Image from ucreative.com

CMP784

DEEP LEARNING

Lecture #06 – Understanding and Visualizing Convolutional Neural Networks

Erkut Erdem // Hacettepe University // Fall 2021

HACETTEPE UNIVERSITY COMPUTER VISION LAB

Previously on CMP784

- convolution layer
- pooling layer
- revolution of depth
- design guidelines
- residual connections
- semantic segmentation networks
- object detection networks

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

Image Representations

$\mathbf{x}$

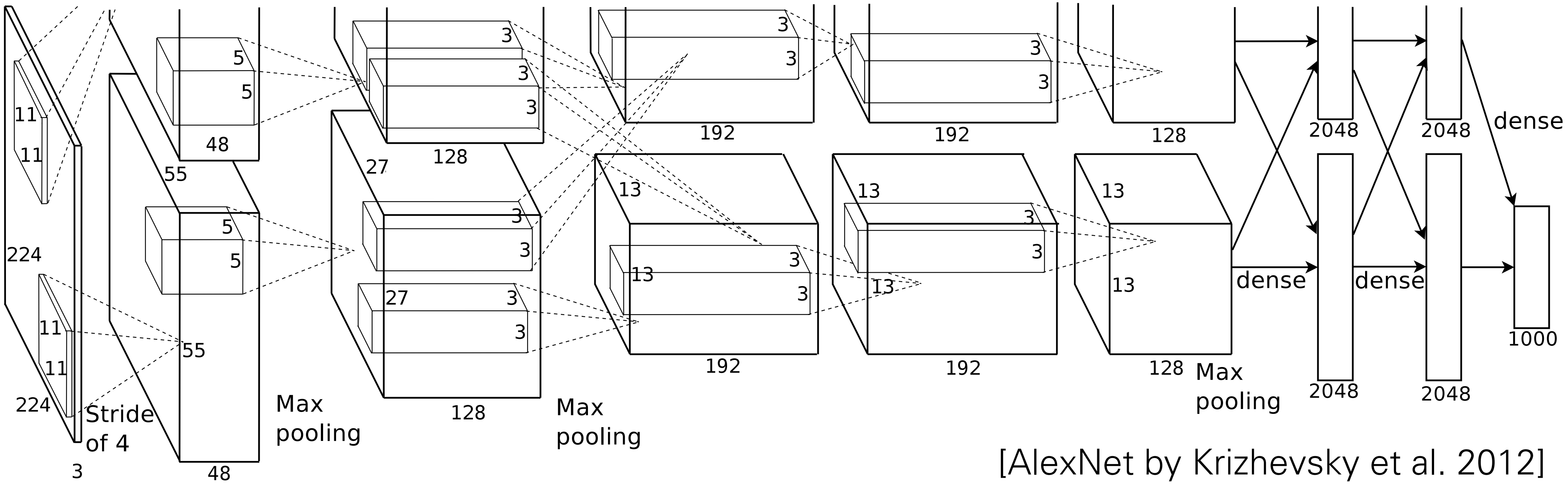
encoder Φ

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]

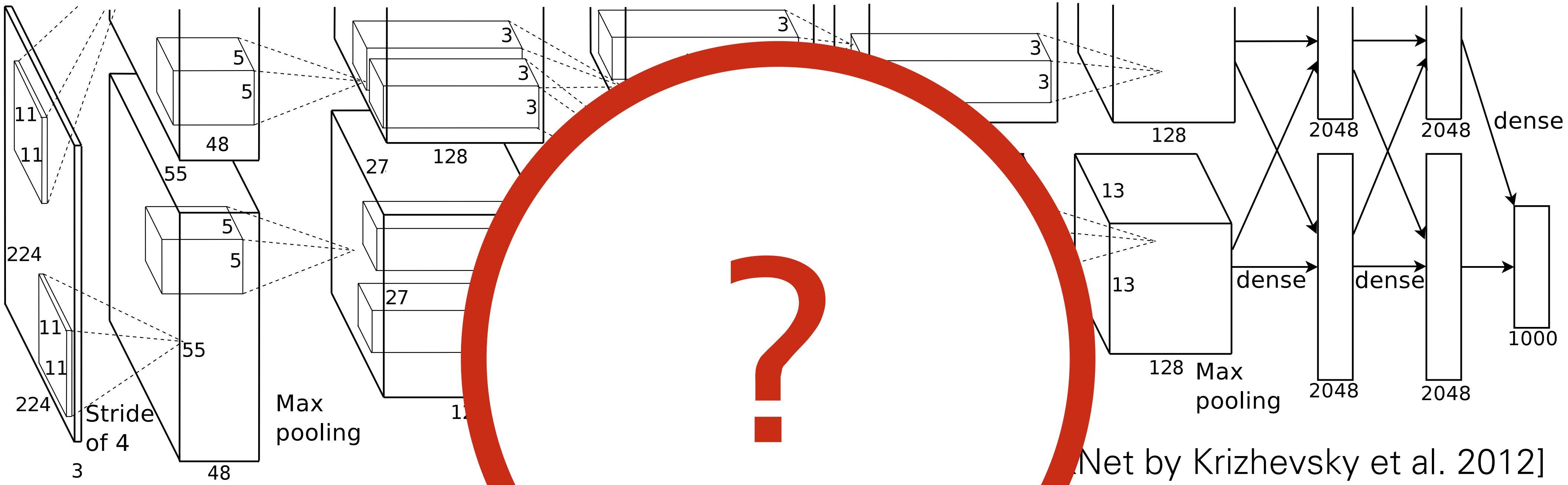
Excellent **performance** in most image understanding tasks

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

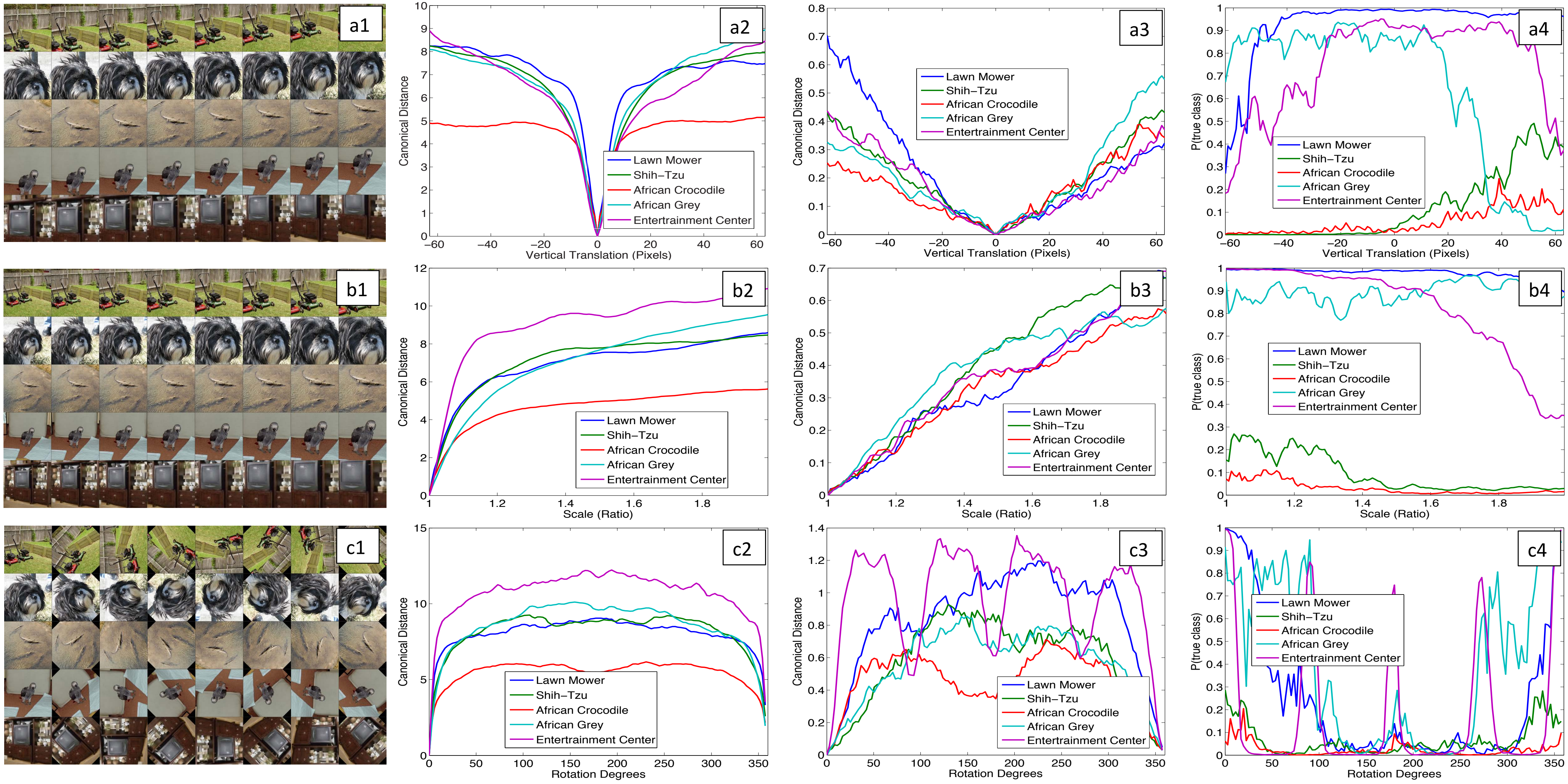
Learn a sequence of **general-purpose representations**

… parameters learned from data

… “**meaning**” of the representation is unclear

Transfer Learning with Deep Networks

Invariance and Covariance

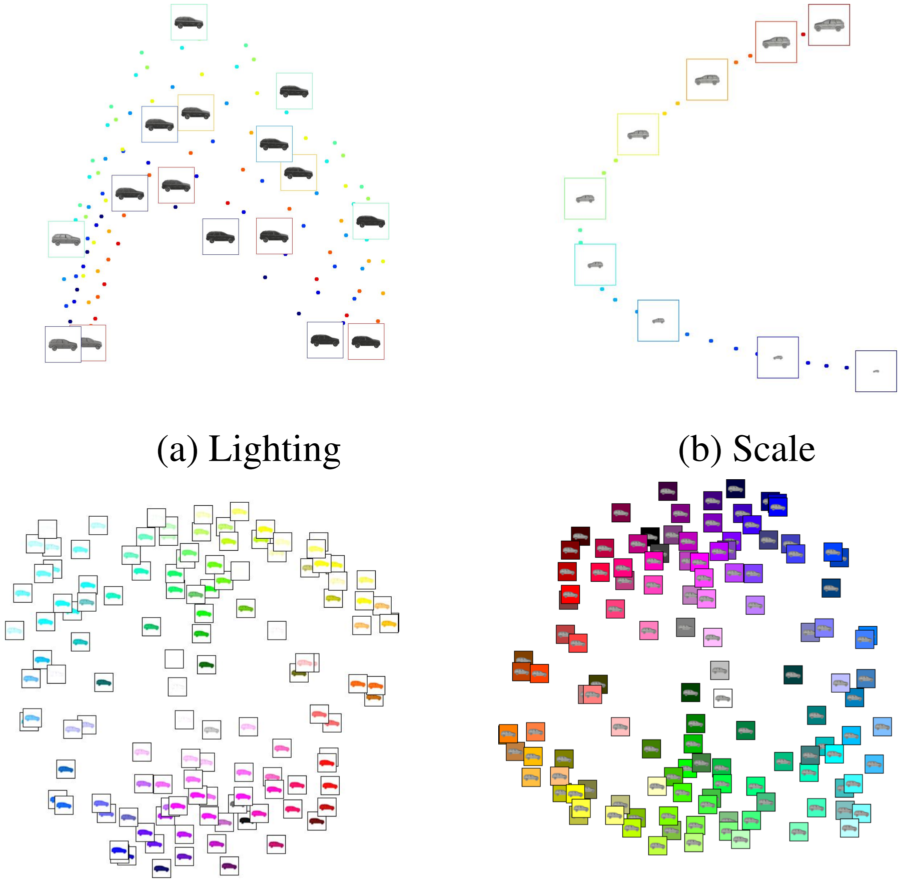


Layer 1 | Layer 7 | Probability Score

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

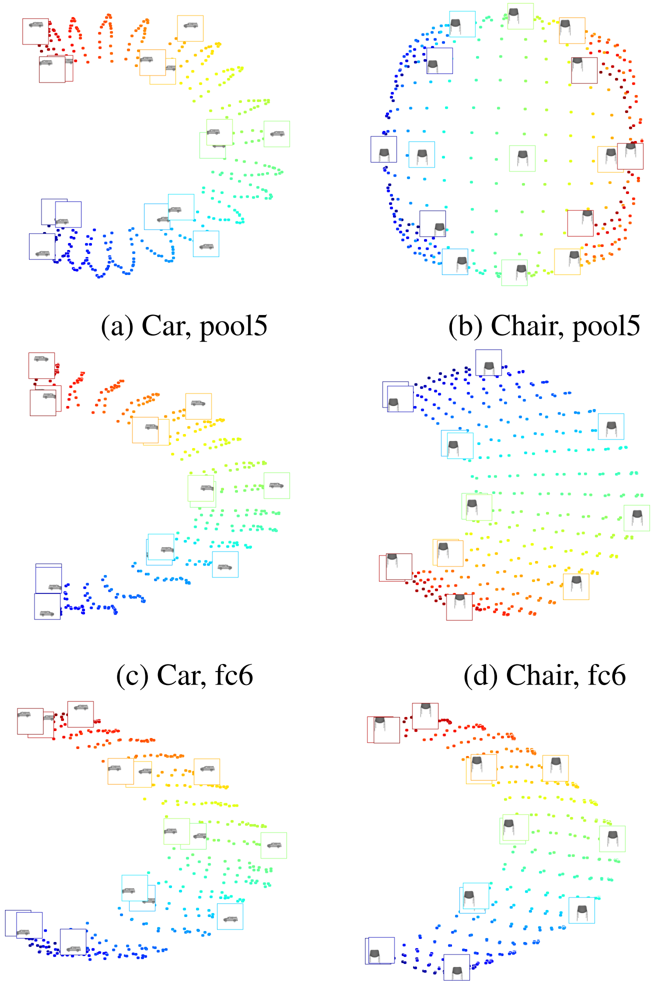
Filter Invariance and Equivariance

- 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

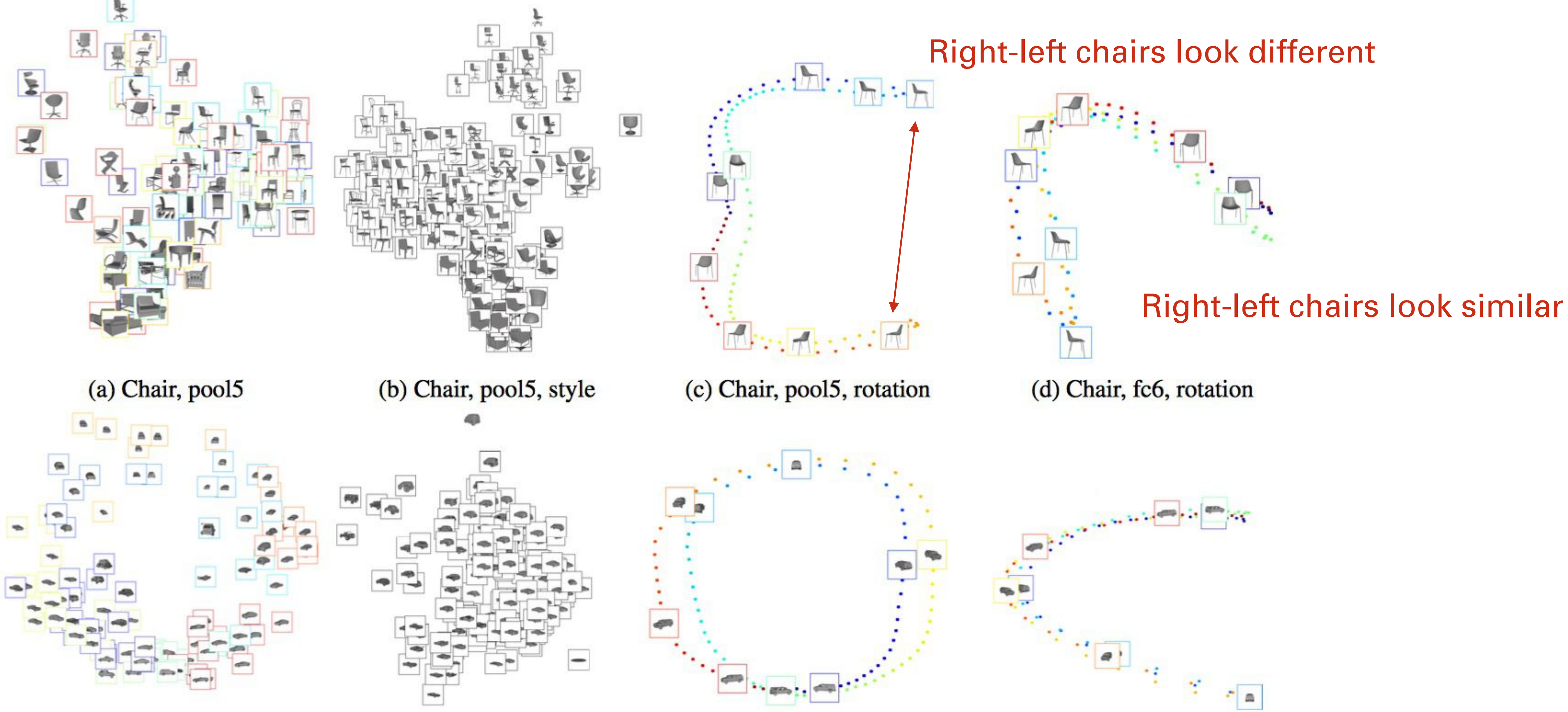
(c) Car, fc6 (d) Chair, fc6

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

		pool5	fc6	fc7
Viewpoint	Places	26.8 %	21.4 %	17.8 %
		8.5	7.0	5.9
	AlexNet	26.4 %	19.4 %	15.6 %
		8.3	7.2	6.0
	VGG	21.2 %	16.4 %	12.3 %
		10.0	7.7	6.2
Style	Places	26.8 %	39.1 %	49.4 %
		136.3	105.5	54.6
	AlexNet	28.2 %	40.3 %	49.4 %
		121.1	125.5	96.7
	VGG	26.4 %	44.3 %	56.2 %
		181.9	136.3	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



(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

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]

Pretrained layers | Fine-tuned layers

representation → **predictor** → **label**

CNN as universal representations

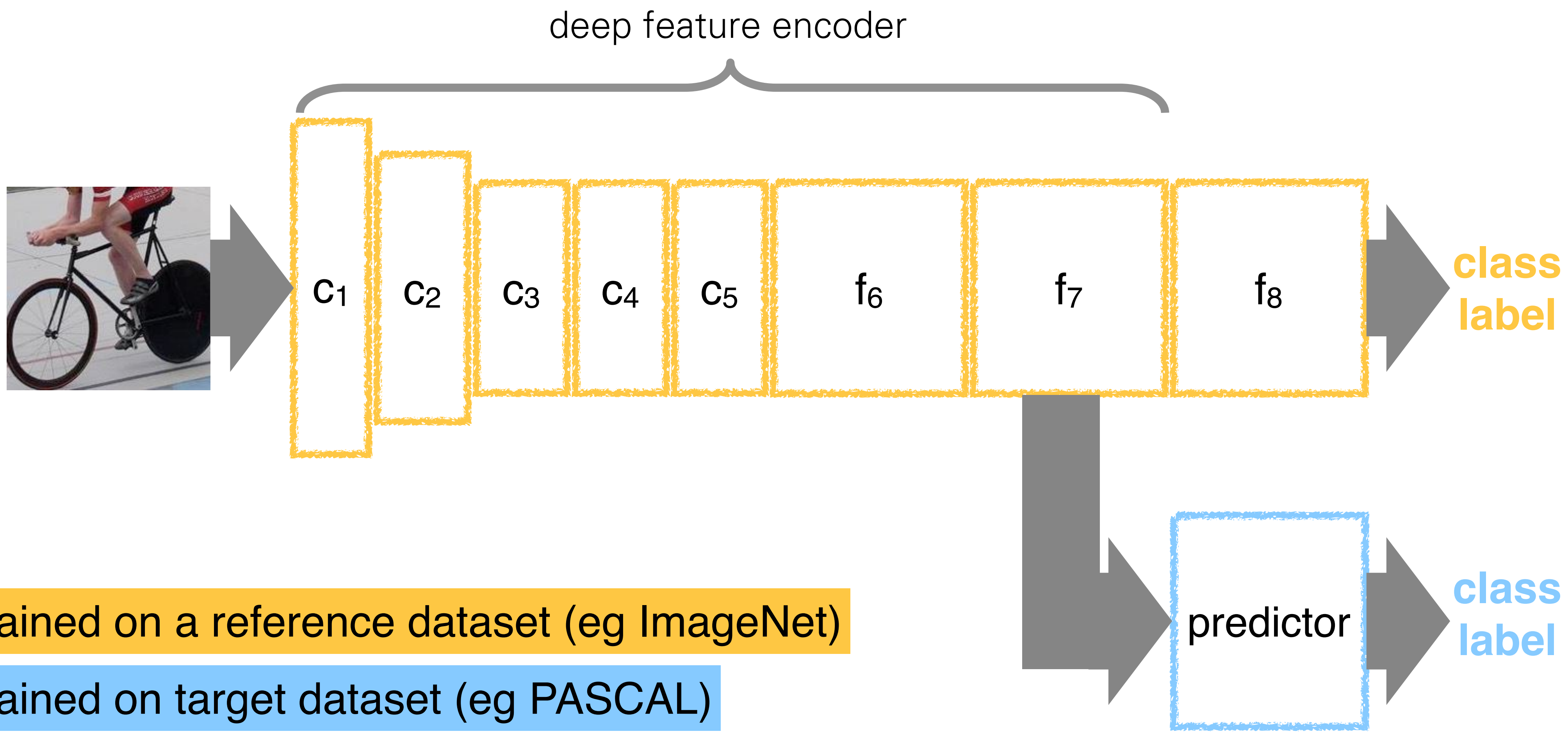
- First several layers in most CNNs are generic
- They can be reused when training data is comparatively scarce.

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



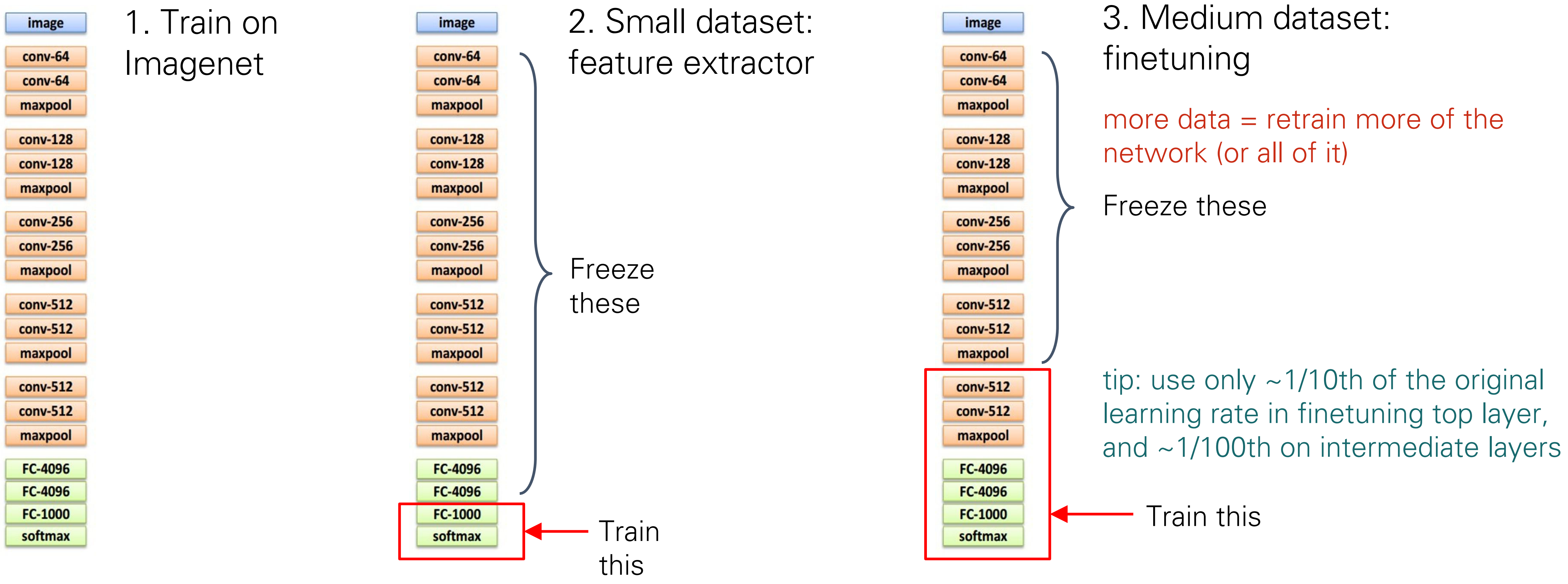
trained on a reference dataset (eg ImageNet)

trained on target dataset (eg PASCAL)

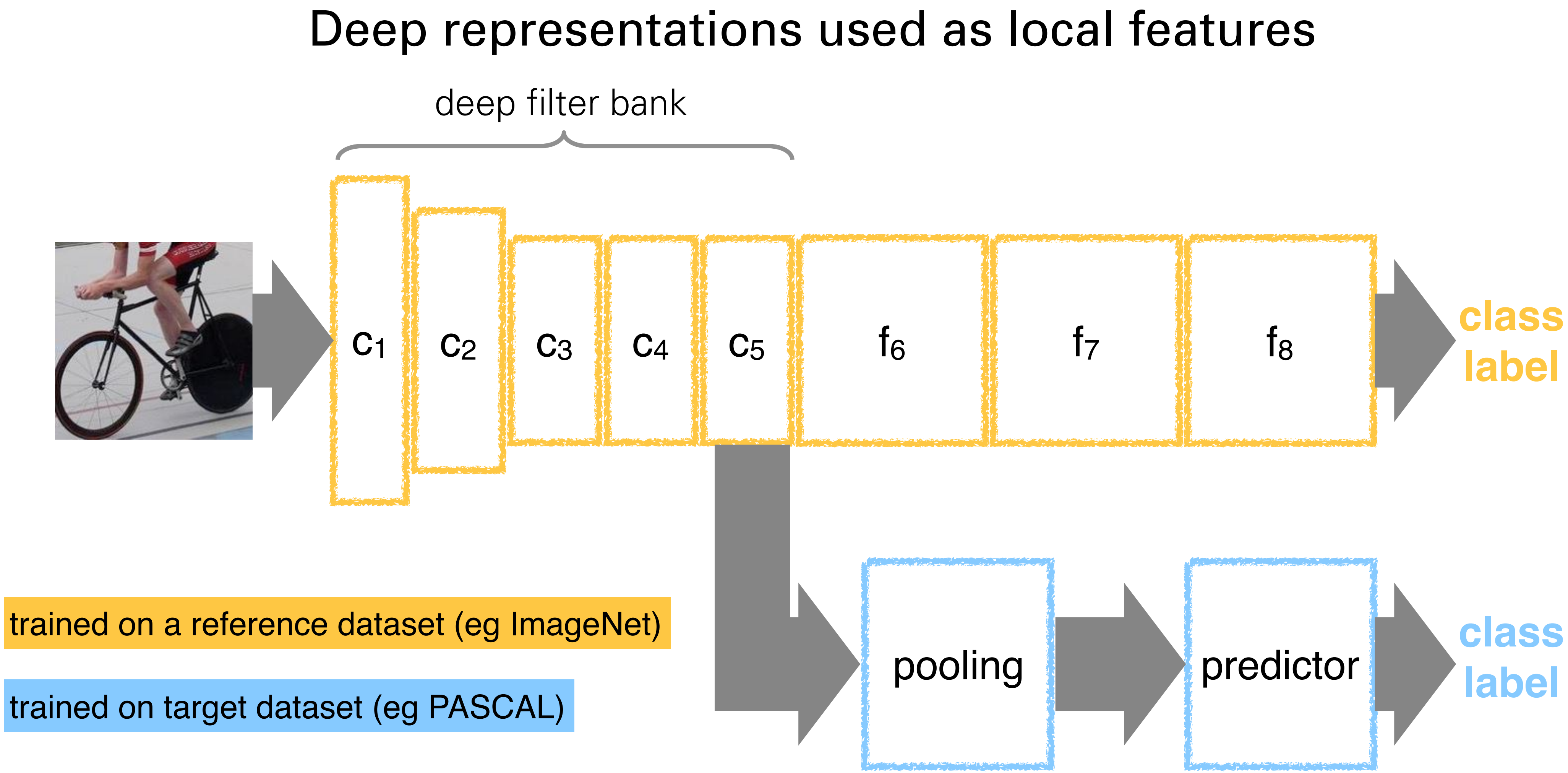
- A general purpose deep encoder is obtained by chopping off the last layers of a CNN trained on a large dataset.

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

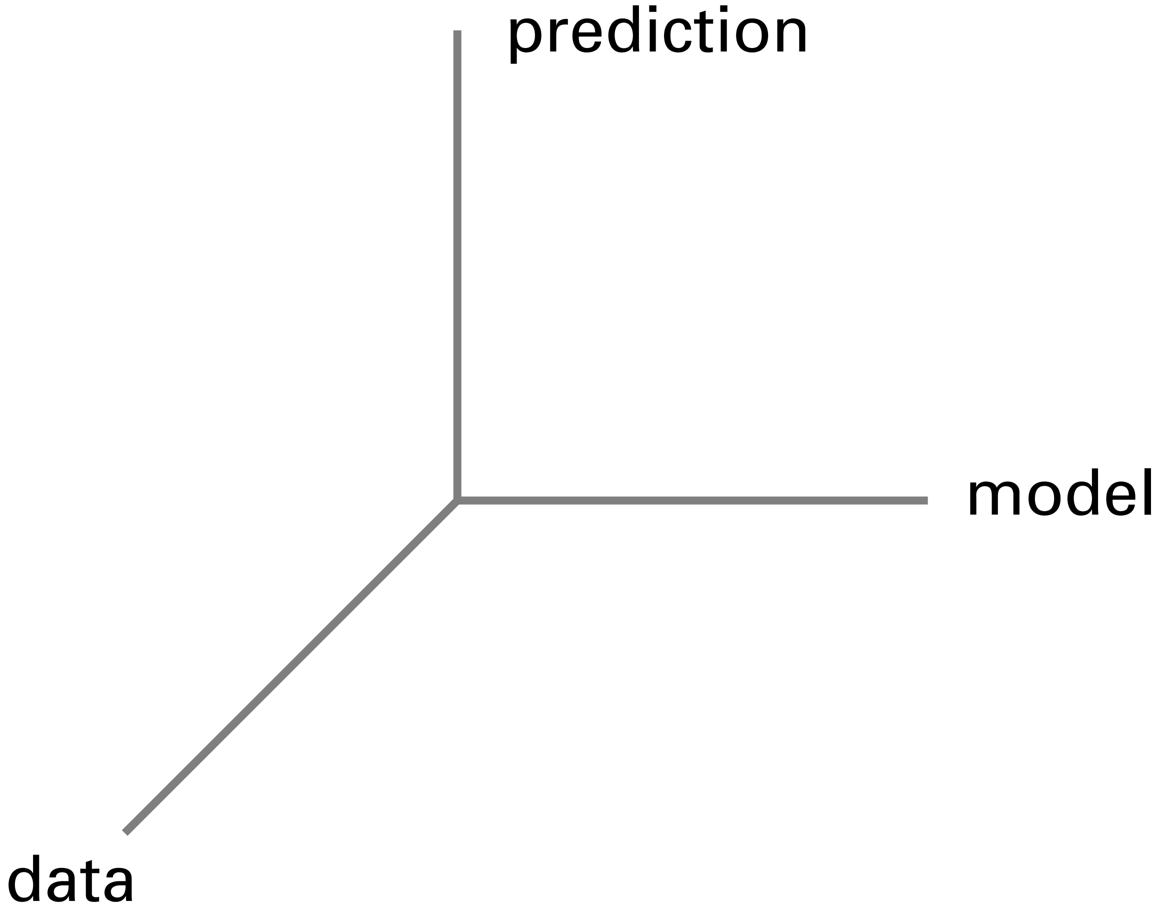


- In R-CNN and similar models, the most important shared component are the convolutional features.

Interpretability

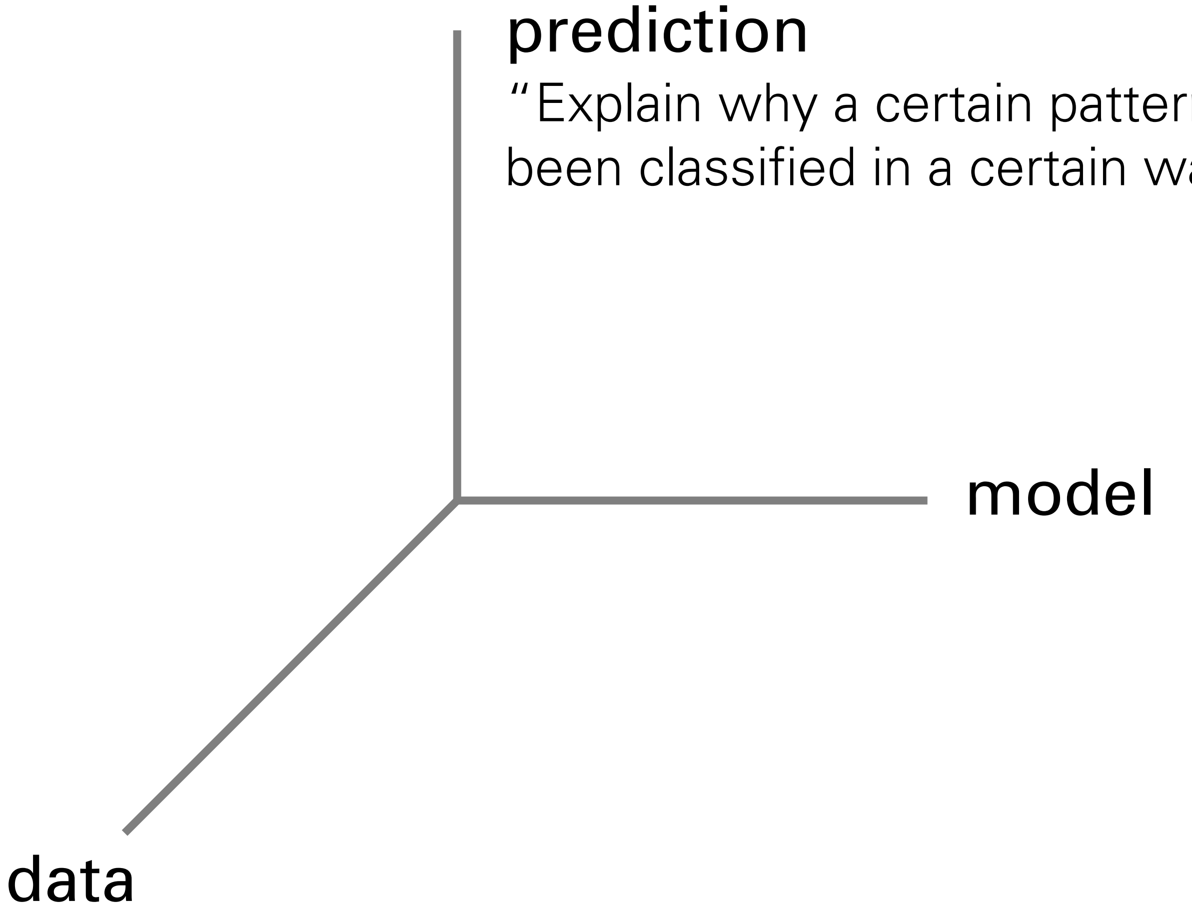
Dimensions of Interpretation

Different dimensions
of “interpretability”

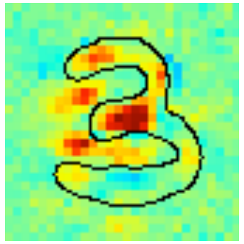


Dimensions of Interpretation

Different dimensions of "interpretability"



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

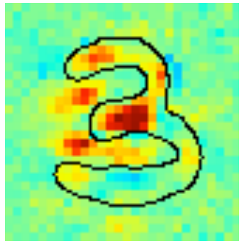


Dimensions of Interpretation

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."

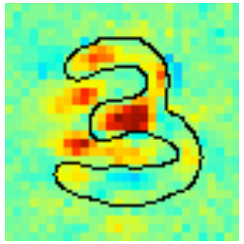
data

Dimensions of Interpretation

Different dimensions of "interpretability"

prediction

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

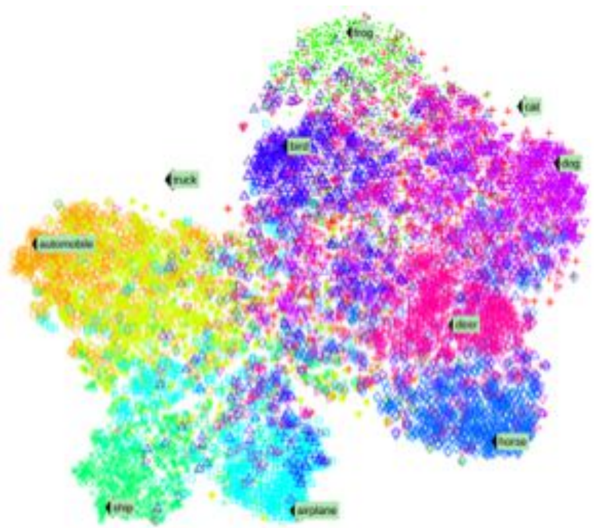


model

"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."



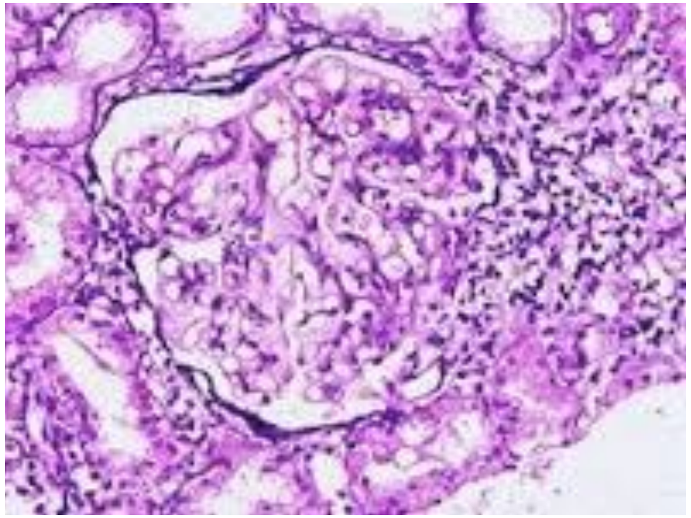
Why Interpretability?

1) Verify that classifier works as expected

Wrong decisions can be costly and dangerous

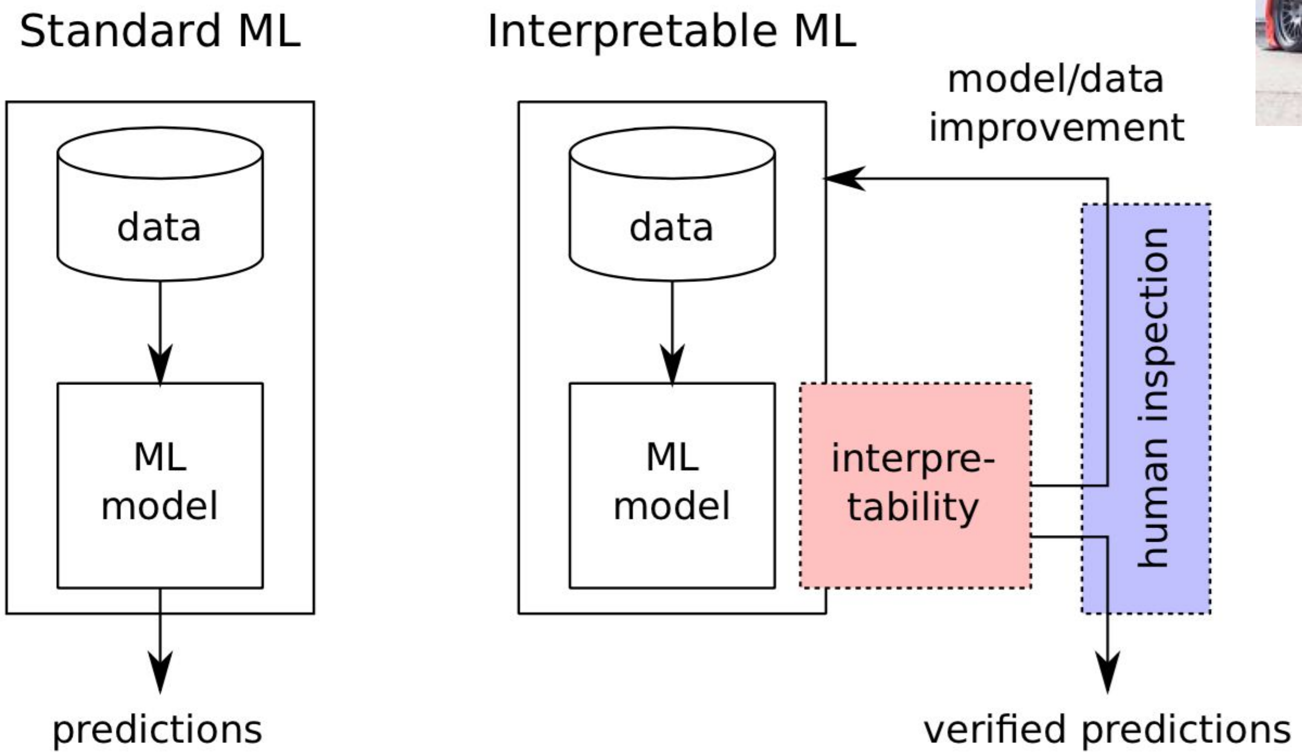
"Autonomous car crashes, because it wrongly recognizes ..."

