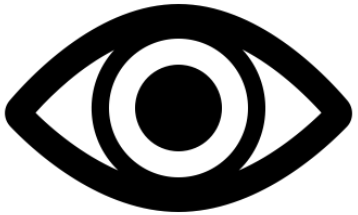
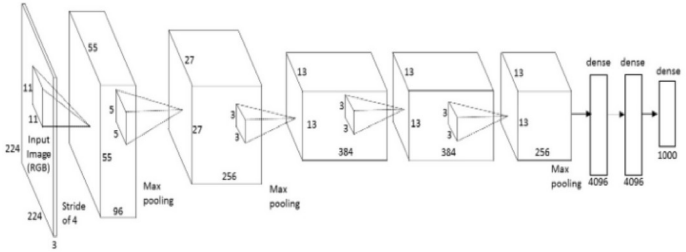


Advanced Image Processing
What does a CNN learn?

SYDE 671

Review of previous content

Training Neural Networks



- Build network architecture and define loss function
- Pick hyperparameters – learning rate, batch size
- Initialize weights + bias in each layer randomly

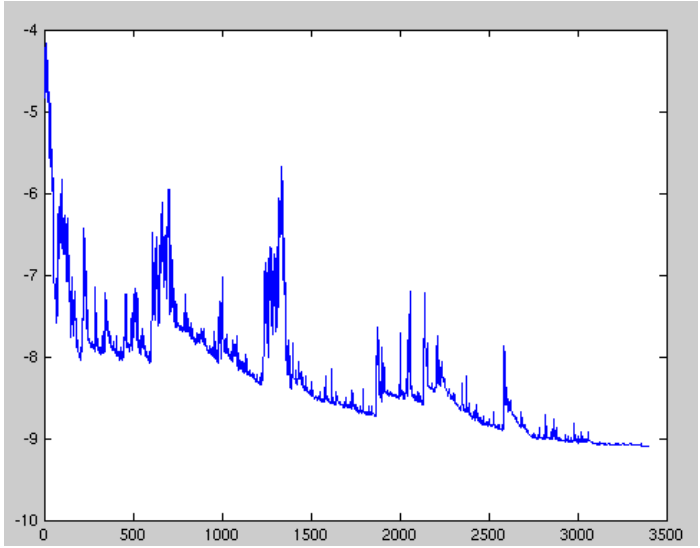
- While loss still decreasing
  - Shuffle training data
  - For each data point *i=1…n (maybe as mini-batch)*
    - *Gradient descent*
  - Check validation set loss

"Epoch"

Stochastic Gradient Descent

Try to speed up processing with random training subsets

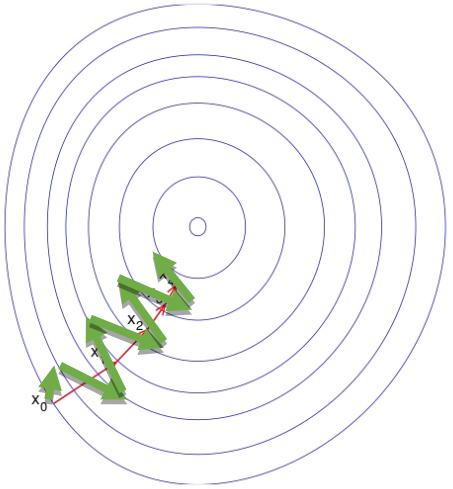
Loss will not always decrease (locally) as training data point is random, but converges over time.



Momentum

Gradient descent step size is weighted combination over time to dampen ping pong.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \left(\alpha \left[\frac{\partial L}{\partial \boldsymbol{\theta}} \right]_{t-1} + \left[\frac{\partial L}{\partial \boldsymbol{\theta}} \right]_t \right)$$



