Implementing ANNs with TensorFlow

Session 08 - More on CNNs

Organizational Stuff

07.01. - 14.01.	Advanced CNNs + Visualization	n.a.	n.a.	n.a.
14.01. - 21.01.	Recurrent Neural Networks	n.a.	n.a.	n.a.
21.01. - 28.01.	Word Embeddings	n.a.	n.a.	n.a.
28.01. - 04.02.	Generative Models	n.a.	n.a.	n.a.
04.02. - 11.02.	Deep Reinforcement Learning	n.a.	n.a.	n.a.
11.02. - 14.02.	Practical Aspects	n.a.	n.a.	n.a.
14.02.	Deadline Fix Topic for Final Project	-	-	-

Agenda

1. Advanced CNNs
2. Feature Visualization
3. Adversarial Examples

Advanced CNNs

1. **AlexNet**
2. **VGG**
3. **InceptionNet**
4. **ResNet**

Object Recognition

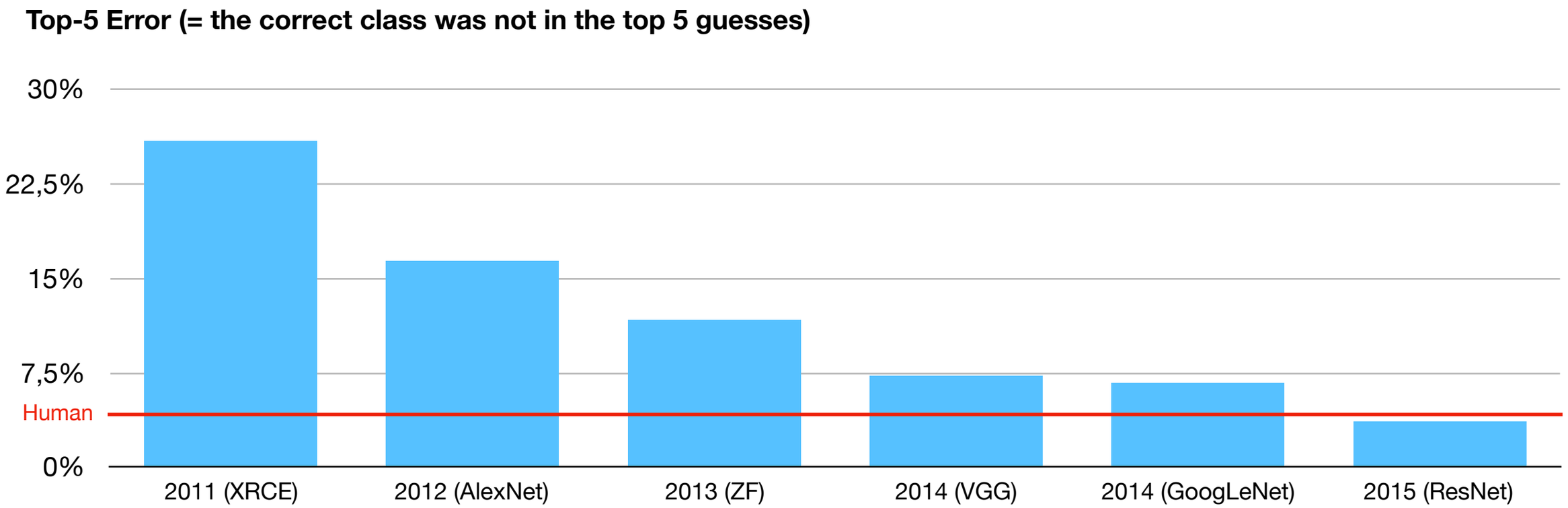
- One popular task for deep neural networks is object recognition.

- Example:

“dog”

ImageNet Benchmark

- A very popular benchmark for object recognition is the ImageNet dataset.
- It contains 1000 classes with 1.2 million images (of various resolutions) in total.
- The dataset is very complex containing for example 120 different breeds of dogs.

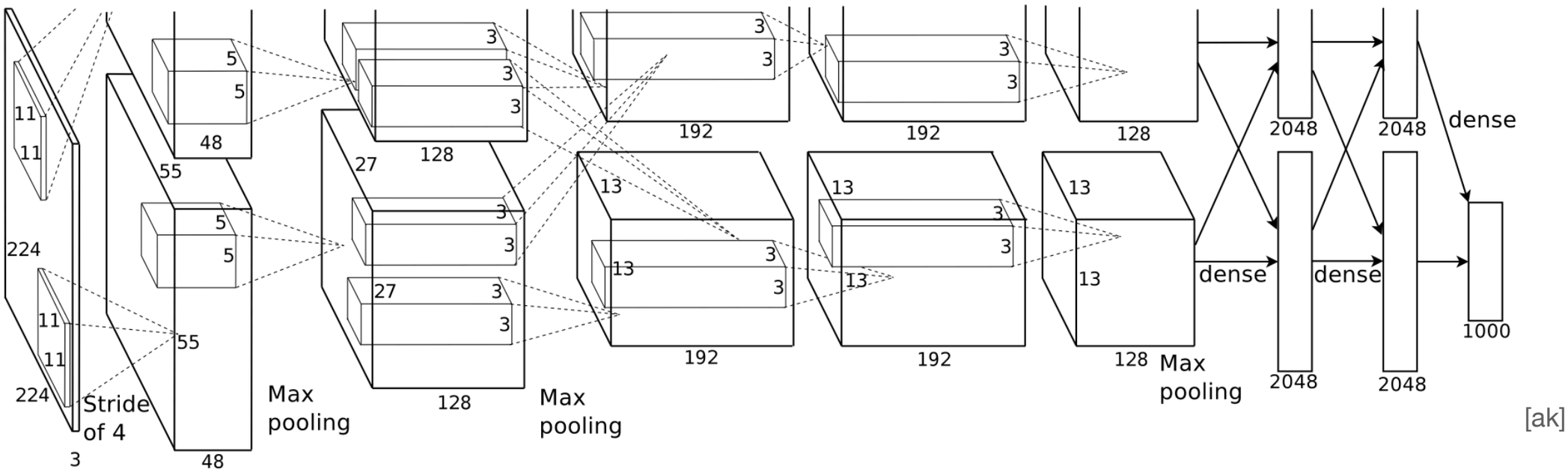


AlexNet

[Paper]

- Until 2012 an application of deep networks for large-scale image recognition was not feasible due to restrictions in computational power.

- AlexNet solved that problem by using two separate GPUs at the same time.