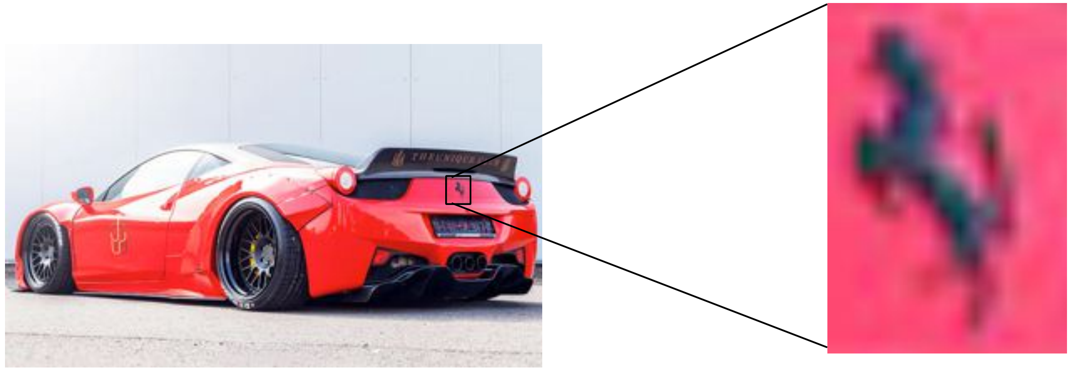
"AI medical diagnosis system misclassifies patient's disease ..."



Why Interpretability?

2) Improve classifier





Generalization error

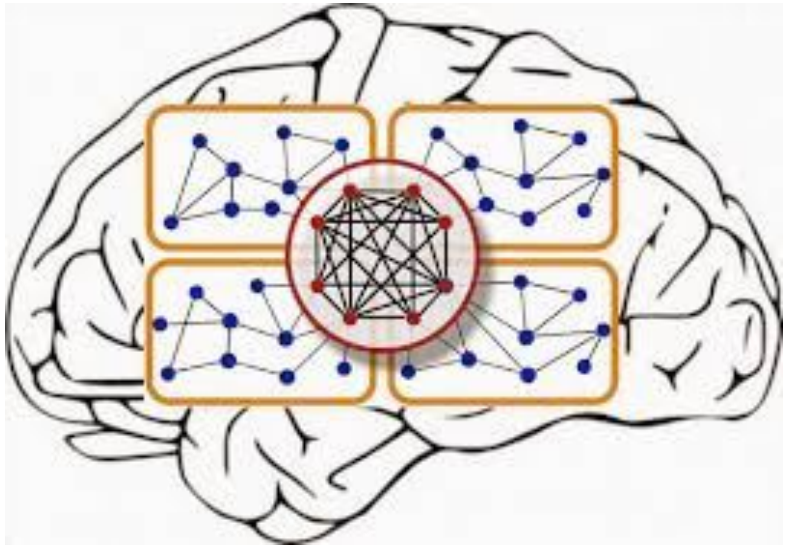
Generalization error + human experience

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)

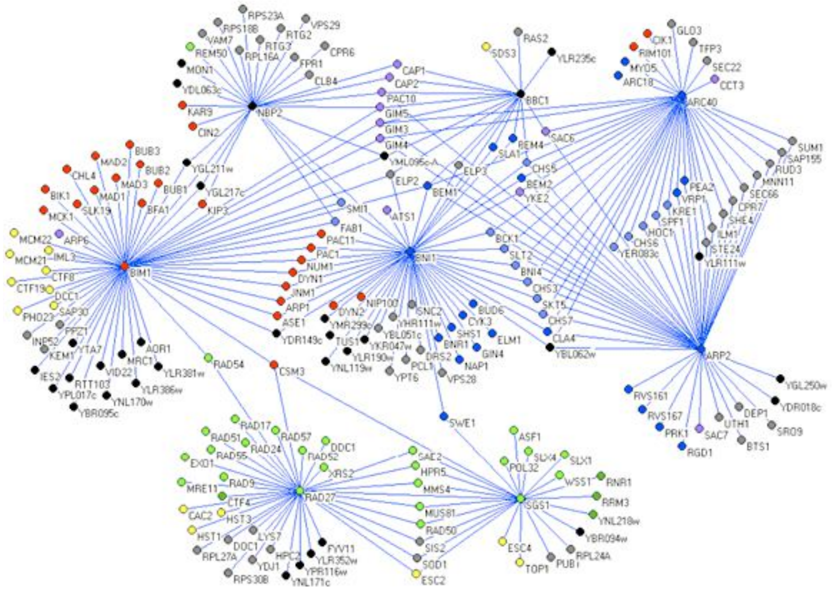
Old promise:
"Learn about the human brain."

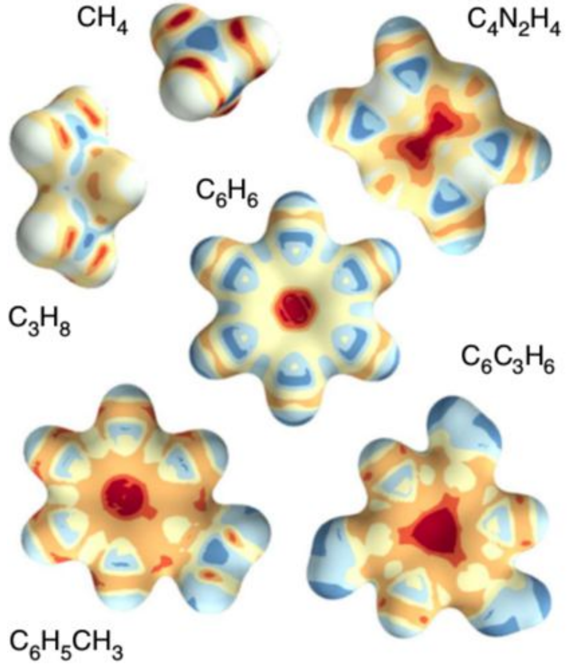


Why Interpretability?

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

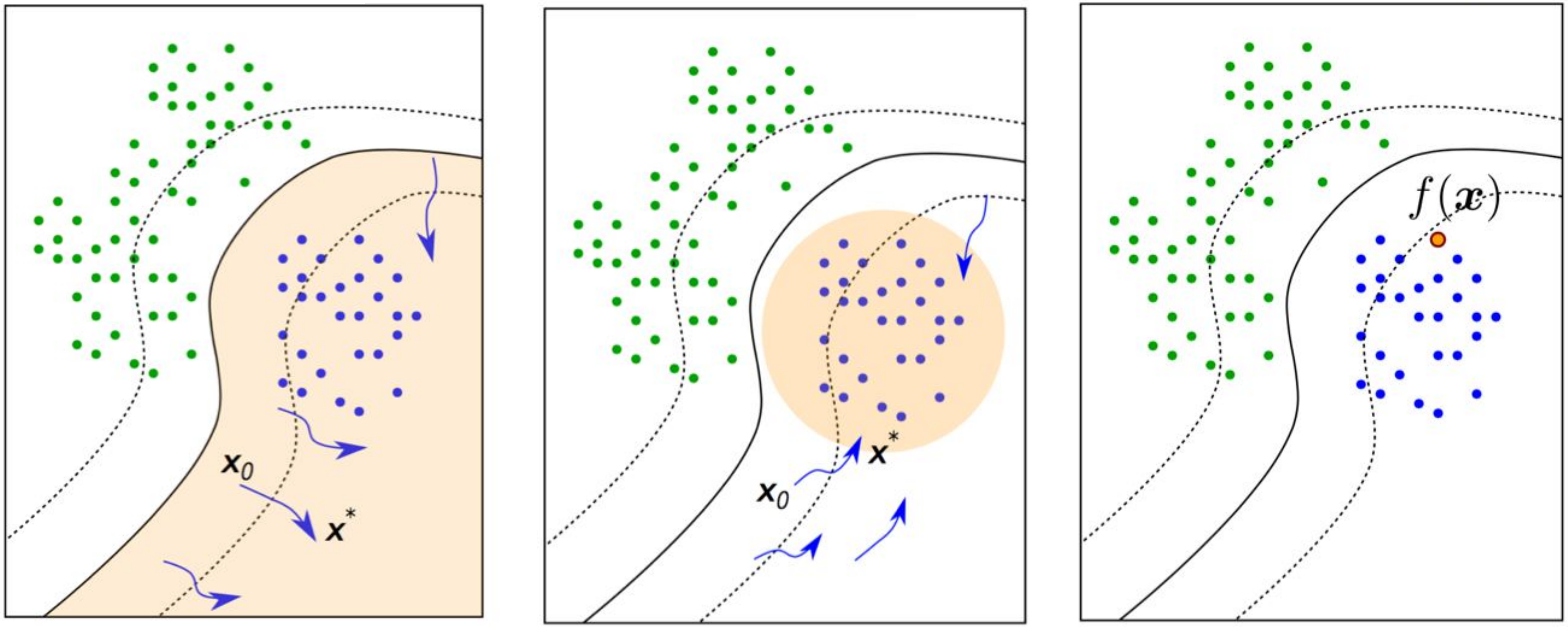
European Union's new General Data Protection Regulation → "right to explanation"

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**



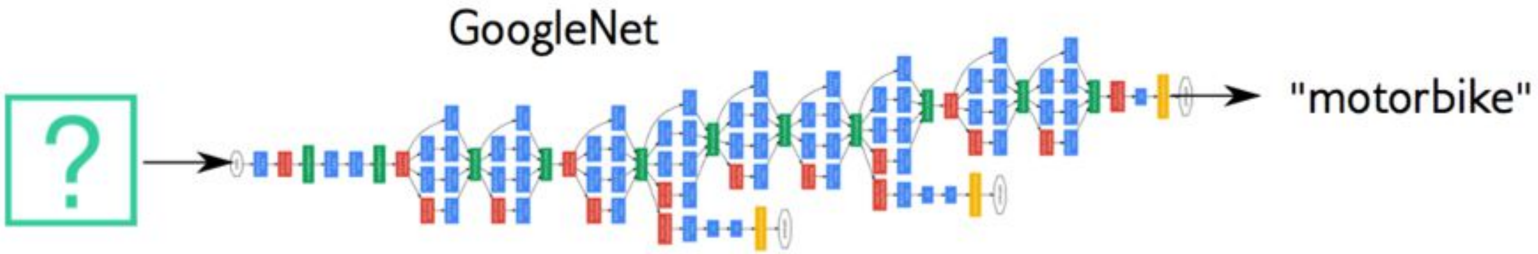
Find the input pattern that maximizes class probability.

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)

Related Work

Analysis tools

Visualizing higher-layer features of a deep network
Ethan et al. 2009
[intermediate features]

Deep inside convolutional networks
Simonyan et al. 2014
[deepest features, aka "deep dreams"]

DeConvNets
Zeiler et al. In ECCV, 2014
[intermediate features]

Understanding neural networks through deep visualisation
Yosinksi et al. 2015
[intermediate features]

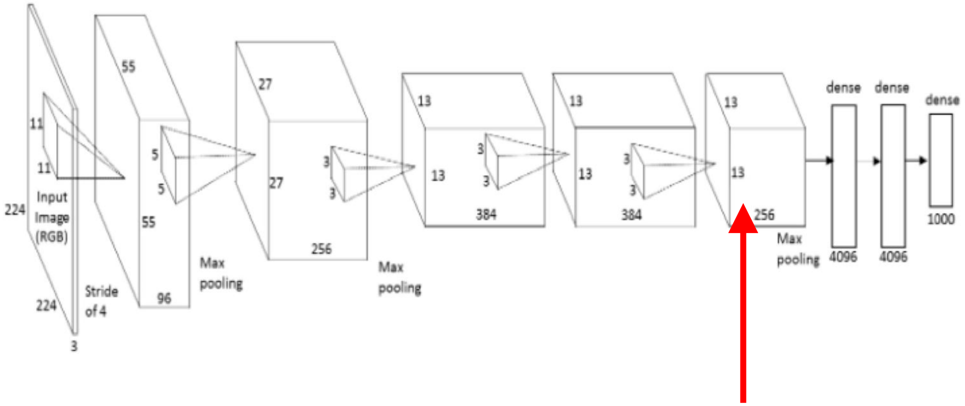
Artistic tools

Google's "inceptionsm"
Mordvintsev et al. 2015

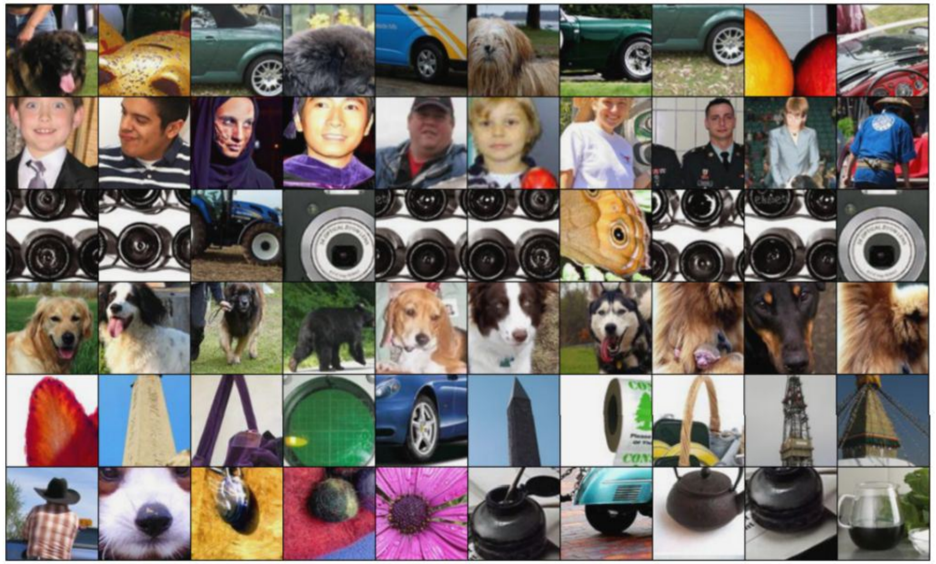
Style synthesis and transfer
Gatys et al. 2015

And many more…

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



Visualize patches that maximally activate neurons

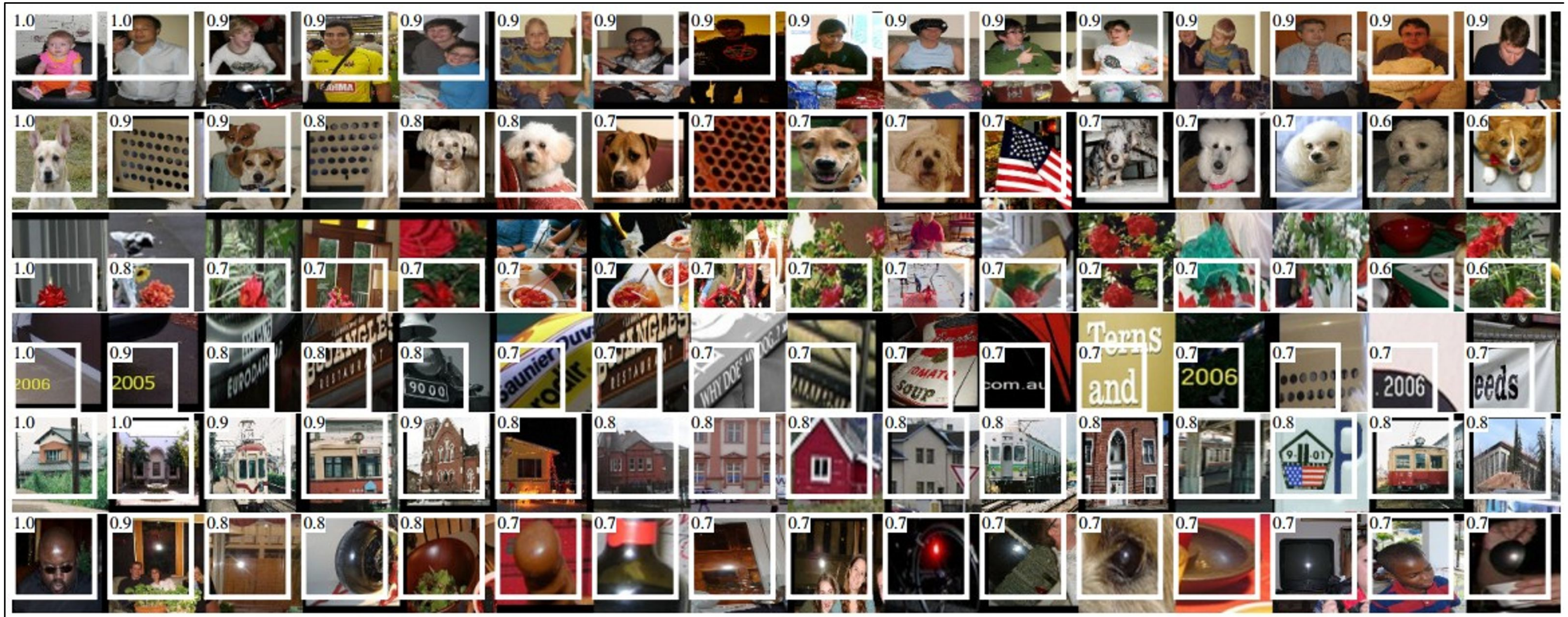
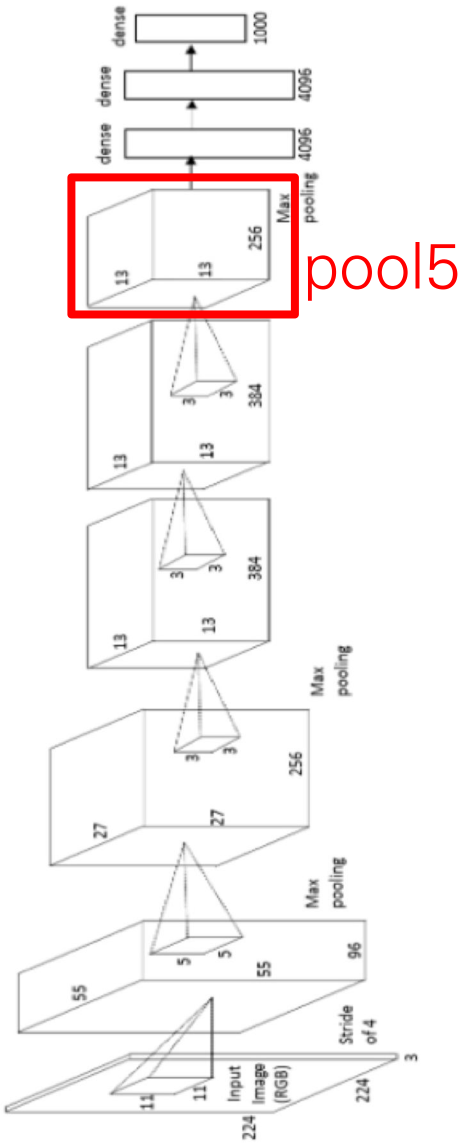


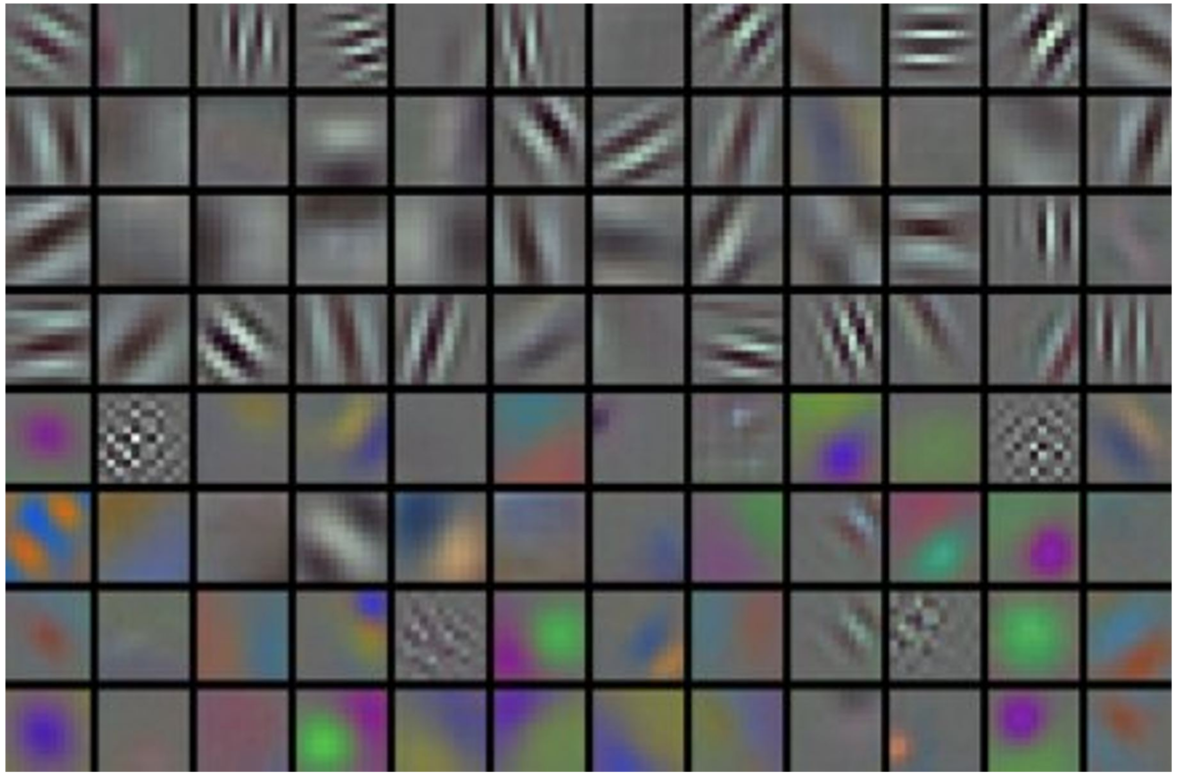
Figure 4: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

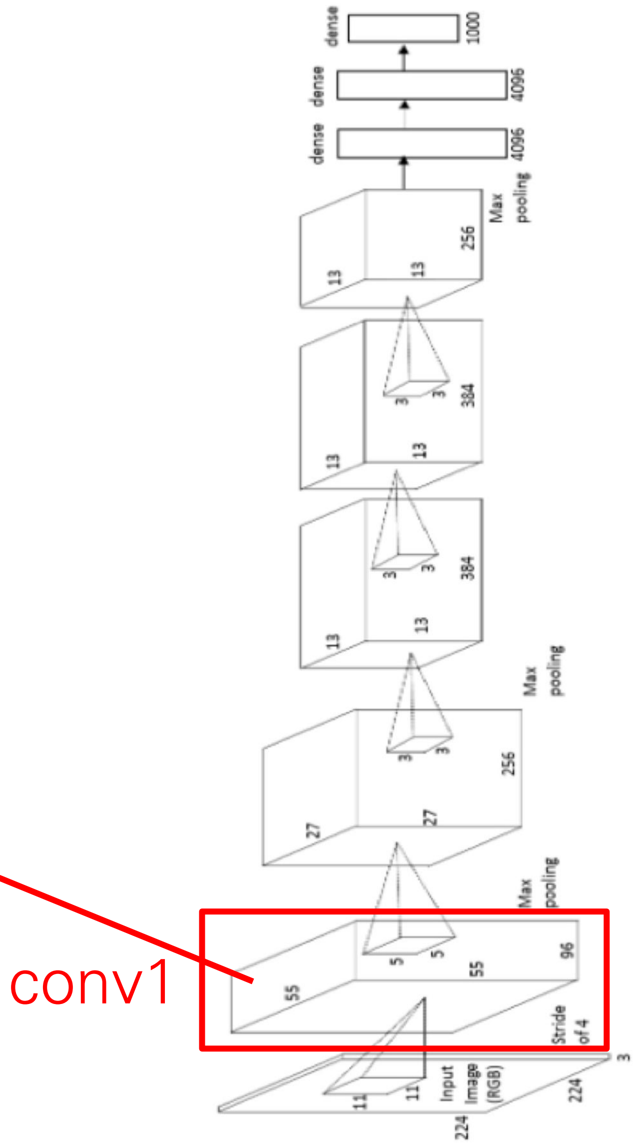
one-stream AlexNet



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

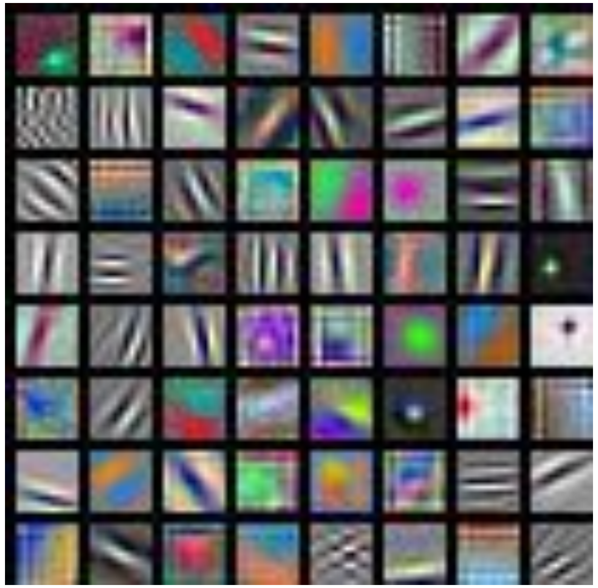
Visualize the filters/kernels (raw weights)





only interpretable on the first layer :(

Visualize the filters/kernels (raw weights)

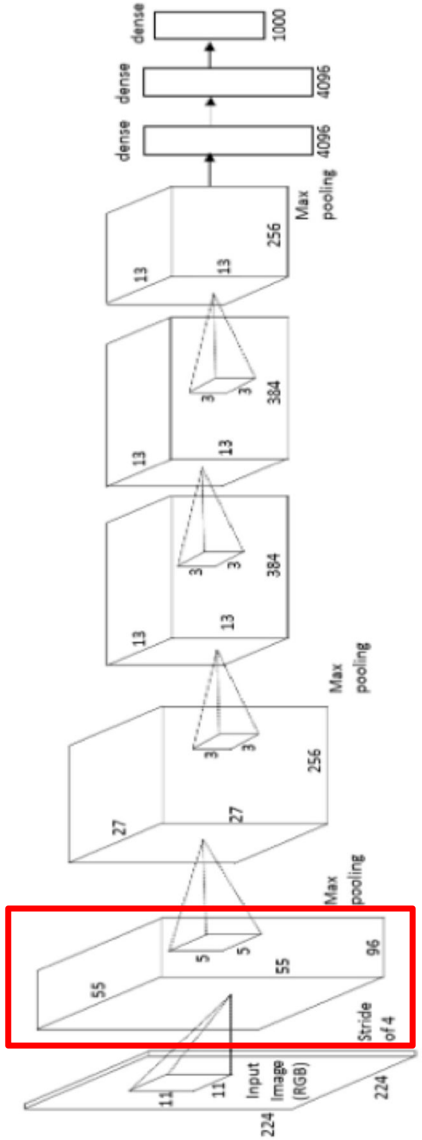


AlexNet:
64 x 3 x 11 x 11

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

Visualize the filters/kernels (raw weights)

layer 1 weights

layer 2 weights

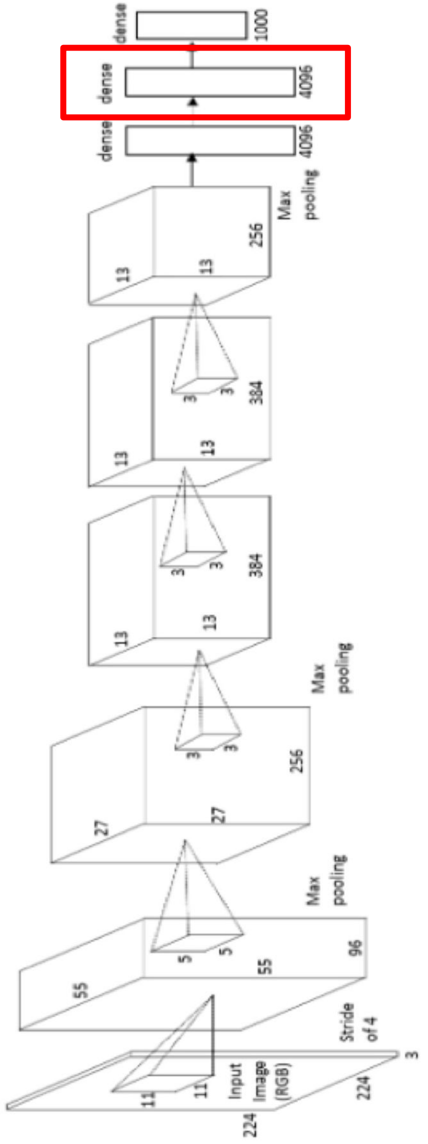
layer 3 weights

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

(these are taken from ConvNetJS CIFAR-10 demo)

Visualizing the representation

fc7
layer



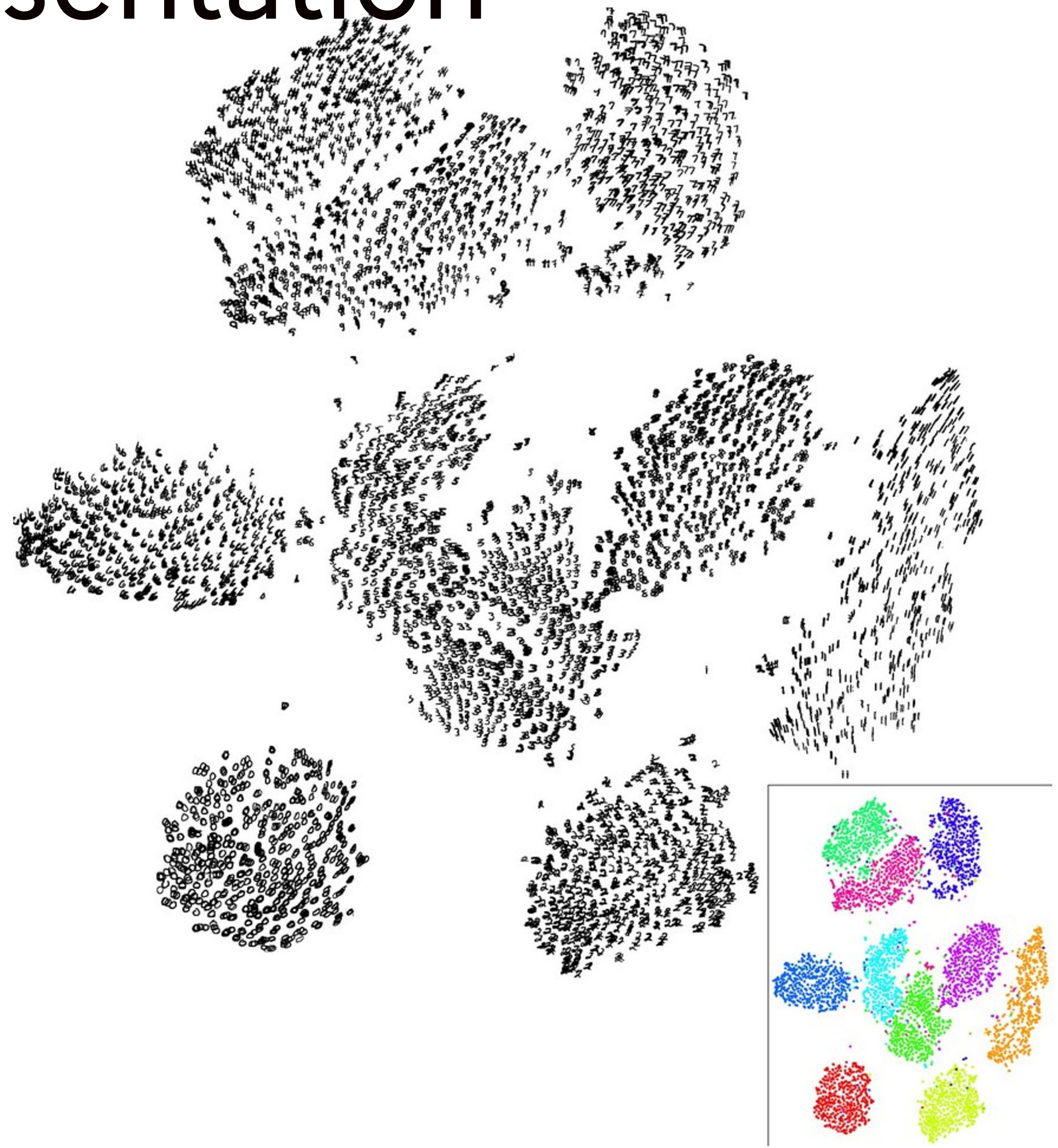
4096-dimensional "code" for an image
(layer immediately before the classifier)

can collect the code for many images

Visualizing the representation

t-SNE visualization
[van der Maaten & Hinton]

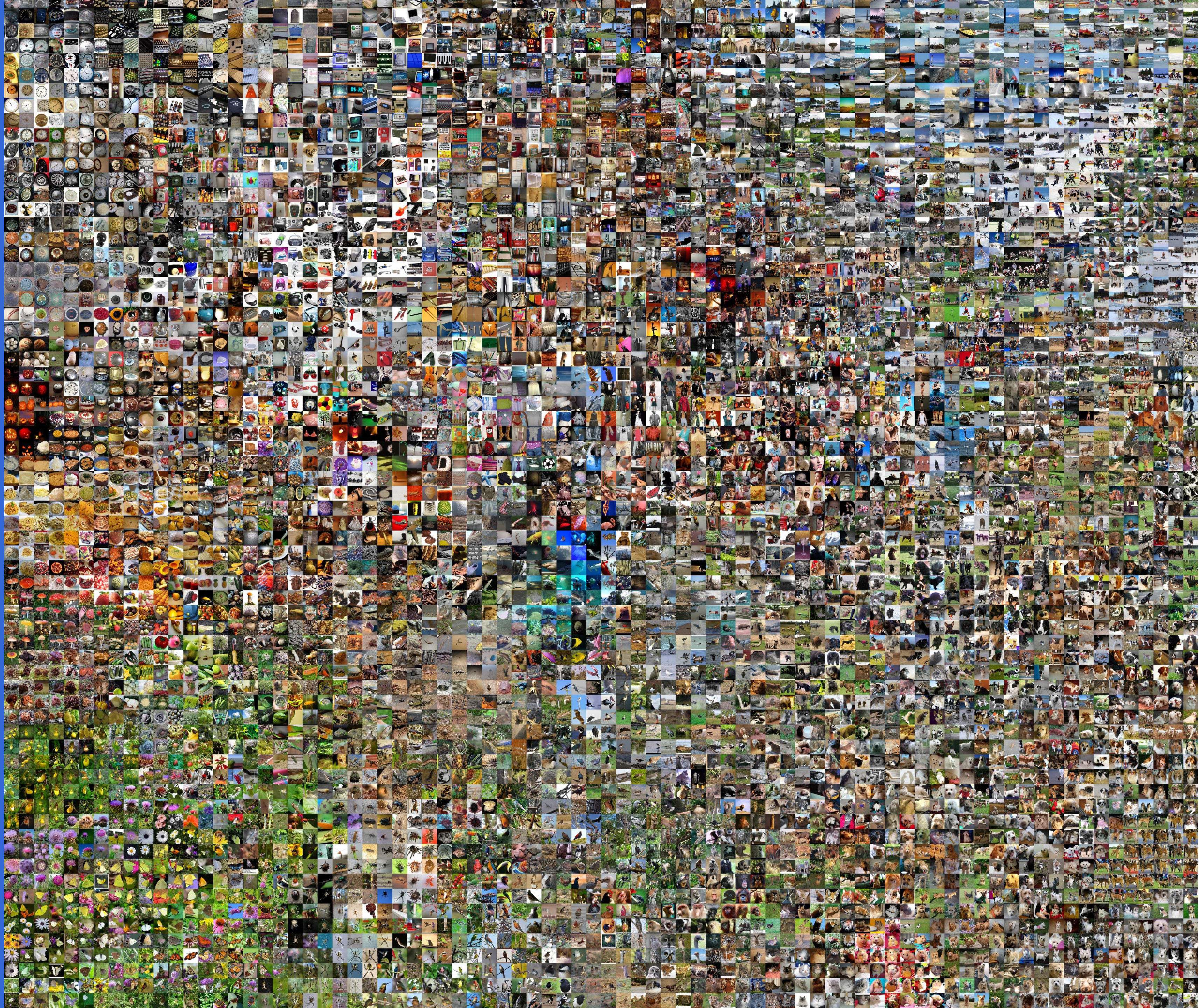
- Embed high-dimensional points so that locally, pairwise distances are conserved
- i.e. similar things end up in similar places. dissimilar things end up wherever
- **Right**: Example embedding of MNIST digits (0-9) in 2D



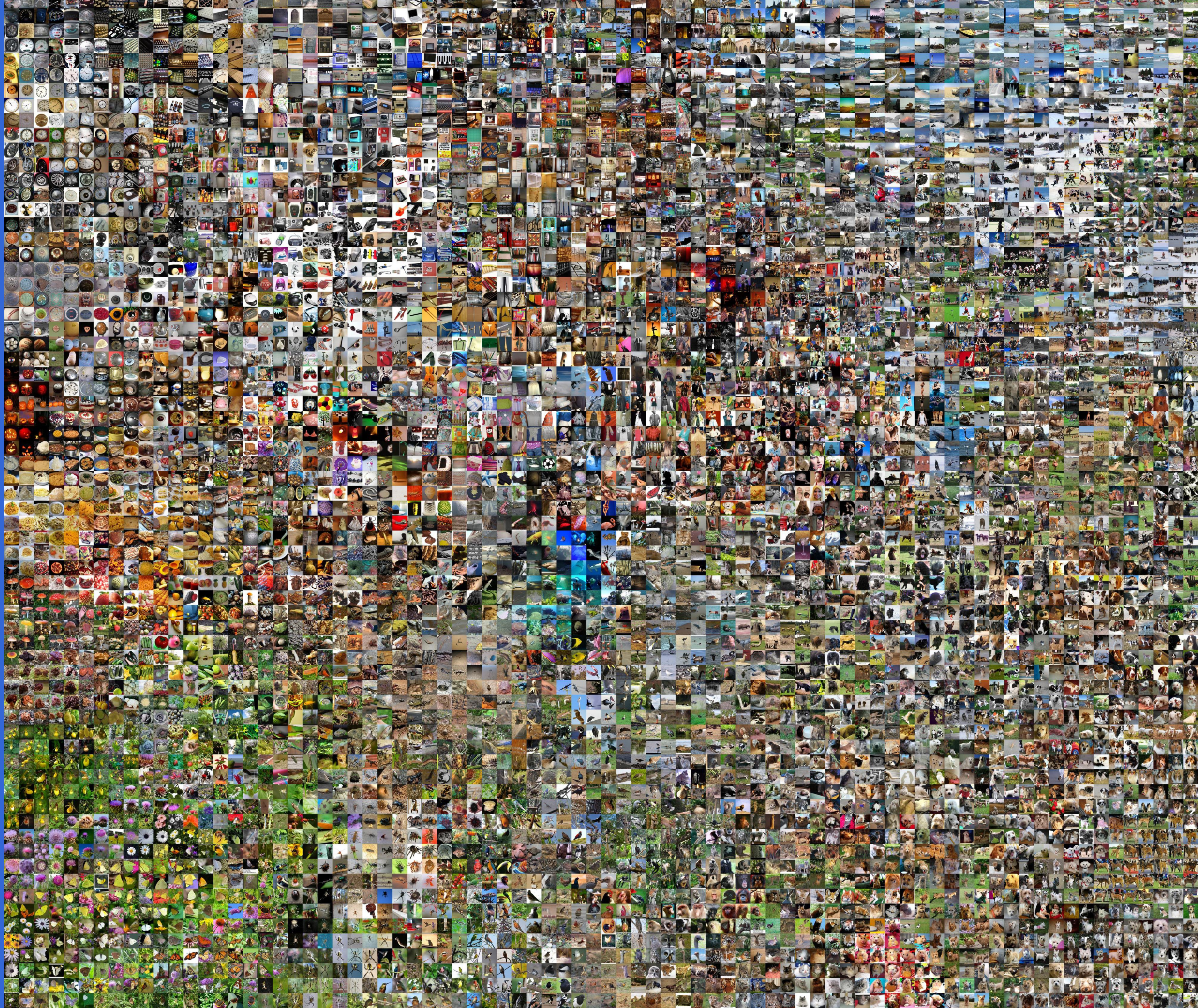
t-SNE visualization:

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

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

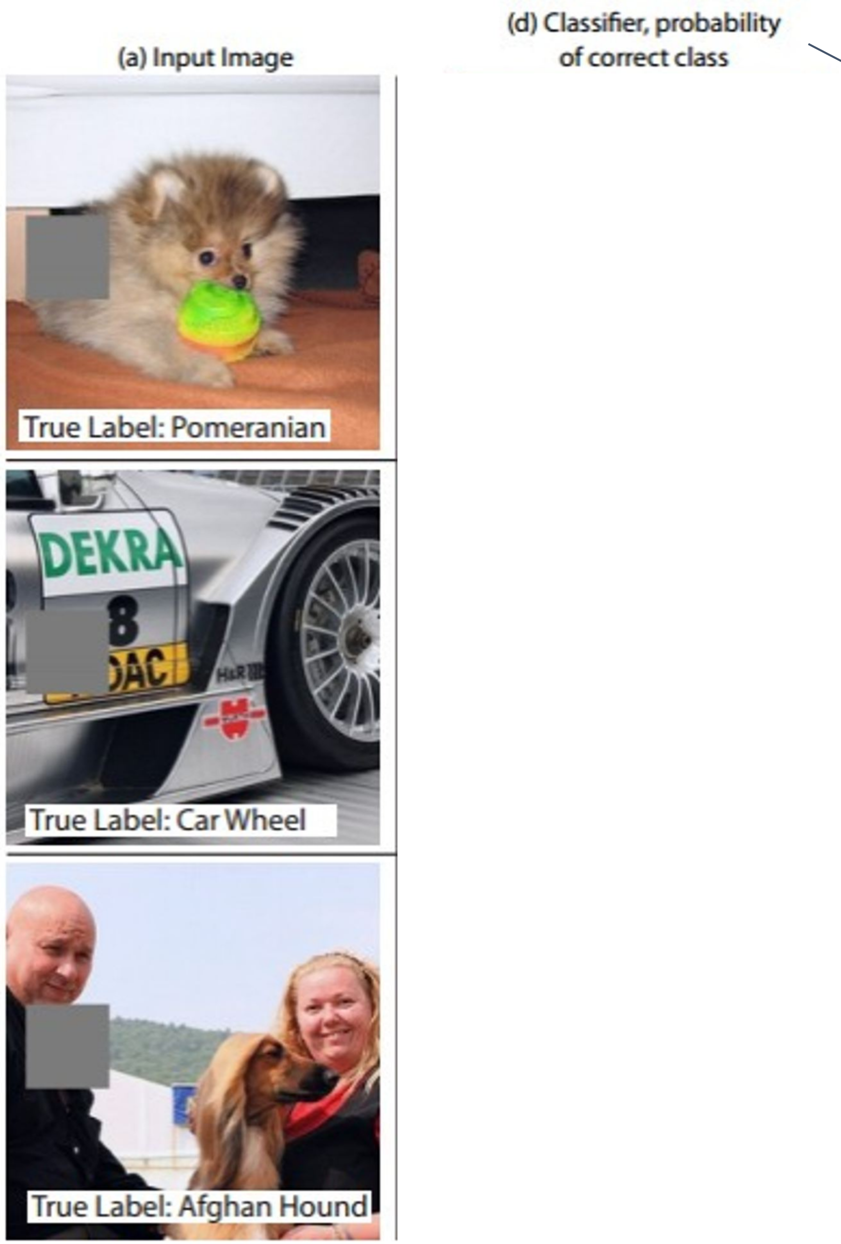


t-SNE visualization:



Occlusion experiments

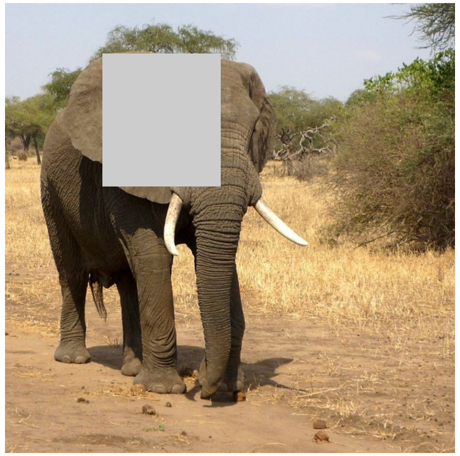
[Zeiler & Fergus 2013]

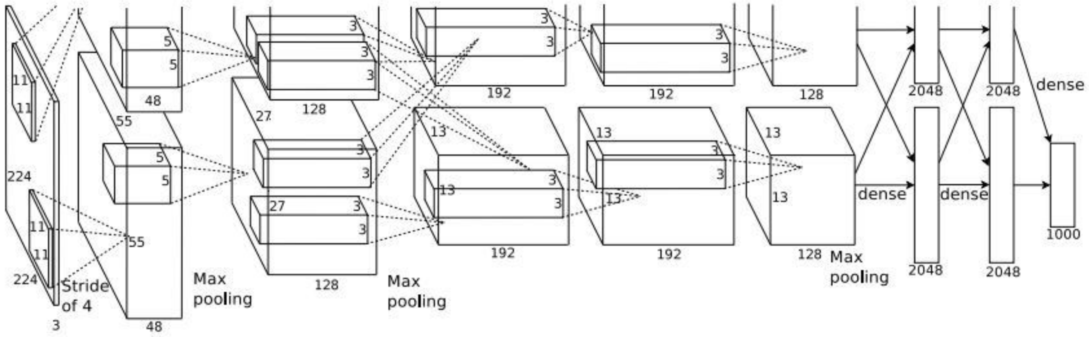


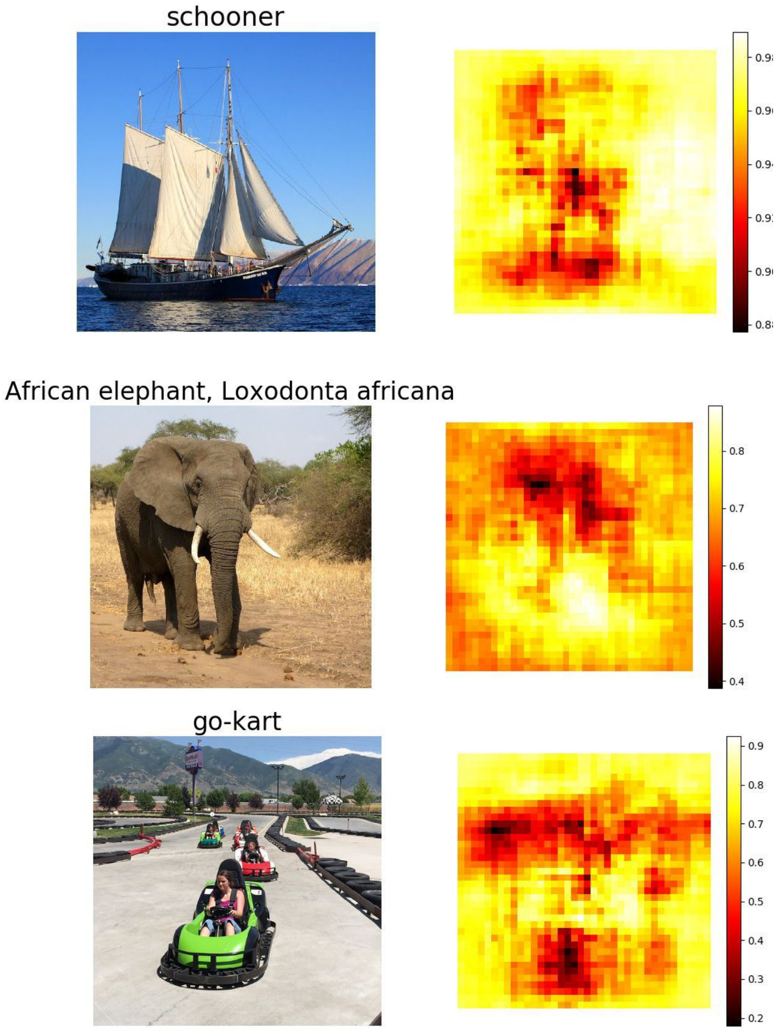
(as a function of the position of the square of zeros in the original image)

Occlusion experiments
[Zeiler & Fergus 2013]

Mask part of the image before feeding to CNN, draw heatmap of probability at each mask location



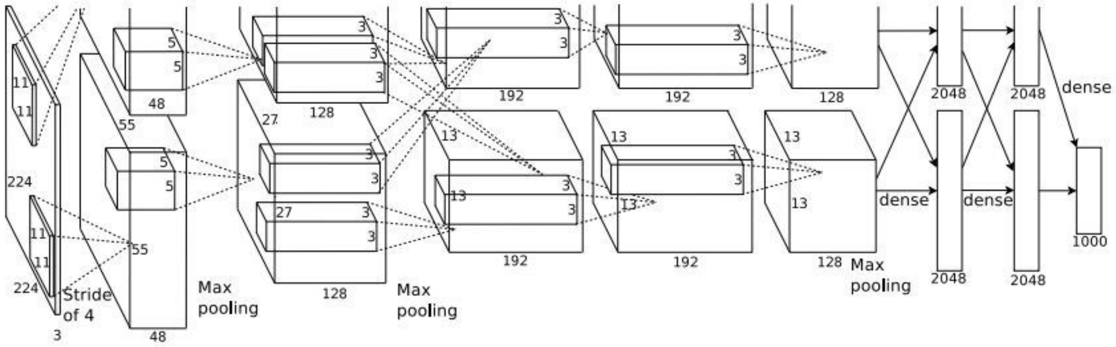




Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Class-specific image saliency

How to tell which pixels matter for classification?

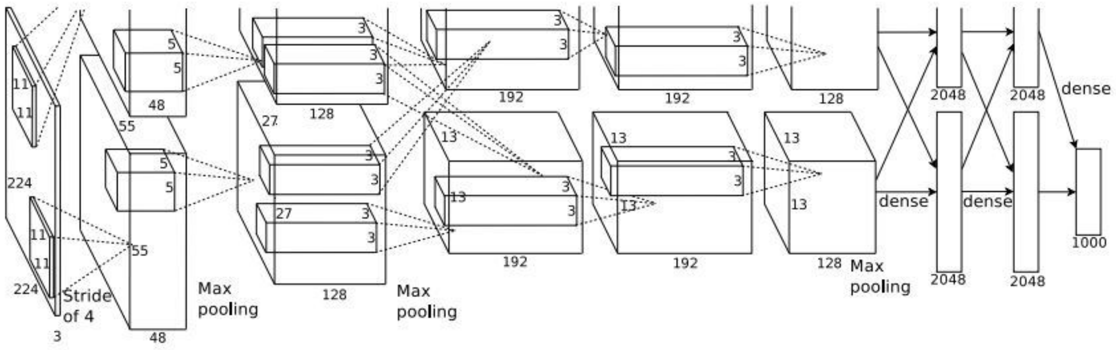


Dog

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

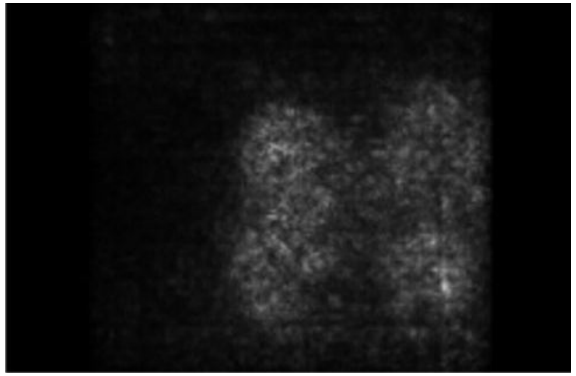
Class-specific image saliency

How to tell which pixels matter for classification?



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

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 $\boldsymbol{I}_0$ with the first order Taylor expansion

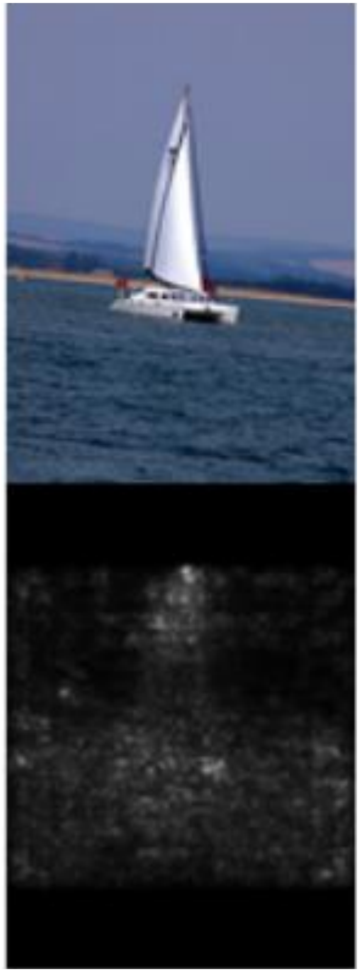
$$S_c(I)|_{I_0} \approx w^T I + b \text{ , where } w = \frac{\partial S_c}{\partial I}|_{I_0}$$

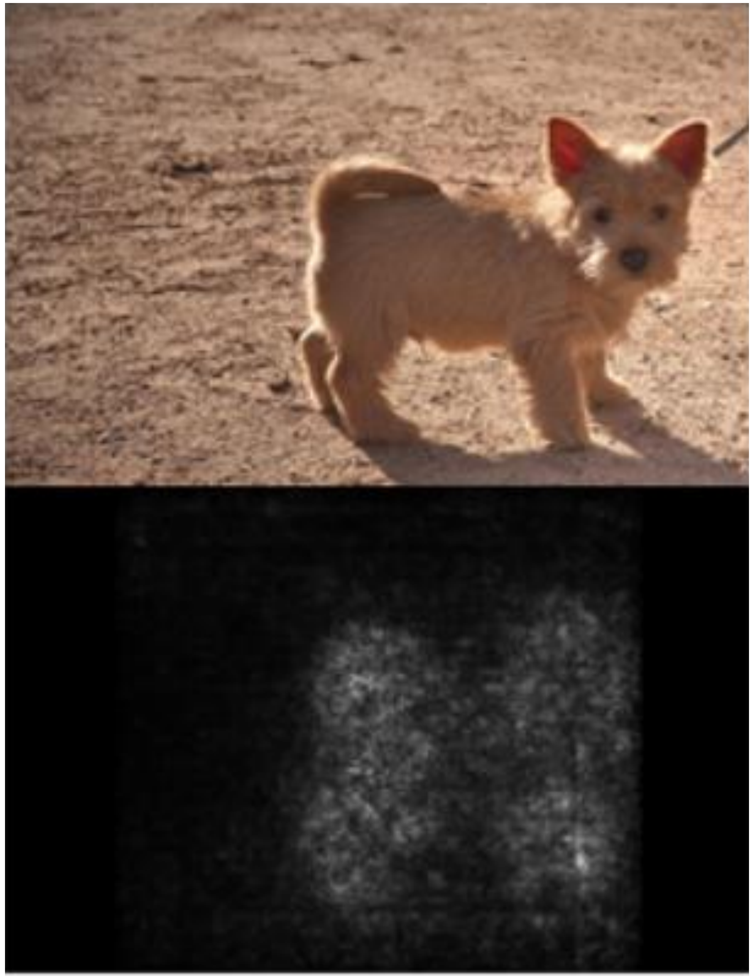
from backpropagation

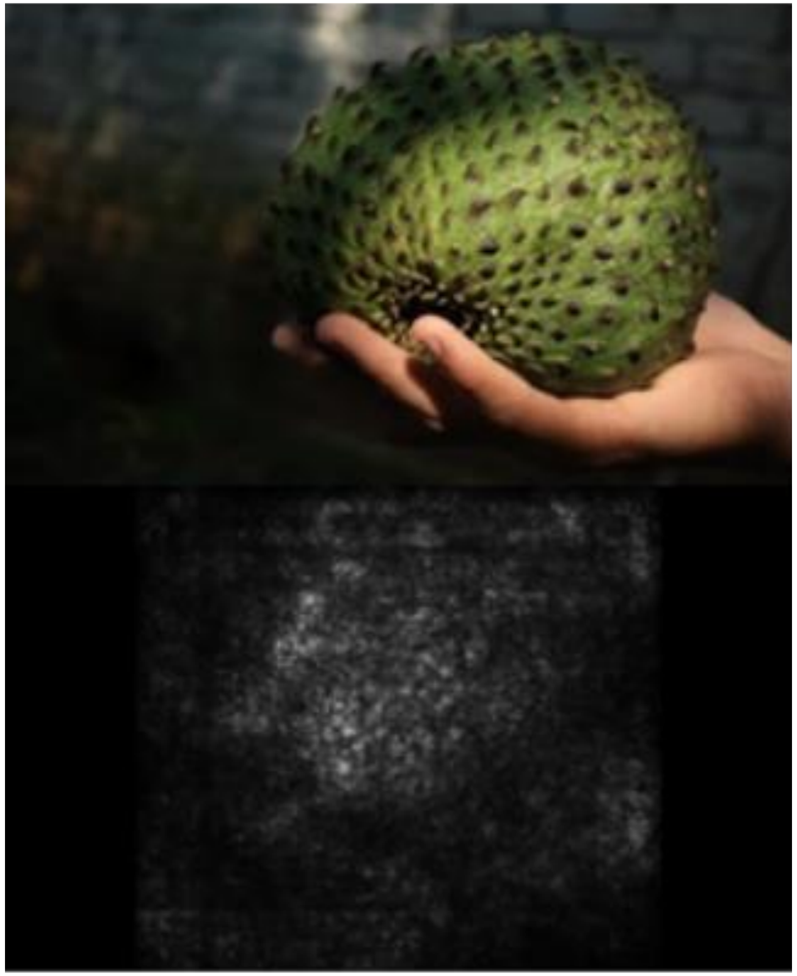
  – Solution is locally optimal

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

Examples







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

Examples

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

Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization
[Selvaraju et al. 2016]

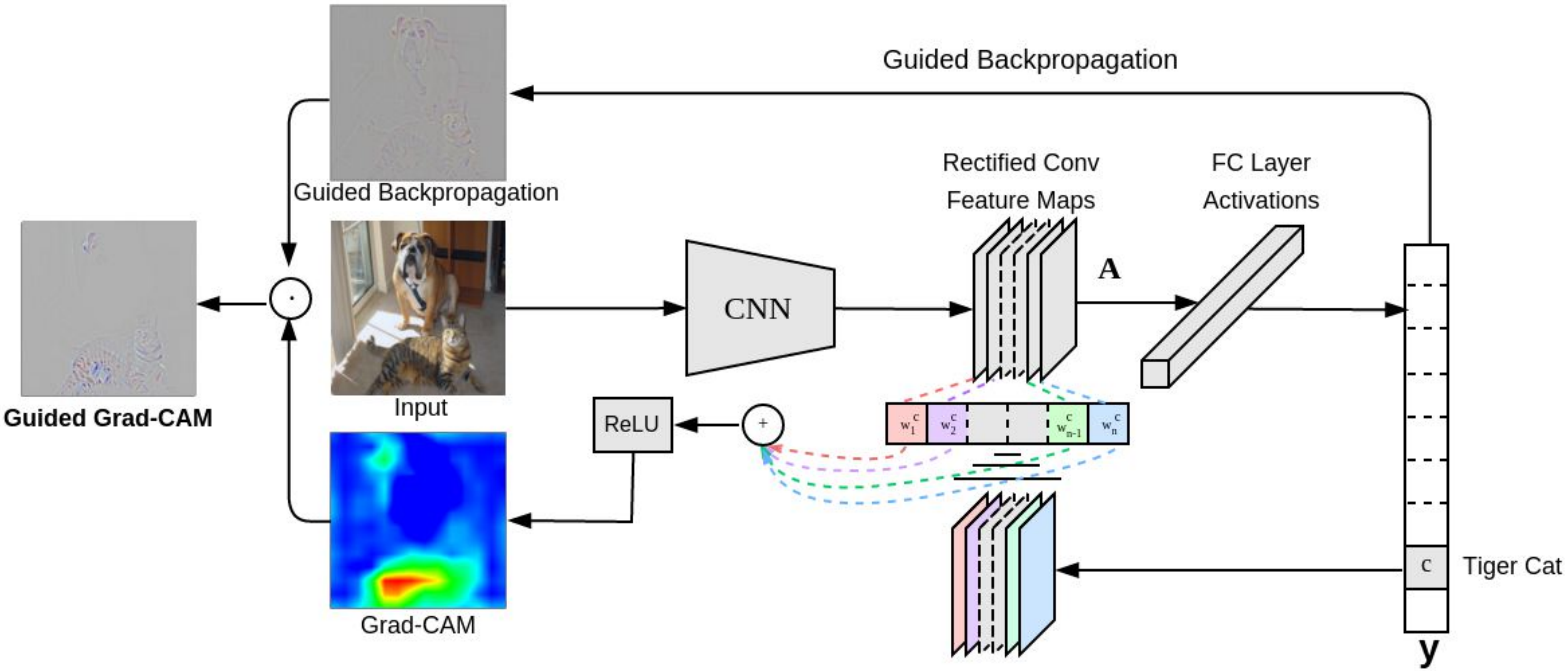
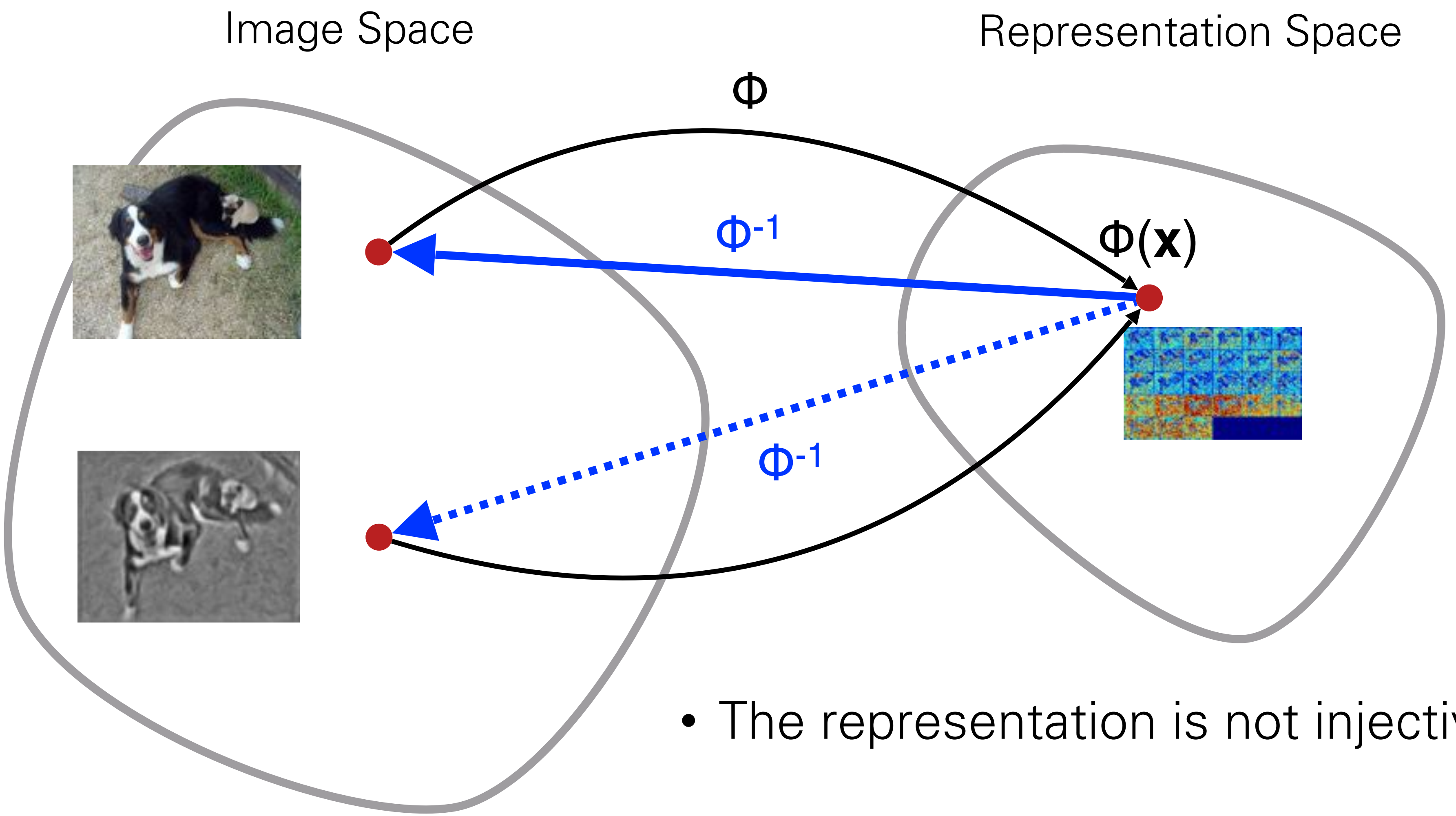


Figure 2: Grad-CAM overview: Given an image, and a category ('tiger cat') as input, we foward propagate the image through the model to obtain the raw class scores before softmax. The gradients are set to zero for all classes except the desired class (tiger cat), which is set to 1. This signal is then backpropagated to the rectified convolutional feature map of interest, where we can compute the coarse Grad-CAM localization (blue heatmap). Finally, we pointwise multiply the heatmap with guided backpropagation to get Guided Grad-CAM visualizations which are both high-resolution and class-discriminative.

Understanding the Model: Pre-Images

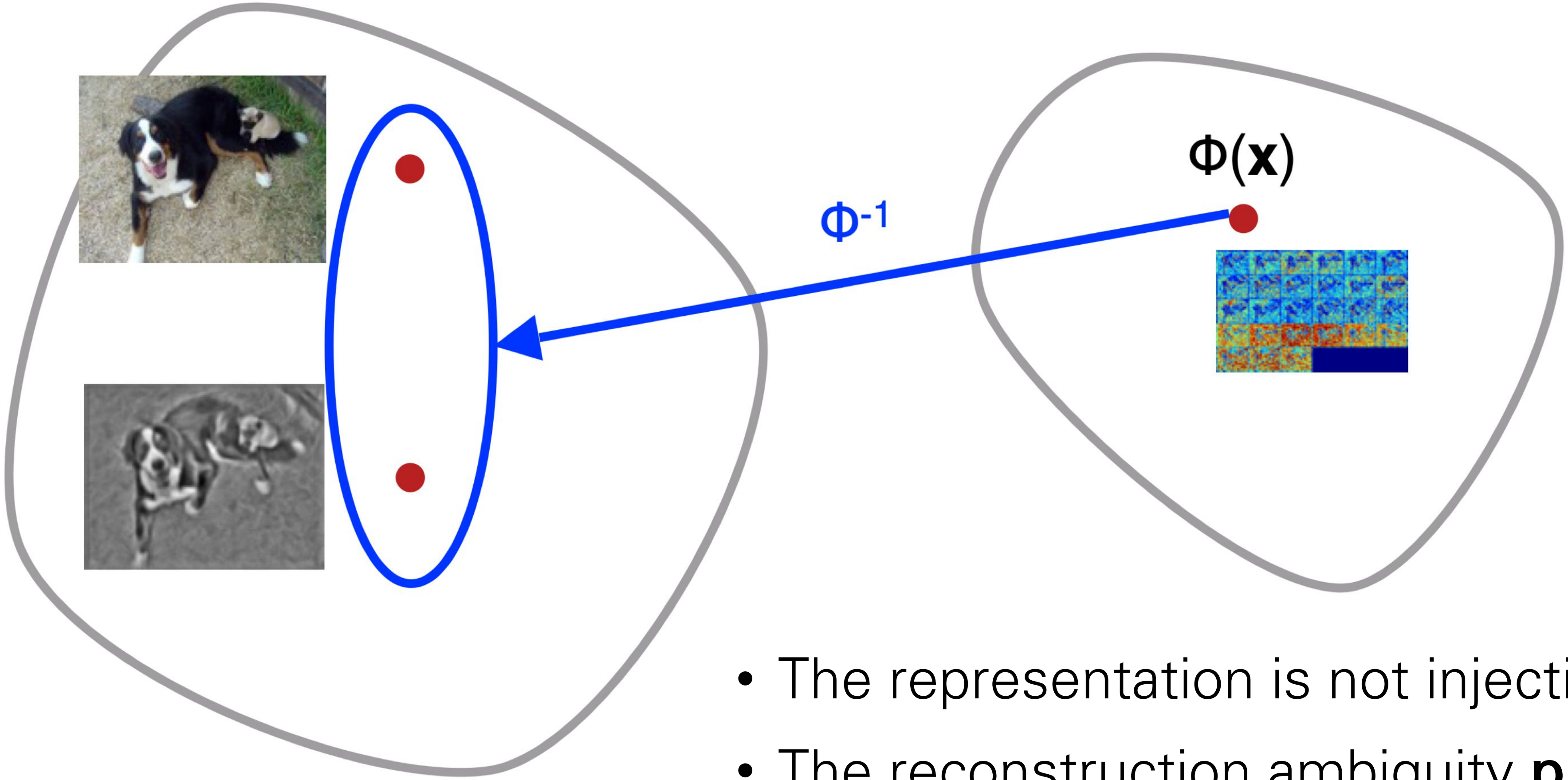


- The representation is not injective

Understanding the Model: Pre-Images

Image Space

Representation Space



- 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

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

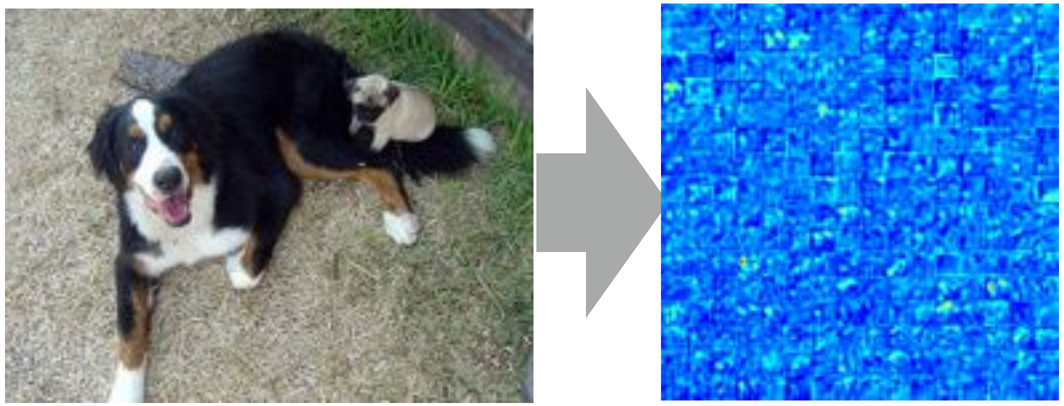


Image Representation

Finding a Pre-Image

A simple yet general and effective method

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

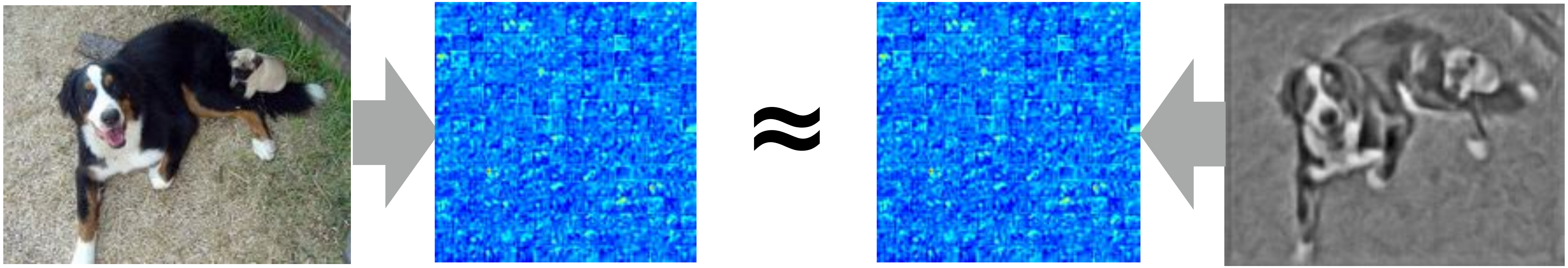


Image Representation Reconstruction Pre-Image

- 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

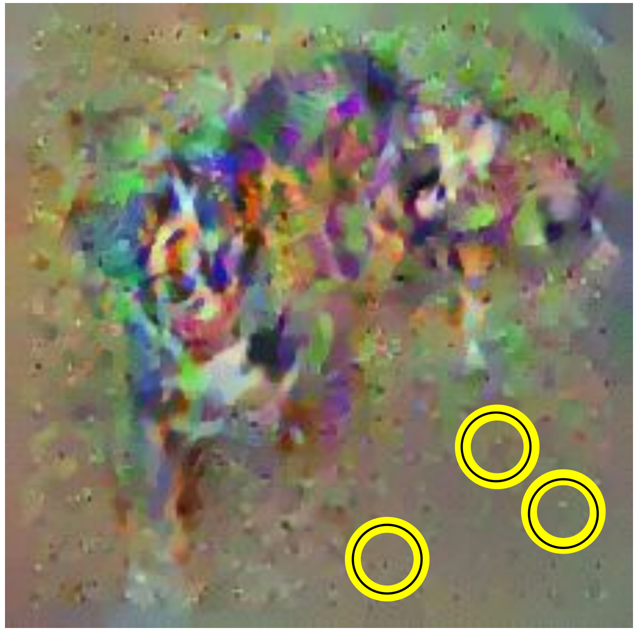
Finding a Pre-Image

A simple yet general and effective method

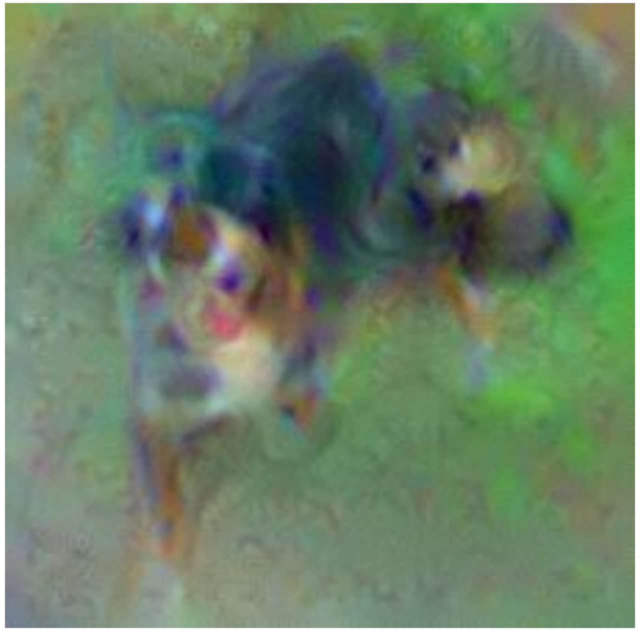
$$\min_{\mathbf{x}} \|\Phi(\mathbf{x}) - \Phi_0\|_2^2 + R_{TV}(\mathbf{x})$$

No prior

TV-norm β = 1

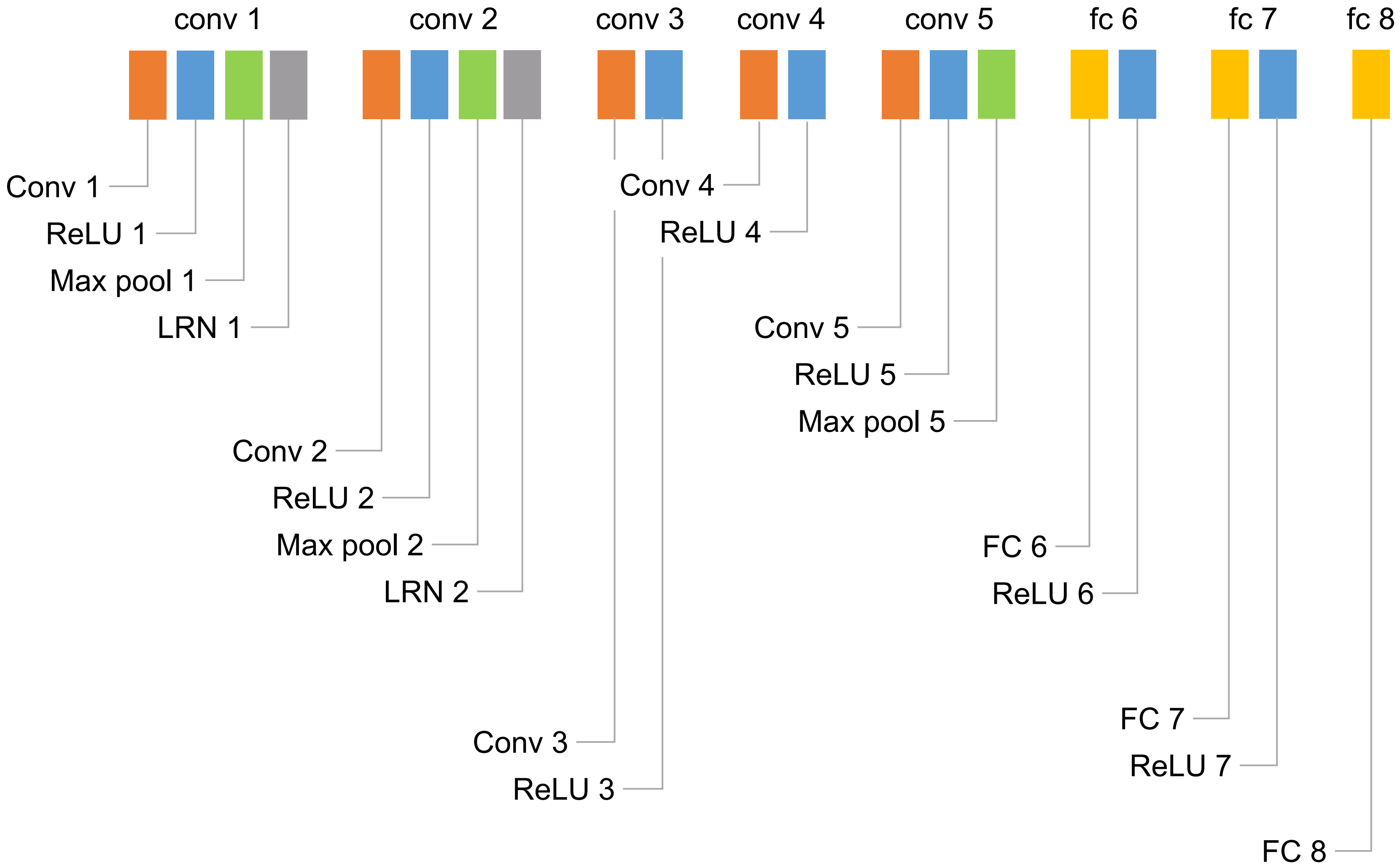


TV-norm β = 2

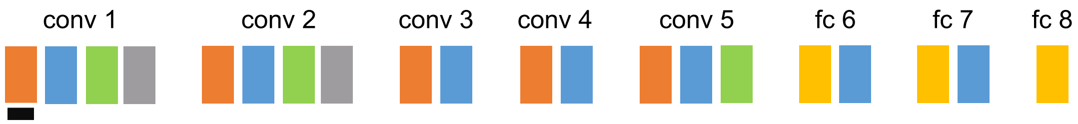


Inverting a Deep CNN

AlexNet [Krizhevsky et al. 2012]

