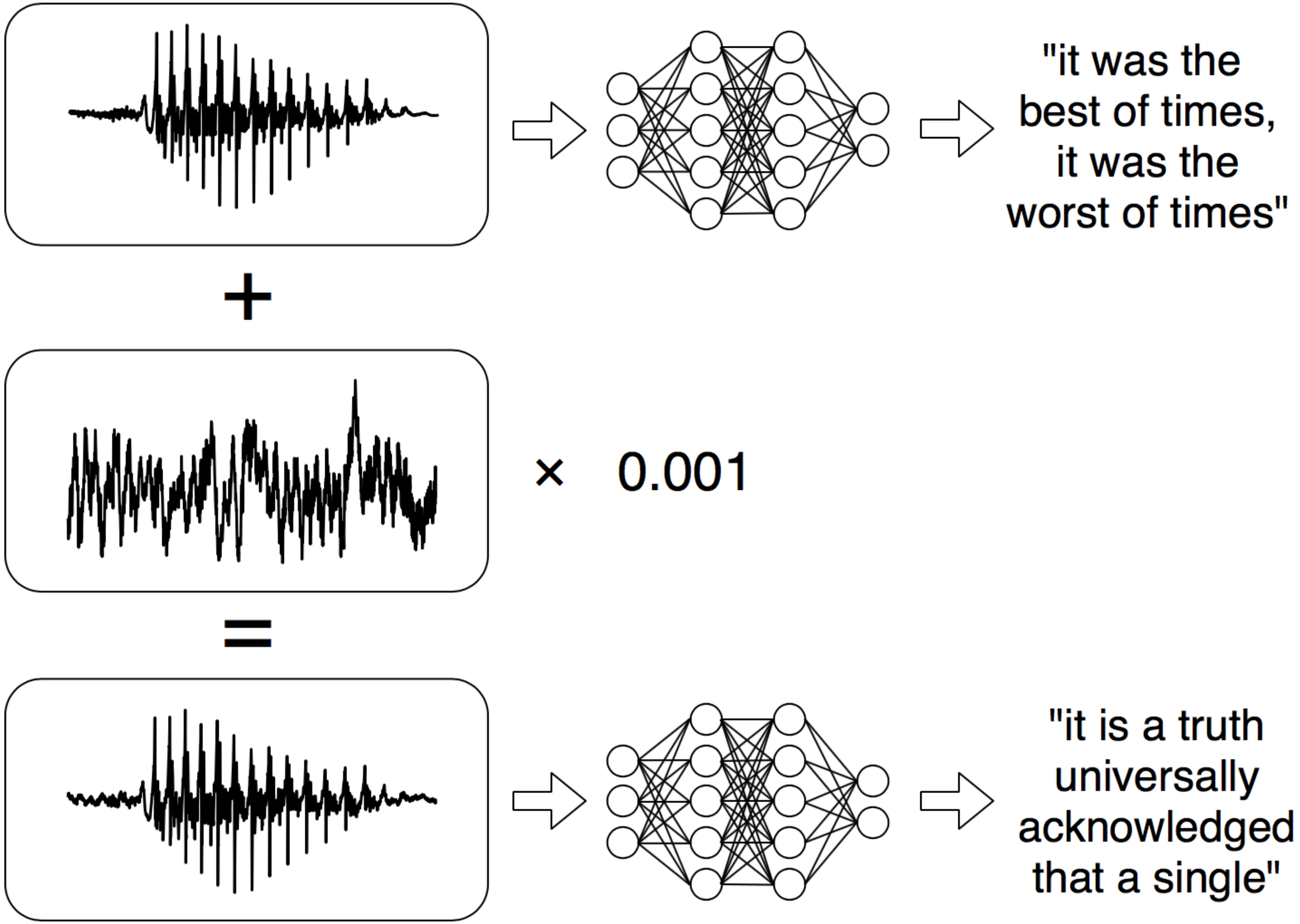
http://bair.berkeley.edu/blog/2017/12/30/yolo-attack/

Audio Adversarial Examples

- targeted audio adversarial examples on speech-to-text transcription neural networks

“without the dataset the article is useless”

“okay google browse to evil dot com”



https://nicholas.carlini.com/code/audio_adversarial_examples/

Figure credit: N. Carlini and D. Wagner

Failed defenses

Generative
pretraining

Removing perturbation
with an autoencoder

Adding noise
at test time

Ensembles

Confidence-reducing
perturbation at test time

Error correcting
codes

Multiple glimpses

Weight decay

Double backprop

Adding noise
at train time

Various
non-linear units

Dropout

Adversarial Training

Labeled as bird

Still has same label (bird)