[ak]

AlexNet

convolutional layer: 96 kernels 11x11

Max pooling

convolutional layer: 256 kernels 5x5

Max pooling

convolutional layer: 384 kernels 3x3

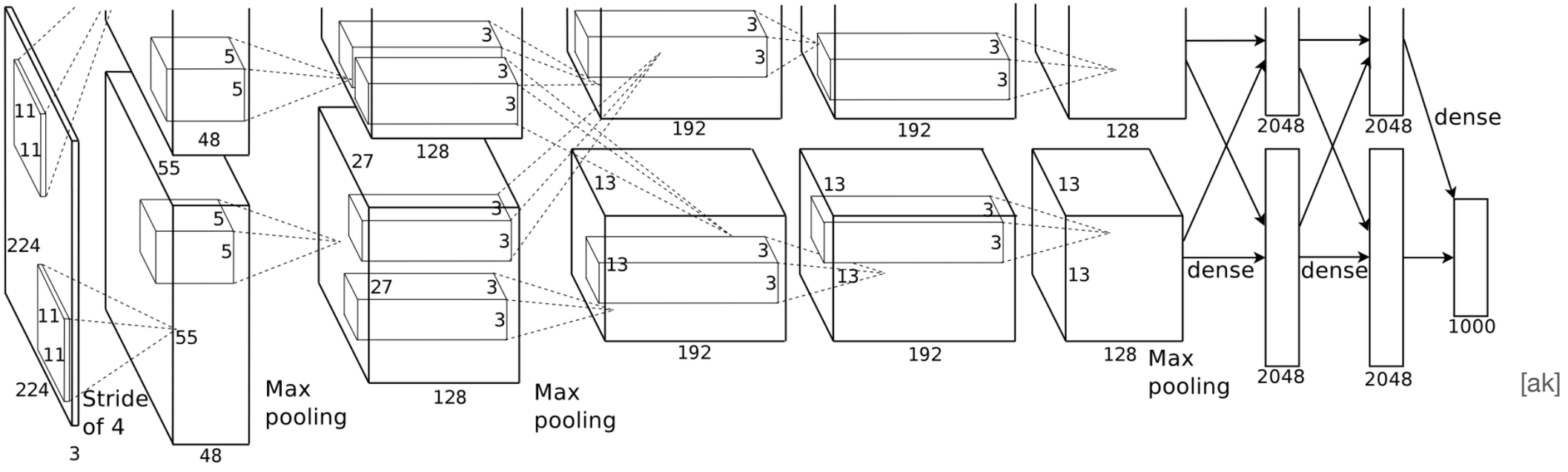
convolutional layer: 384 kernels 3x3

convolutional layer: 256 kernels 3x3

fully connected layer: 4096 neurons

fully connected layer: 1000 neurons (output)

8 layers
60M parameters



[ak]

AlexNet

- Further tricks used:
  - ReLU
  - Data Augmentations
  - Overlapping Pooling (size:3, stride:2)
  - DropOut
  - Weight Decay

VGG

VGG16 VGG19

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Paper

16/19 layers
138/144M parameters

[ks]

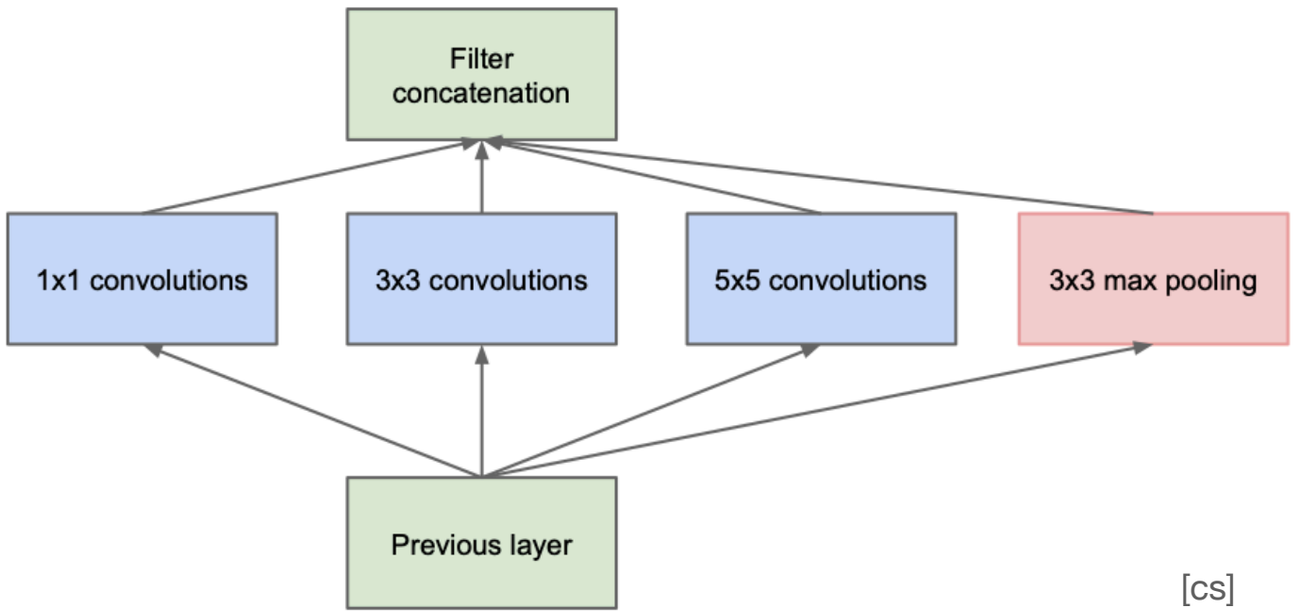
VGG

- Main contribution: Multiple small kernels instead of one large kernel.
- A large kernel (e.g. 7x7) has an accordingly large receptive field.
- A small kernel (e.g. 3x3) only has a small receptive field.
- But two 3x3 kernels have a receptive field of 5x5 and three 3x3 kernels have a receptive field of 7x7.
- What is the gain?
    - More layers → more non-linearities, more discriminative activation function.
    - Less weights: $3 \cdot (3^2C^2) = 27C^2$ vs. $7^2C^2 = 49C^2$, where C is the number of input and output channels.

GoogLeNet

[Paper]

Inception Module (naive version)



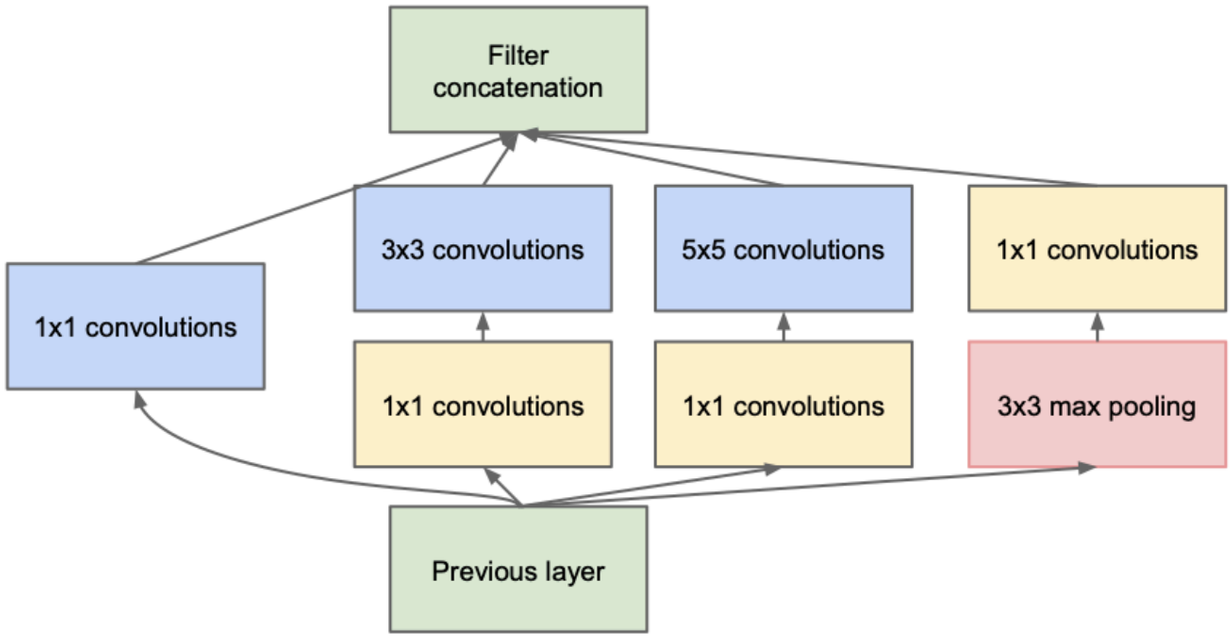
[cs]

1. Apply kernels of different size to one layer.

2. Concatenate the resulting feature maps.

GoogLeNet

Inception Module (clever version)



[cs]

Include 1x1 convolutions to reduced depth of feature map and thus reduce computation.