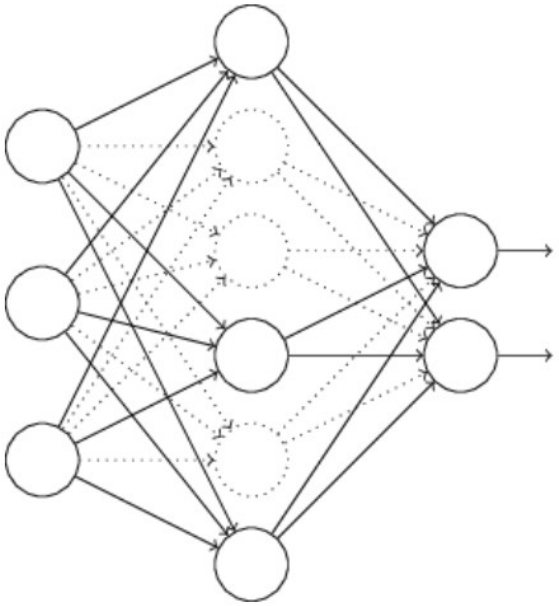
Wikipedia

Regularization

- Penalize weights for simpler solution
  - Occam's razor

$$C = C_0 + \lambda \sum_W W^2$$

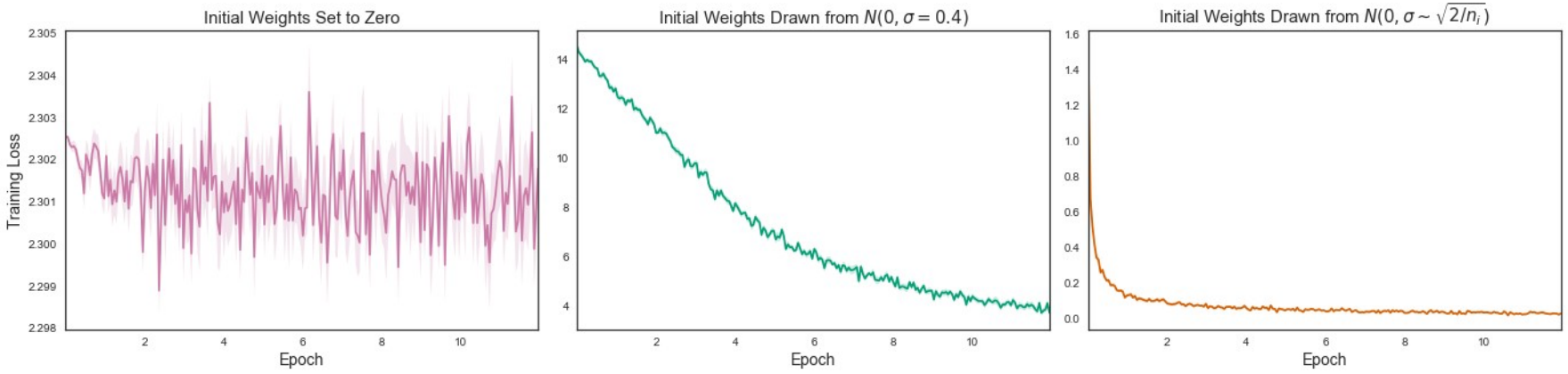
- Dropout half of neurons for each minibatch
  - Forces robustness



Weight initialization

Setting zero weights makes all neurons equivalent as there is no difference in the gradient computed across neurons. Called “symmetric updates”.

Setting zero bias is OK; often we standardize the data beforehand (subtract mean, divide by std. dev) so a zero bias is a good initialization.



Good strategy: He et al. 2015: https://arxiv.org/pdf/1502.01852.pdf

(First to surpass human-level performance on ImageNet)

For ReLU, draw random weights from Gaussian distribution with variance = 2 / # inputs to layer

[Image Andre Perunicic]

What is activation function for?

To allow multiple layers; to avoid resulting composition of linear functions collapsing to a single layer.

Difference between CNN and convolution in feature extraction?

No difference! Same operation [correlation/convolution]

Why do we shave off pixels?

We only use the valid region of convolution; typically no padding.

Some recent works special case these edge convolutions.

Why multidimensional kernels?

We wish to convolve over the outputs of many other learned kernels -> ‘integrating’ information via weighted sum of previous layer outputs.

How to know which kernels to use in 2nd+ convolution layers?

Gradient descent + back propagation learns them.

How to set weights on fully connected layers?

Gradient descent + back propagation learns them.

What even is back propagation again?

Assignment 4 Q5 has some good references.

How do we decide the best parameters for network architecture?
For less complicated situations, we can use 'guess & check' [trial and error]. Is there any method?

'Grid search' -> trial and error

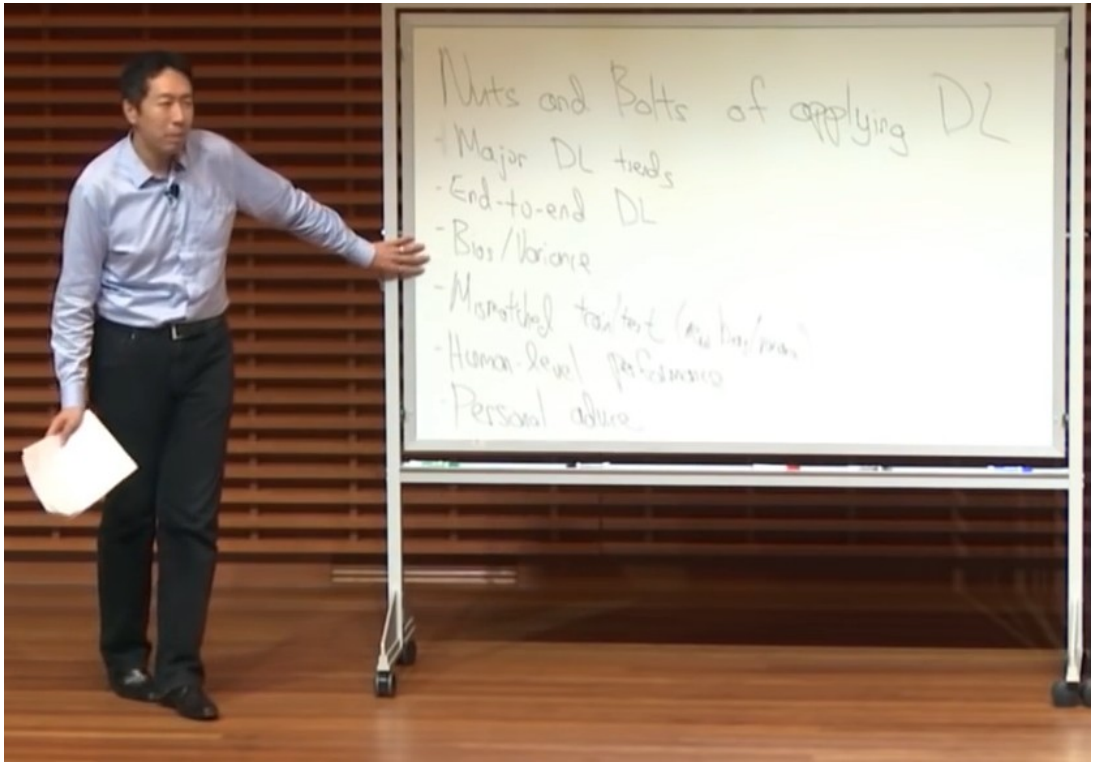
'Bayesian optimization' -> meta-learning; optimize the hyperparameters

General architecture strategy:

Architectures with "Bottleneck" -> extract information (through the learned kernels) and learn to squeeze representation into a smaller number of parameters ("distilling the features").