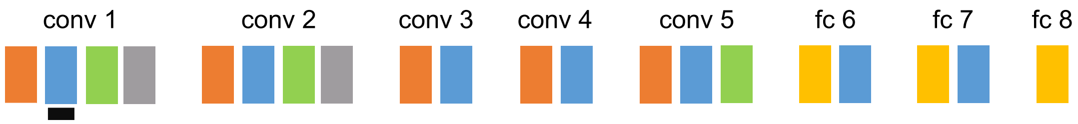


Inverting a Deep CNN



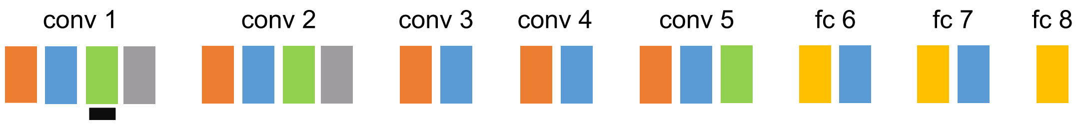
Original Image

Inverting a Deep CNN



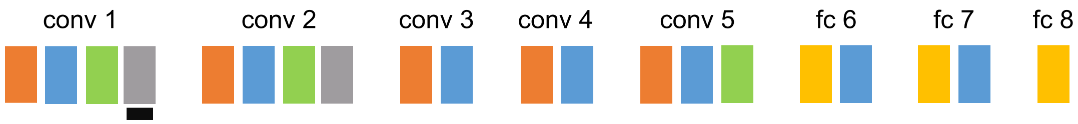
Original Image

Inverting a Deep CNN

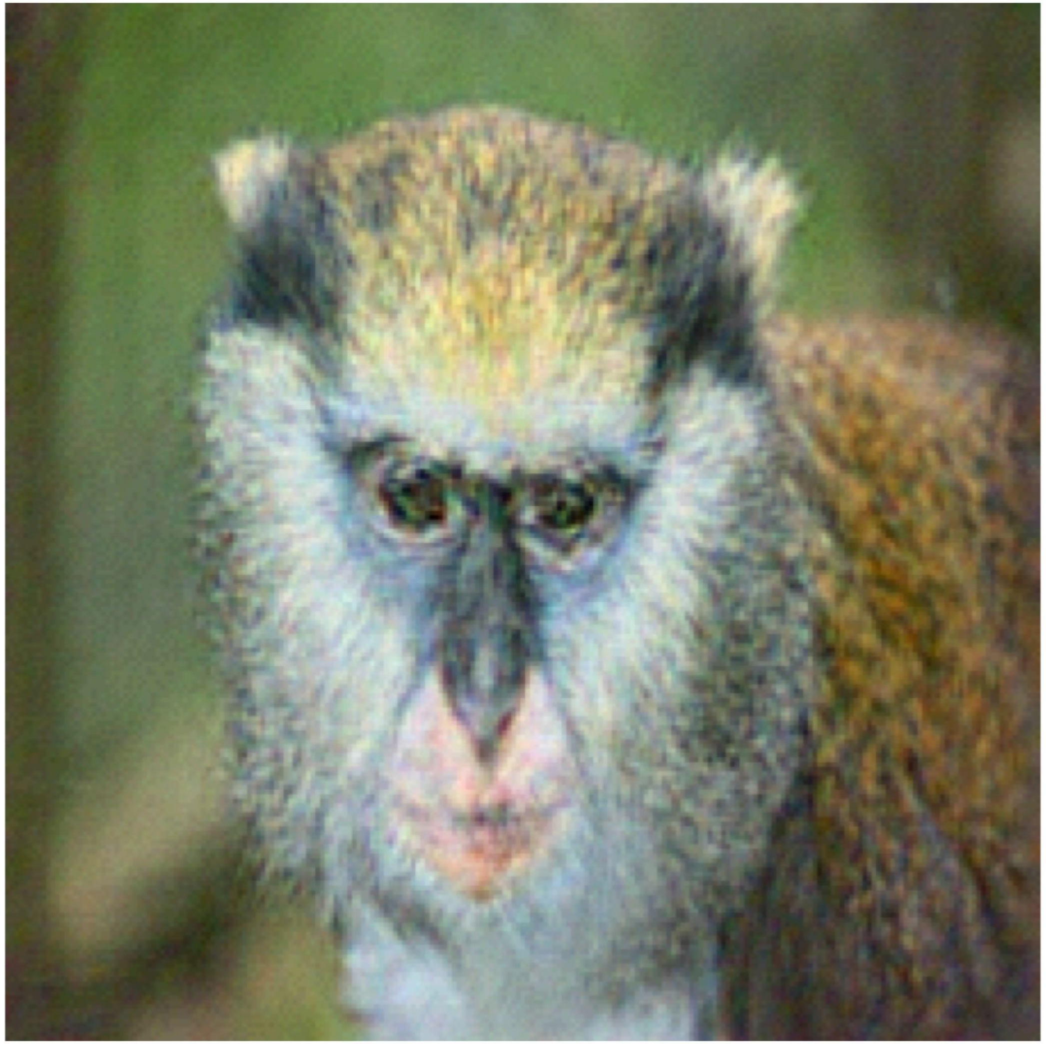


Original Image

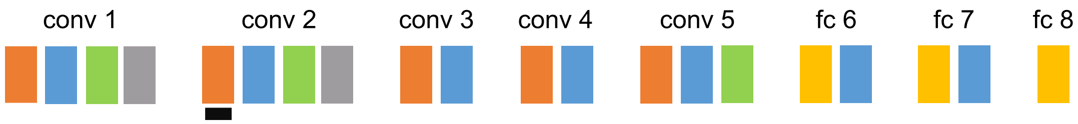
Inverting a Deep CNN



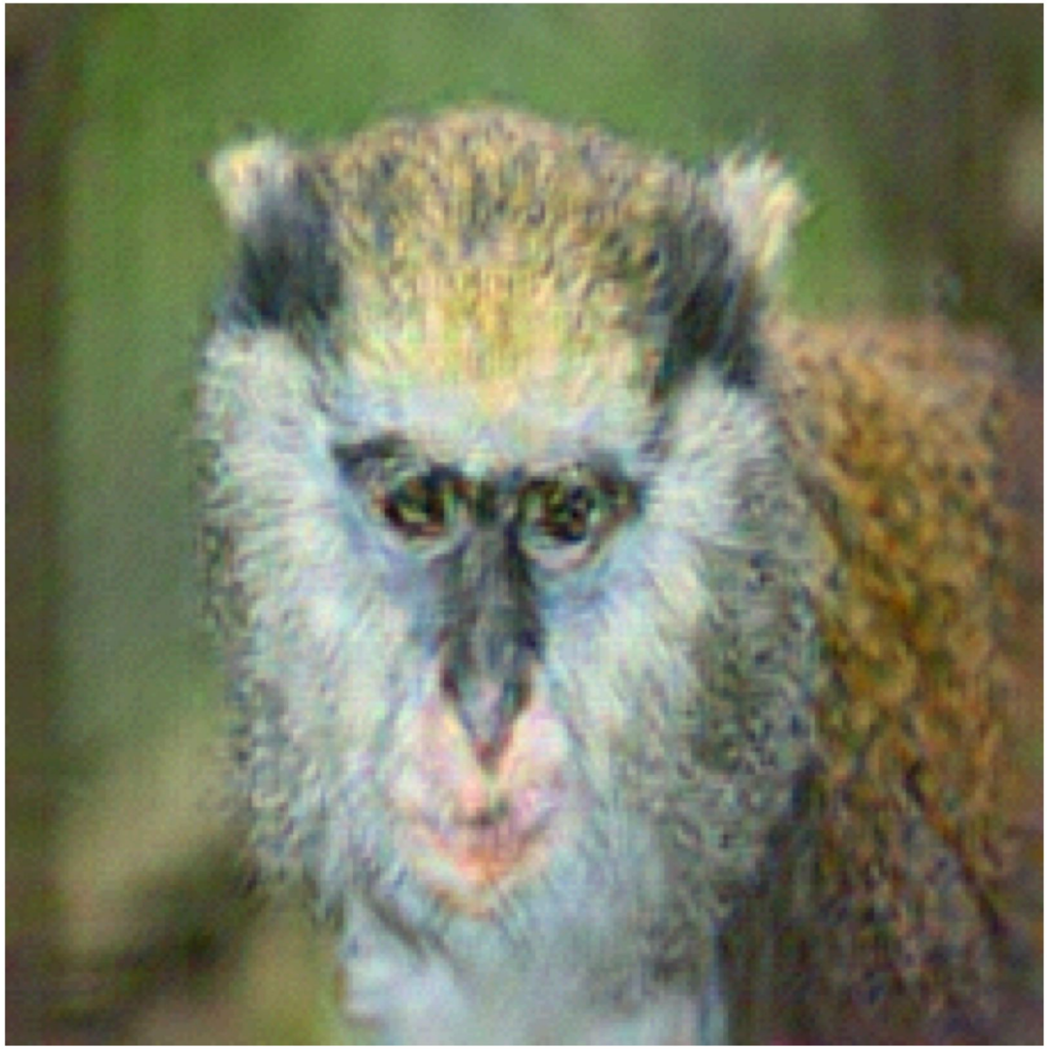
Original Image



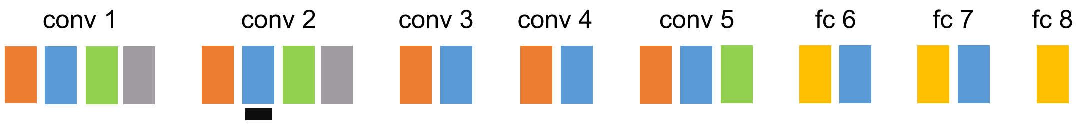
Inverting a Deep CNN



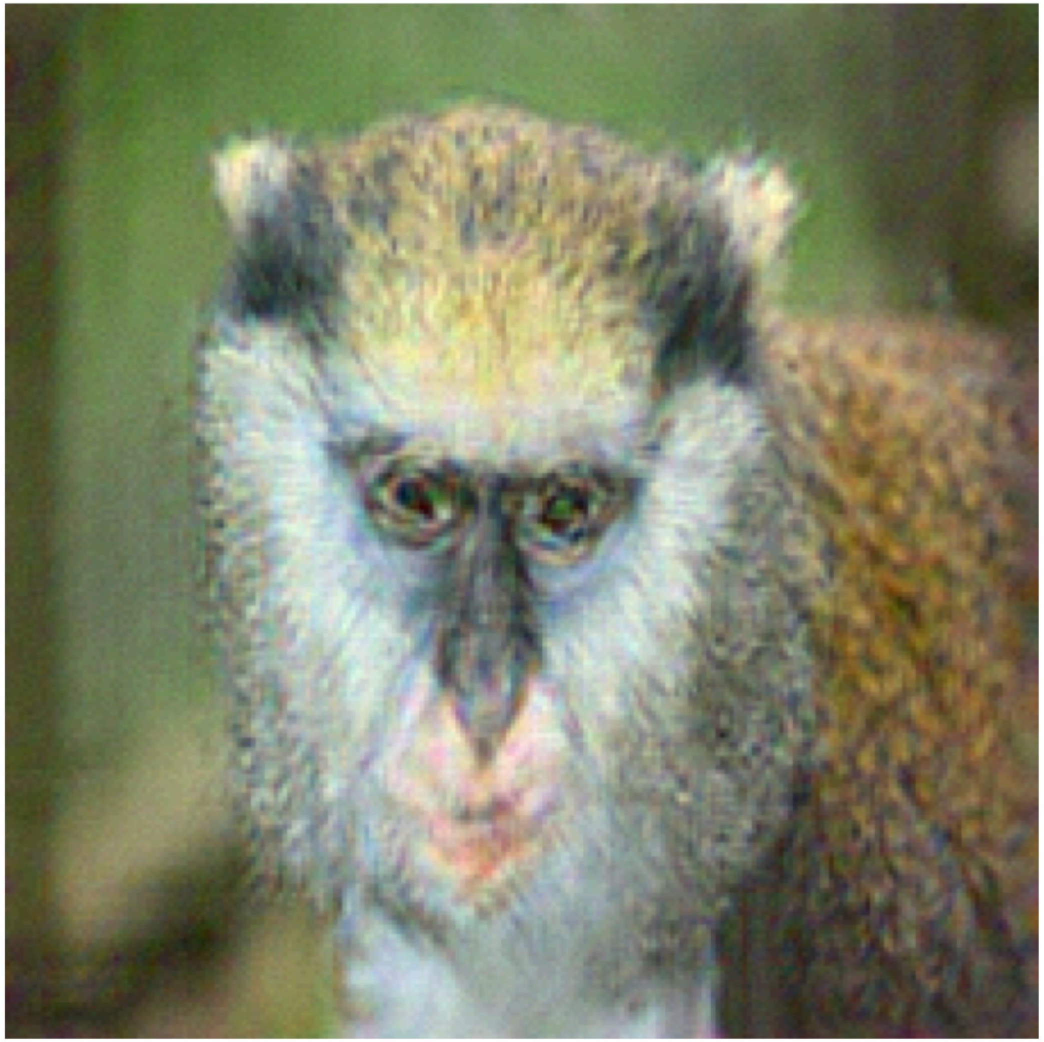
Original Image



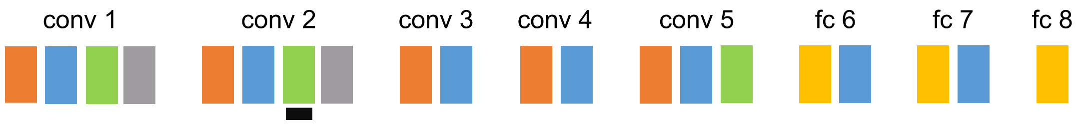
Inverting a Deep CNN



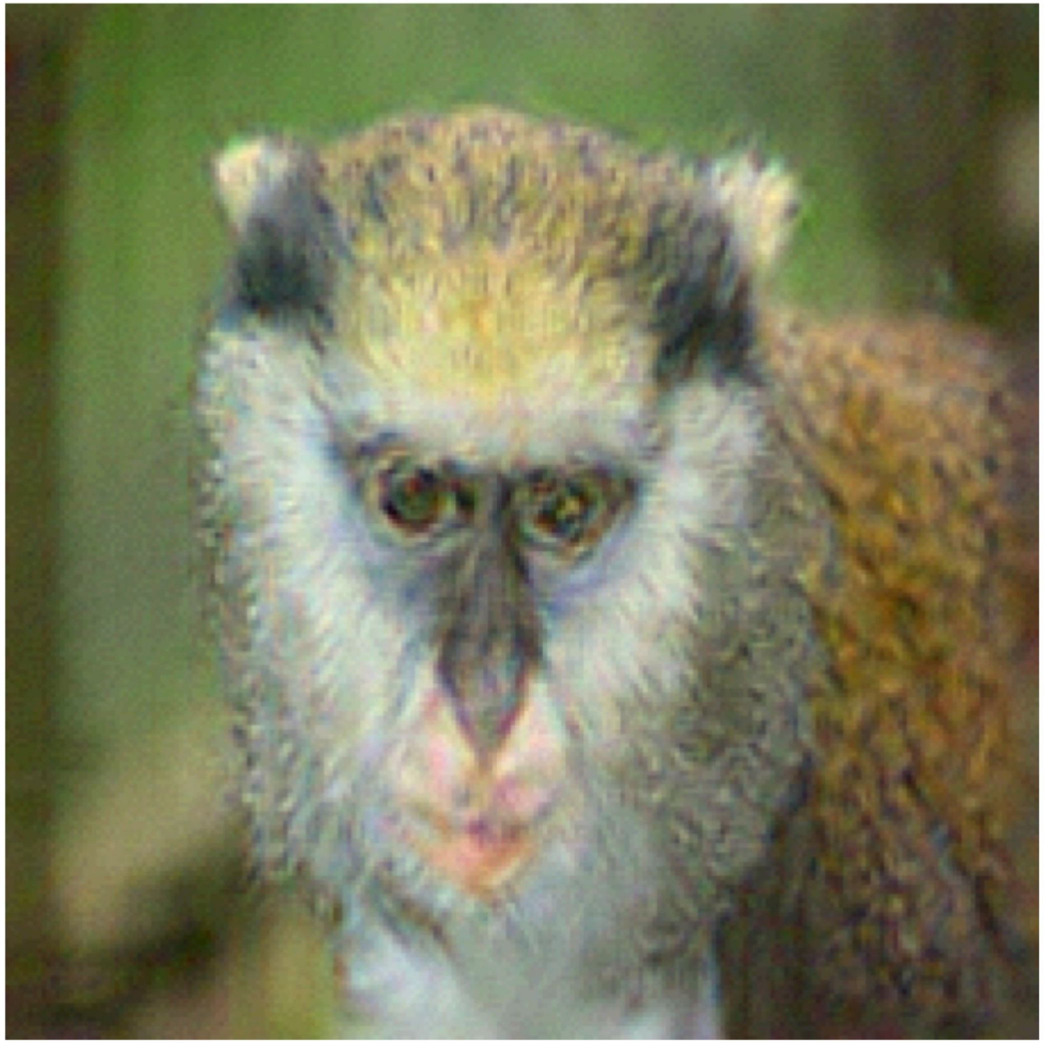
Original Image



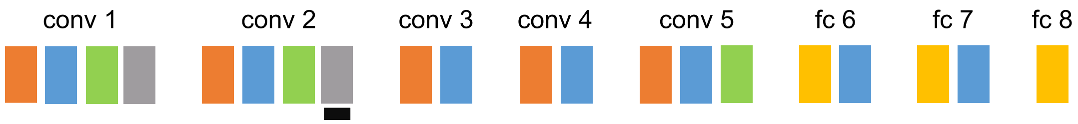
Inverting a Deep CNN

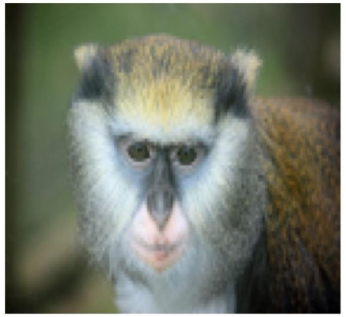


Original Image

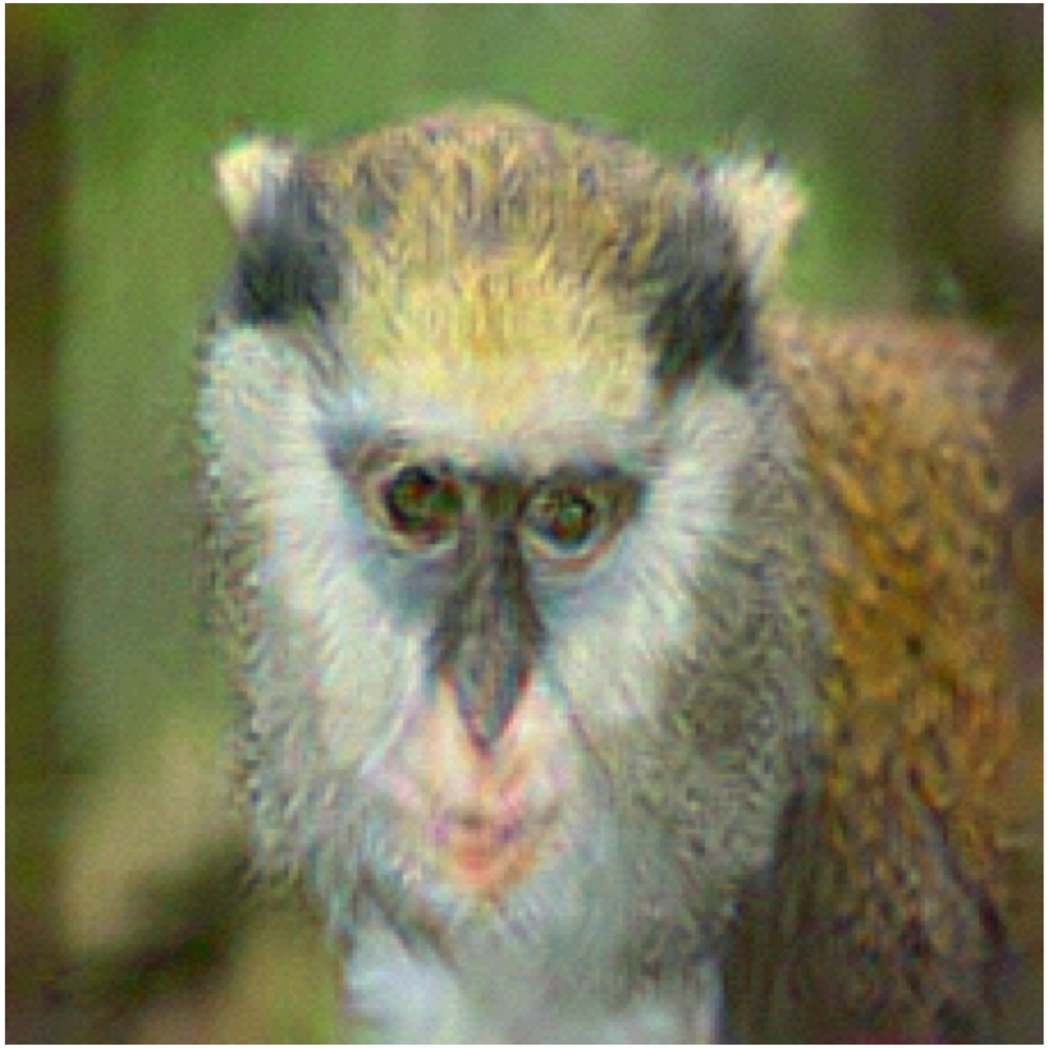


Inverting a Deep CNN

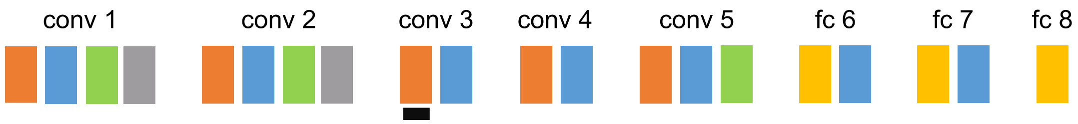


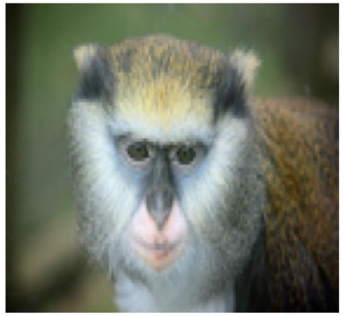


Original Image

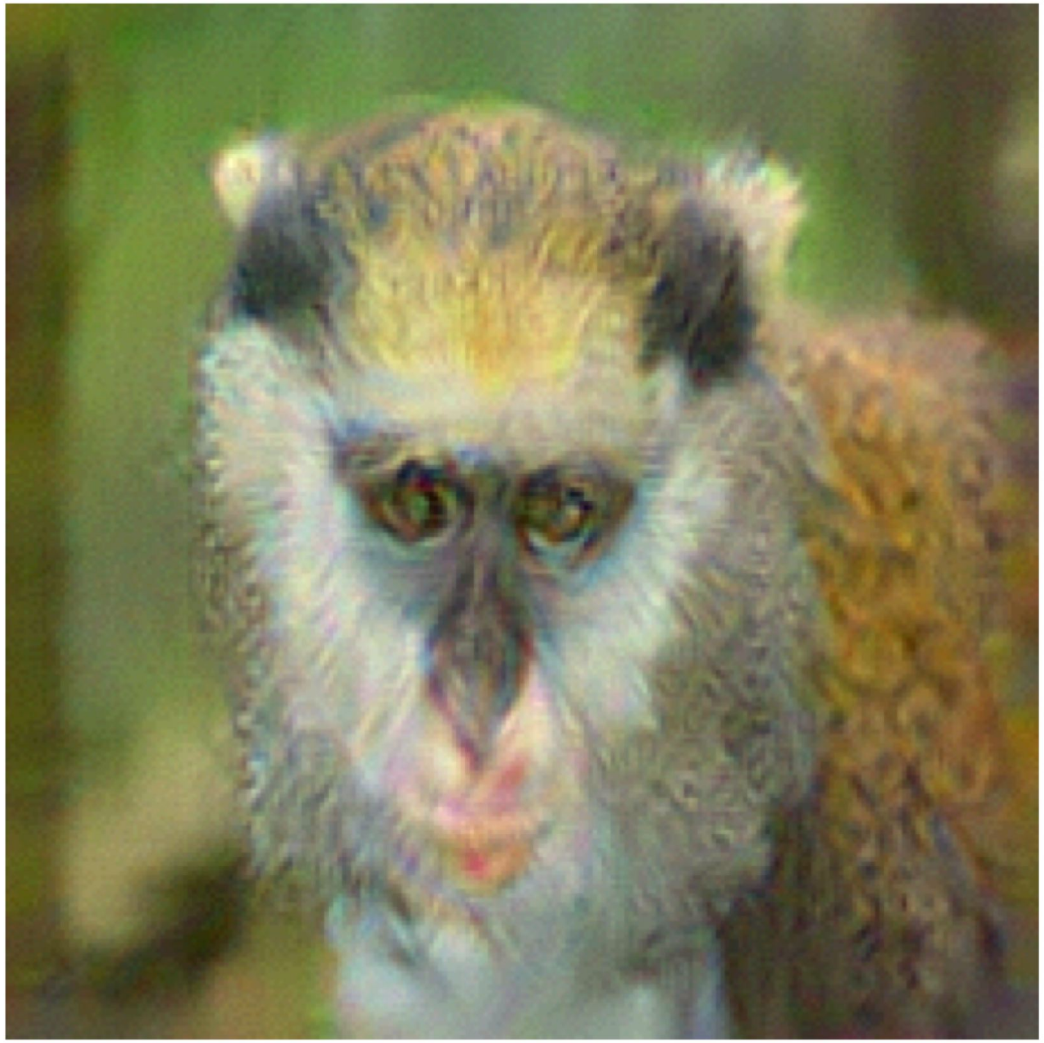


Inverting a Deep CNN

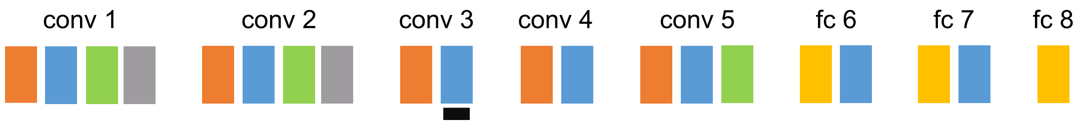




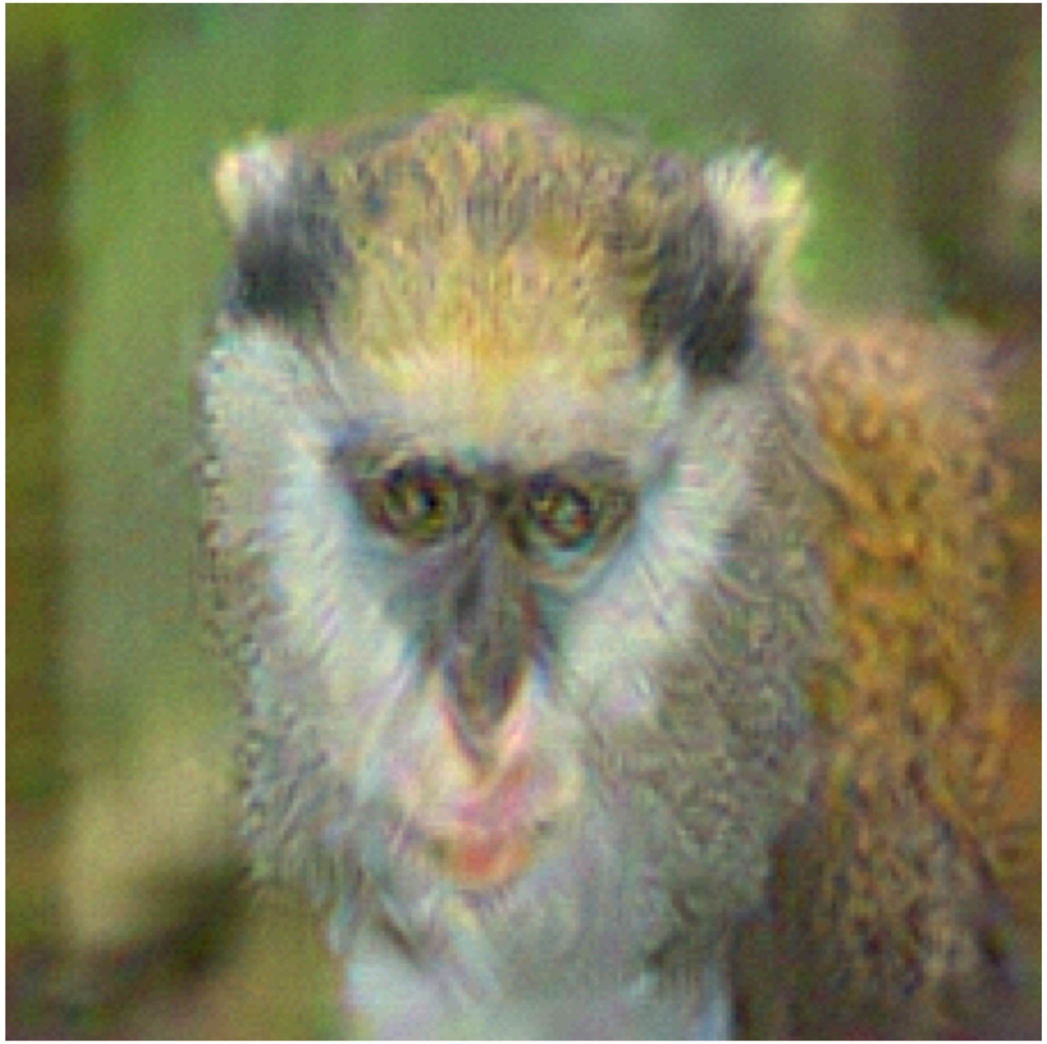
Original
Image



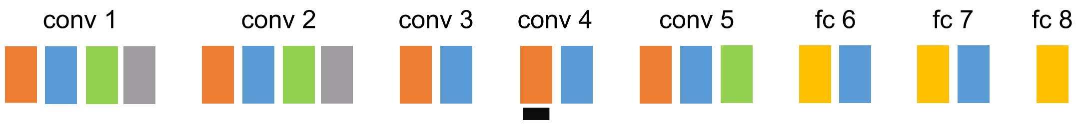
Inverting a Deep CNN



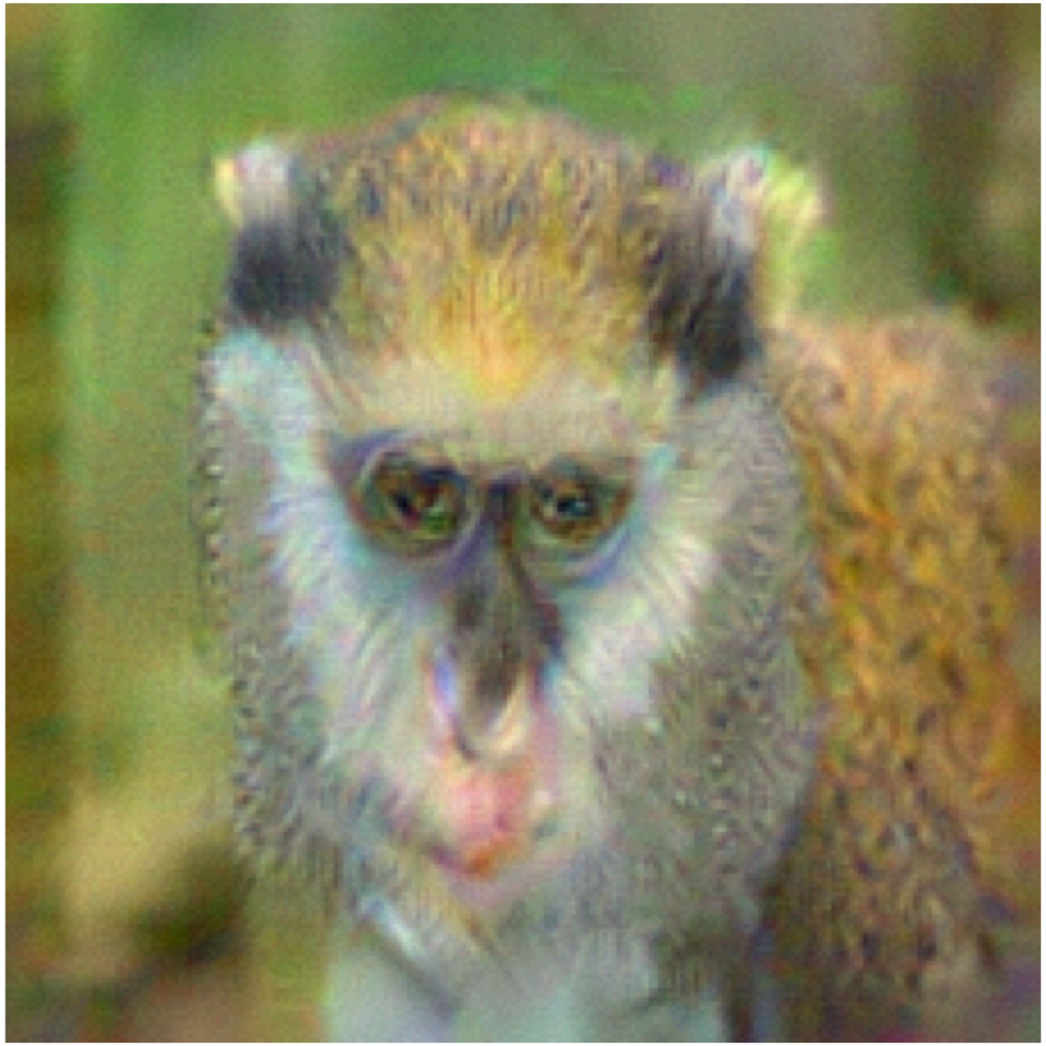
Original
Image



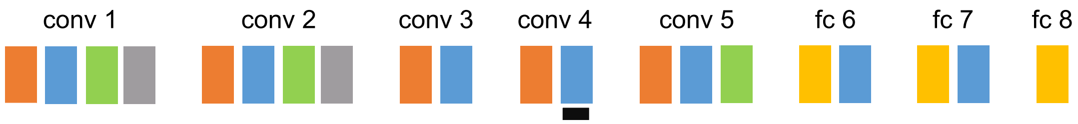
Inverting a Deep CNN



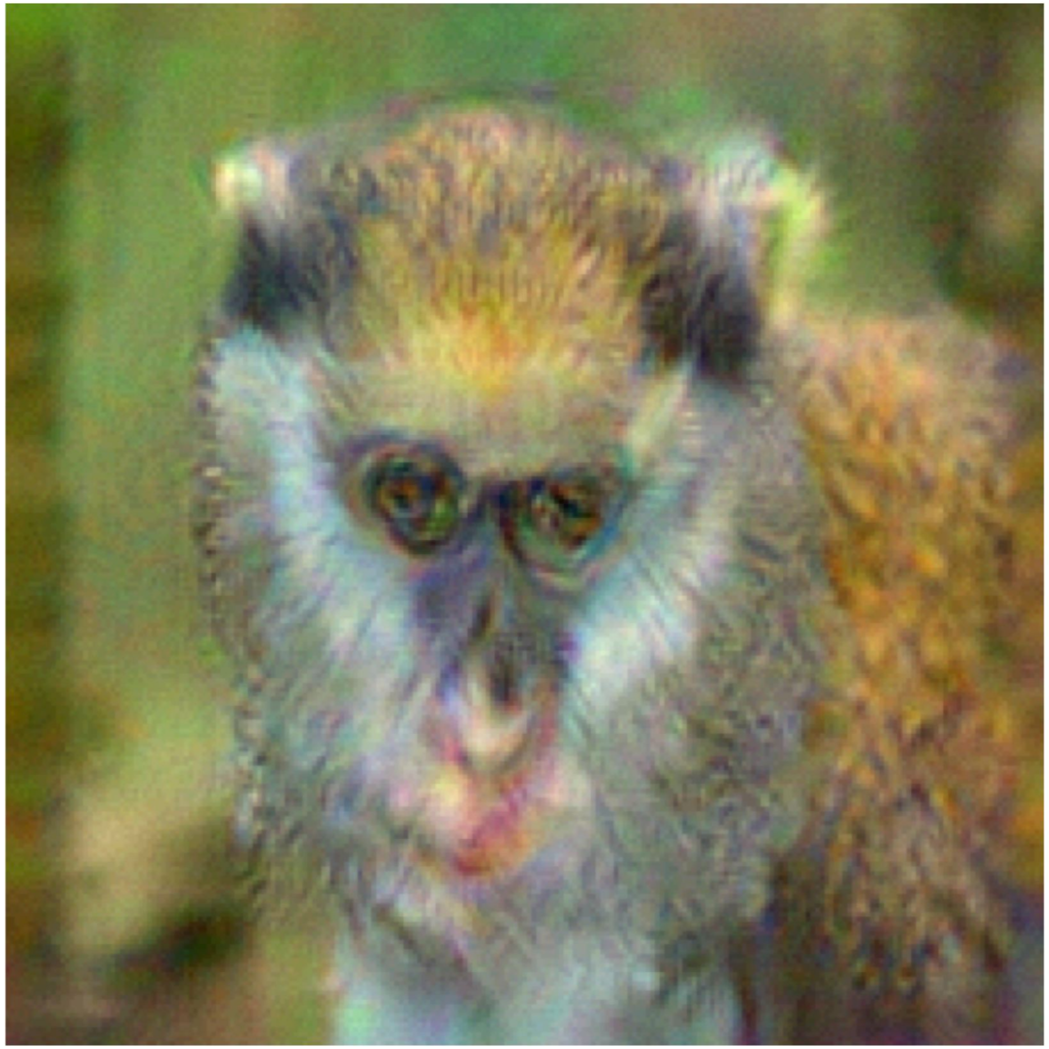
Original Image



Inverting a Deep CNN

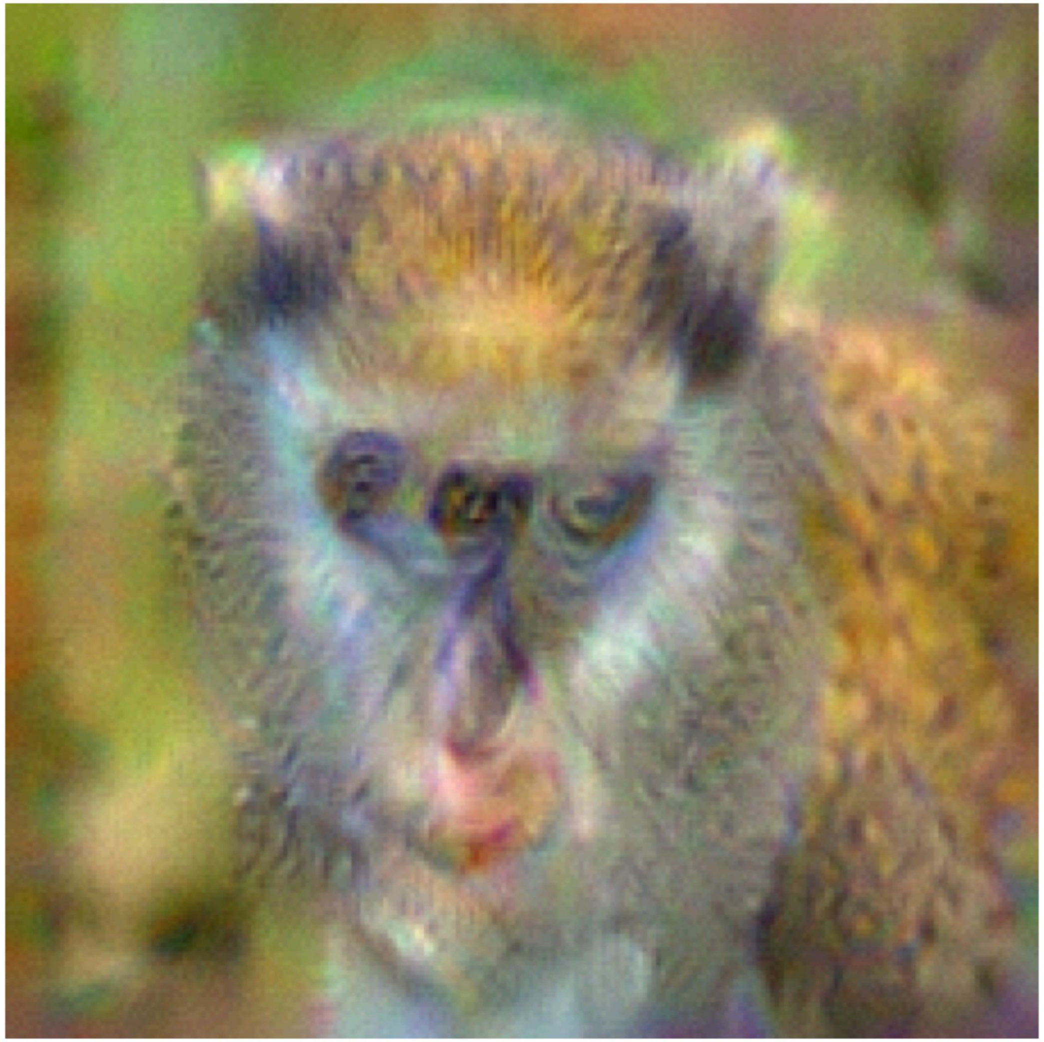


Original Image

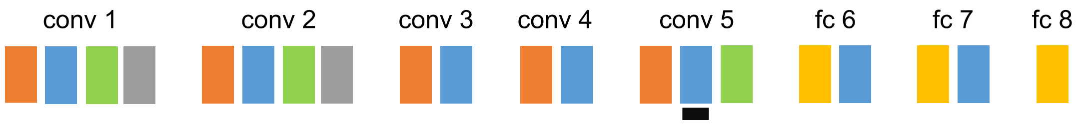


Inverting a Deep CNN

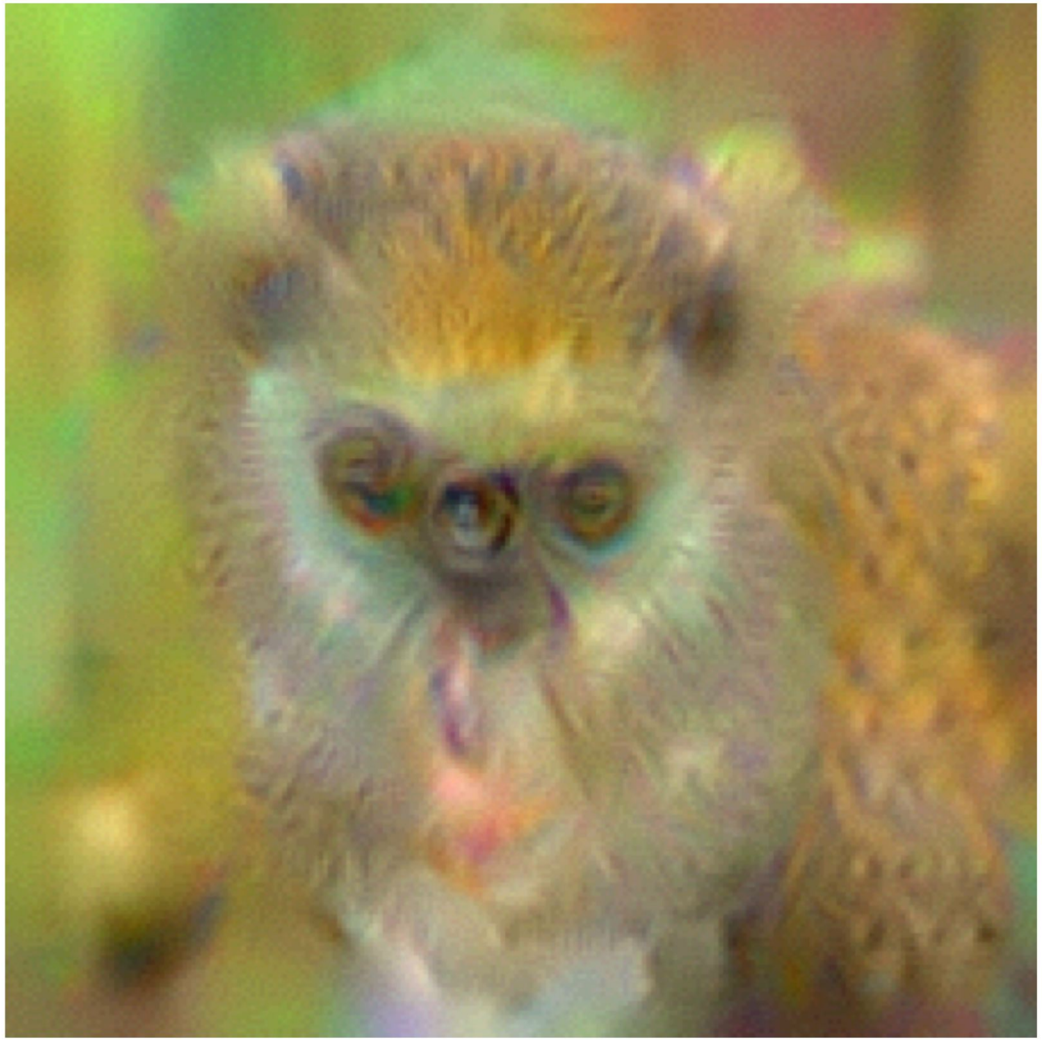

Original Image



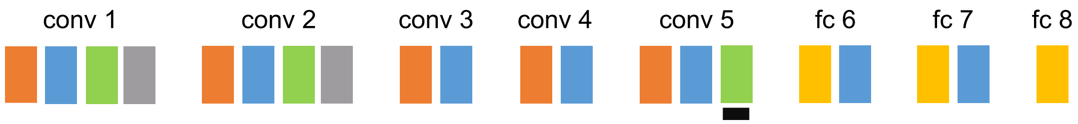
Inverting a Deep CNN



Original Image