GoogLeNet

22 layers
6.7M parameters

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

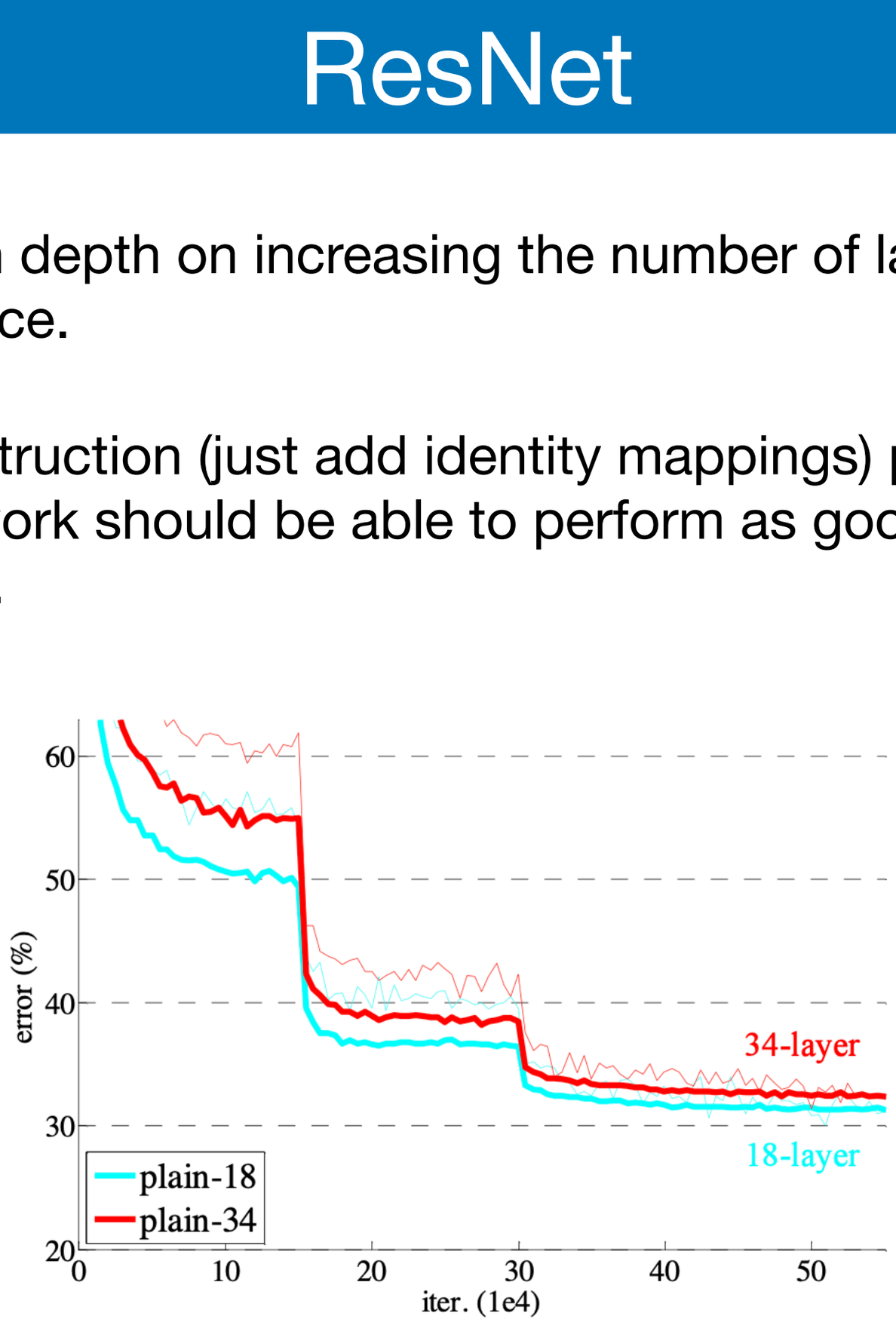
[cs]

ResNet

[Paper]

- From a certain depth on increasing the number of layers hurts the performance.

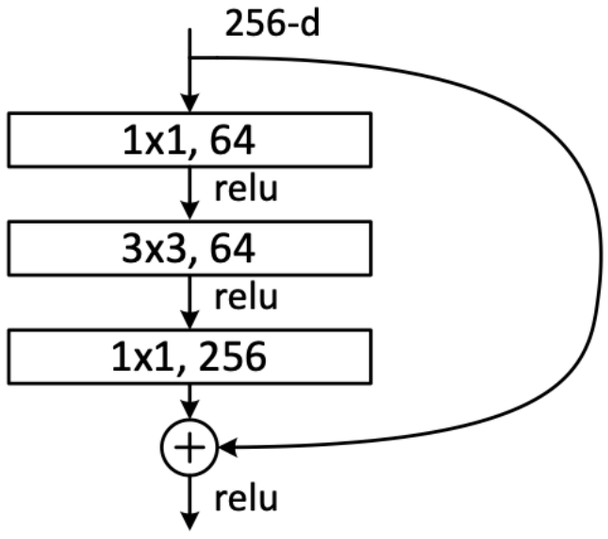
- A simple construction (just add identity mappings) proves that a deeper network should be able to perform as good as the shallower one.



[kh]

ResNet

Residual Block



[kh]

- After performing a convolutional block (e.g. depth reduction, 3x3 convolution, depth increase) the input and the output feature maps are **summed up**.
- This way it will be easier to approximate the identity mapping, which might be advantageous sometimes.

ResNet

Up to 152 layers
23M parameters (in popular ResNet50)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

[kh]

Others

- Inception V2/V3/V4
- ResNext
- NASNet
- Squeeze and Excitation
- ...

Feature Visualization

Feature Visualization

- Deep neural networks are black boxes.
- There is no direct way to ask a neural network why it made a certain decision.
- In the case of CNNs however there are different techniques that try to visualize what different parts of the network are looking for.

Most Activating Inputs

- A very simple way to visualize what a certain neuron looks for is to look at the input that excited this neuron the most

[mz]