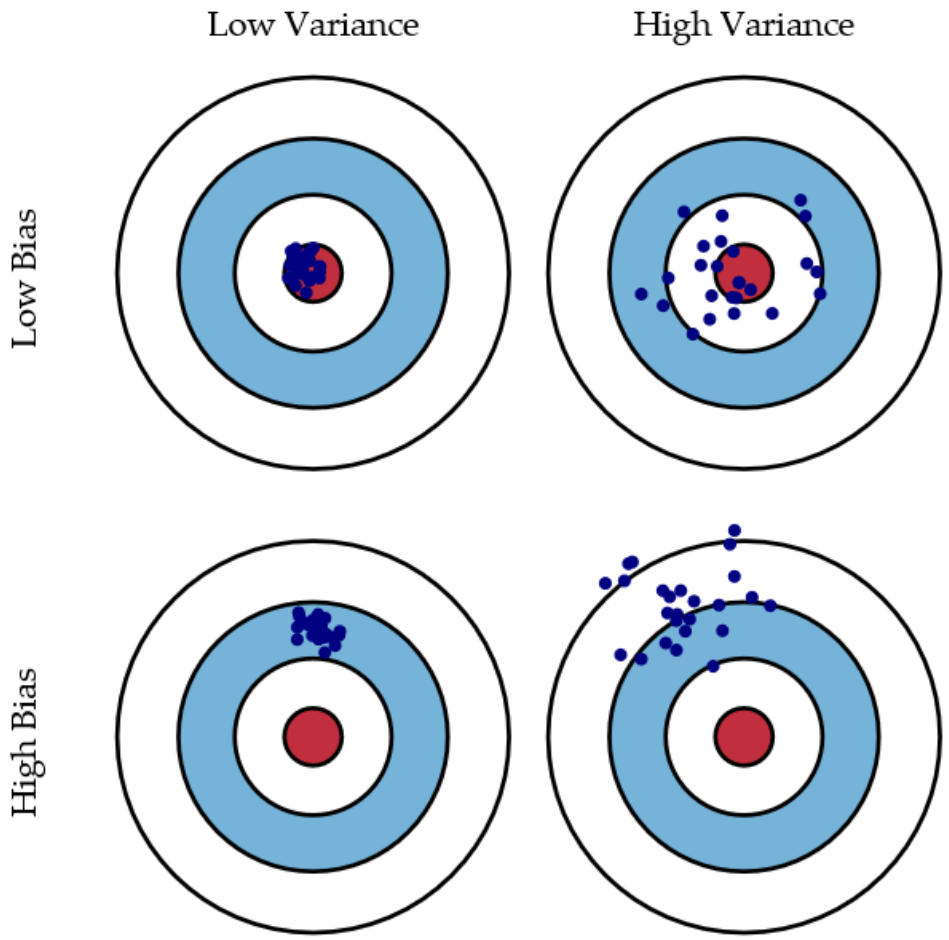
The Nuts and Bolts of Building Applications using Deep Learning