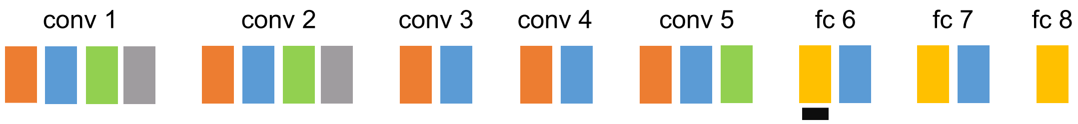


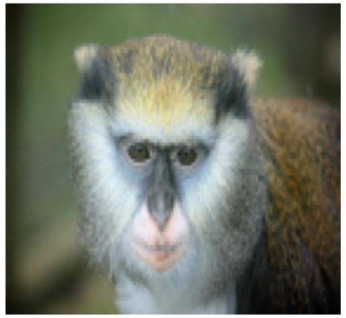
Inverting a Deep CNN



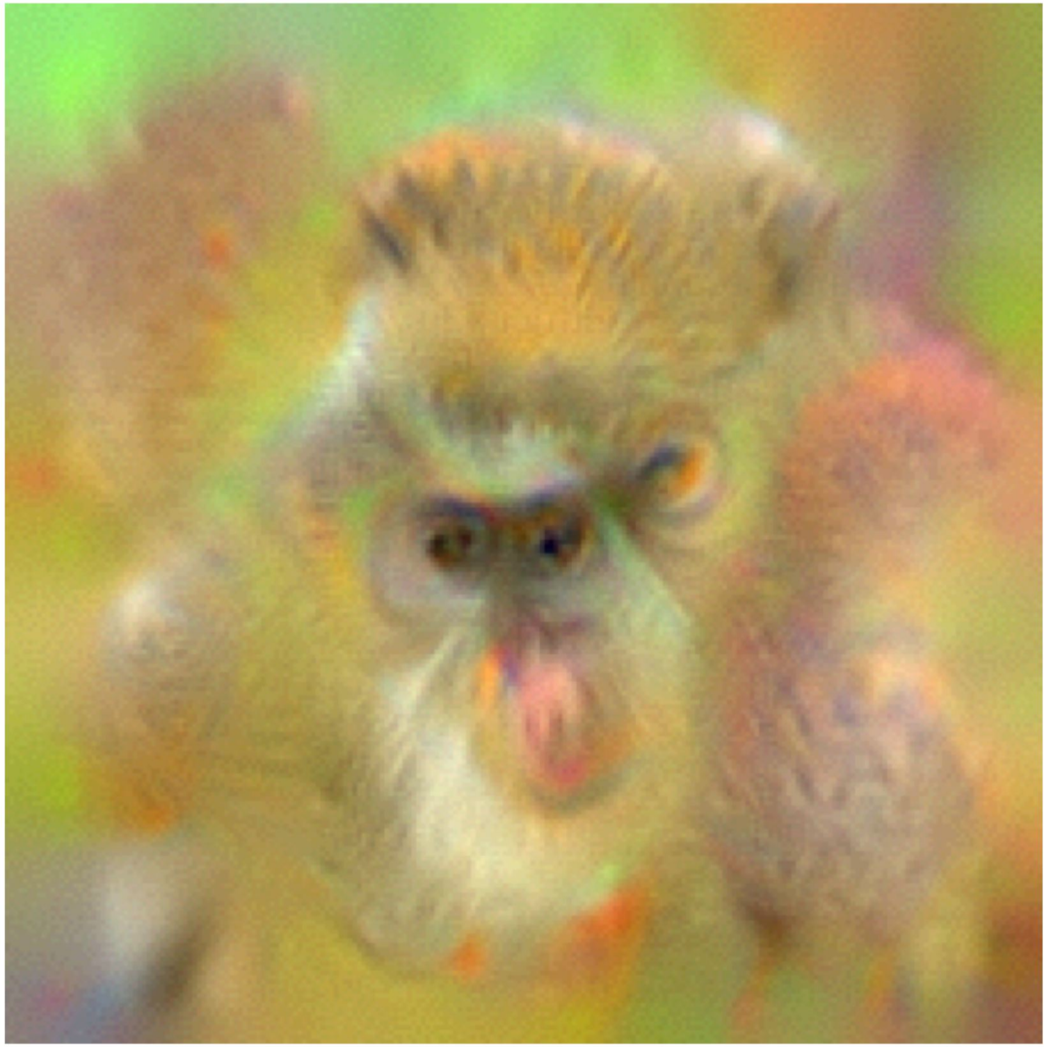
Original Image

Inverting a Deep CNN

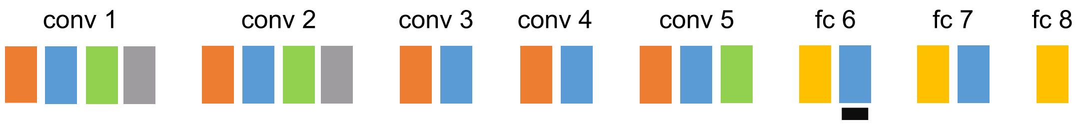




Original Image

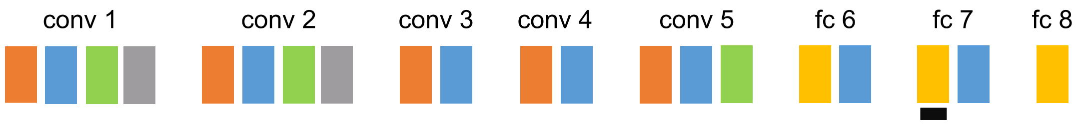


Inverting a Deep CNN

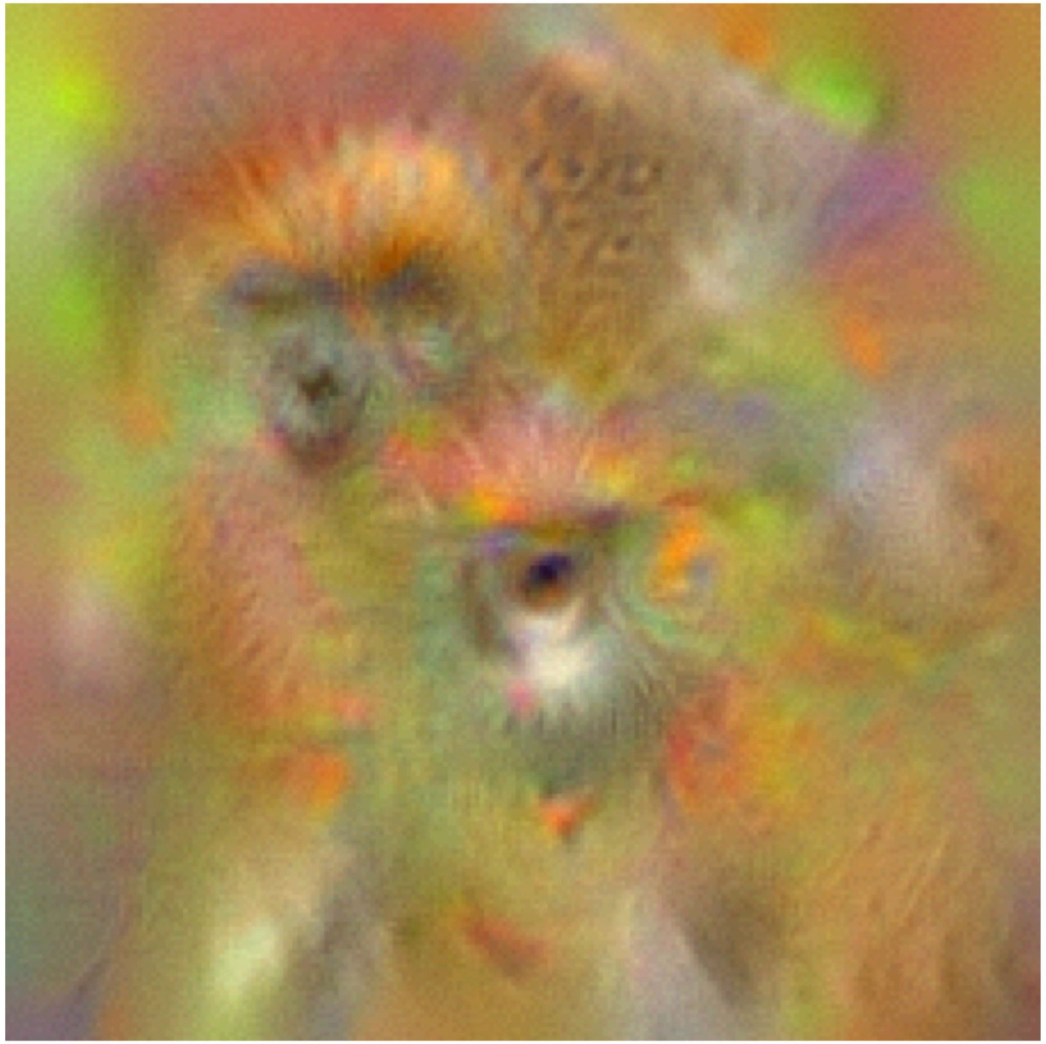


Original
Image

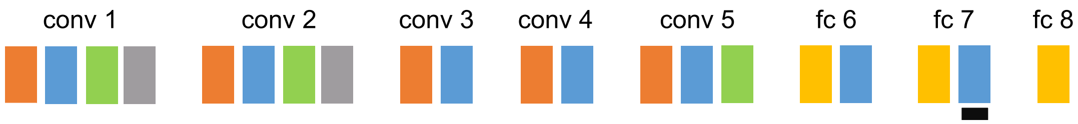
Inverting a Deep CNN



Original Image

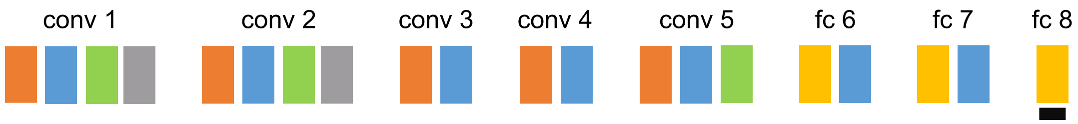


Inverting a Deep CNN



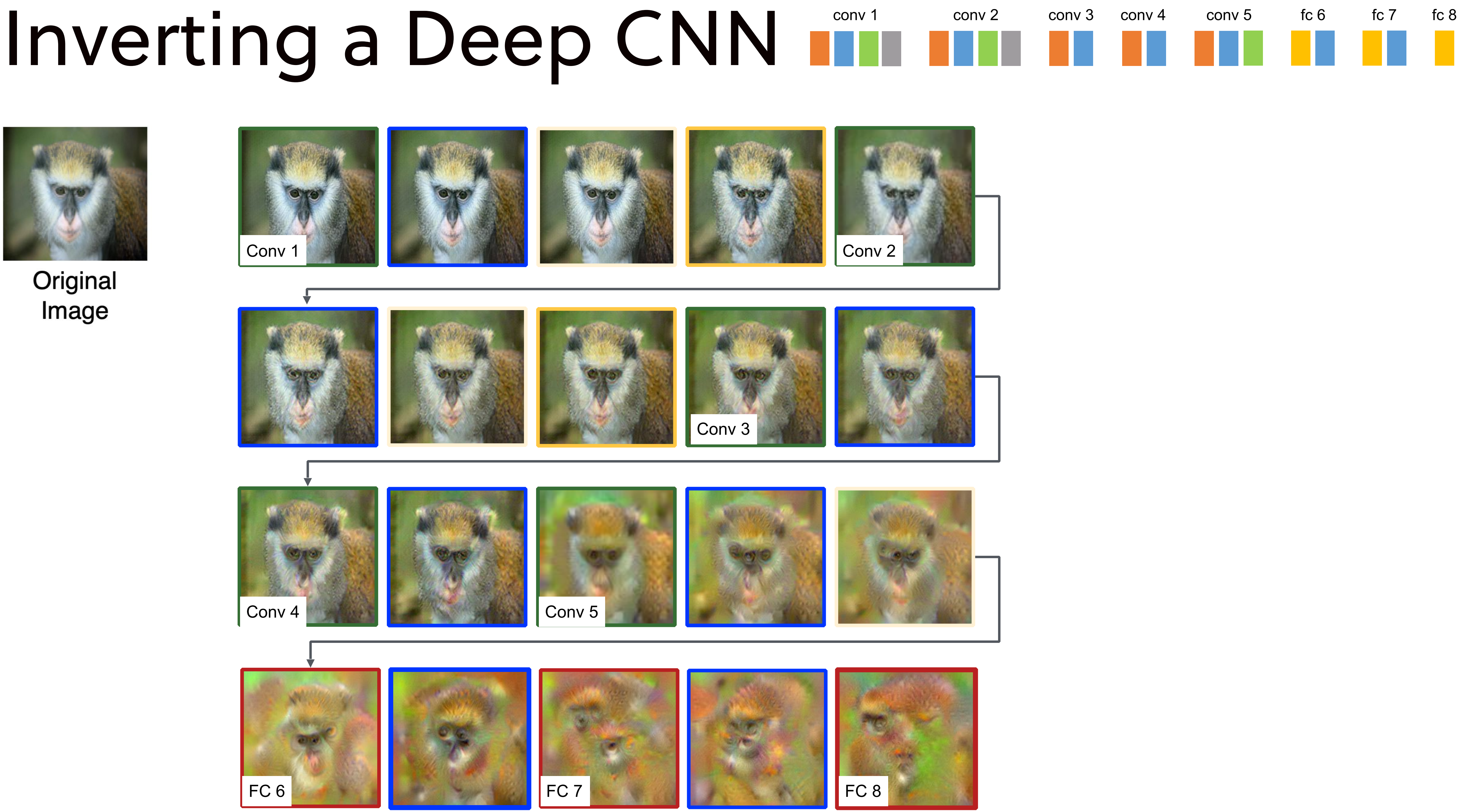
Original Image

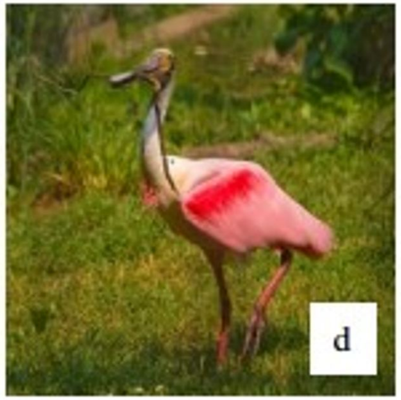
Inverting a Deep CNN



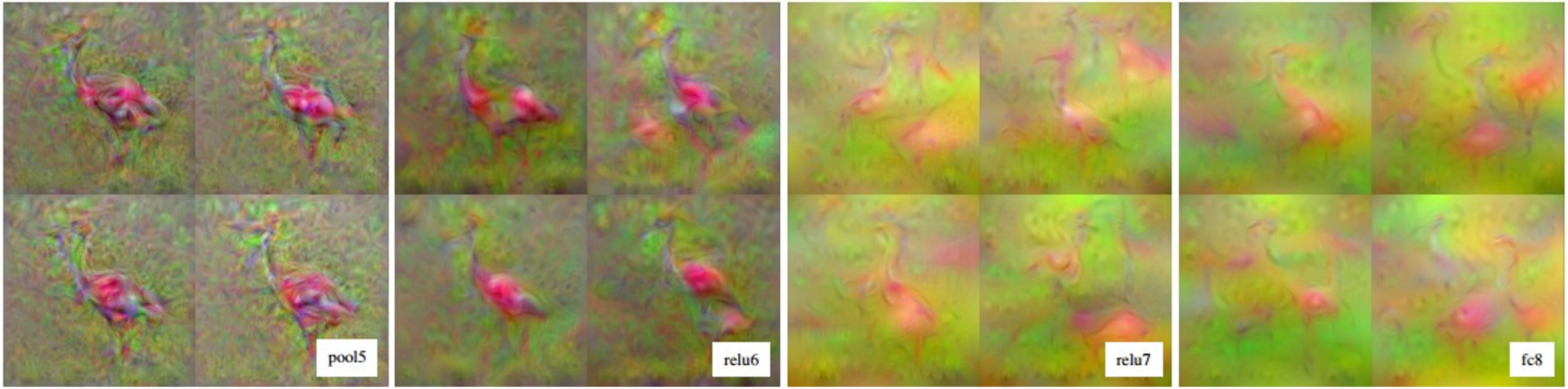
Original Image

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:

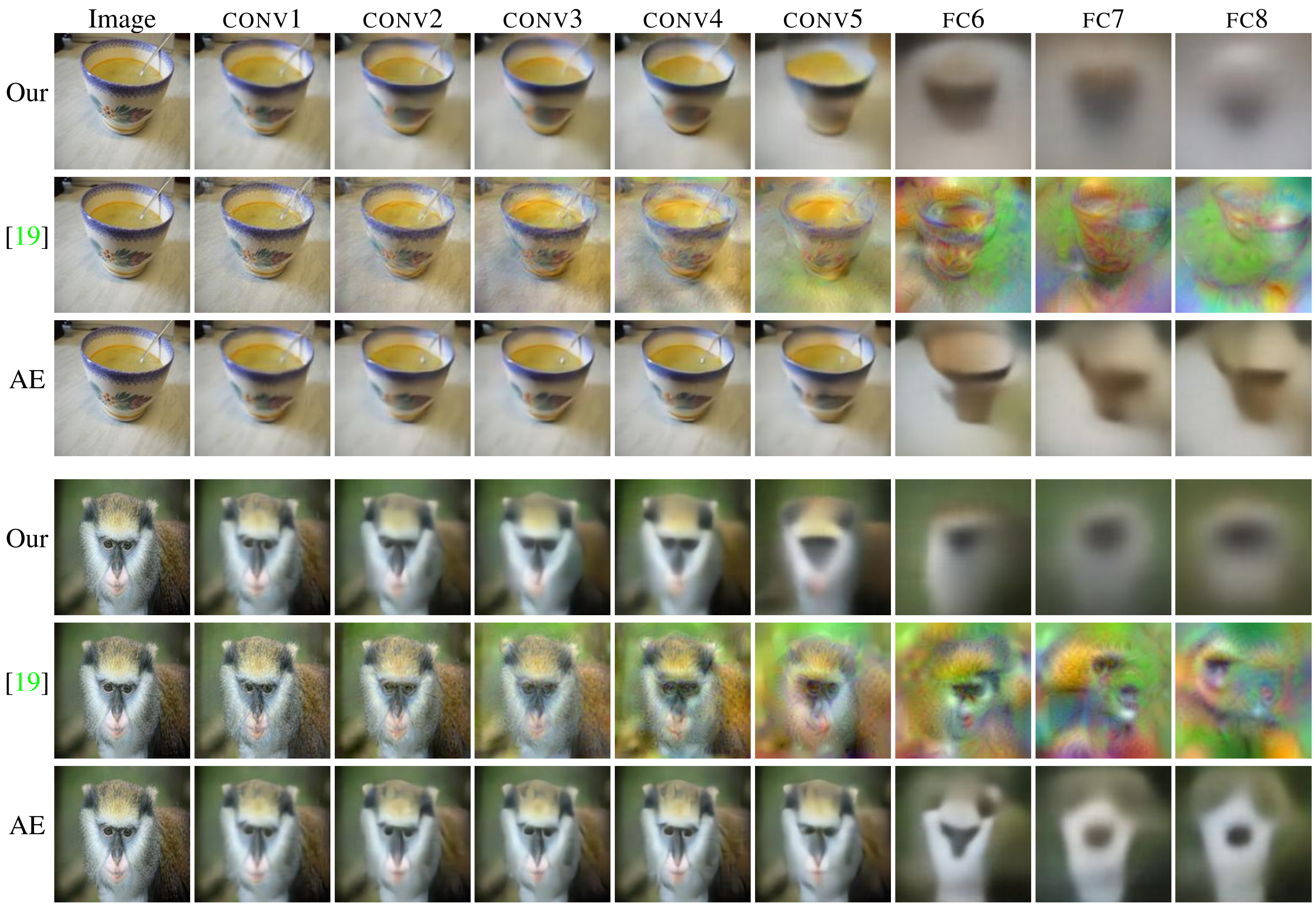
$$\mathbb{E}_{\mathbf{x},\boldsymbol{\phi}}\,||\mathbf{x} - f(\boldsymbol{\phi})||^2$$

Pre-image as the conditional expectation:

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

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

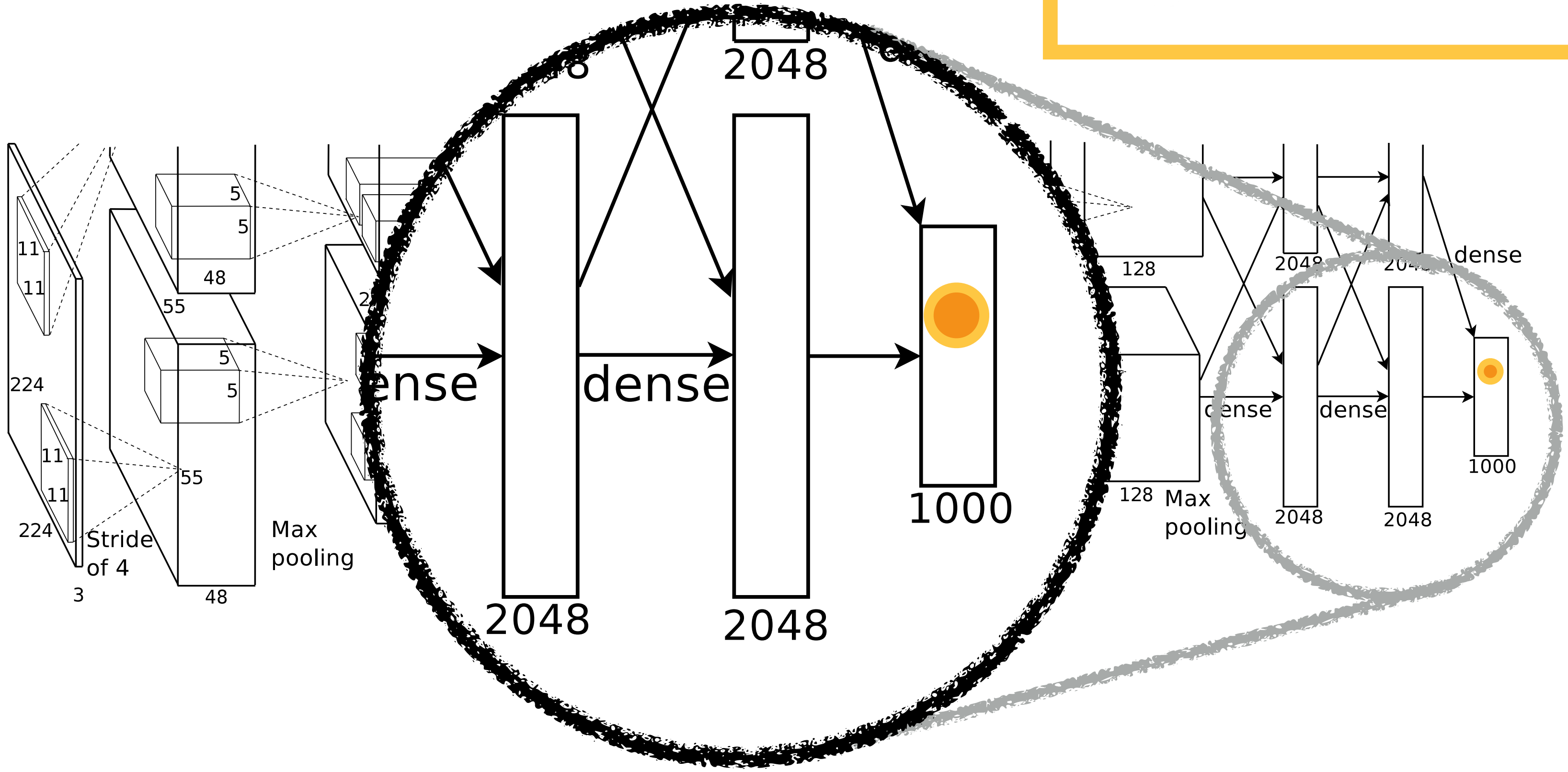
$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_i ||\mathbf{x}_i - f(\boldsymbol{\phi}_i, \mathbf{w})||_2^2.$$

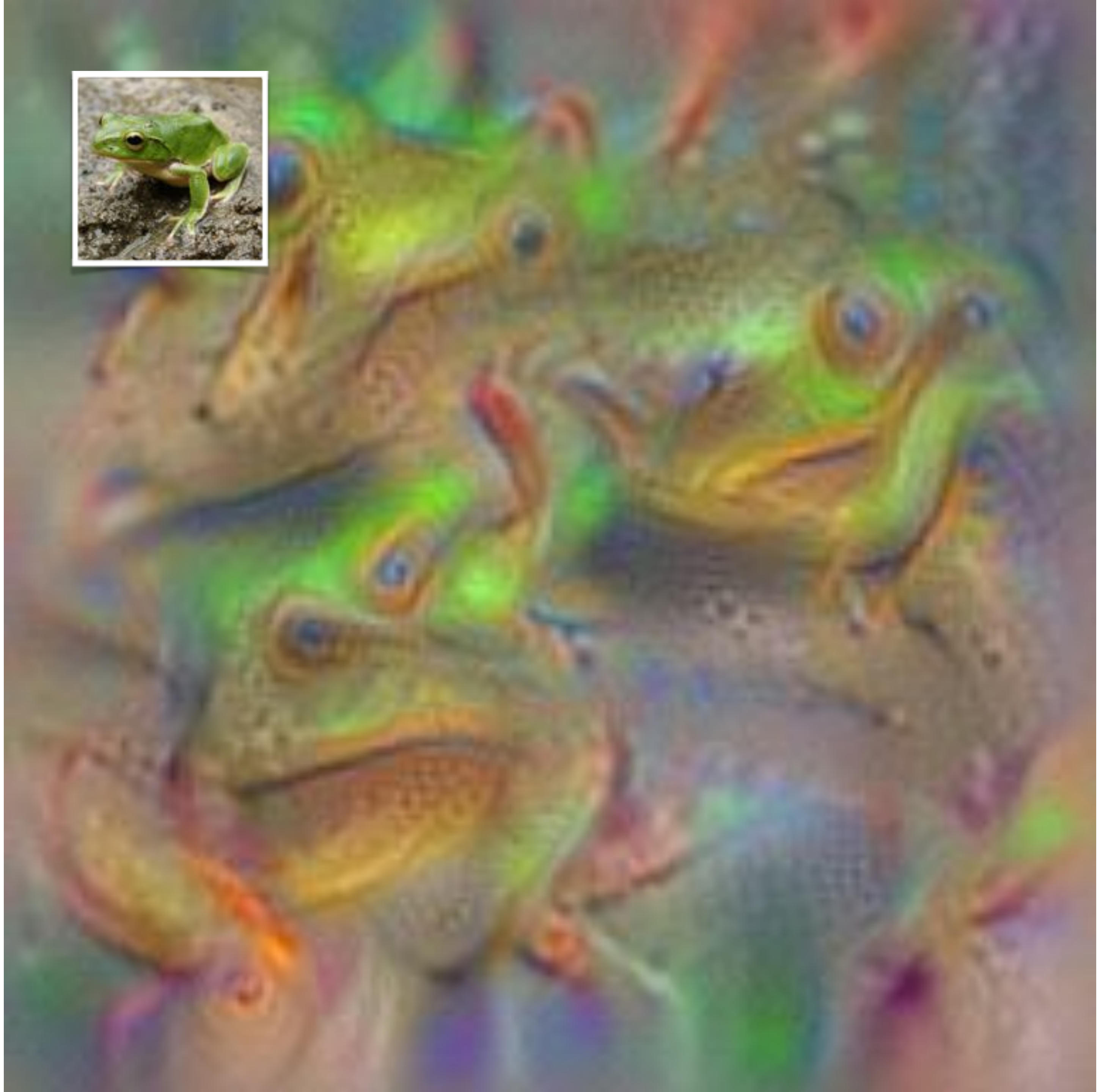


Activation Maximization

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

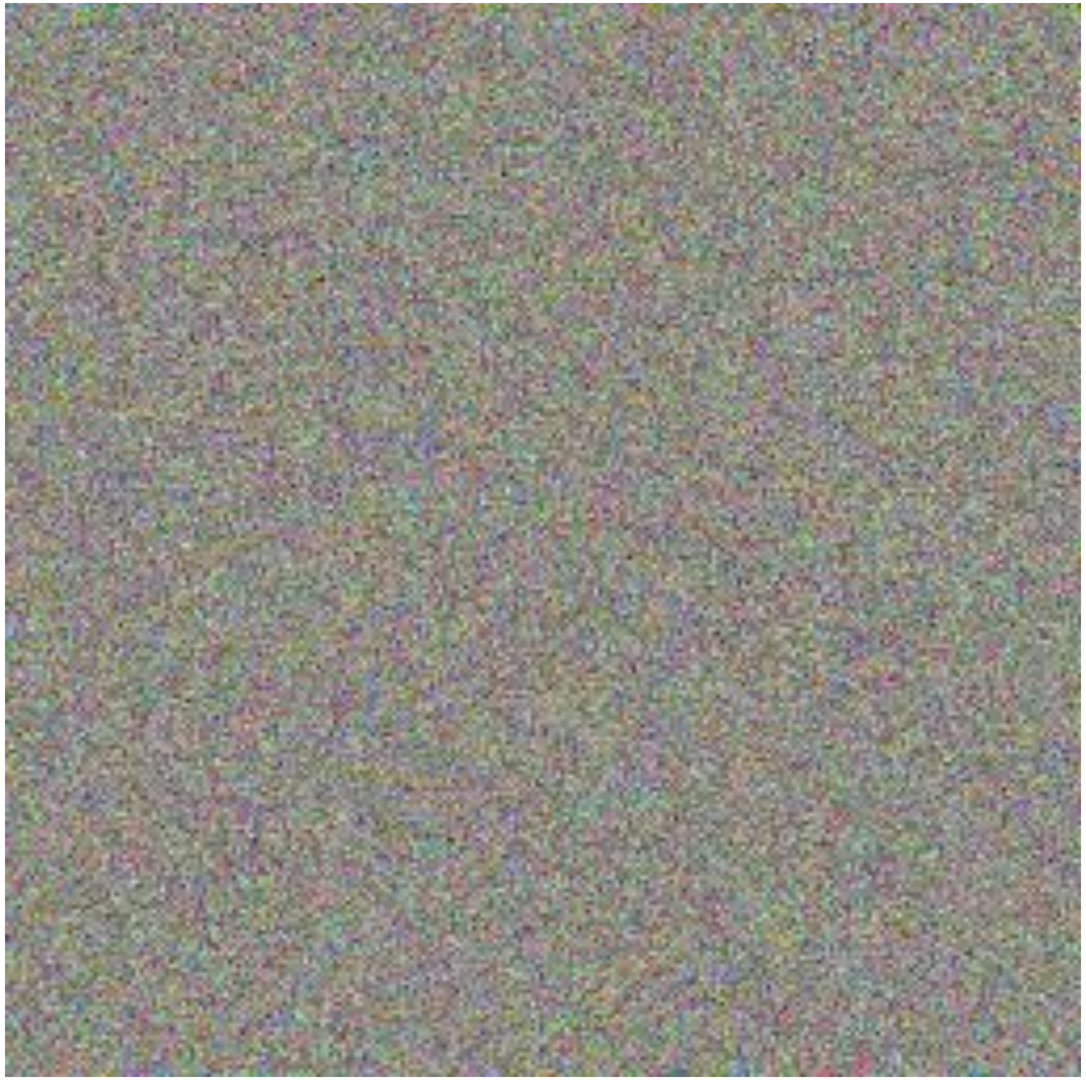
$$\min_{\mathbf{x}} -\langle \mathbf{e}_k, \Phi(\mathbf{x}) \rangle + R_{TV}(\mathbf{x}) + R_\alpha(\mathbf{x})$$



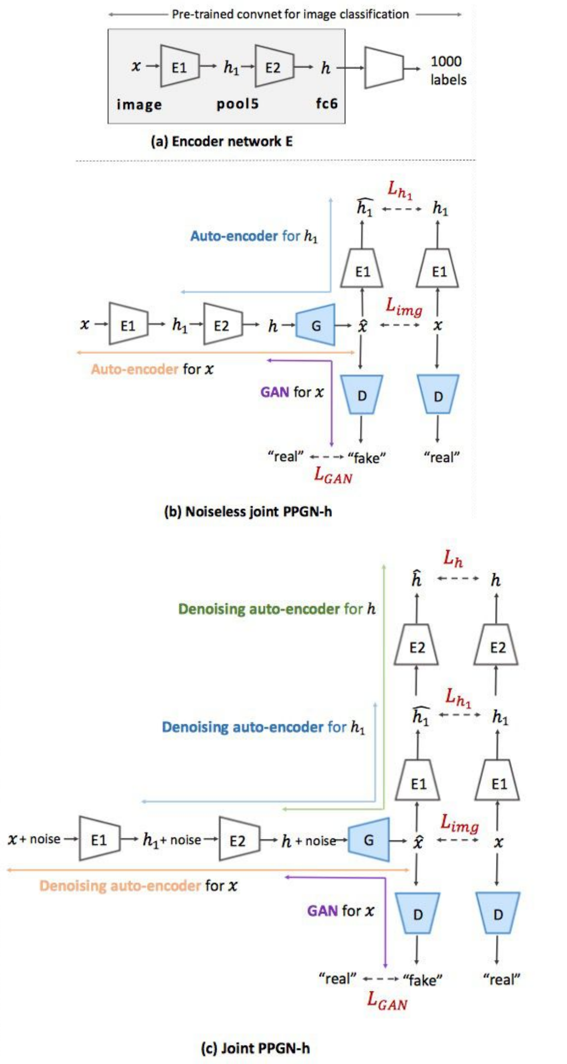


Recall Mahendran and Vedaldi's pre-images: The starting point is white noise

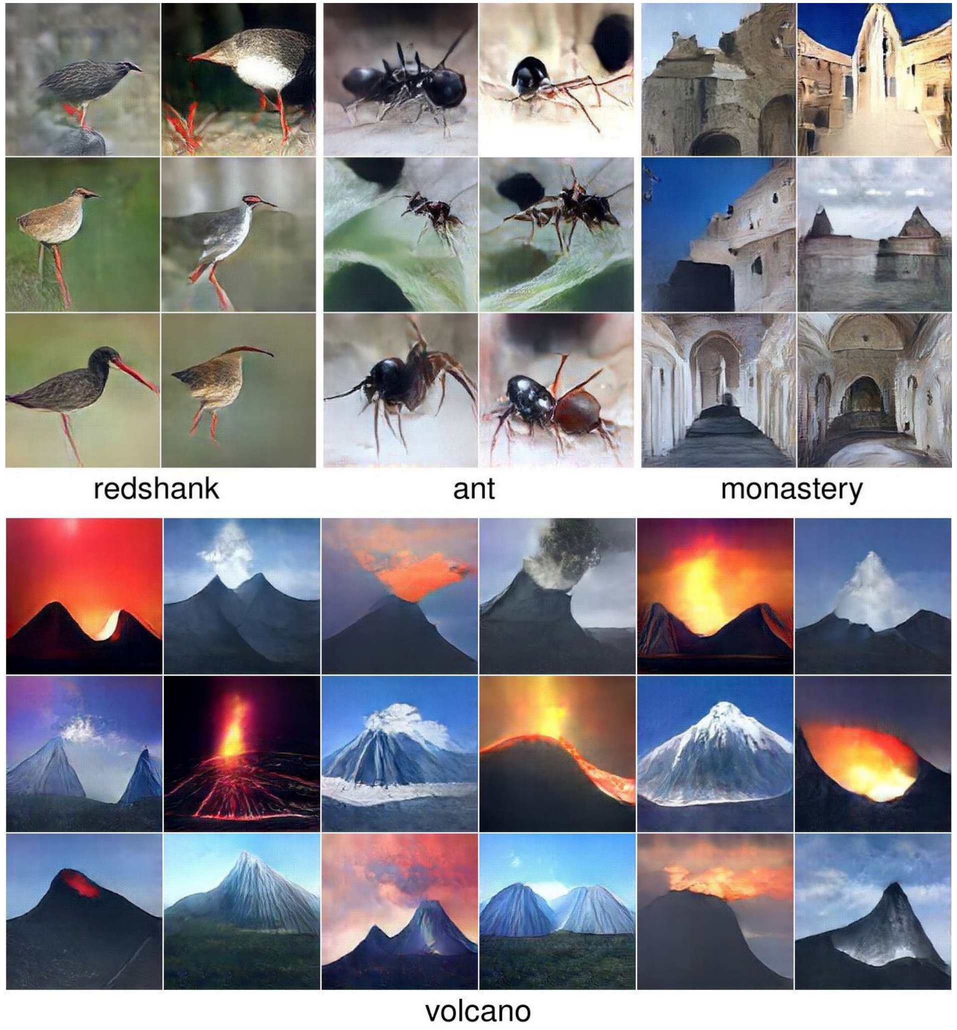
- Not an image!