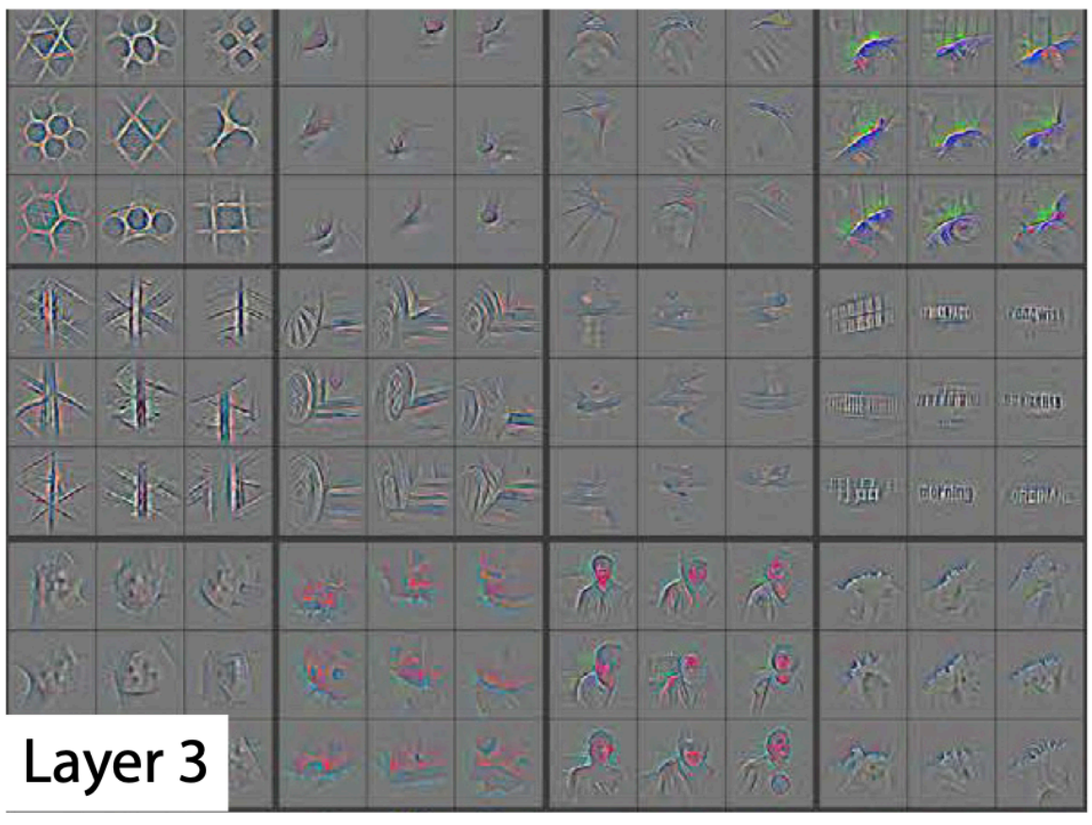
- But this only partly reveals the neurons preference.

Deconvolutional Neural Networks

Deconvolutional Neural Network

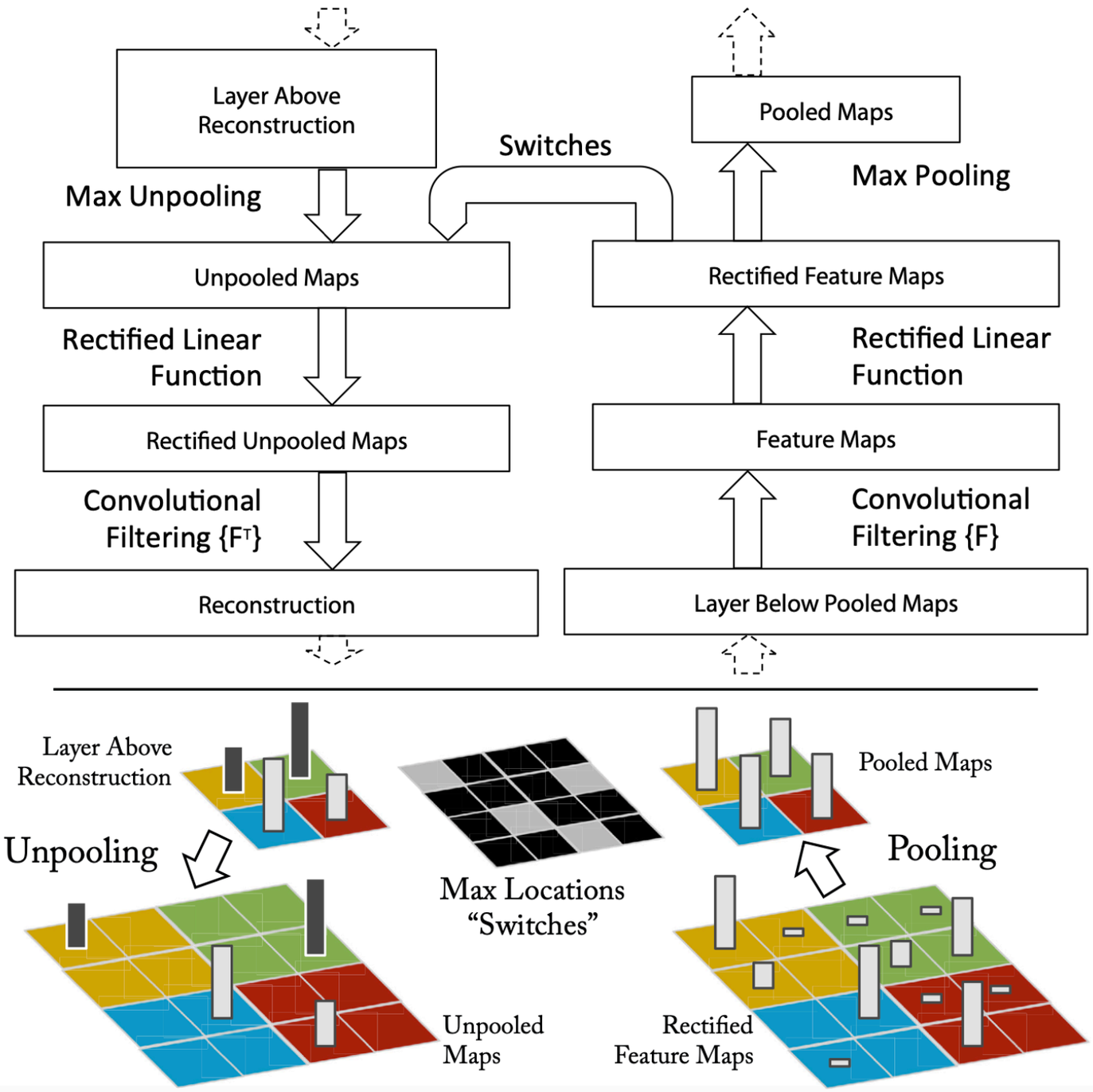
- Zeiler & Fergus (2014) introduced the deconvolutional operation, which can map the activations of a feature map back to the image.

- This visualizes which parts of the input excited the neuron.