- Andrew Ng - NIPS 2016
- https://youtu.be/F1ka6a13S9I

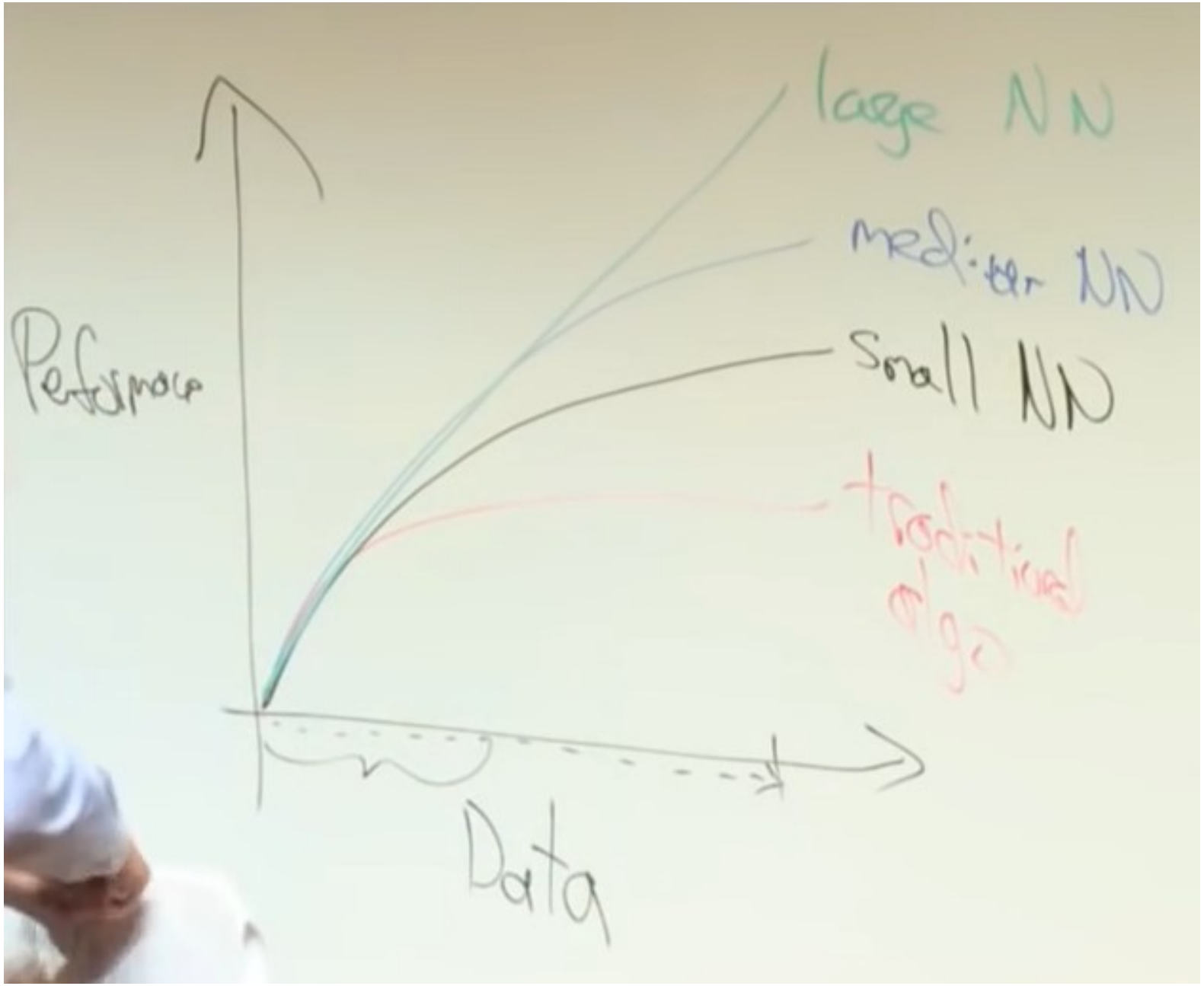


Bias/variance trade-off

"It takes surprisingly long time to grok bias and variance deeply, but people that understand bias and variance deeply are often able to drive very rapid progress." *--Andrew Ng*



Bias = accuracy
Variance = precision


large NN
medium NN
Small NN
traditional algo
Performance
Data

Go collect a dataset

- Most important thing:
  - Training data must represent target application!

- Take all your data
  - 60% training
  - 40% testing
    - 20% testing
    - 20% validation (or ‘development’)

Properties

- Human level error = 1%
- Training set error = 10%
- Validation error = 10.2%
- Test error = 10.4%

"Bias"

"Variance"