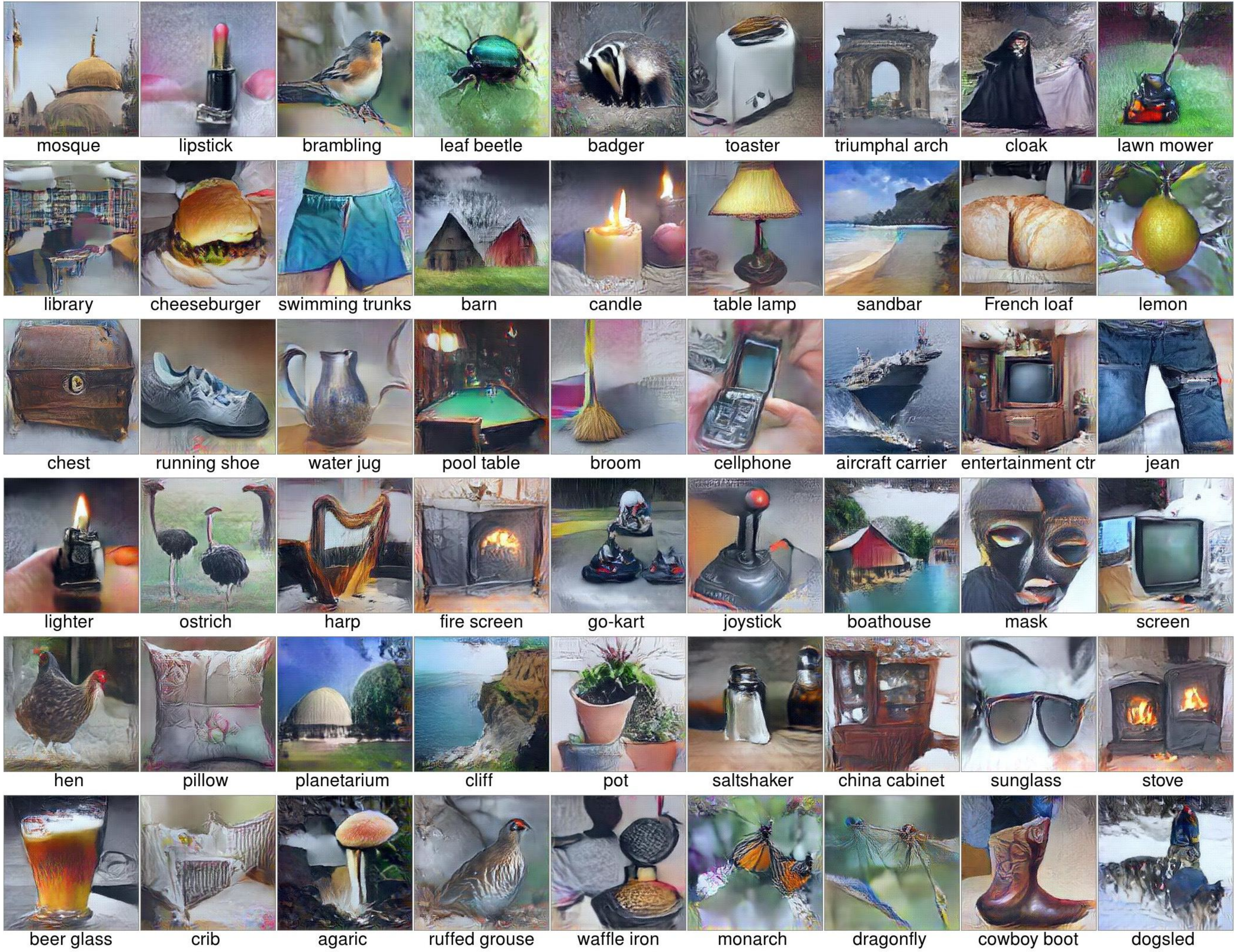
Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space
[Nguyen et al. 2016]

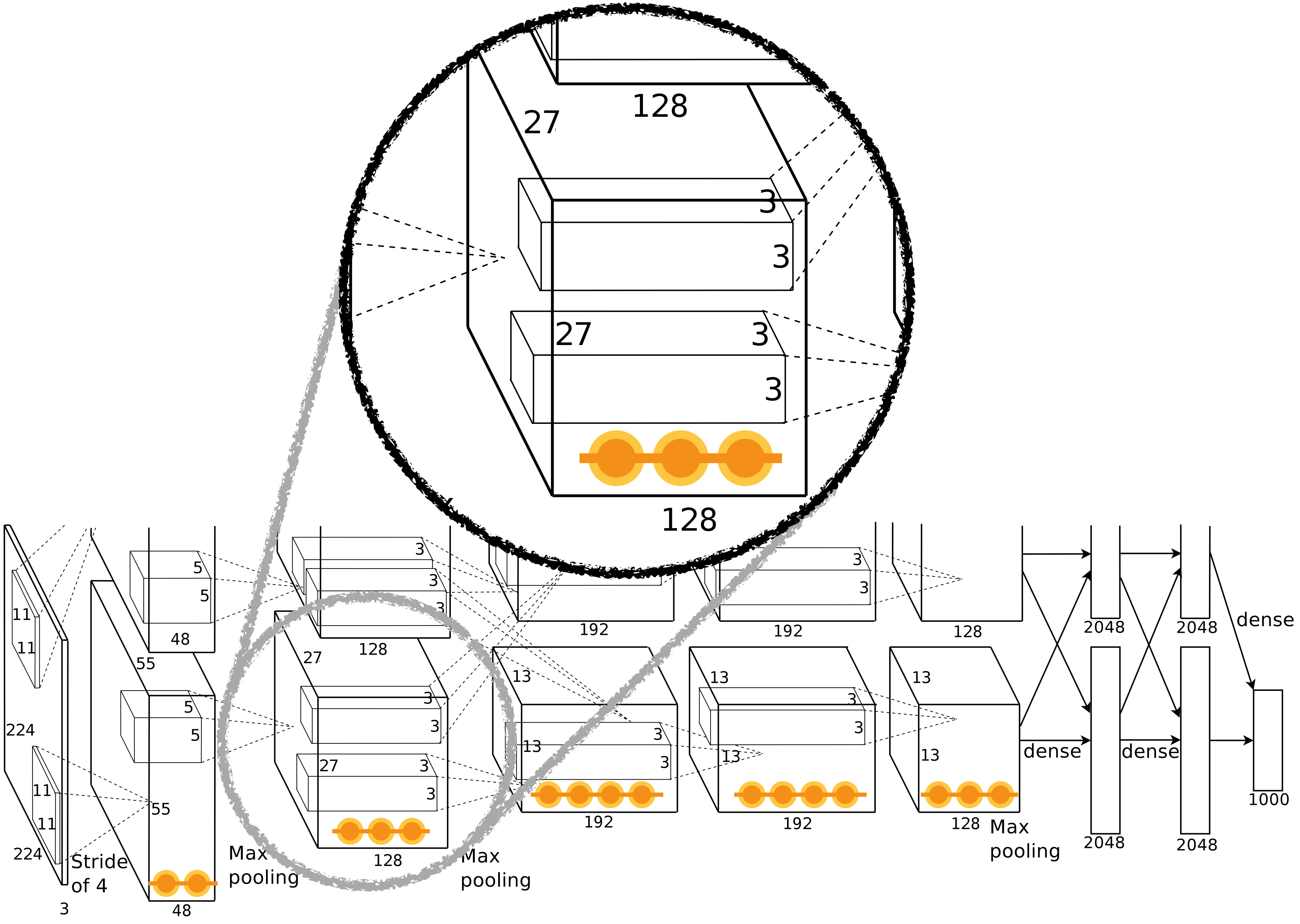


Employs auto-encoder and generative adversarial network components



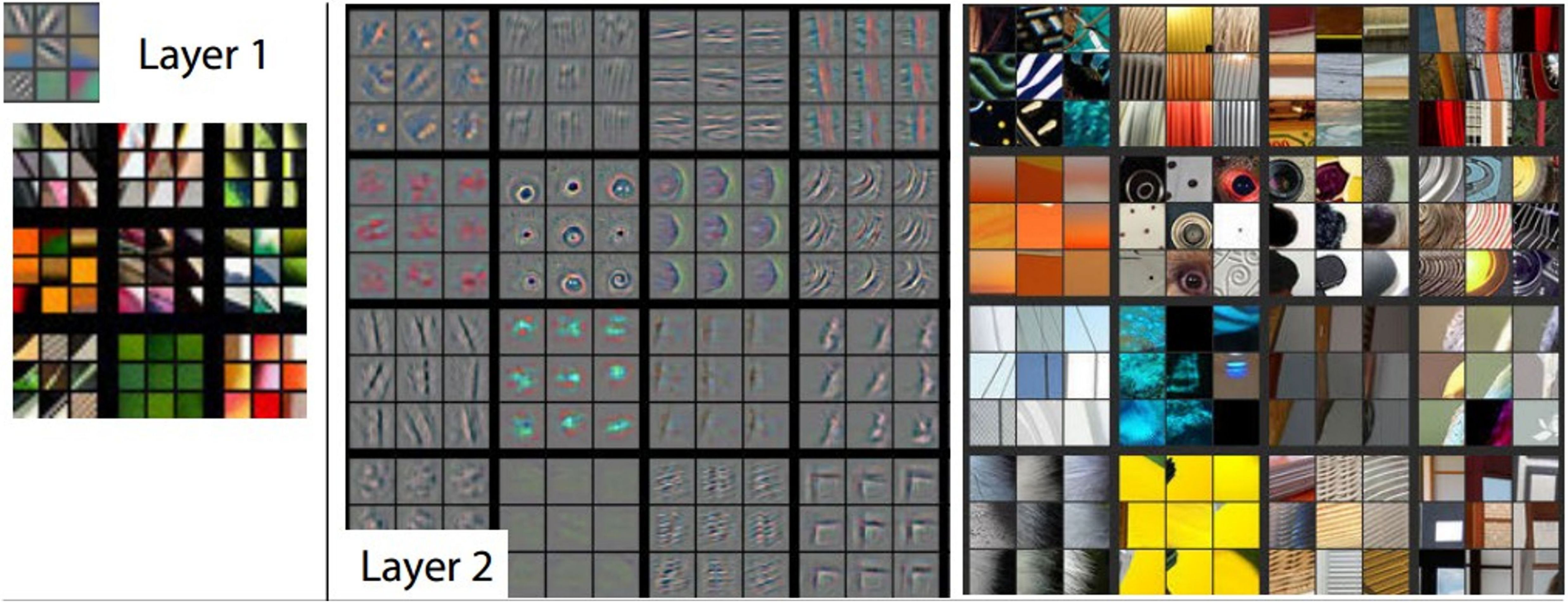
Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space [Nguyen et al. 2016]





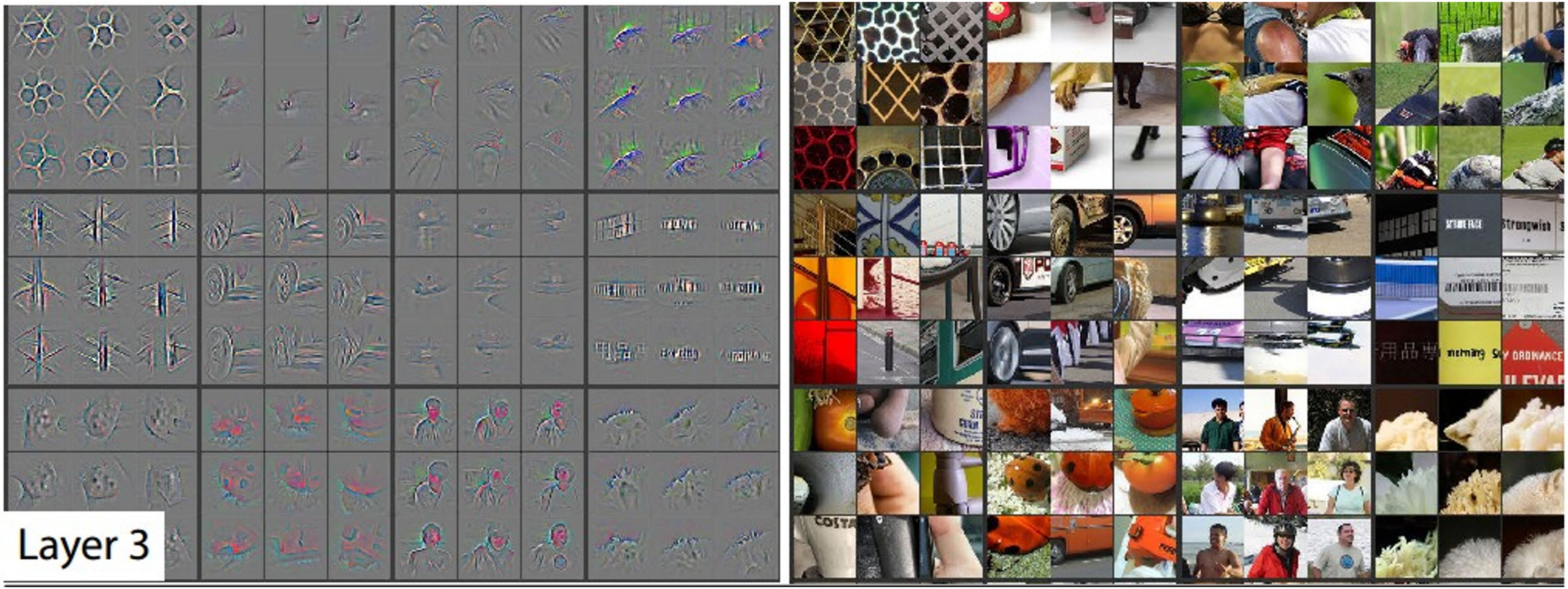
128
27
3
3
27
3
3
128
11
11
224
224
11
11
Stride
of 4
3
5
5
48
55
5
5
55
48
Max
pooling
3
3
3
128
27
3
3
27
3
3
128
Max
pooling
192
13
13
3
3
192
192
13
3
3
13
3
3
192
128
13
13
128
Max
pooling
dense
dense
dense
2048
2048
2048
2048
1000

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



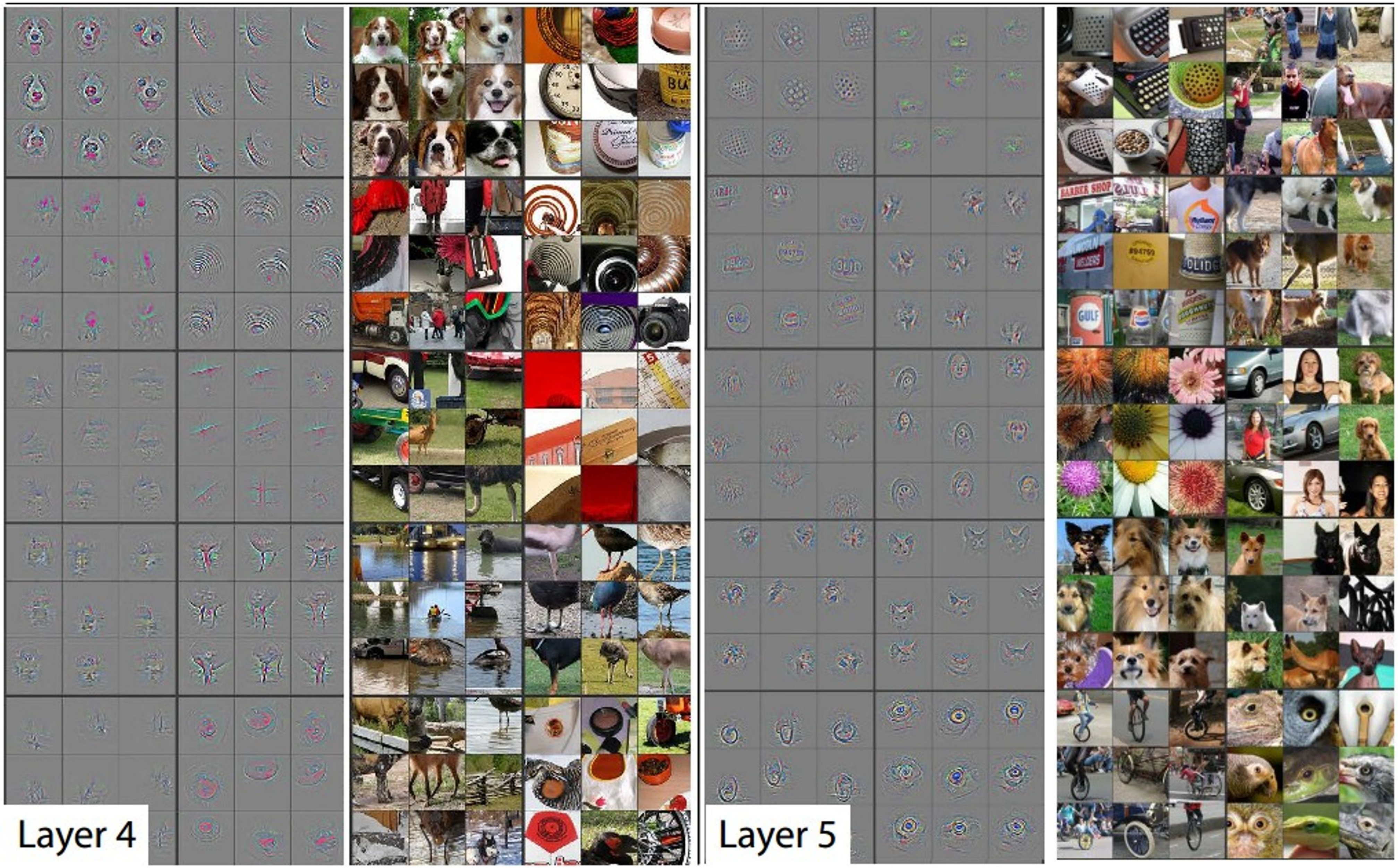
Visualizing and Understanding Convolutional Networks
[Zeiler & Fergus, 2013]

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



Visualizing and Understanding Convolutional Networks
[Zeiler & Fergus, 2013]

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

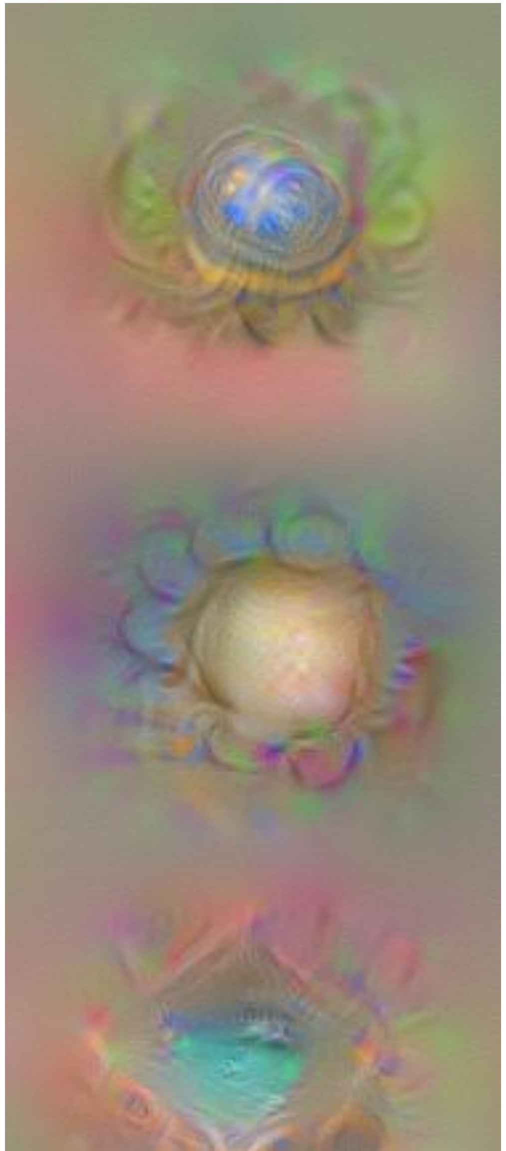


Visualizing and Understanding Convolutional Networks
[Zeiler & Fergus, 2013]

Network Comparison

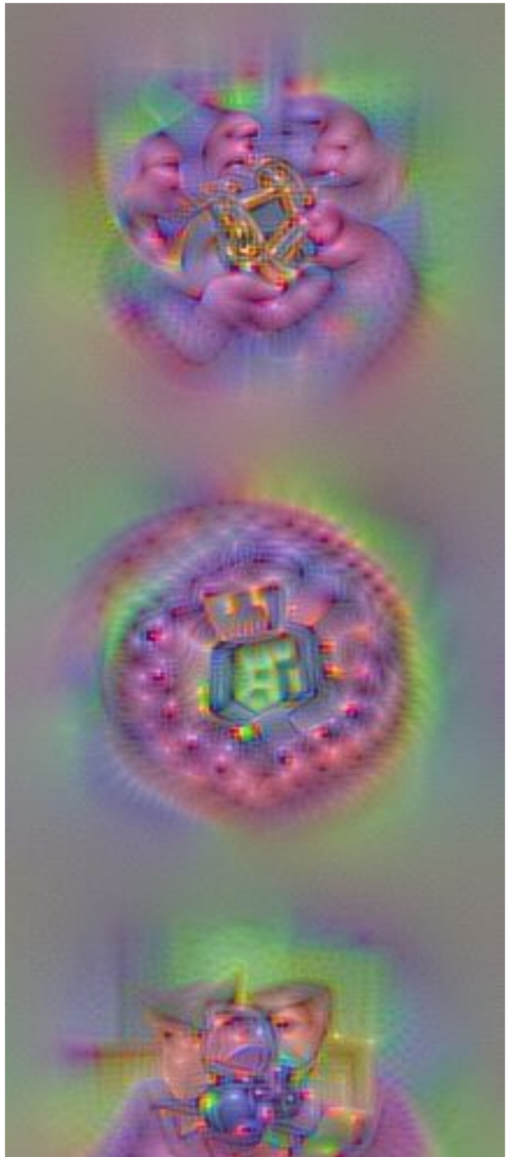
“conv5” features

AlexNet



VGG-M

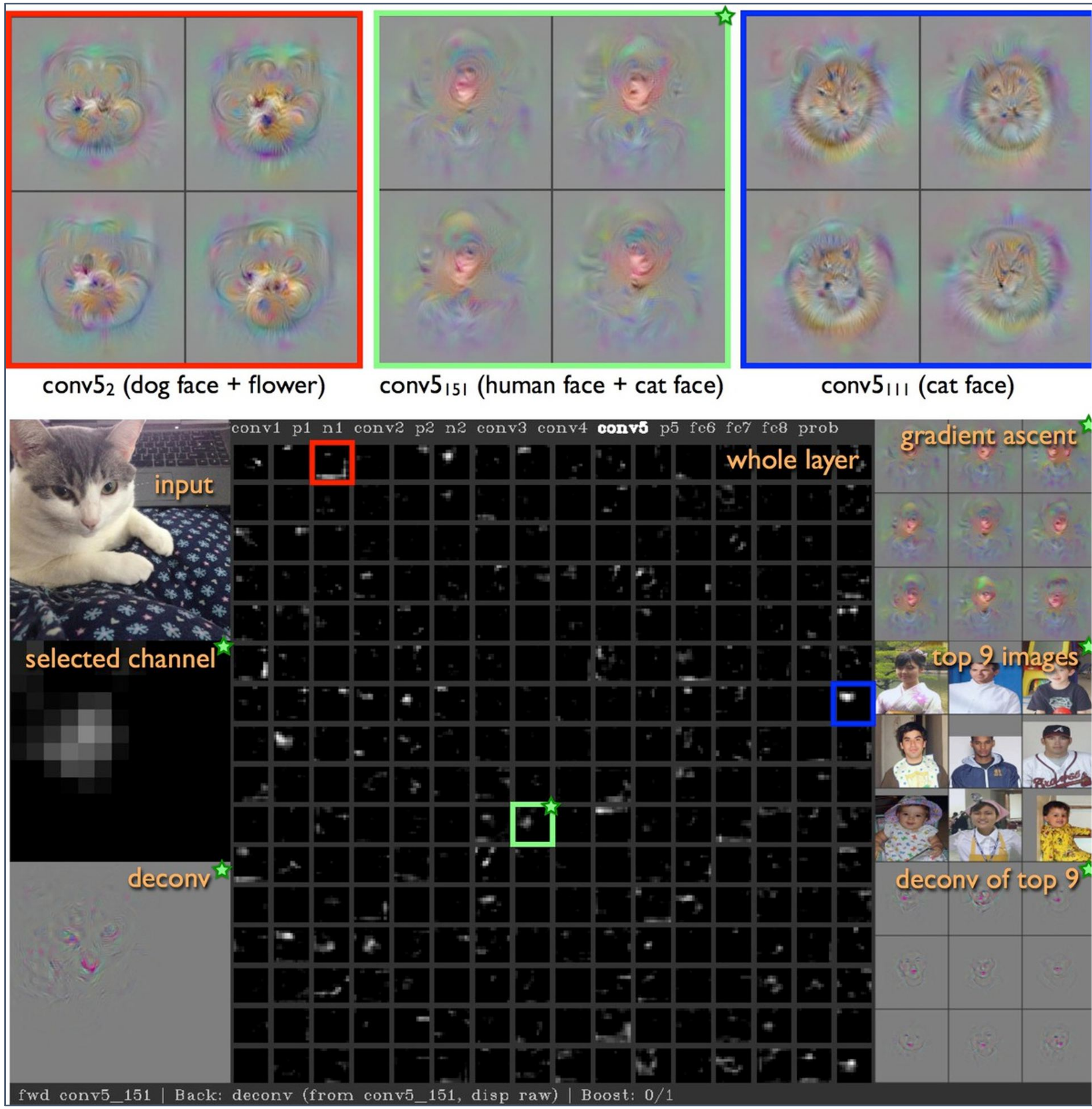
VGG-VD



Visualizing Activations

http://yosinski.com/deepvis

YouTube video
https://www.youtube.com/watch?v=AgkfIQ4IGaM
(4min)



Deep Visualization Toolbox

yosinski.com/deepvis

#deepvis


Jason Yosinski


Jeff Clune


Anh Nguyen


Thomas Fuchs


Hod Lipson

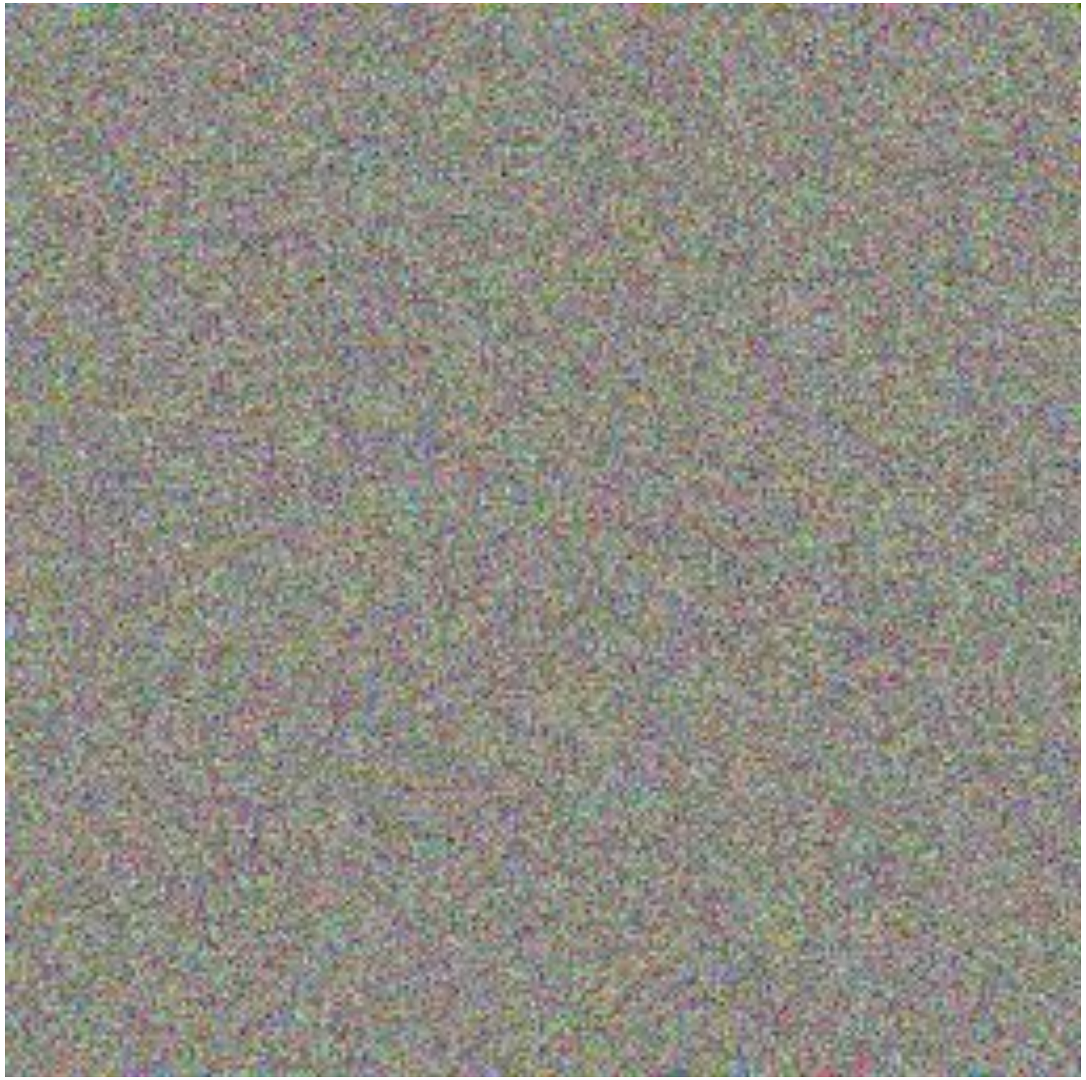
Cornell University

University of Wyoming

Jet Propulsion Laboratory
California Institute of Technology

Recall Mahendran and Vedaldi's pre-images: The starting point is white noise

- Not an image!



Caricaturization

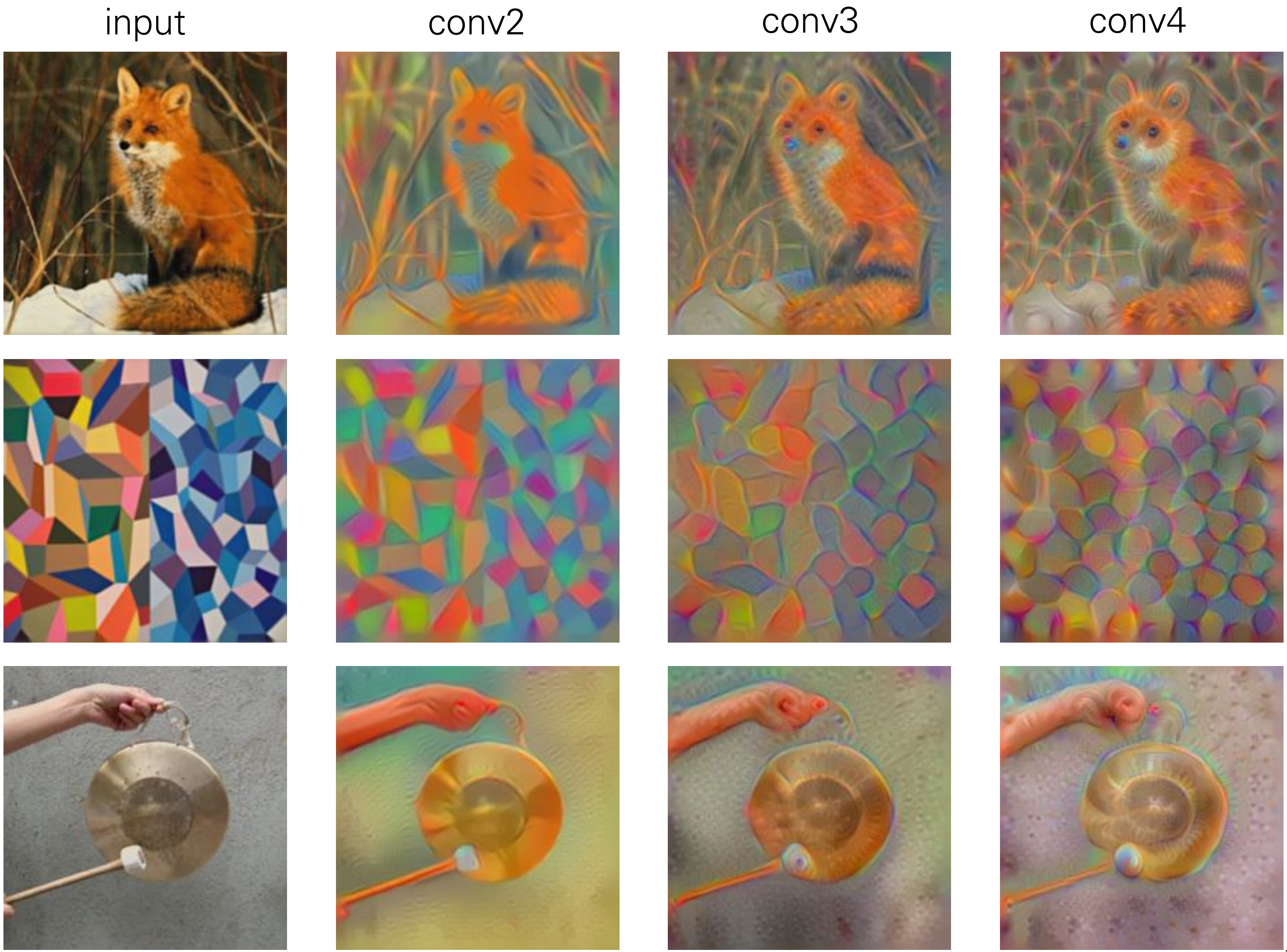
[Google Inceptionism 2015, Mahendran et al. 2015]

- Emphasize patterns that are detected by a certain representation

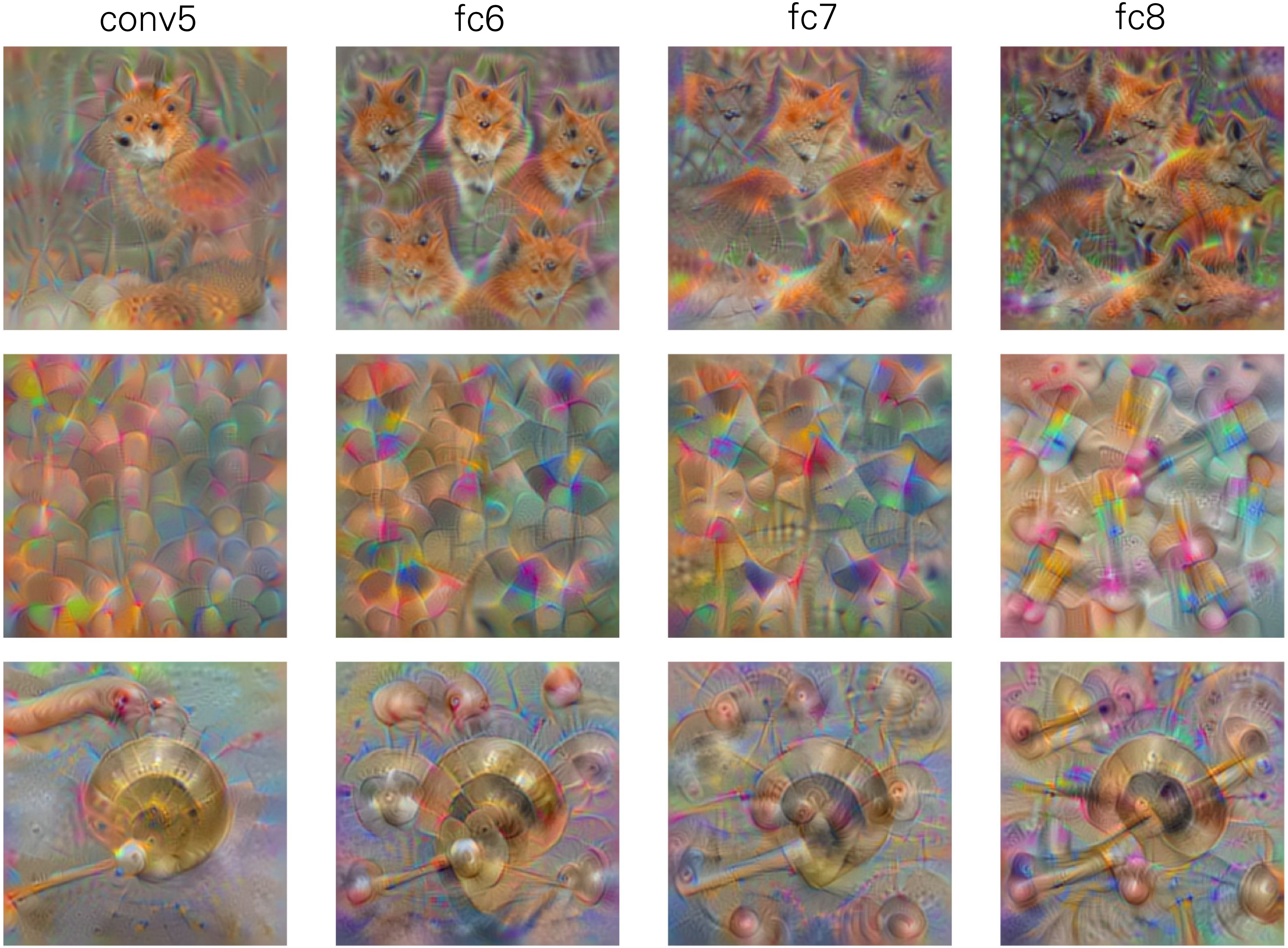
$$\boxed{\min_{\mathbf{x}} -\langle \Phi(\mathbf{x}_0), \Phi(\mathbf{x}) \rangle + R_{TV}(\mathbf{x}) + R_{\alpha}(\mathbf{x})}$$

- 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.

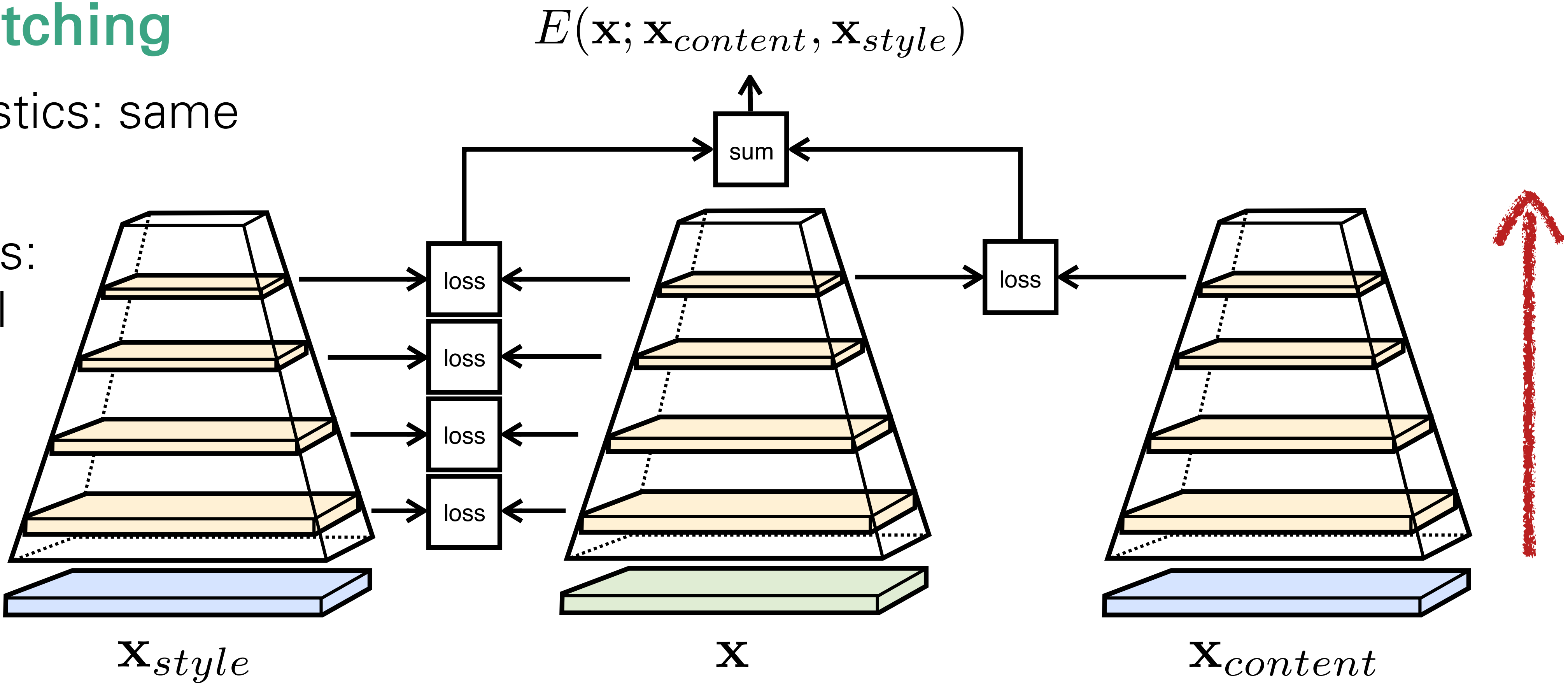
Feed-forward neural network equivalents

- 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.