Decrease
probability
of bird class

Adversarial examples in the physical world - Kurakin, et al - 2016

Virtual Adversarial Training

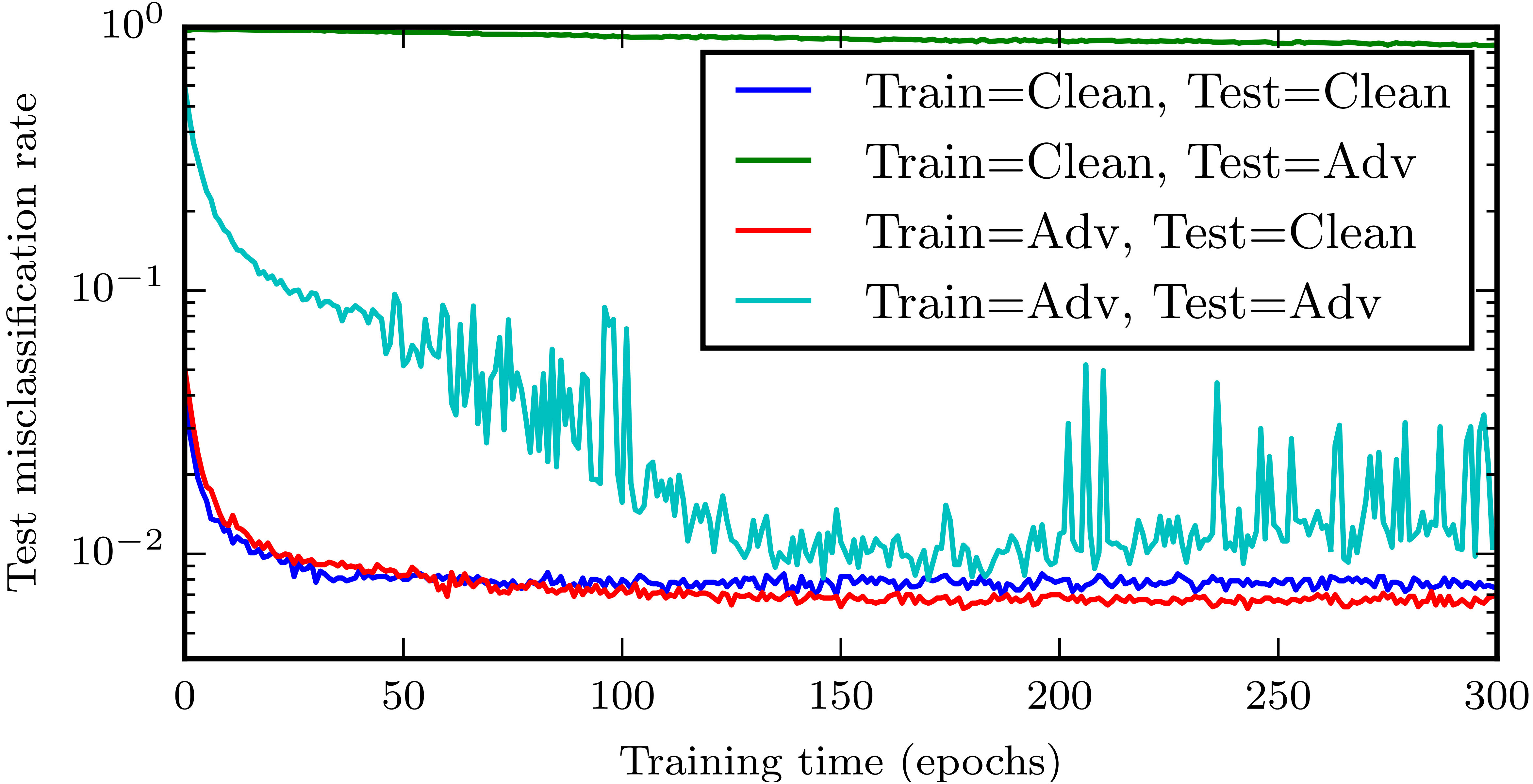
Unlabeled; model guesses it’s probably a bird, maybe a plane

New guess should match old guess (probably bird, maybe plane)

Adversarial perturbation intended to change the guess

Adversarial examples in the physical world - Kurakin, et al - 2016

Training on Adversarial Examples



Adversarial Training of other Models

- Linear models: SVM / linear regression cannot learn a step function, so adversarial training is less useful, very similar to weight decay

- k-NN: adversarial training is prone to overfitting.

- Takeway: neural nets can actually become more secure than other models. Adversarially trained neural nets have the best empirical success rate on adversarial examples of any machine learning model.

Next lecture:
Recurrent Neural Networks