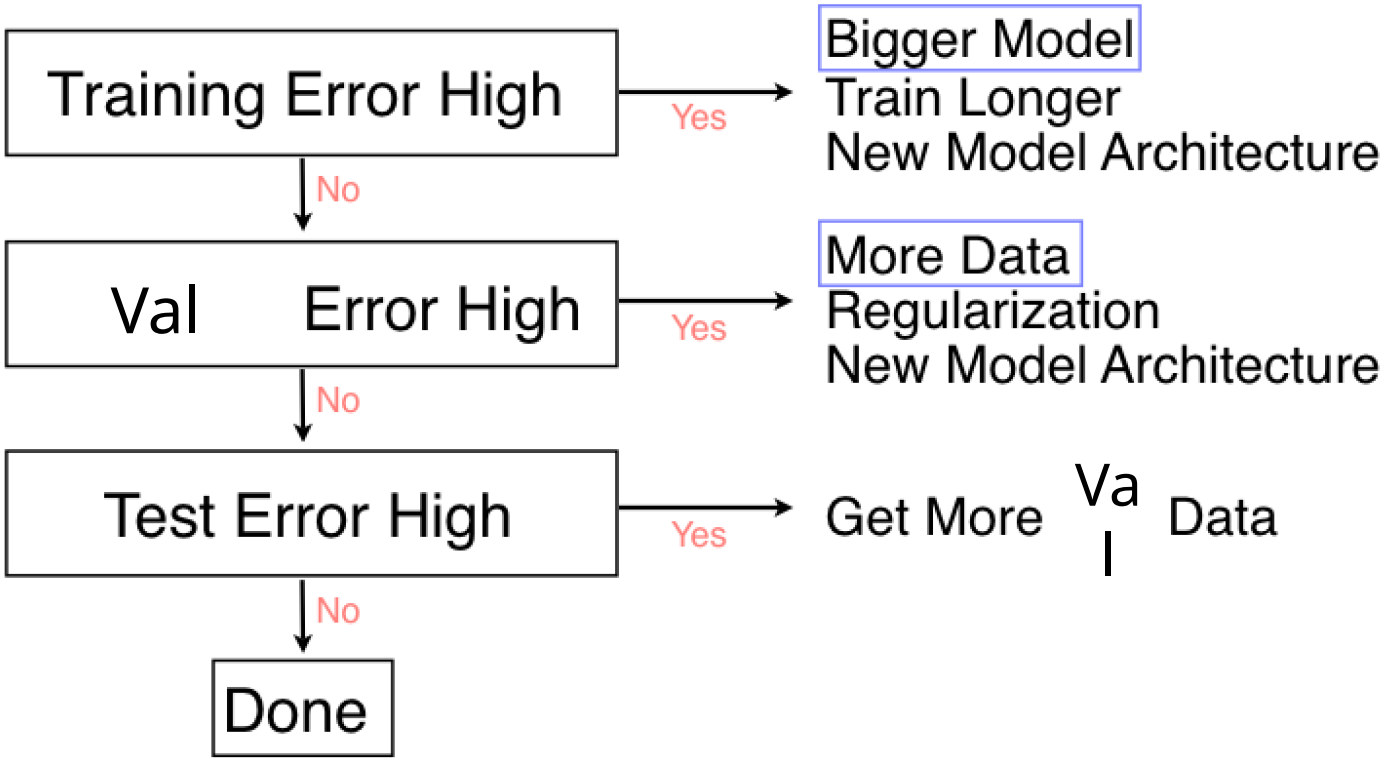
Overfitting to validation

In this case, I have high bias – my data isn't right.
If I had High variance = needs regularization

When something is not working…

…how do I know what to do next?

The Nuts and Bolts of Building Applications Using Deep Learning



[Andrew

My Neural Network isn't working! What should I do?

Created on Aug. 19, 2017, 5:56 p.m.

So you're developing the next great breakthrough in deep learning but you've hit an unfortunate setback: your neural network isn't working and you have no idea what to do. You go to your boss/supervisor but they don't know either - they are just as new to all of this as you - so what now?

Well luckily for you I'm here with a list of all the things you've probably done wrong and compiled from my own experiences implementing neural networks and supervising other students with their projects:

1. You Forgot to Normalize Your Data
2. You Forgot to Check your Results
3. You Forgot to Preprocess Your Data
4. You Forgot to use any Regularization
5. You Used a too Large Batch Size
6. You Used an Incorrect Learning Rate
7. You Used the Wrong Activation Function on the Final Layer
8. Your Network contains Bad Gradients
9. You Initialized your Network Weights Incorrectly
10. You Used a Network that was too Deep
11. You Used the Wrong Number of Hidden Units

Daniel Holden

http://theorangeduck.com/page/neural-network-not-working

Some online neural nets demos

Digit recognition

Neural Nets Playground

Neural Style Transfer

What does CNN learn?

Short cuts to AI

With billions of images on the web, it’s often possible to find a close nearest neighbor.

We can shortcut hard problems by “looking up” the answer, stealing the labels from our nearest neighbor.

So what is intelligence?

Weak AI:
The simulation of a ‘mind’ is a model for the ‘mind’.

Strong AI:
The simulation of a ‘mind’ is a ‘mind’.

"Can machines fly?"
Yes, aeroplanes exist.

"Can machines fly like a bird?"
No, because aeroplanes don’t flap.

"Can machines perceive?”
“Can machines understand?"
Are these question like the first, or like the second?

The Chinese room argument: https://plato.stanford.edu/entries/chinese-room/

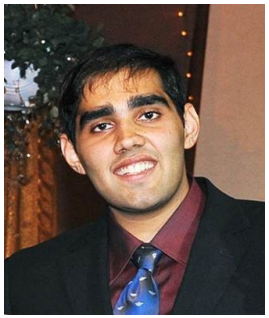
[Adapted from Norvig]

https://youtu.be/nnR8fDW3Ilo

Festo SmartBird [2011]