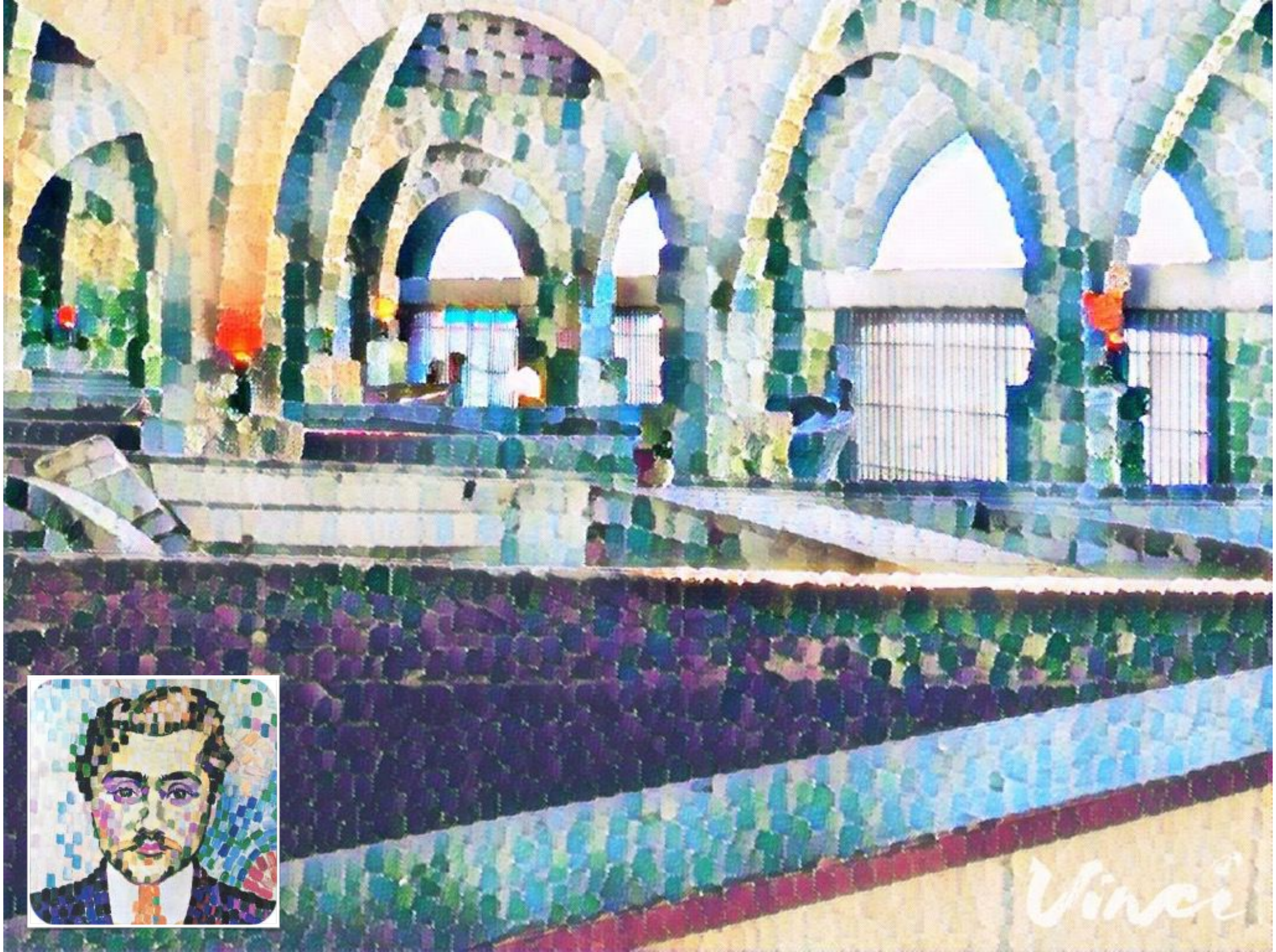
Generation by Moment Matching

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

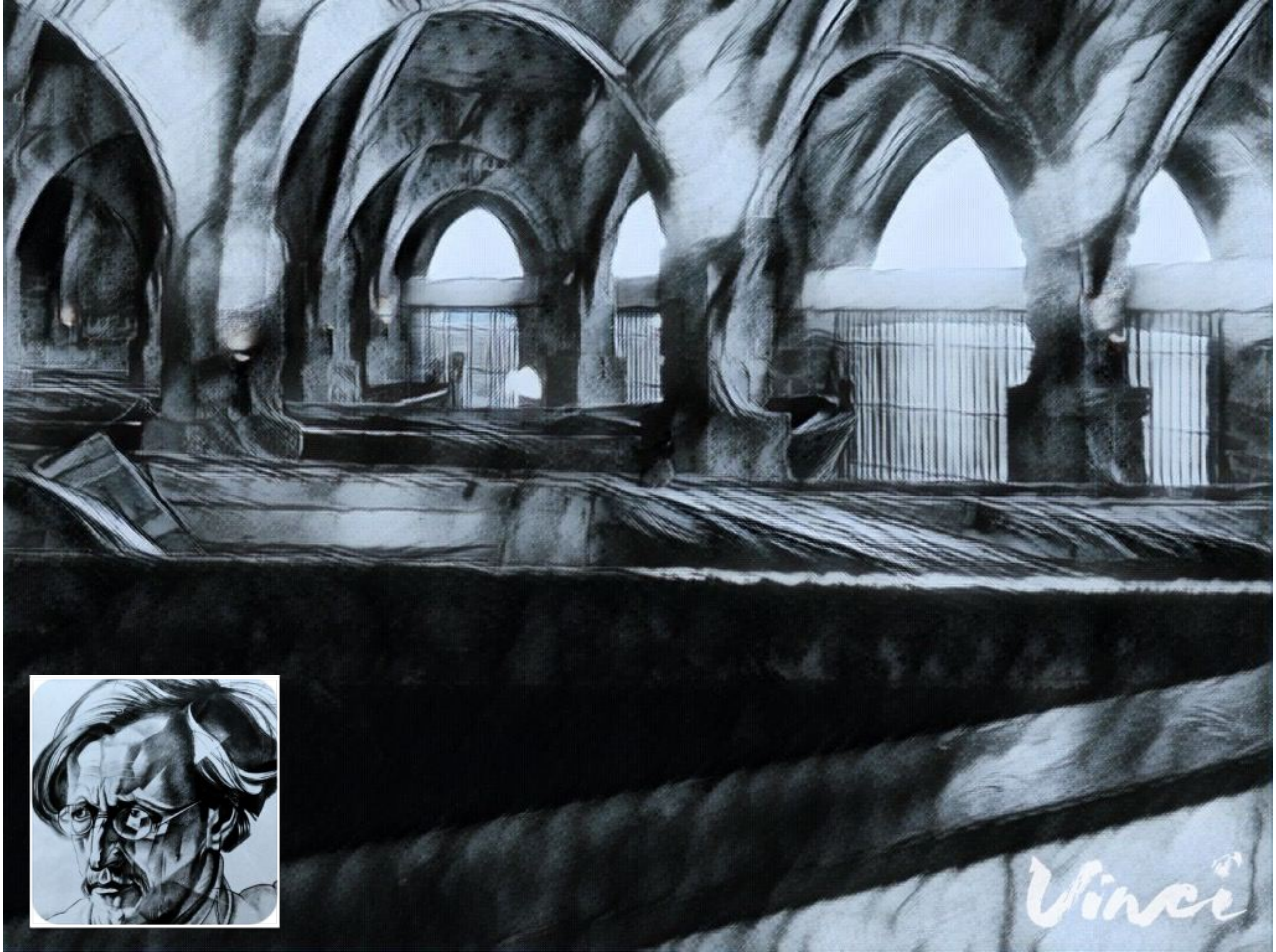

Vinci


Vinci


Vinci


Vinci


Vinci




Vinci


Vinci

Artistic style transfer for videos

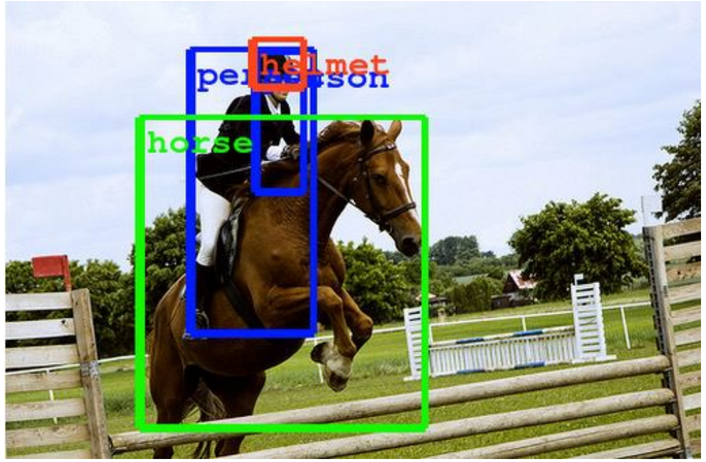
Manuel Ruder
Alexey Dosovitskiy
Thomas Brox

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...

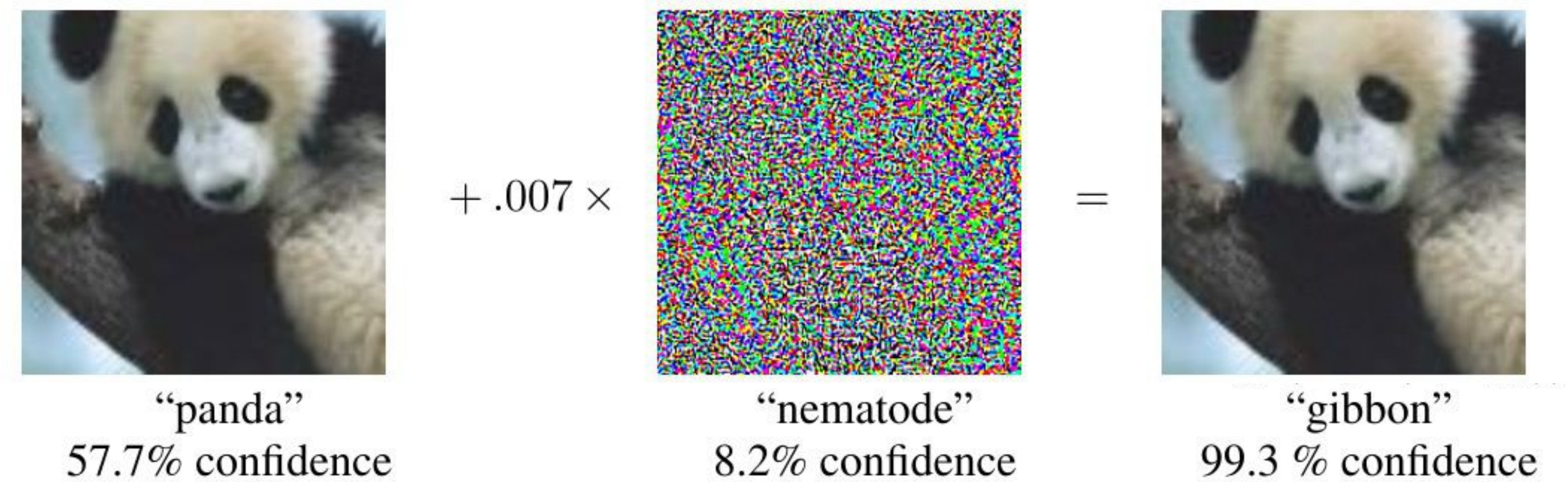
(Goodfellow et al, 2013)

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



Early Timeline:

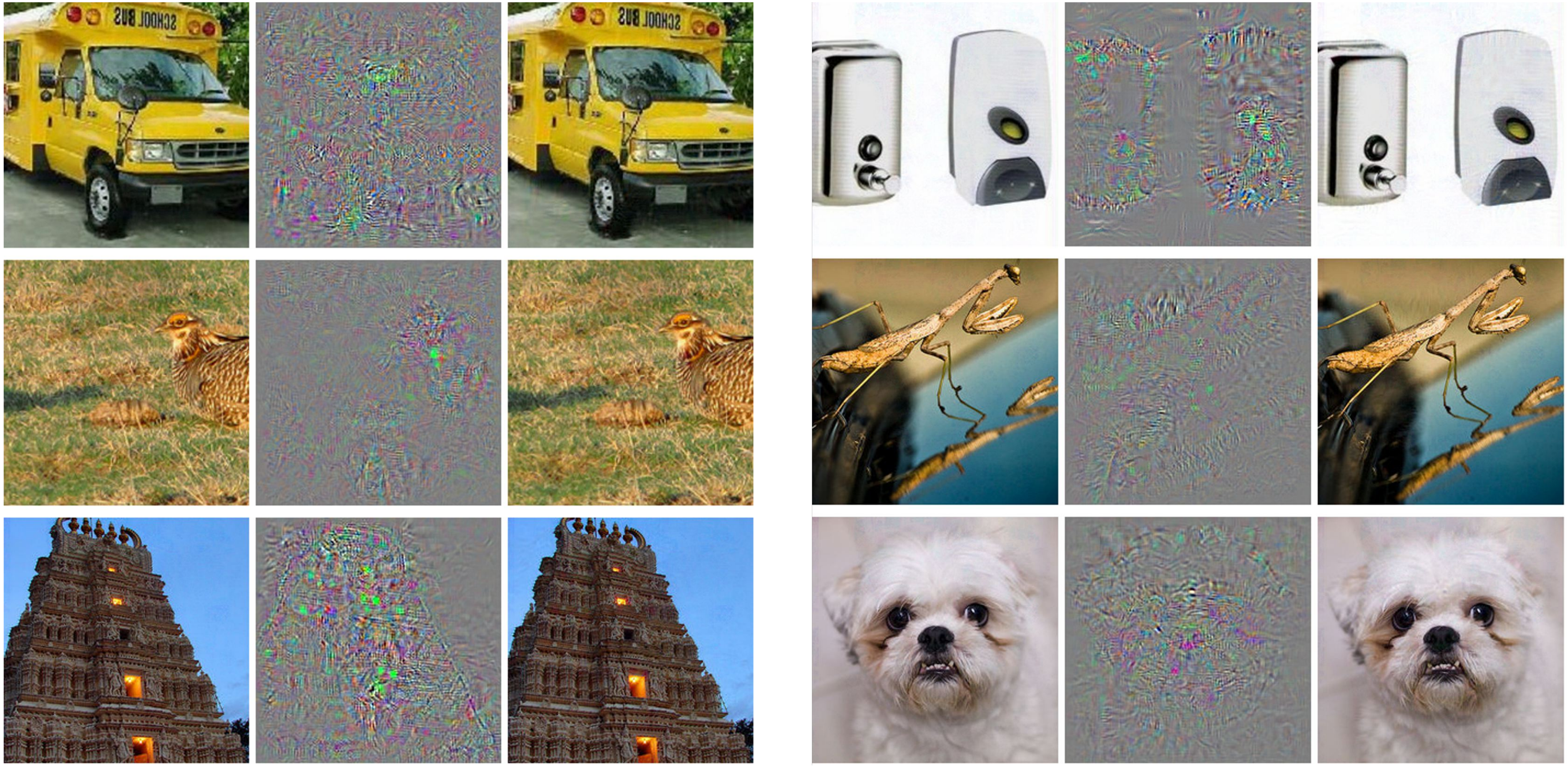
"Adversarial Classification" Dalvi et al 2004: fool spam filter

"Evasion Attacks Against Machine Learning at Test Time" Biggio 2013: fool neural nets

Szegedy et al 2013: fool ImageNet classifiers imperceptibly

Goodfellow et al 2014: cheap, closed form attack

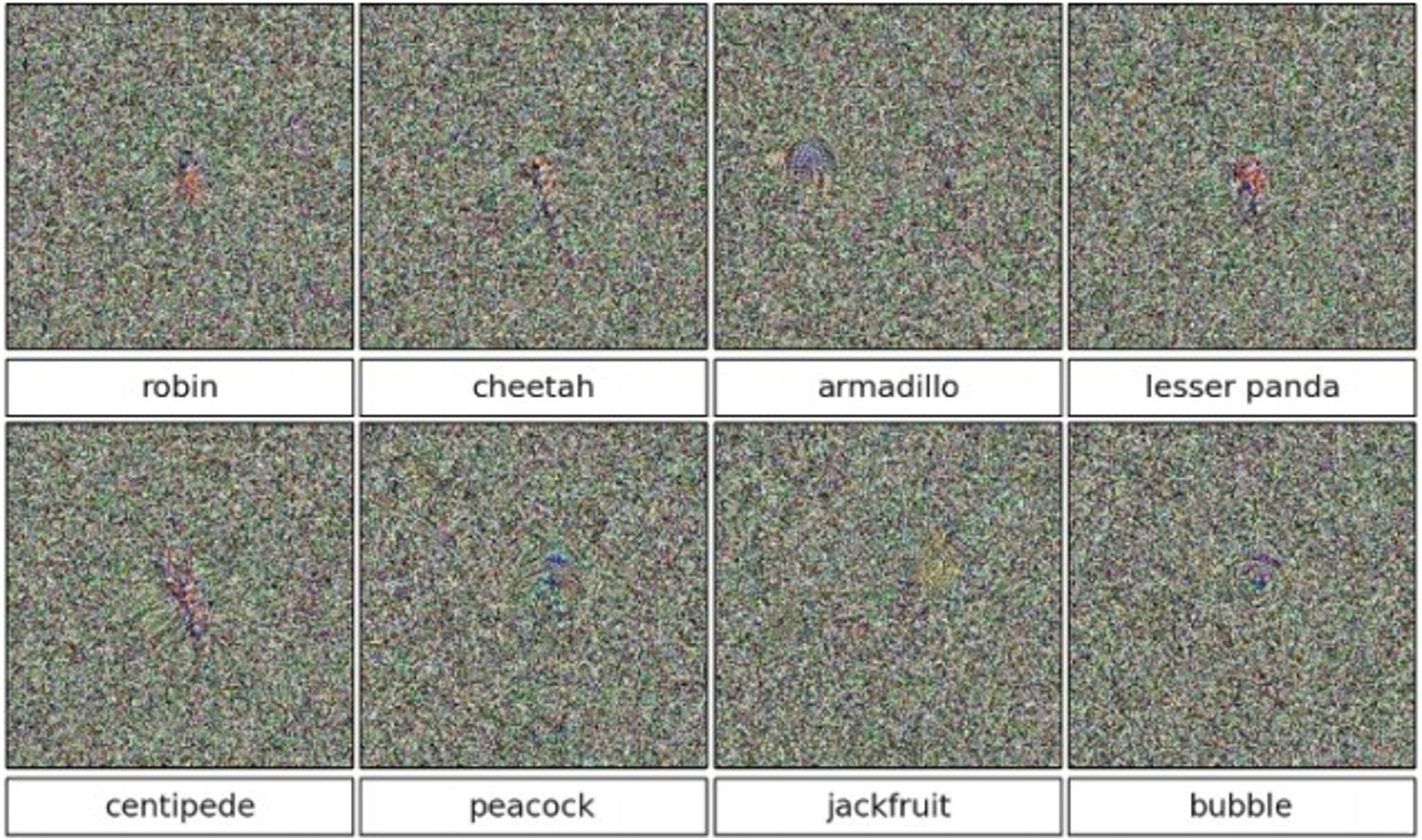
Intriguing properties of neural networks
[Szegedy et al., 2013]



correct +distort ostrich correct +distort ostrich

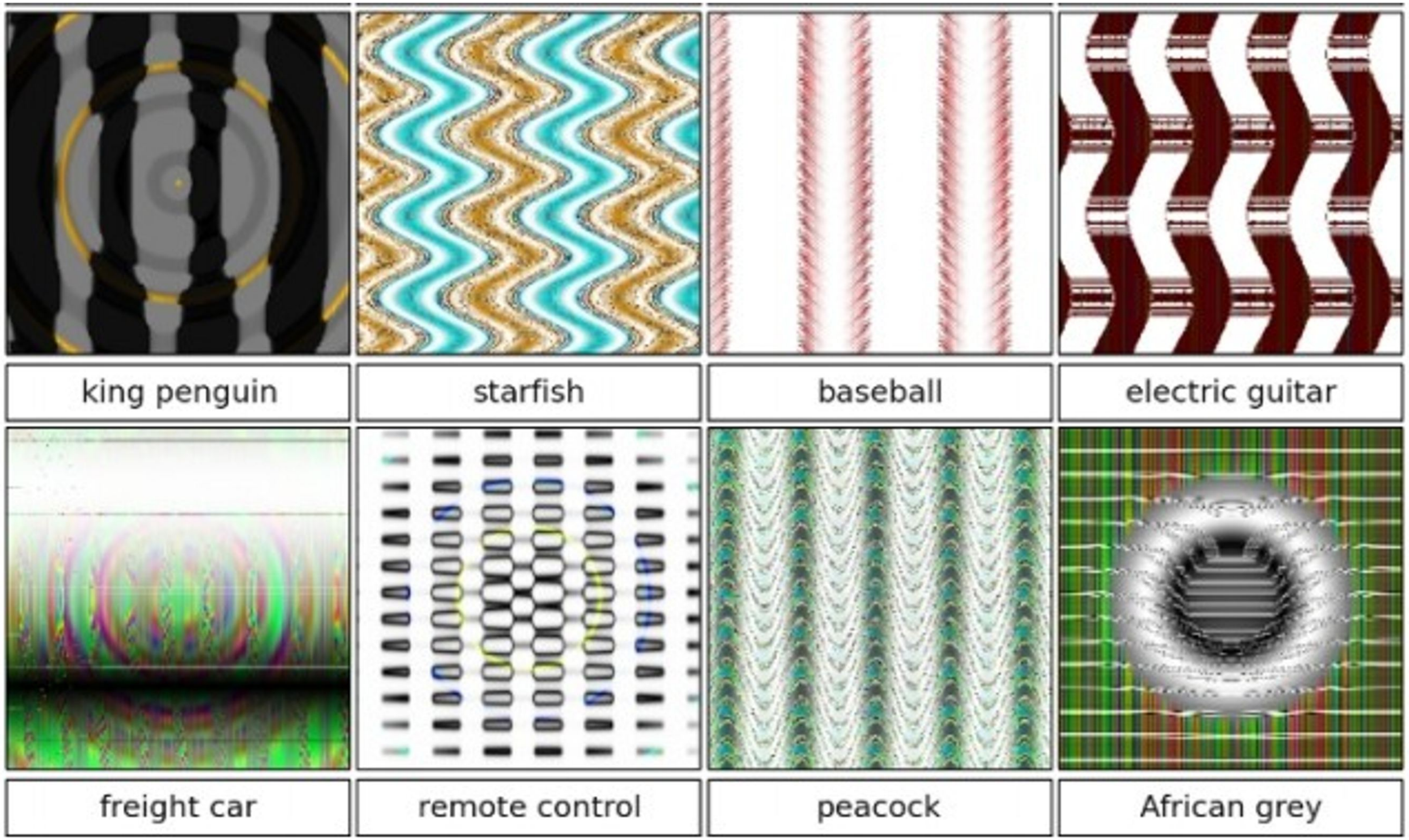
Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images
[Nguyen, Yosinski, Clune, 2014]

>99.6% confidences



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

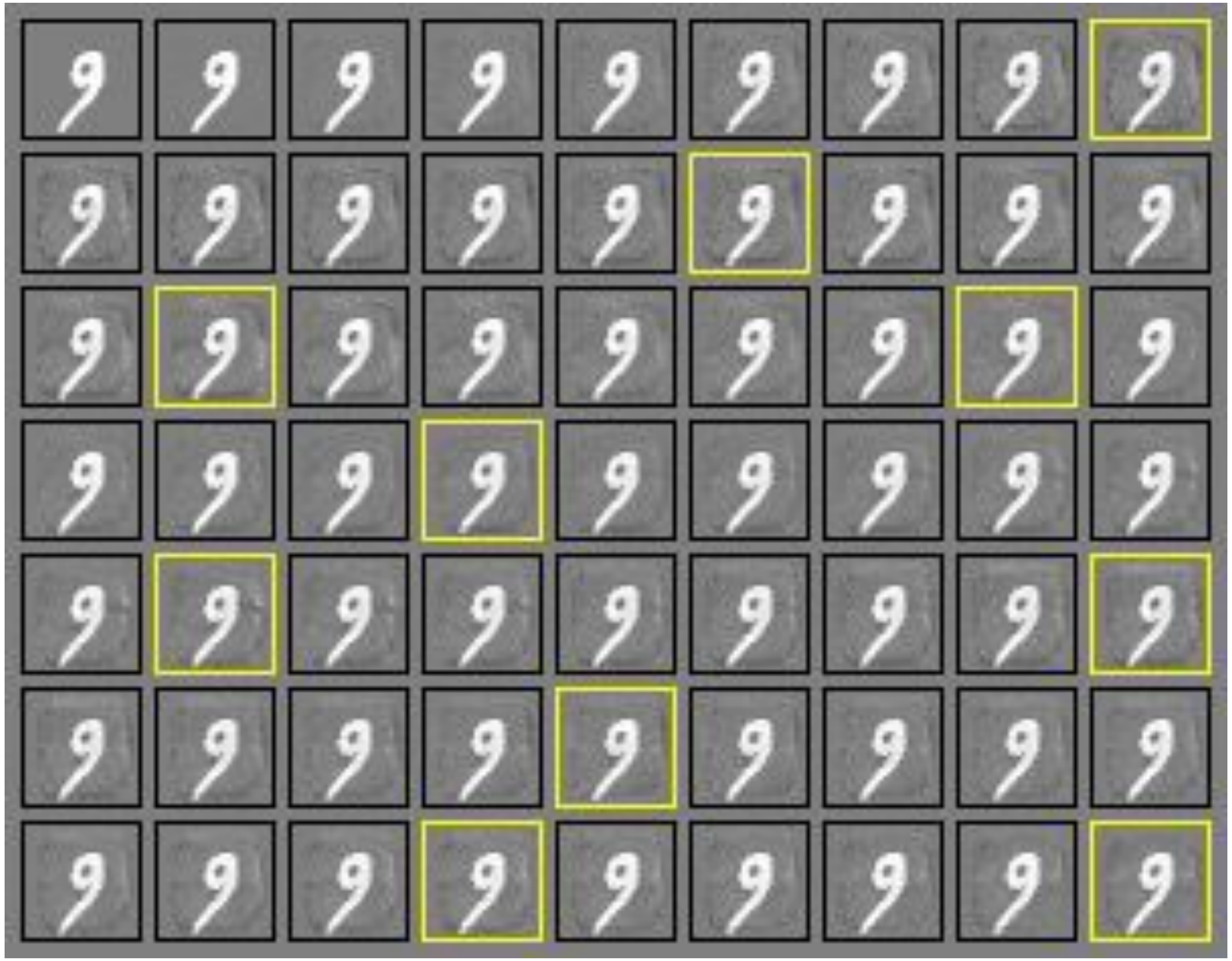
>99.6% confidences



Not just for neural nets

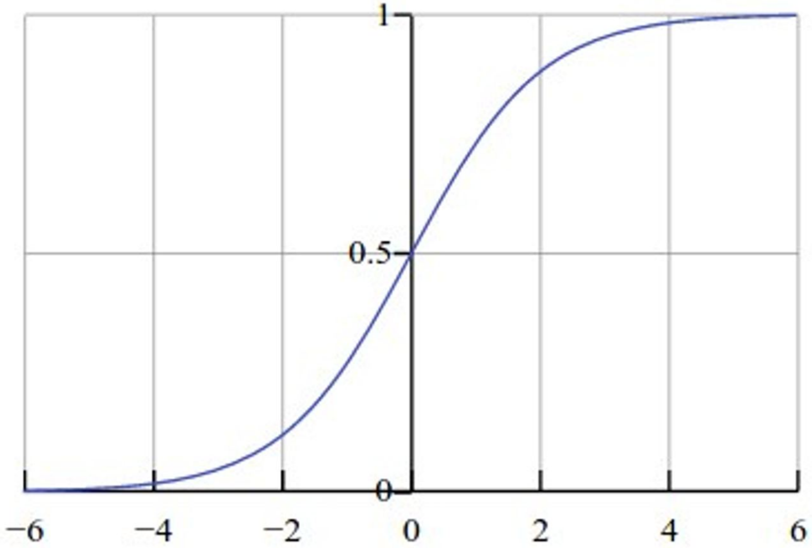
- Linear models
  - Logistic regression
  - Softmax regression
  - SVMs
- Decision trees
- Nearest neighbors

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%

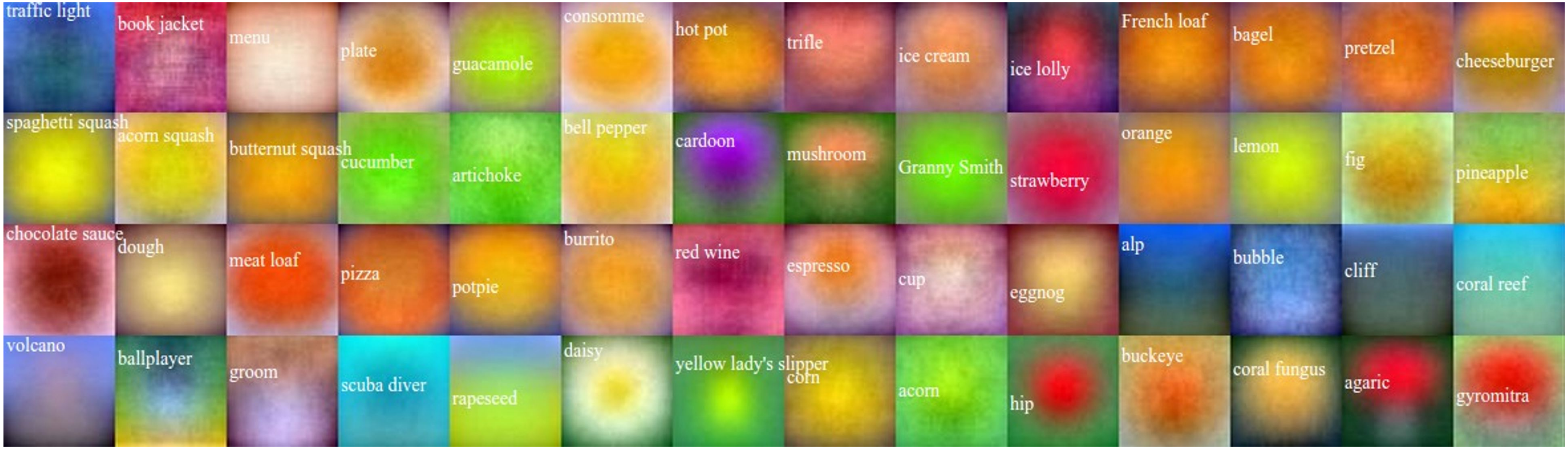
This was only with 10 input dimensions. A 224x224 input image has 150,528.

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

Blog post: Breaking Linear Classifiers on ImageNet

Recall CIFAR-10 linear classifiers:

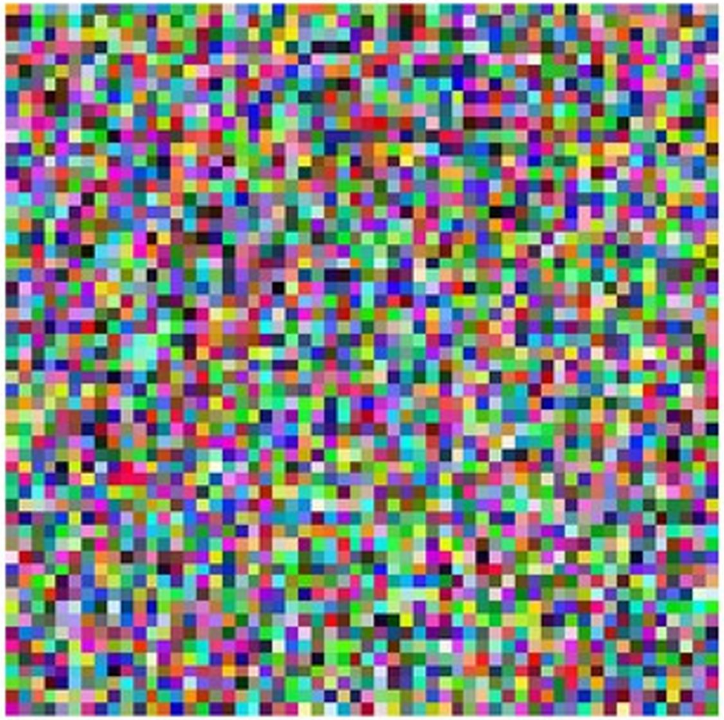
ImageNet classifiers:



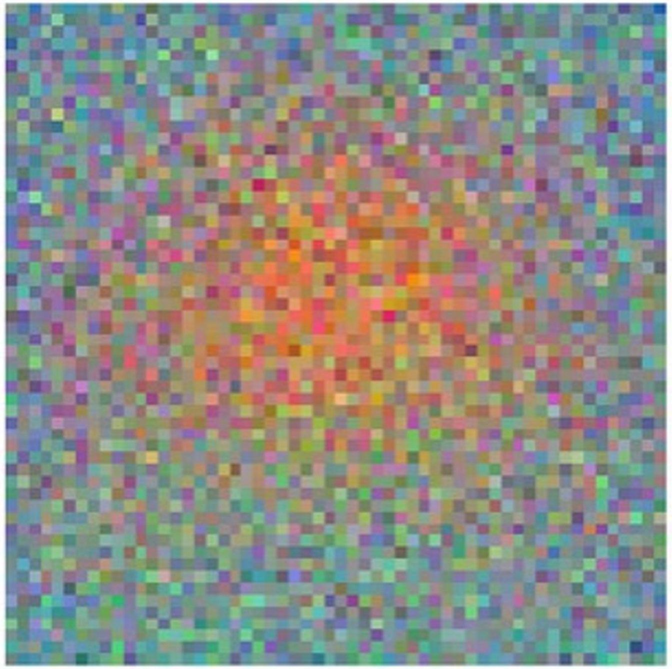
http://karpathy.github.io/2015/03/30/breaking-convnets/