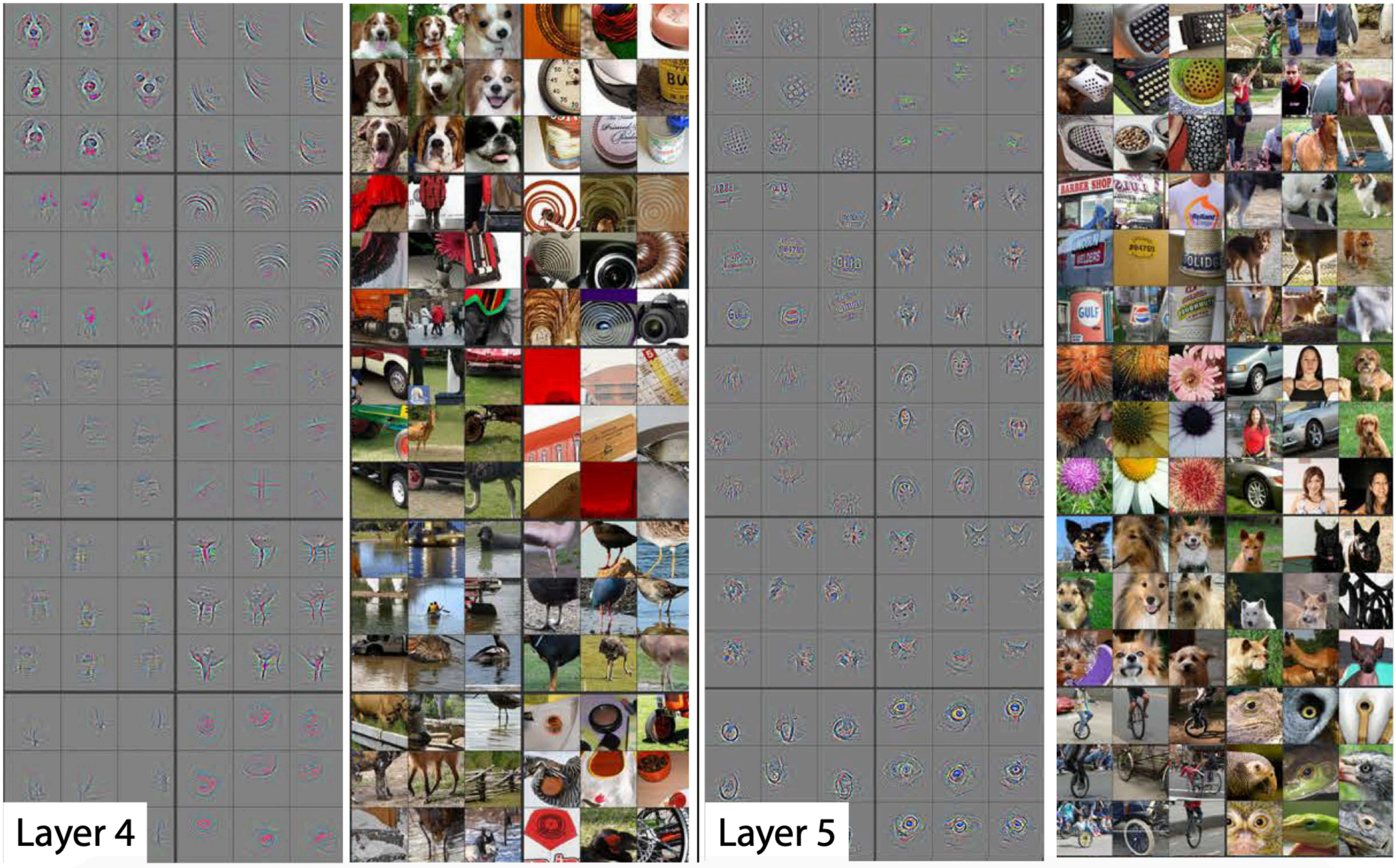
[mz]

Deconvolutional Neural Network



[mz]

Deconvolutional Neural Network



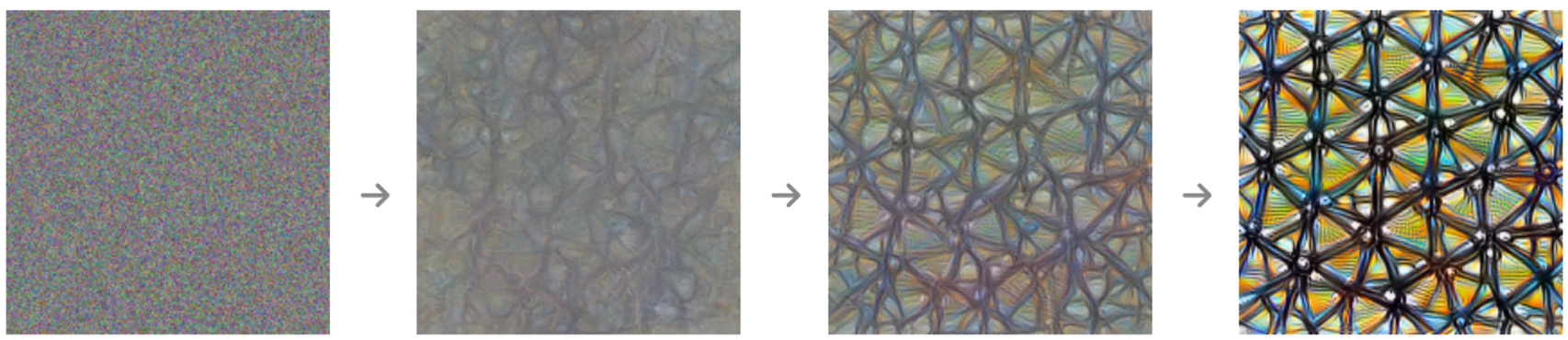
[mz]

Visualization by Optimization

Visualization by Optimization

Paper

- The more popular visualization technique today is that of optimization.
- Starting with a random noise image we optimize this image until it excites a certain neuron.



[co]

Visualization by Optimization

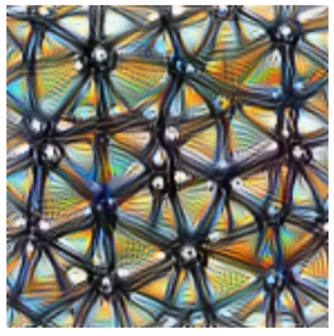
- This optimization works exactly as the training.
- However now the network weights are fixed and the input are the changeable parameters.
- The loss function is the excitement of the neuron and is maximized.

Visualization by Optimization

- With this technique we can visualize a lot of different parts of the neural network:

Neuron

`layer`$_n$`[x,y,z]`



Channel

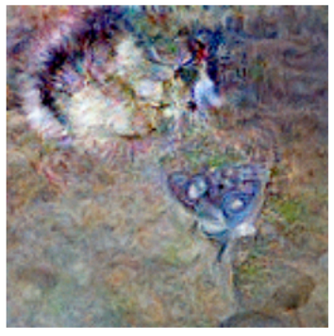
`layer`$_n$`[:,:,z]`

Layer/DeepDream

`layer`$_n$`[:,:,:]`^{2}

Class Logits

`pre_softmax[k]`



Class Probability

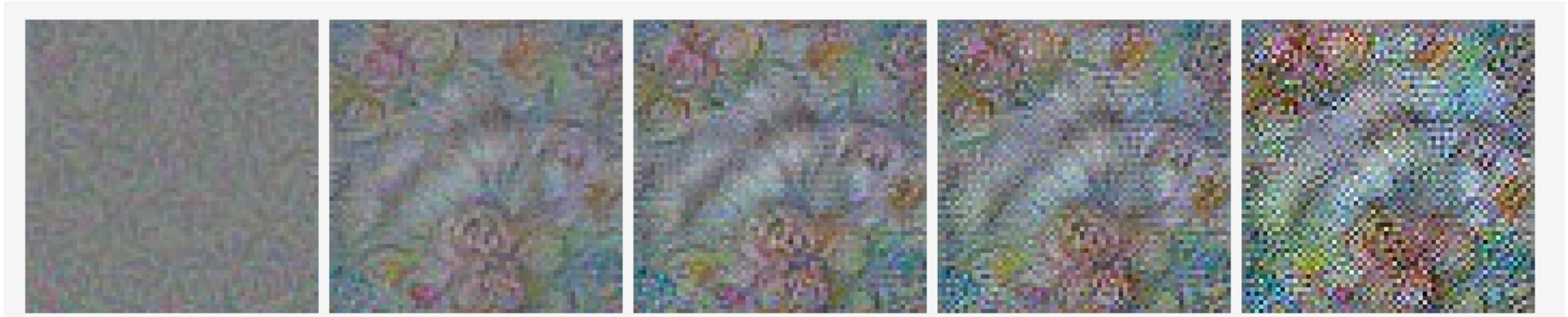
`softmax[k]`

[co]

Visualization by Optimization

- It is important to understand that these beautiful images are only possible through regularization.

- Without regularization the resulting images are mainly very high frequency blurs.