Object Detectors Emerge in Deep Scene CNNs

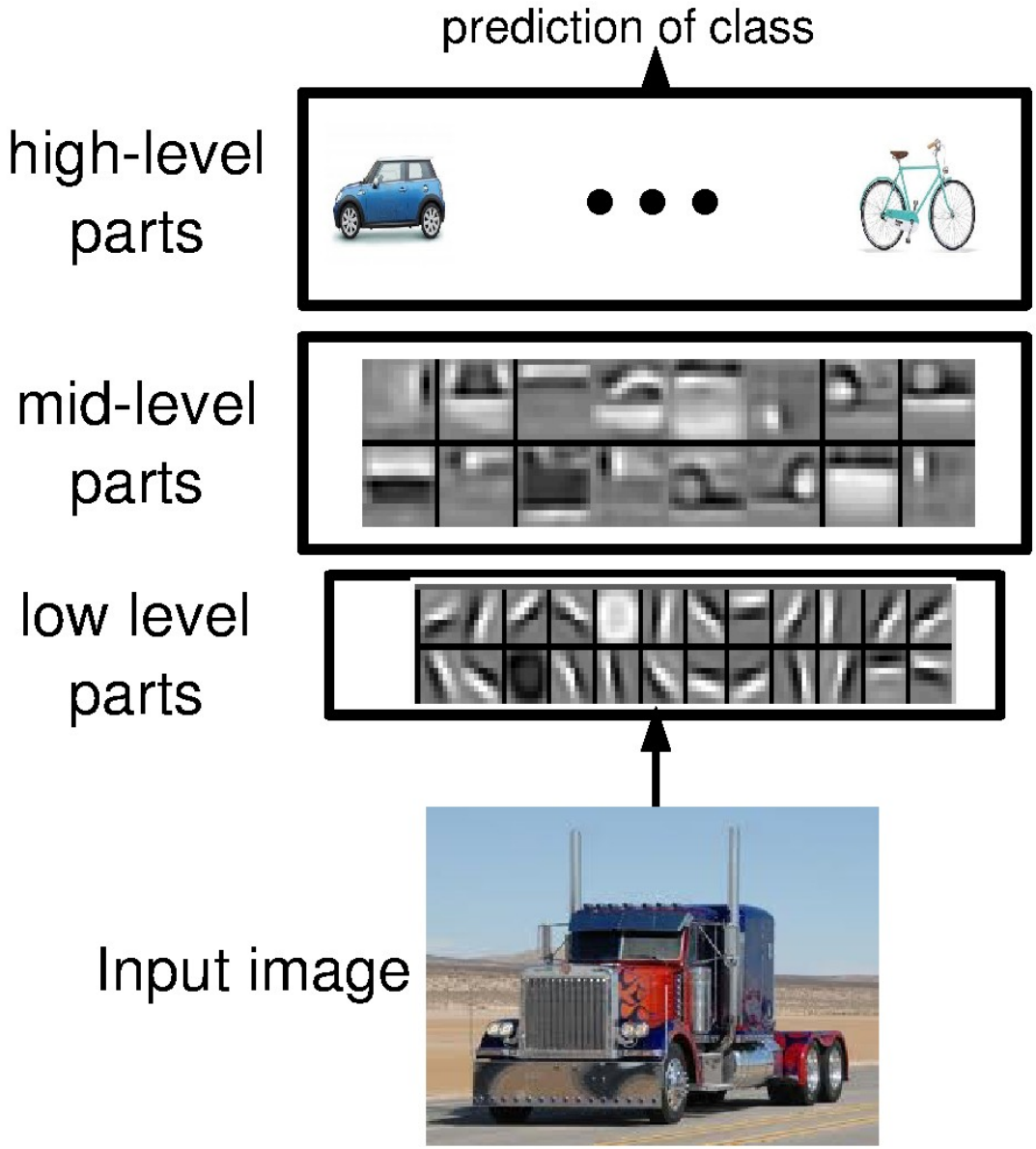
Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba




Massachusetts Institute of Technology

ICLR 2015

Interpretation

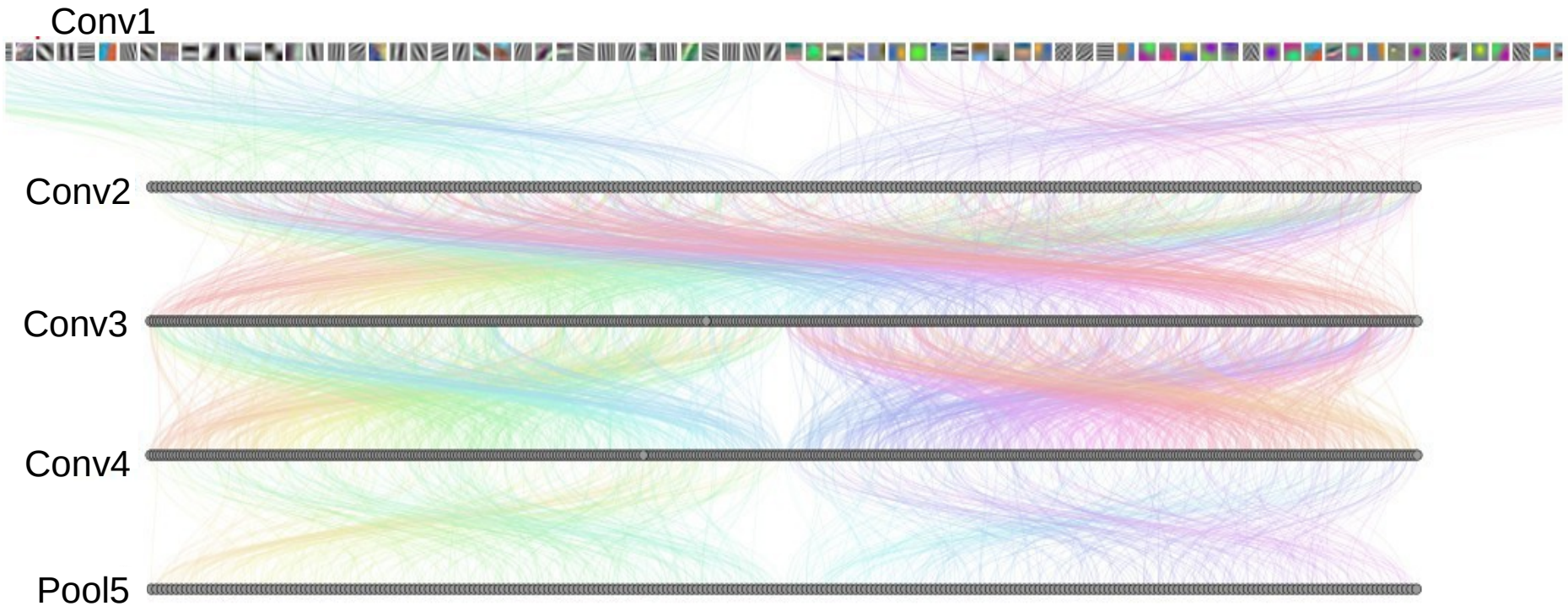


- distributed representations
- feature sharing
- compositionality

Lee et al. “Convolutional DBN's ...” ICML 2009

How Objects are Represented in CNN?