mix in a tiny bit of
Goldfish classifier weights

0.9% bobsled

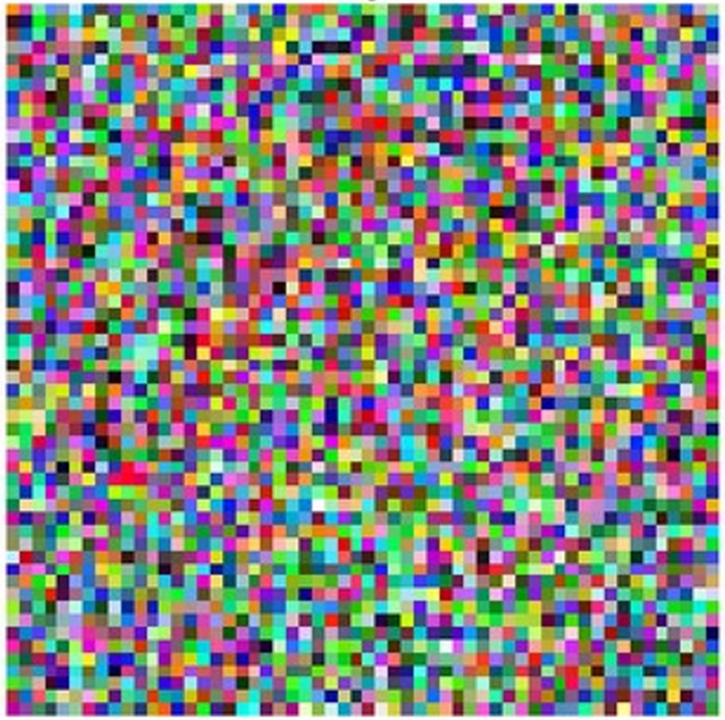


+



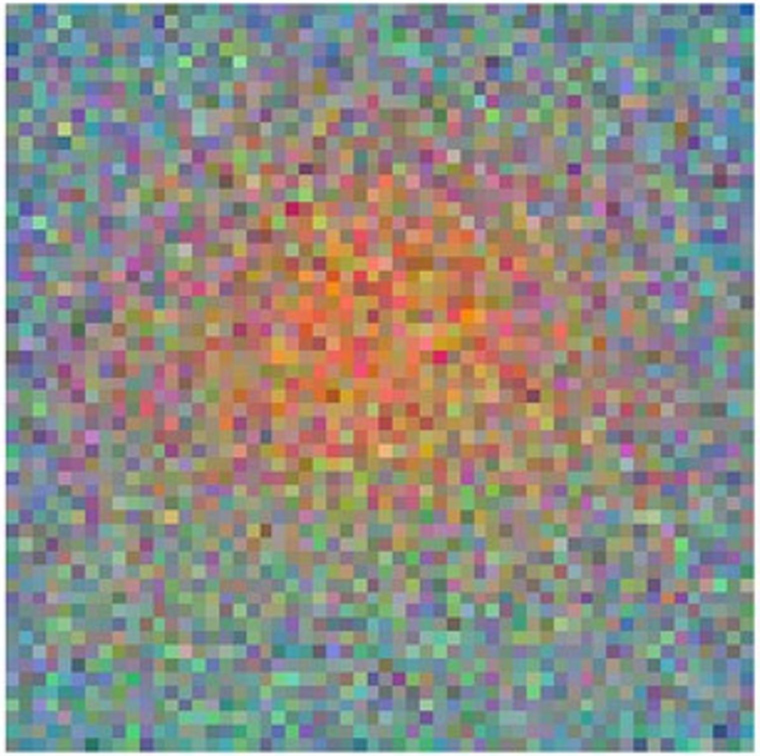
=

100.0% goldfish

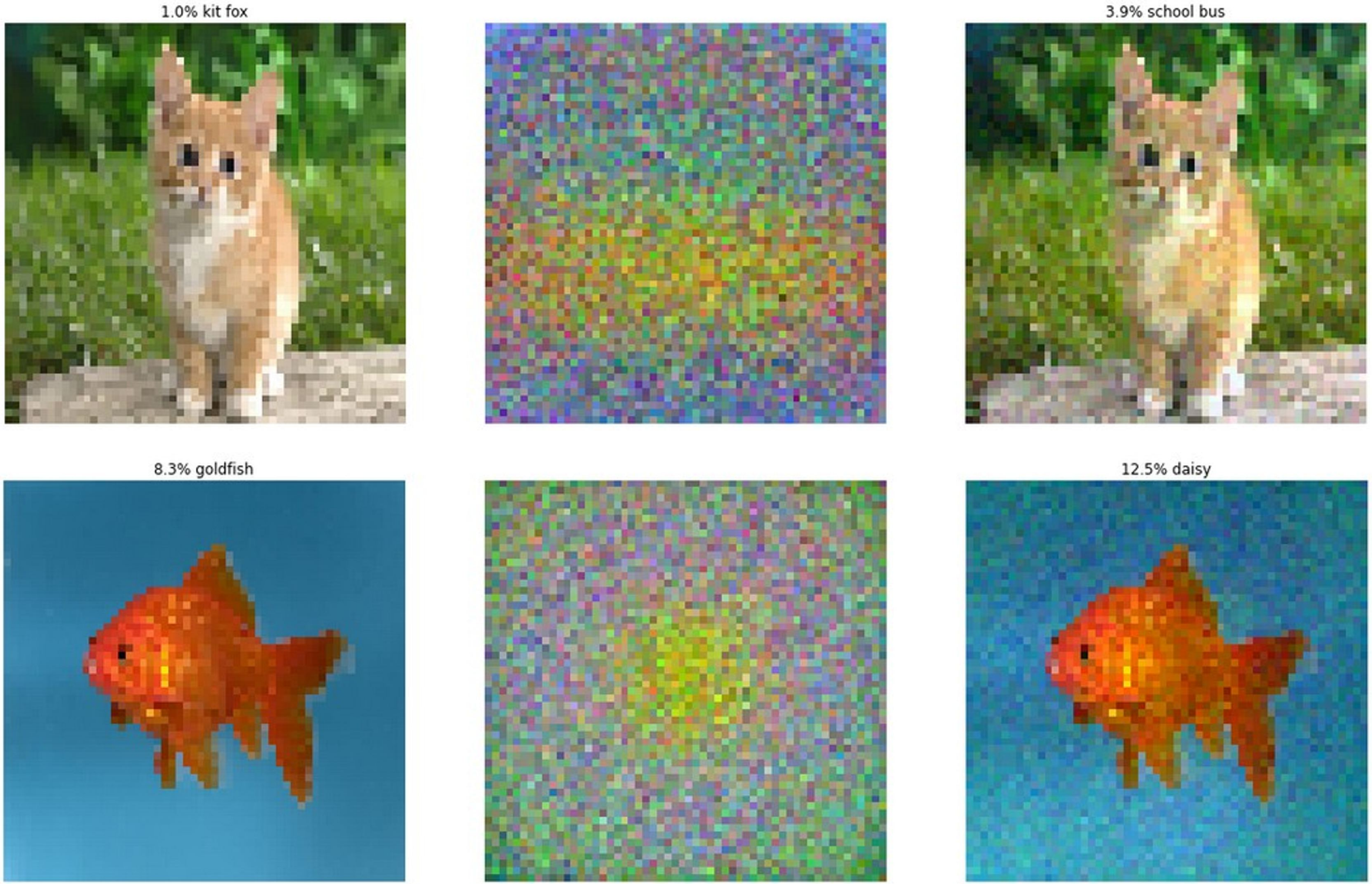


100% Goldfish

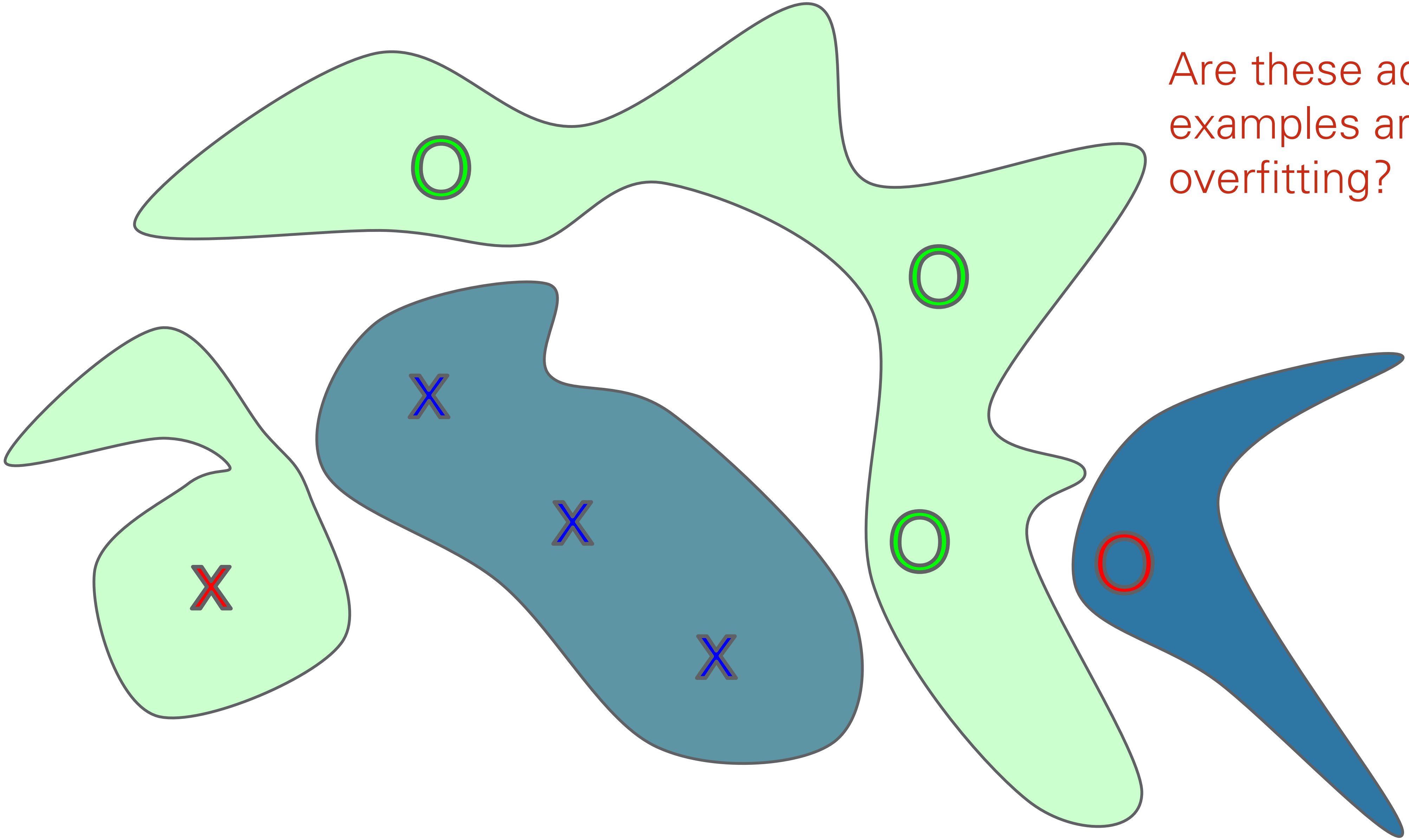
1.0% kit fox



8.0% goldfish

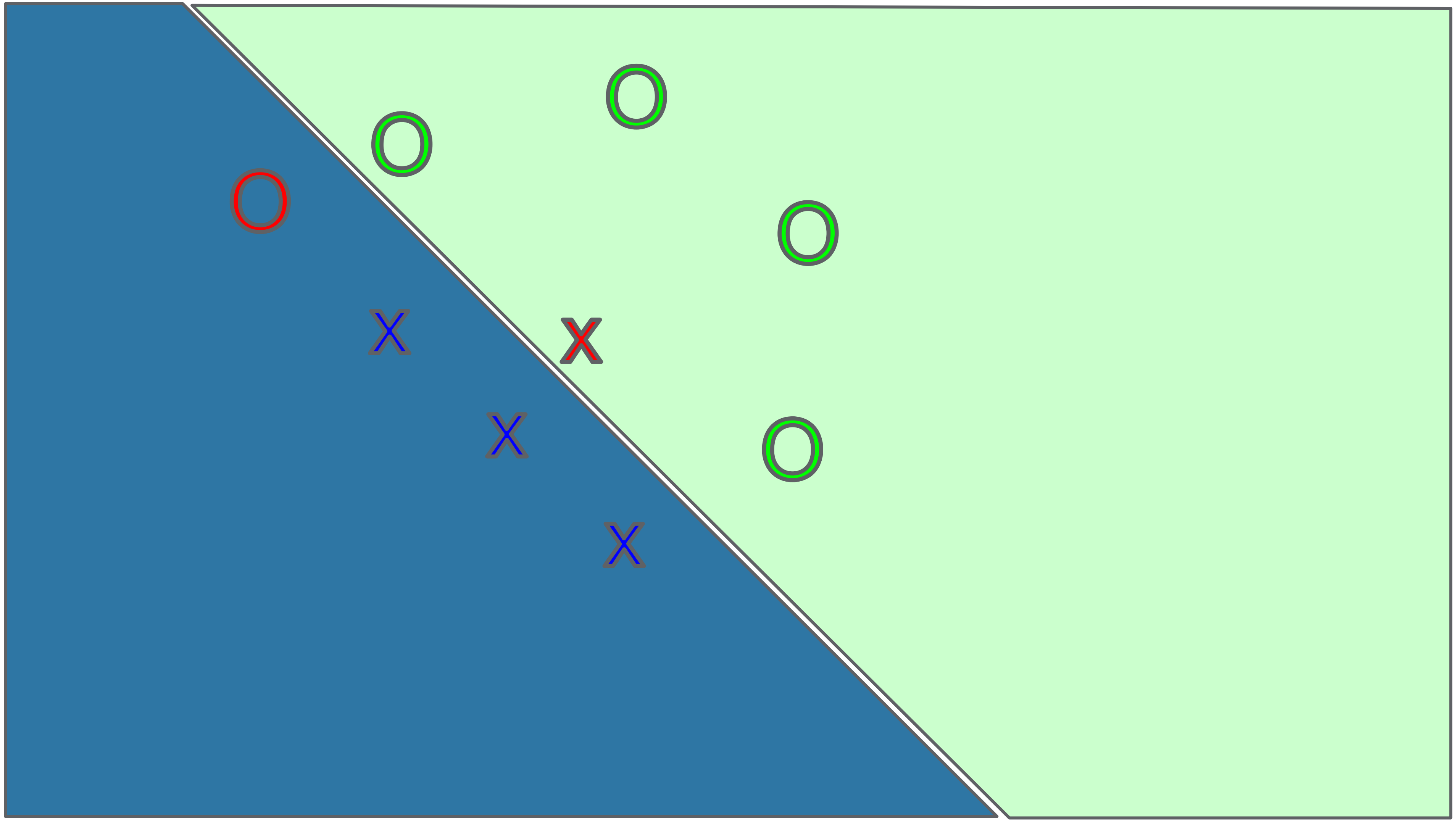

1.0% kit fox
3.9% school bus
8.3% goldfish
12.5% daisy

Adversarial Examples from Overfitting



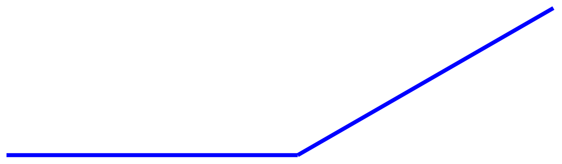
Are these adversarial examples are related to overfitting?

Adversarial Examples from Excessive Linearity

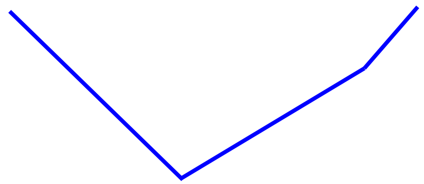


Modern deep nets are very piecewise linear

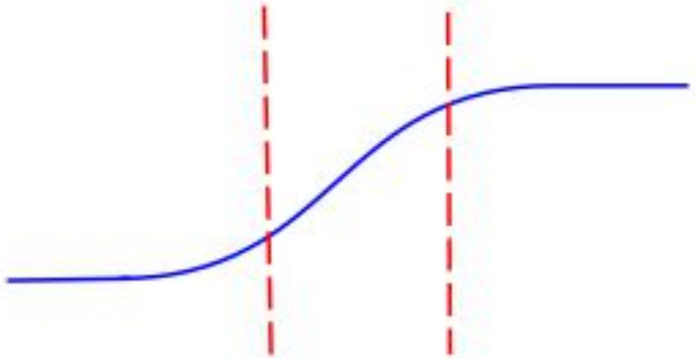
Rectified linear unit



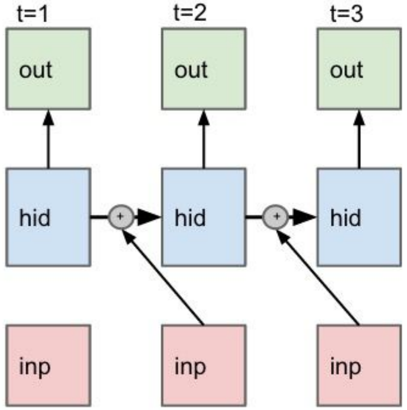
Maxout



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

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

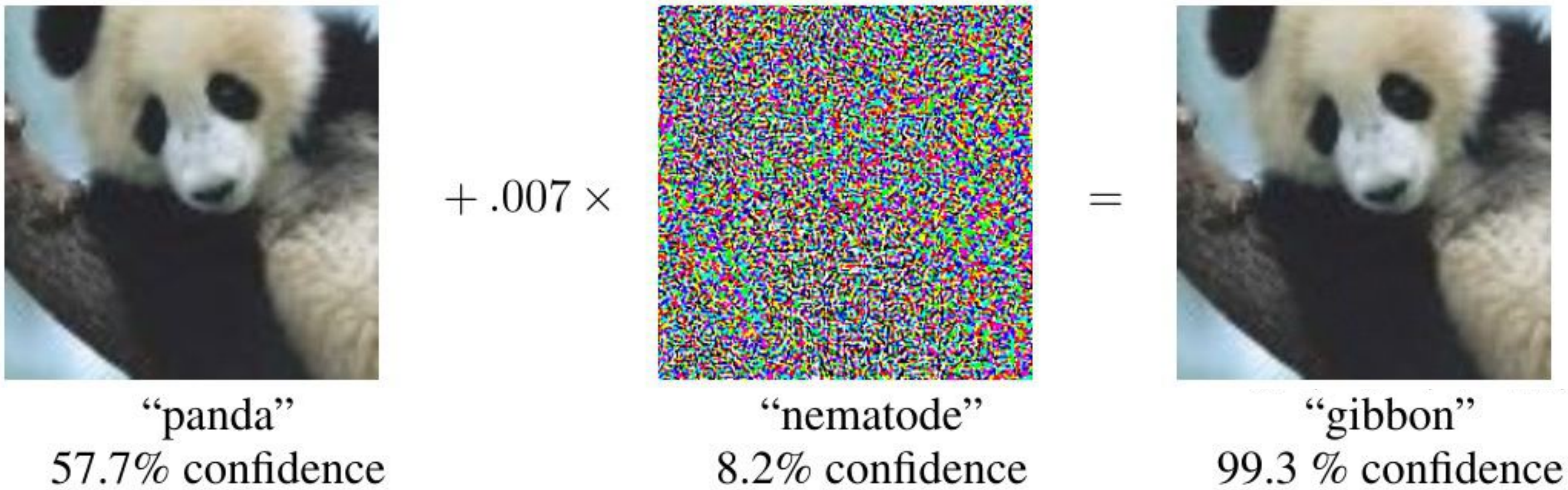
subject to

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

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

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

Adversarial Examples

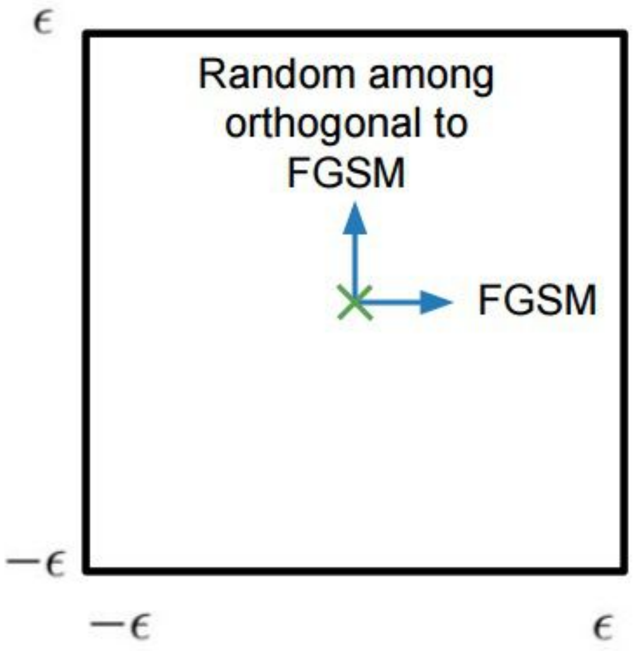


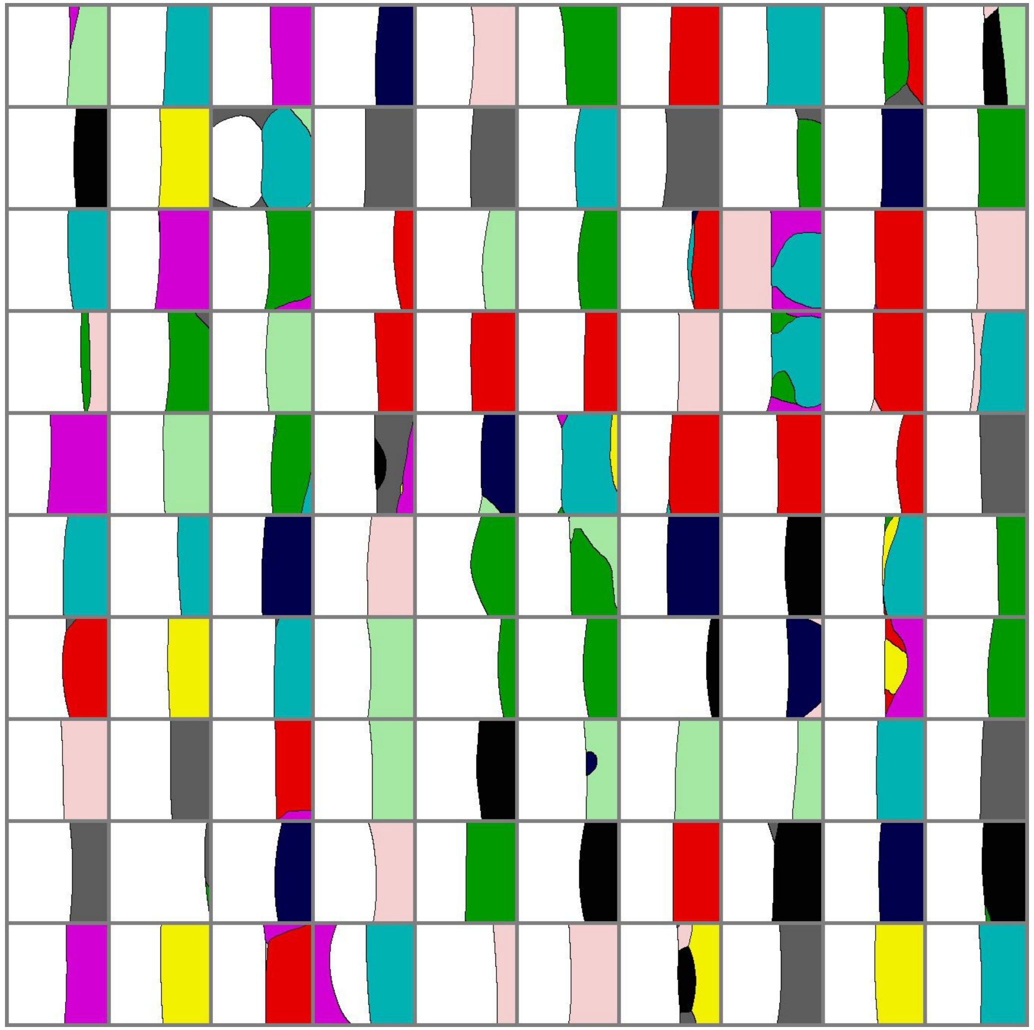
$$X^{adv} = X + \epsilon\,\mathrm{sign}\left(\nabla_X J(X, y_{true})\right)$$

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

Maps of Adversarial and Random Cross-Sections

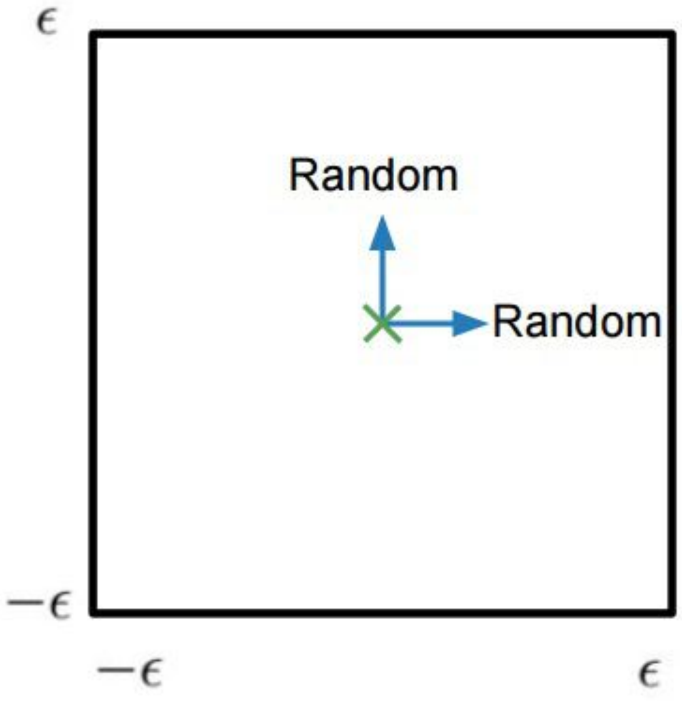


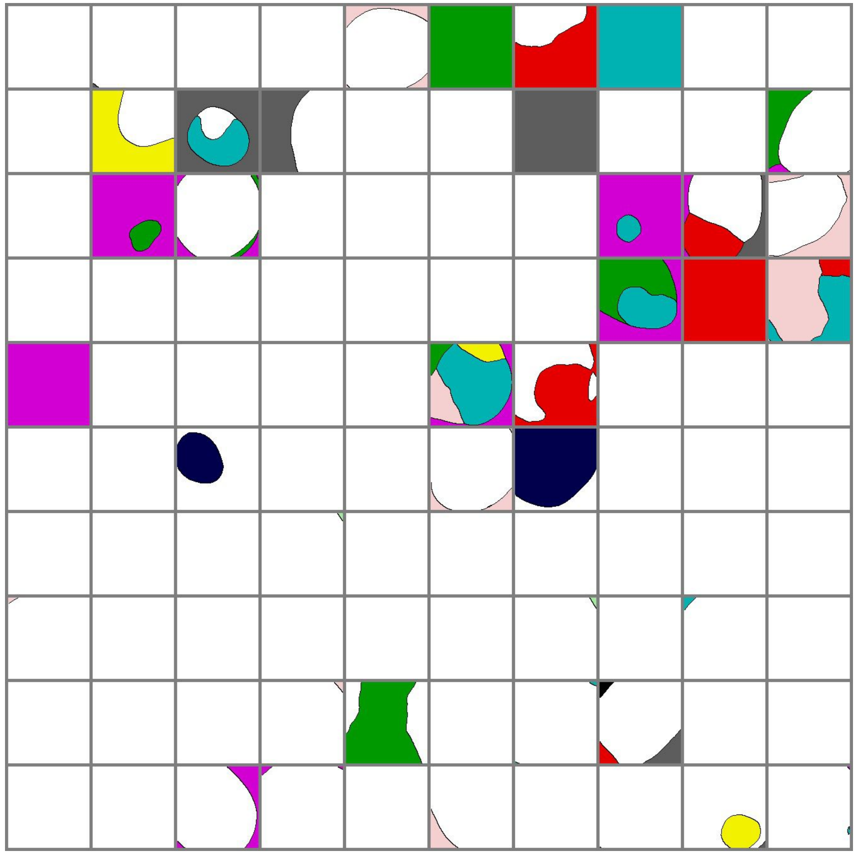


- Trace out input space for CIFAR10 with a deep CNN to see how it classifies different points in space on the above up-down axis

Maps of Random Cross-Sections

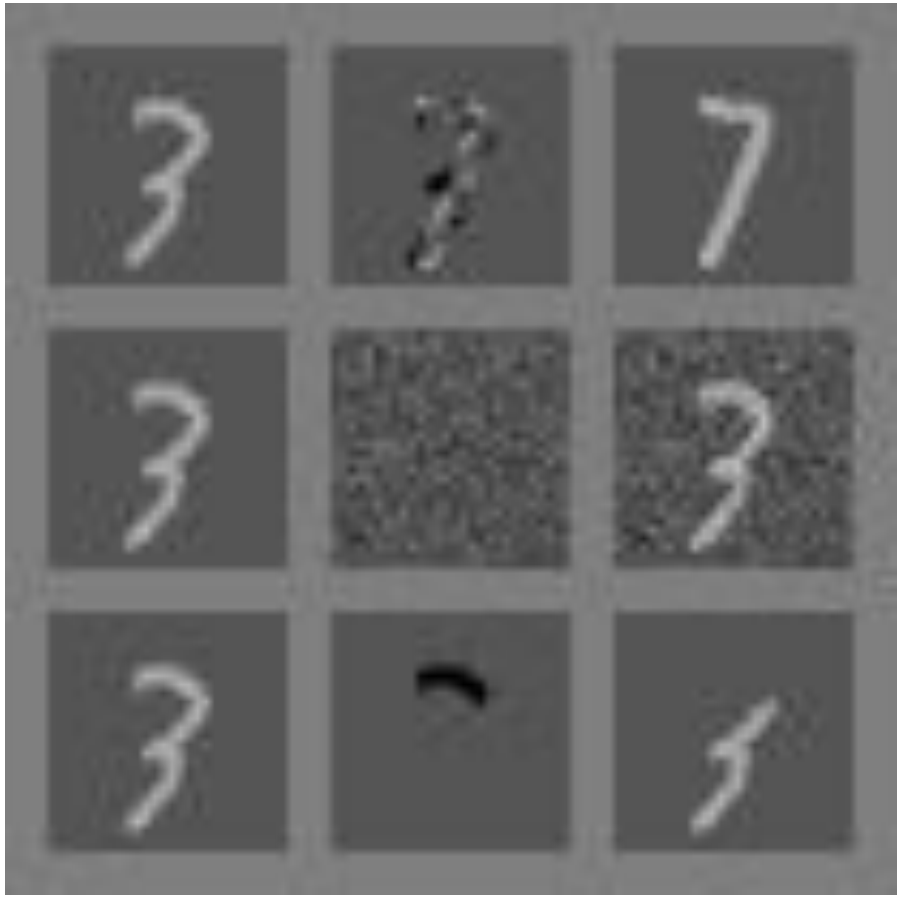
Adversarial examples are not noise





Small inter-class distances

Clean example	Perturbation	Corrupted example



Perturbation changes the true class

Random perturbation does not change the class

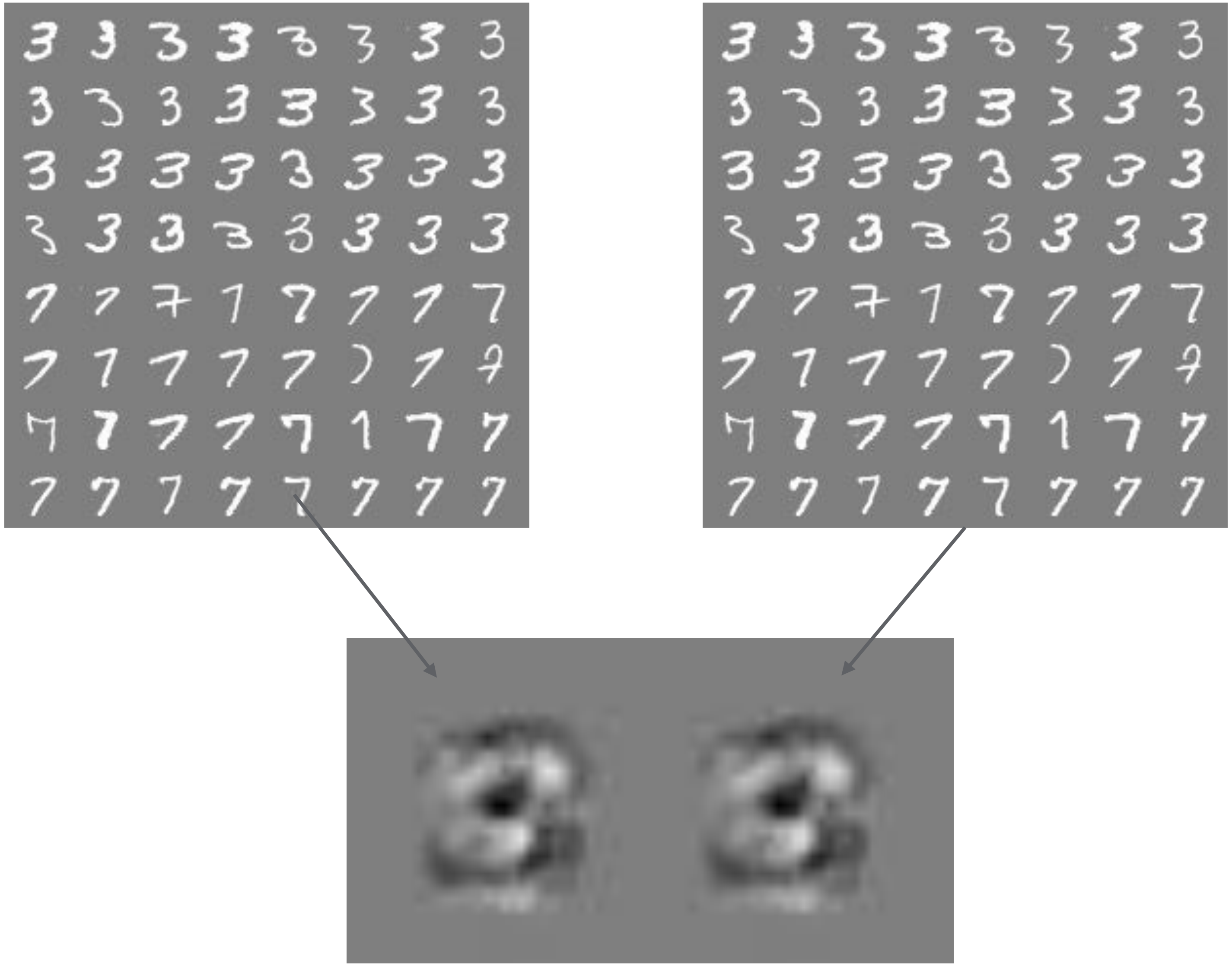
Perturbation changes the input to “rubbish class”

All three perturbations have L2 norm 3.96

This is actually small. We typically use 7!

weight decay does not prevent adversarial examples

Cross-model, cross-dataset generalization



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

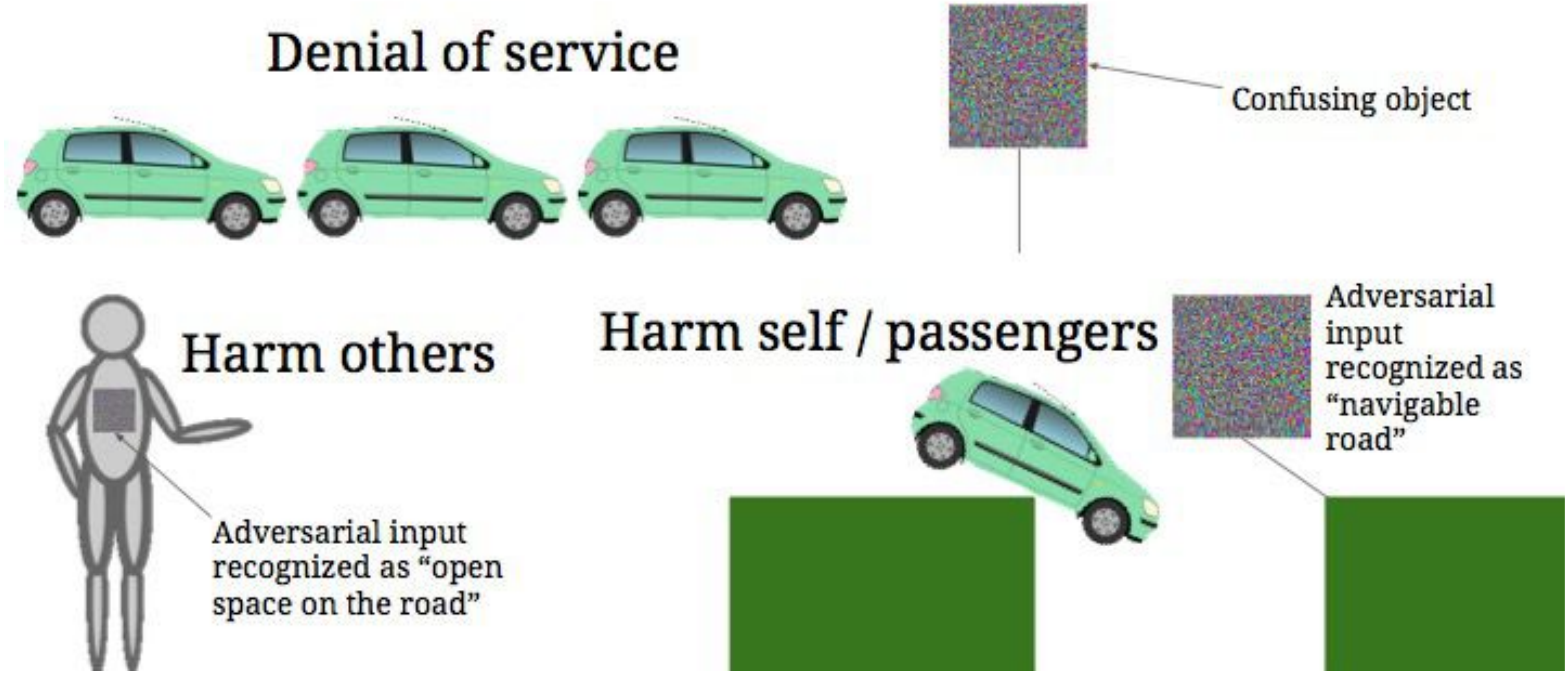
- Display adversarial examples in the physical world and fool machine learning systems that perceive them through a camera

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

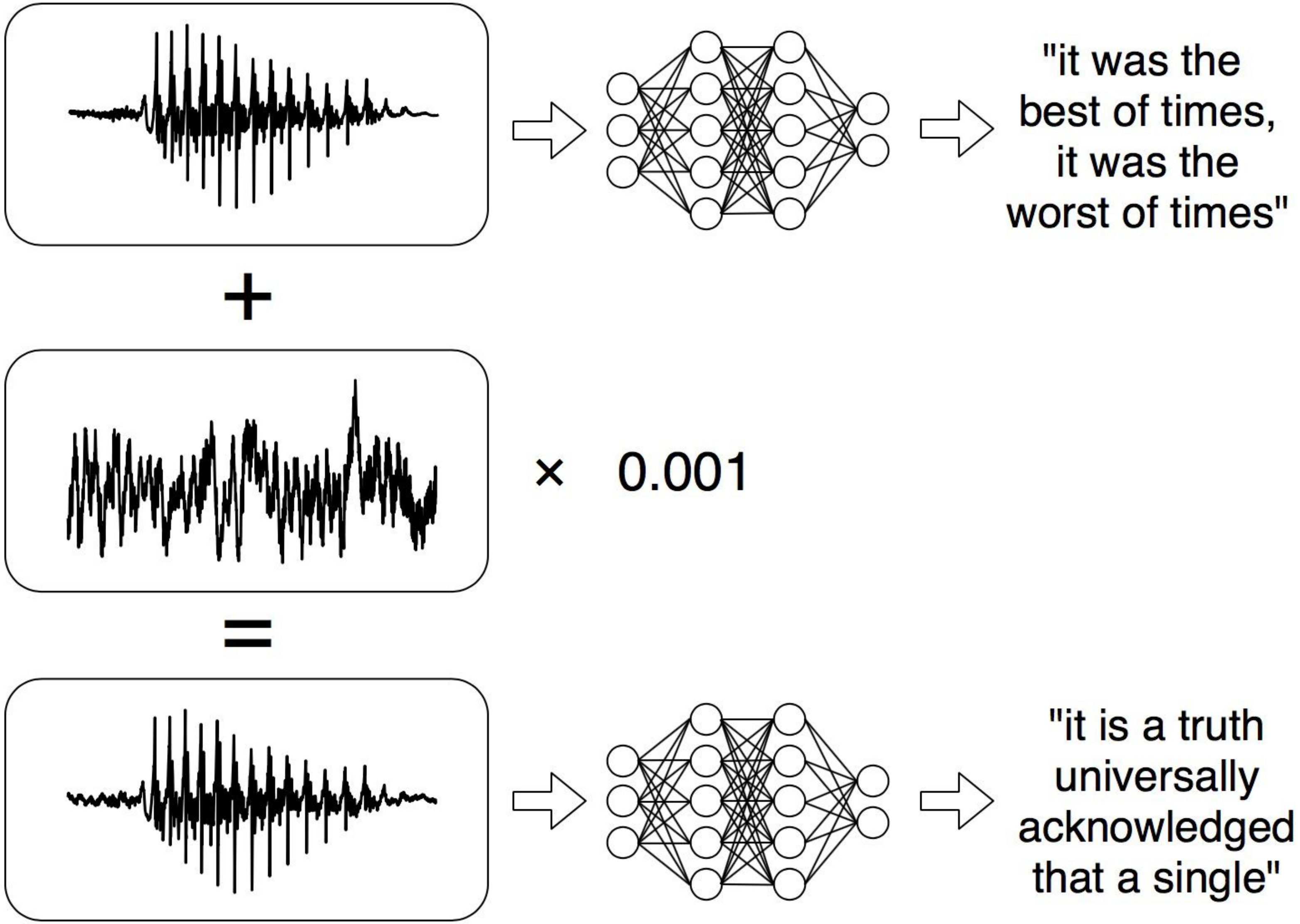
http://bair.berkeley.edu/blog/2017/12/30/yolo-attack/

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

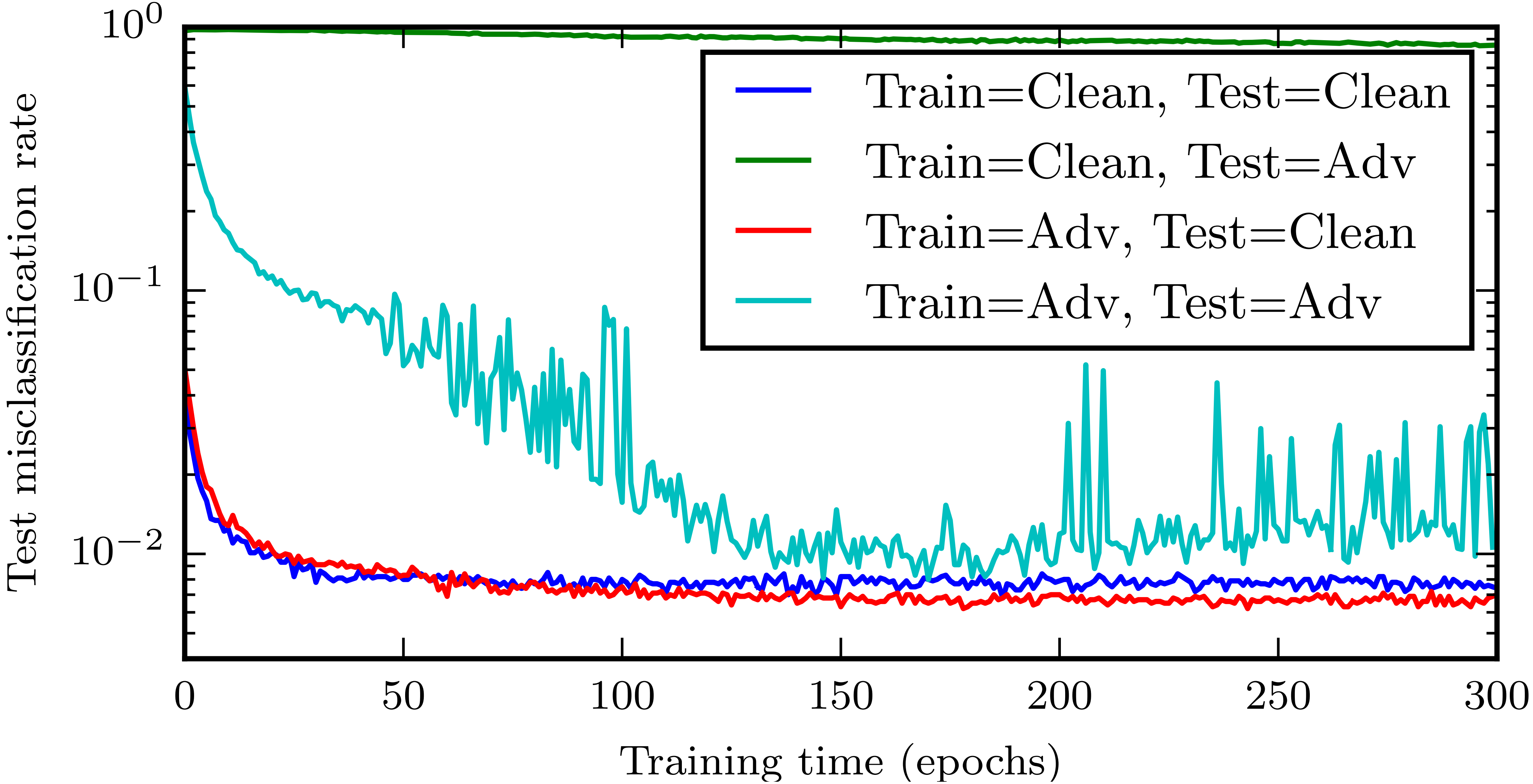
Unlabeled; model guesses it's probably a bird, maybe a plane

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