[co]

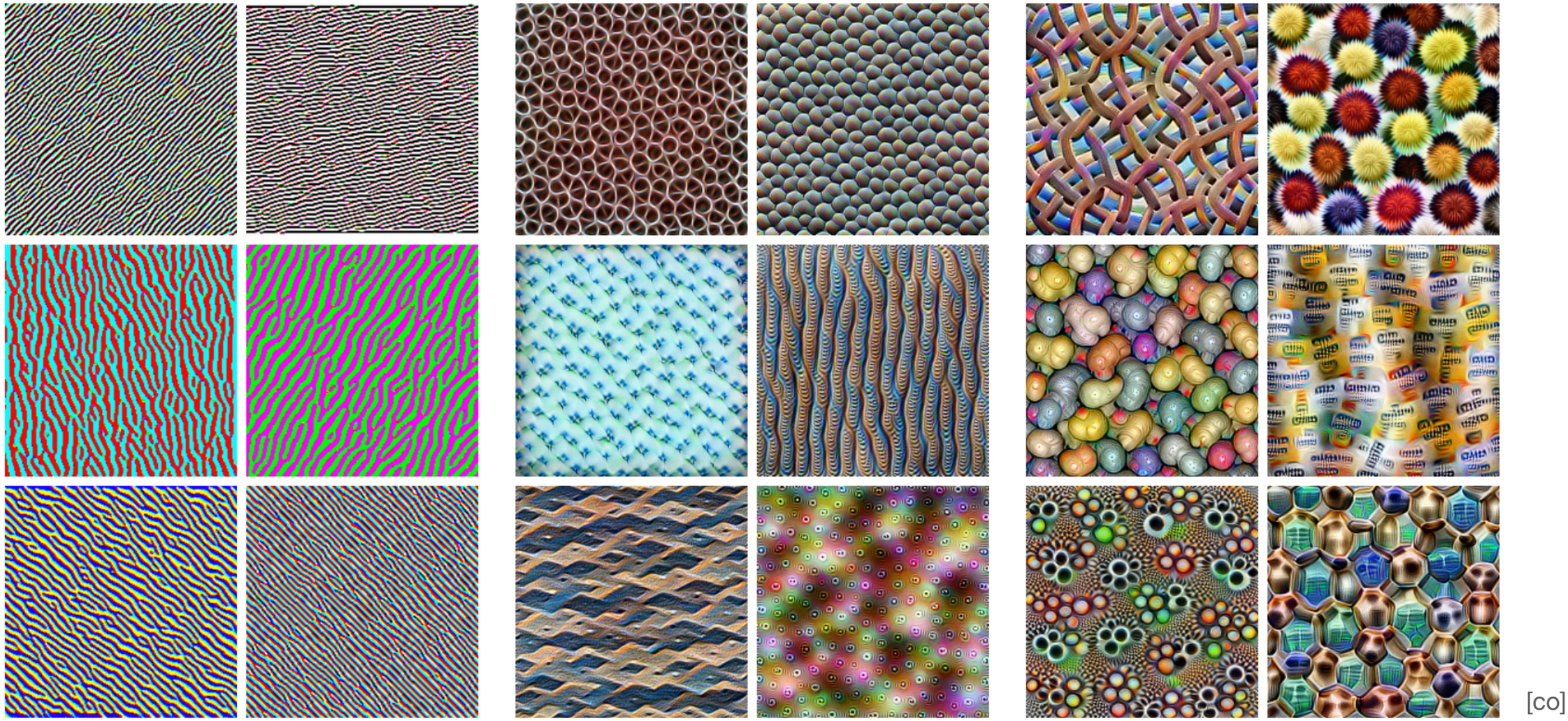
- There are two principle ways to do this:

  - Blur (gaussian) the image in each step.

  - Include a high frequency penalty into the loss function.

Visualization by Optimization

But if done right this technique can produce beautiful images.



Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a)

[co]

Visualization by Optimization

But if done right this technique can produce beautiful images.

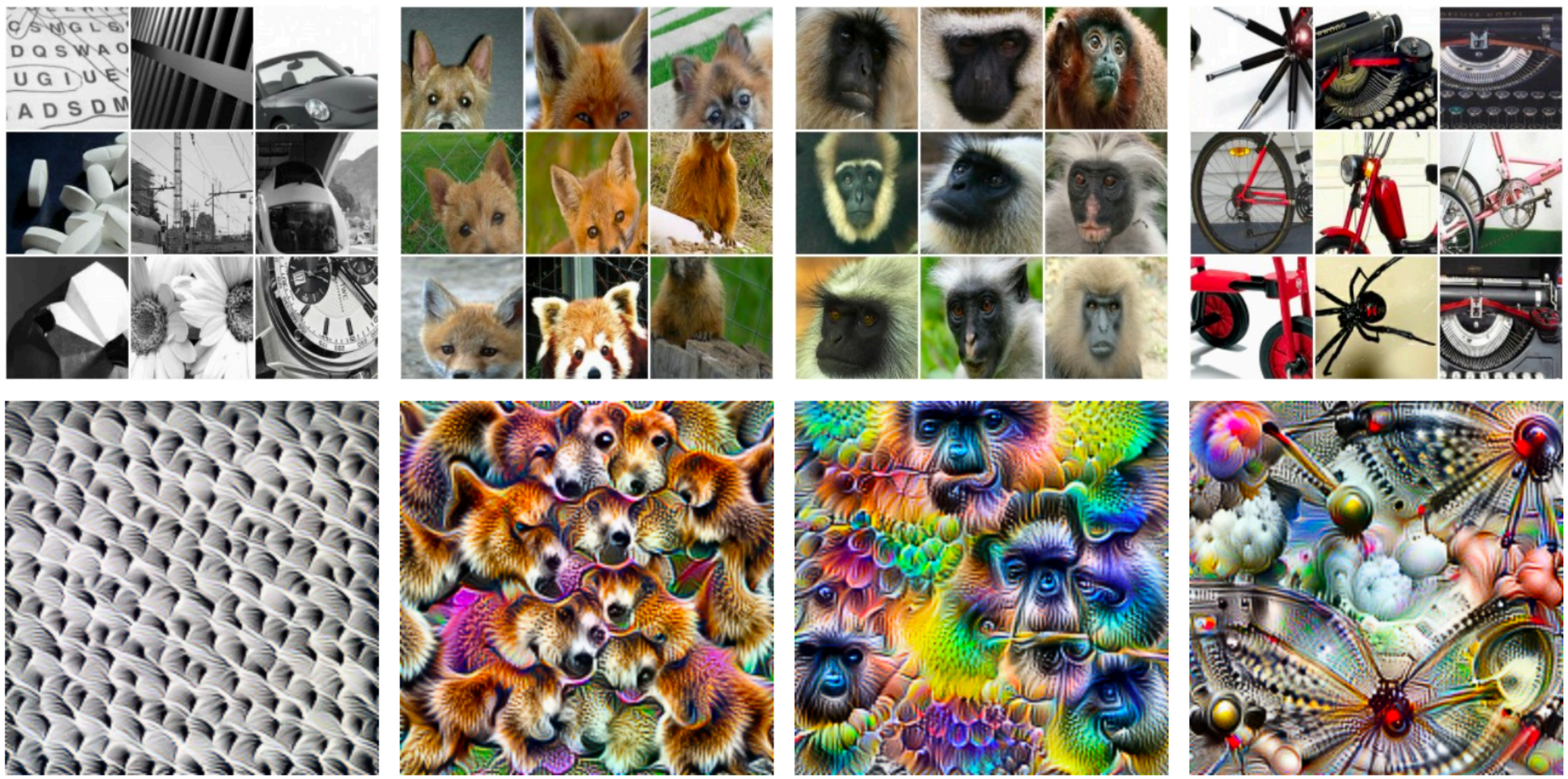
Parts (layers mixed4b & mixed4c)