CNN uses **distributed code** to represent objects.

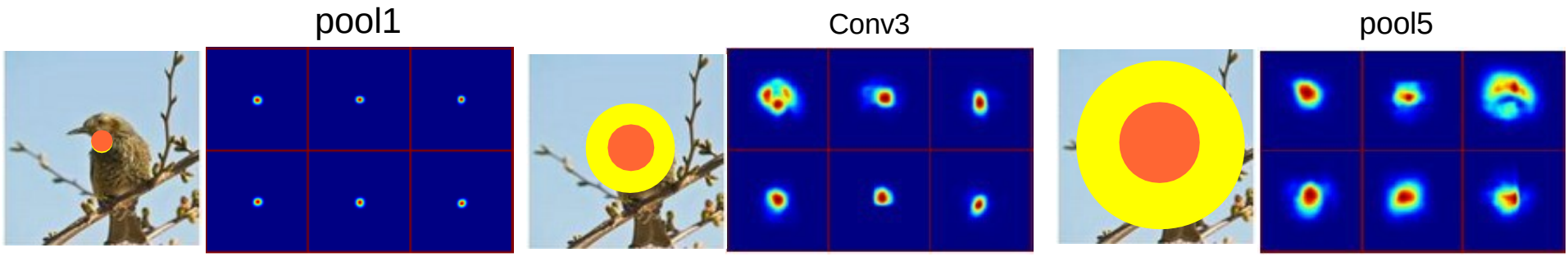


Agrawal, et al. Analyzing the performance of multilayer neural networks for object recognition. ECCV, 2014
Szegedy, et al. Intriguing properties of neural networks.arXiv preprint arXiv:1312.6199, 2013.
Zeiler, M. et al. Visualizing and Understanding Convolutional Networks, ECCV 2014.