Objects (layers mixed4d & mixed4e)

[co]

Visualization by Optimization

- We can also visualize other directions in activation space (e.g. the fifth feature map should be excited twice as much as the sixteenth feature map).



mixed3a, random direction *mixed4c, random direction* *mixed4d, random direction* *mixed5a, random direction* [co]

More on Interpretability

- The Building Blocks of Interpretability
- Activation Atlas

Fooling Deep Neural Networks

Fooling Deep Neural Networks

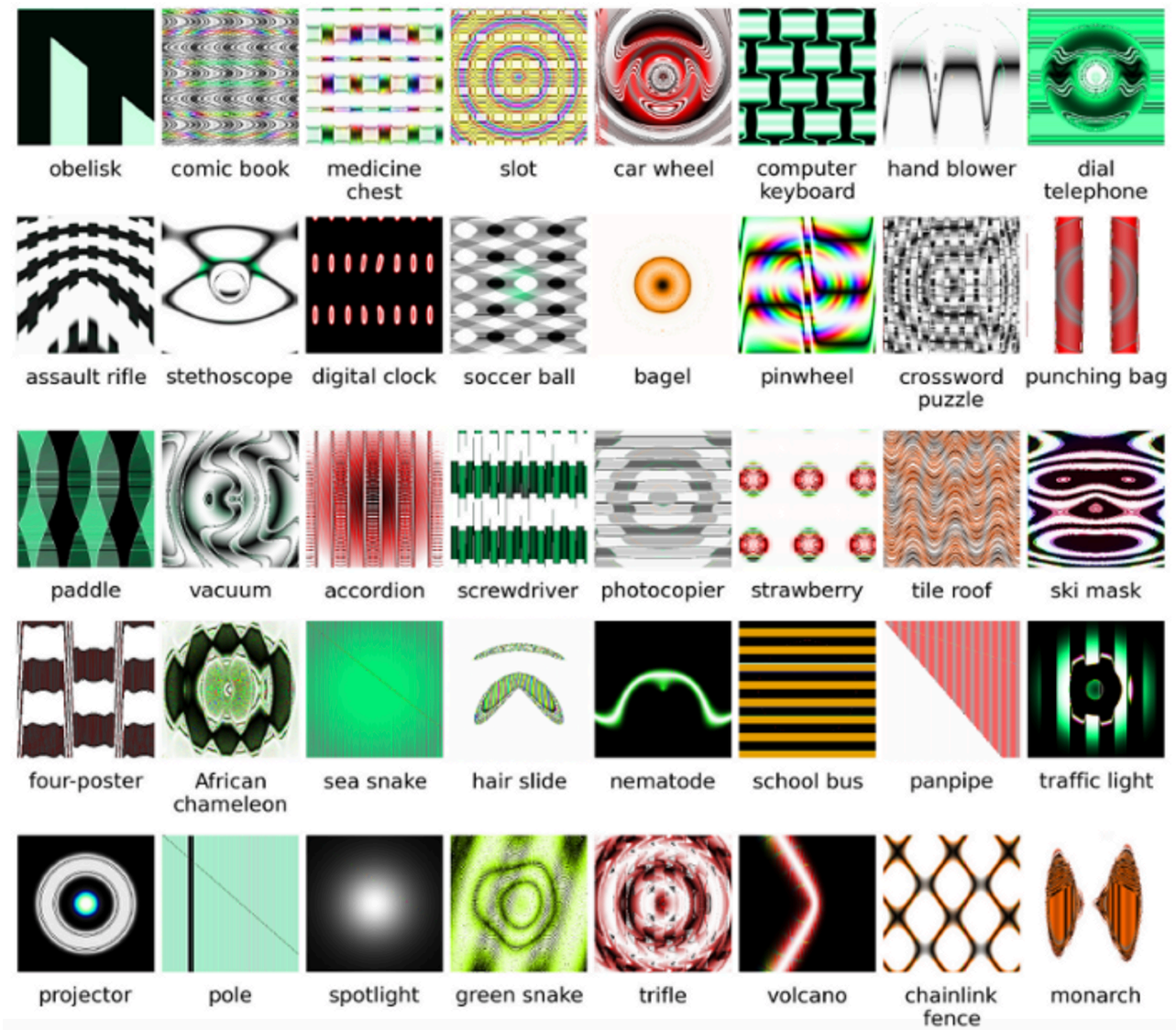
- All these approaches try to understand what neural networks are seeing and how they solve complex visual recognition tasks.

- But other experiments show that neural networks are easily fooled.

- It seems they are not as intelligent as one might think.

Fooling Deep Neural Networks

- Similar to feature visualization genetic algorithms can be used to produce images that highly activate an output neuron, but are not recognizable to humans.

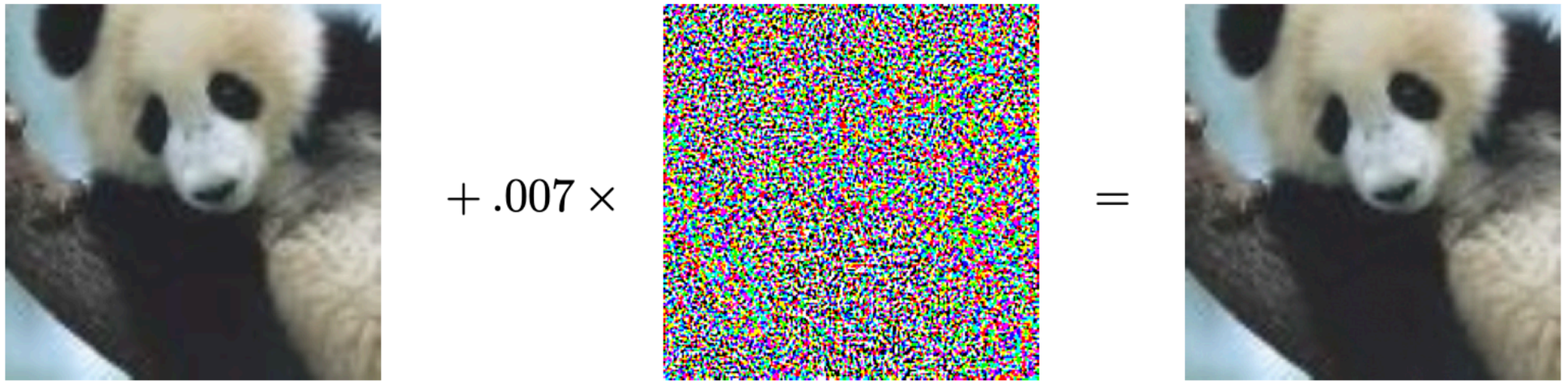


[an]

Adversarial Examples

- Adversarial examples are images that are produced by so called adversarial attacks.

- In an adversarial attack, given an image, one computes a noise that if added to the image completely fools the network.



[ig]

Example: Fast Gradient Sign Method

- First an input x with label y is fed to the network.
- $J(\theta, x, y)$ is the categorical cross entropy, i.e. the negative log probability of the class y.
- The gradient for x is computed $\nabla_x J(\theta, x, y)$.
- We compute the $sign$ of the gradient (i.e. +1 for positive values, -1 for negative values).
- To find the smallest possible update we multiply with the smallest ϵ such that the noise actually produces a new image (e.g. $\epsilon = 0.07$ in the case of real-valued images).

Other Methods

- DeepFool
- Projected Gradient Descent
- Logit-space Projected Gradient Ascent

These methods are so called white-box attacks.
The attacker needs access to the model.

Black-Box Adversarial Attacks

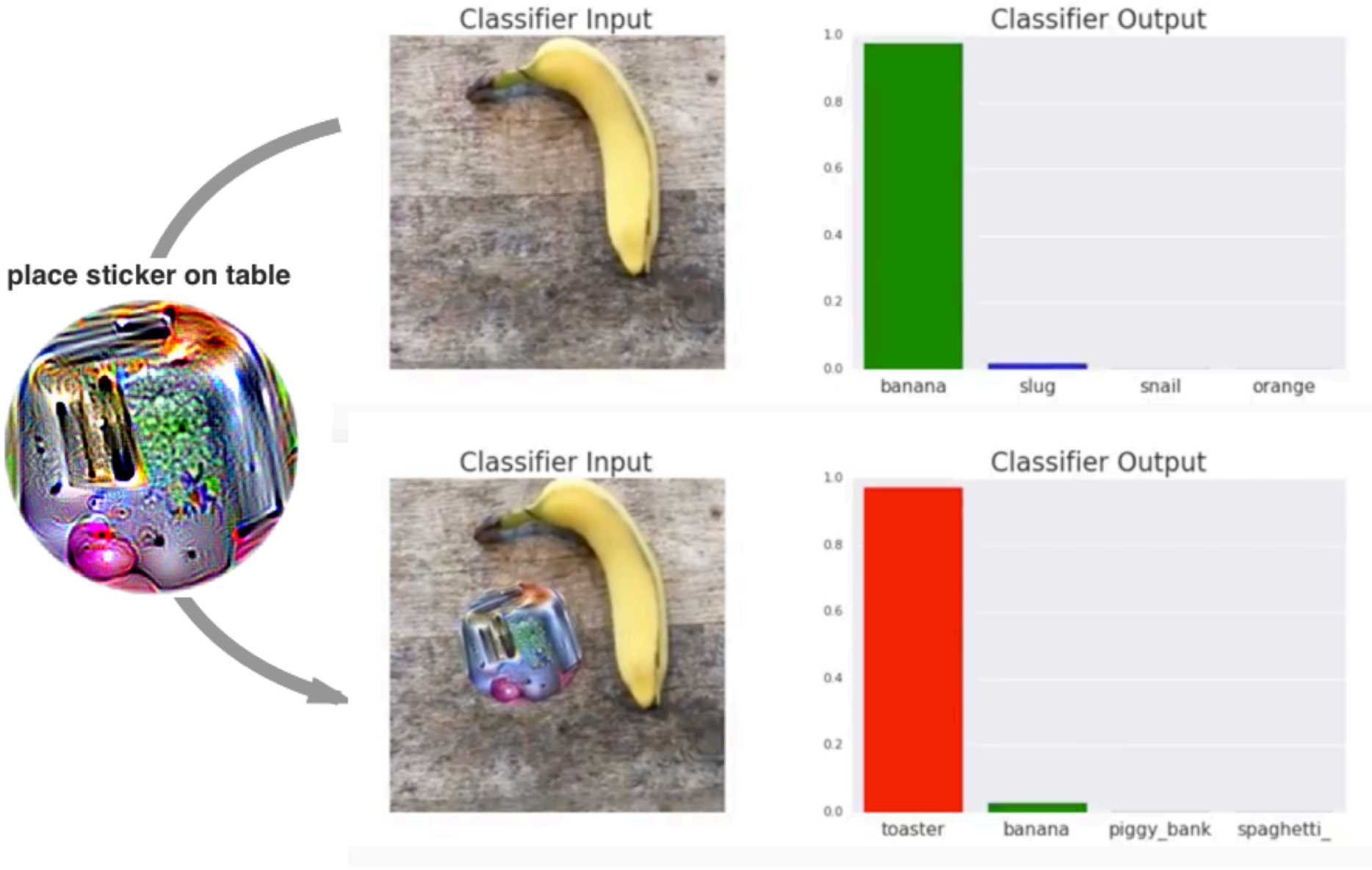
- How to create an adversarial example, if one does not have access to the model?

- In general the idea is to build a new model, which approximates the underlying model and then use it to create adversarial examples.

- See for example: Practical Black-Box Attacks against Machine Learning

Adversarial Patch

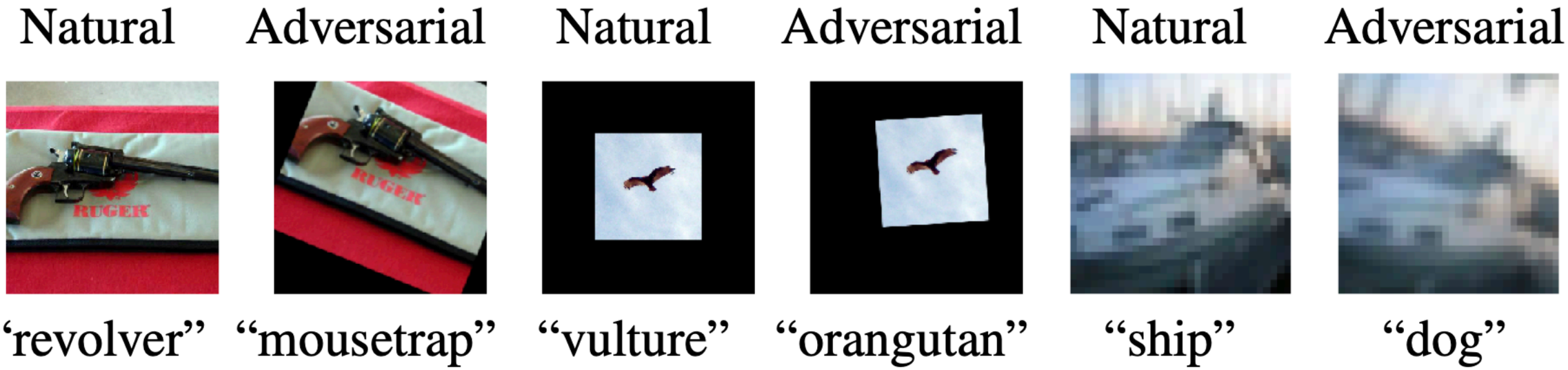
- An adversarial patch can be optimized to be applied to any image and even real objects to fool a neural network into a specific prediction.



[tb]

Adversarial Transformations

- Even simple transformations can fool deep neural networks:



[Ie]

Resources

[ak] A. Krizhevsky et al., "Imagenet classification with deep convolutional neural networks" *Proceedings of the NeurIPS 2012,* 2012.

[mz] M. Zeiler & R. Fergus, “Visualizing and understanding convolutional networks" *Computer Vision - ECCV 2014,* 2014.

[ks] K. Symonian & A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, *ICLR 2015*, 2015.

[cs] C. Szegedy et al., “Going deeper with Convolutions”, *CVPR 2015,* 2015.

[kh] K. He et al., “Deep Residual Learning for Image Recognition”, *CVPR 2016*, 2016.

[co] C. Olah et al., “Feature Visualization”, *Distill,* 2017.

[an] A. Nguyen et al., “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images”, *CVPR 2015*, 2015.

[ig] I. Goodfellow et al., “Explaining and Harnessing Adversarial Examples”, *ICLR,* 2014.

[tb] T. Brown et al., “Adversarial Patch”, *Proceedings of the NeurIPS 2017,* 2017.

[le] L. Engstrom et al., “Exploring the Landscape of Spatial Robustness ”, *arXiv:1712.02779*, 2019.