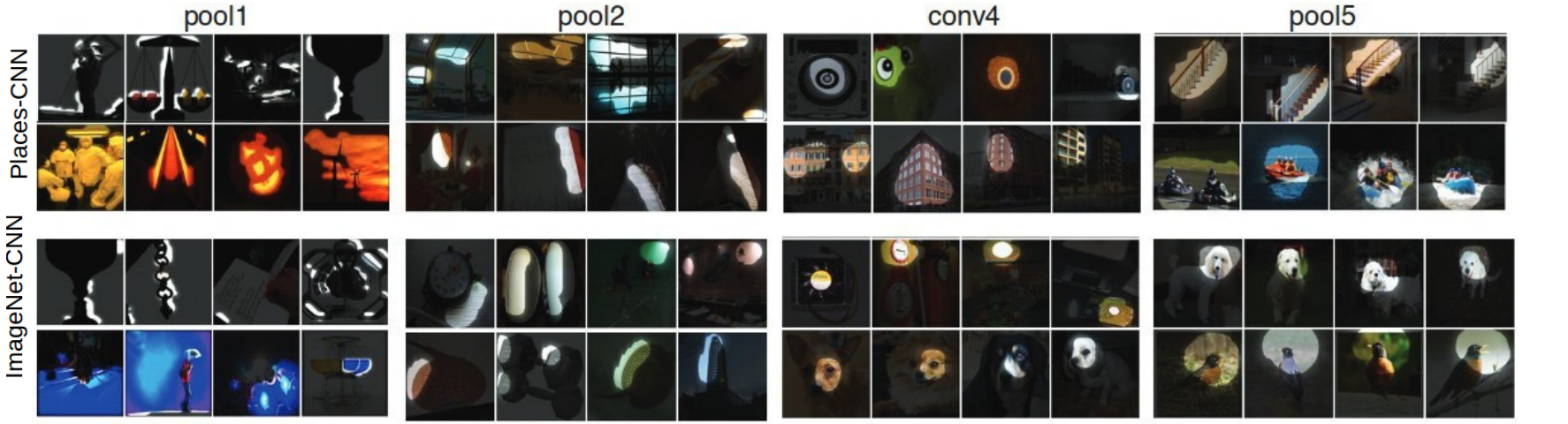
Estimating the Receptive Fields

Estimated receptive fields

Actual size of RF is much smaller than the theoretic size



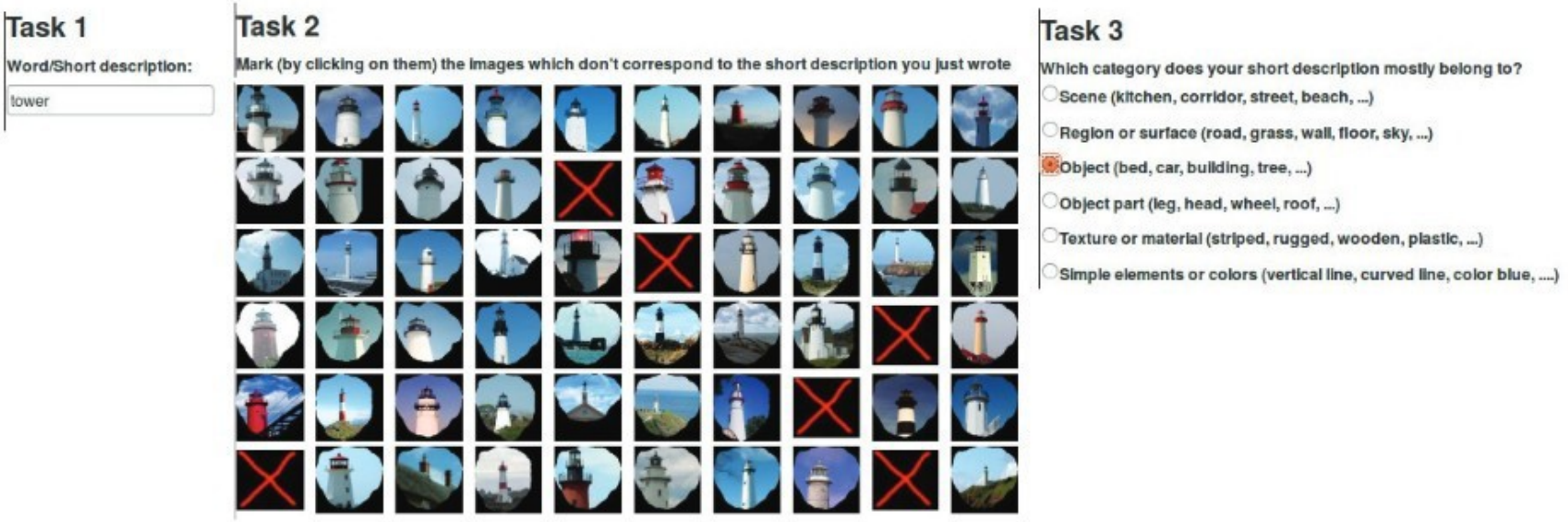
Segmentation using the RF of Units



More semantically meaningful

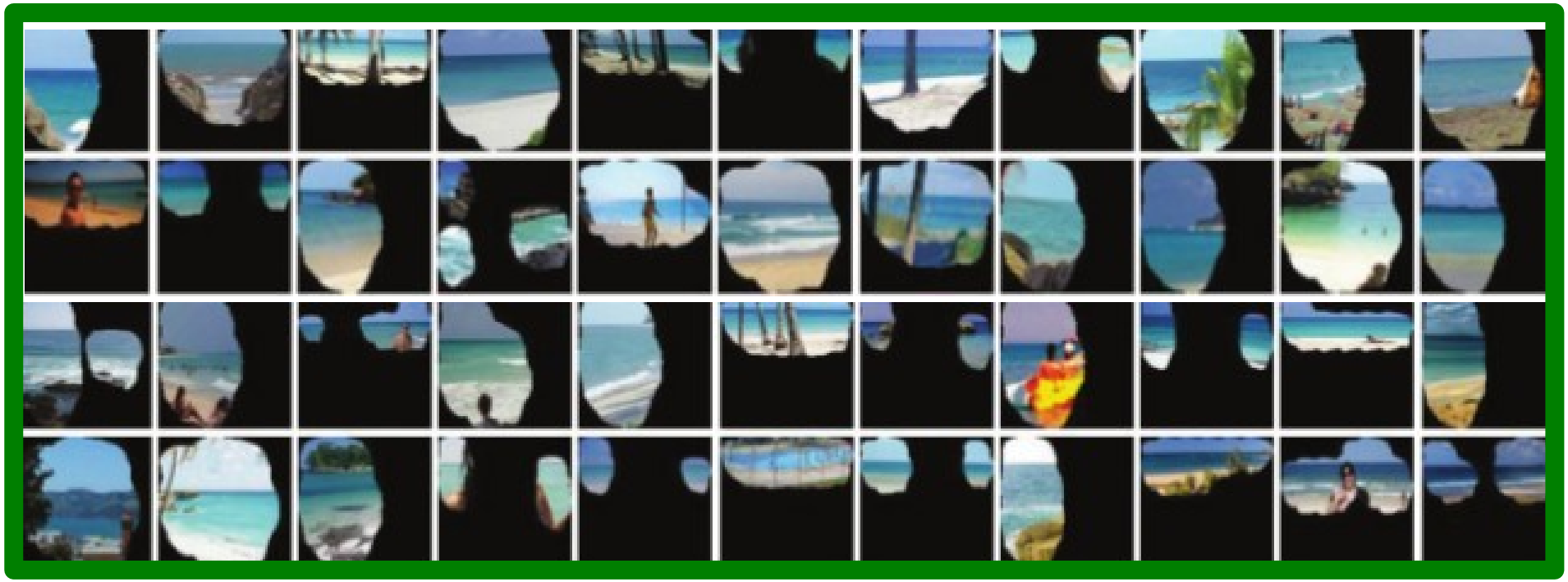
Annotating the Semantics of Units

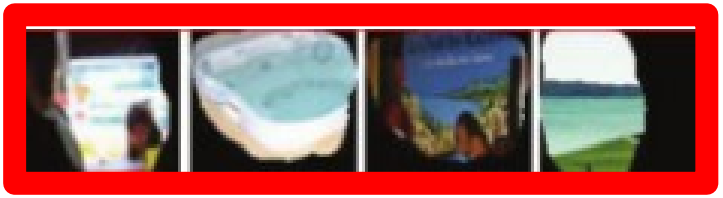
Top ranked segmented images are cropped and sent to Amazon Turk for annotation.



Annotating the Semantics of Units

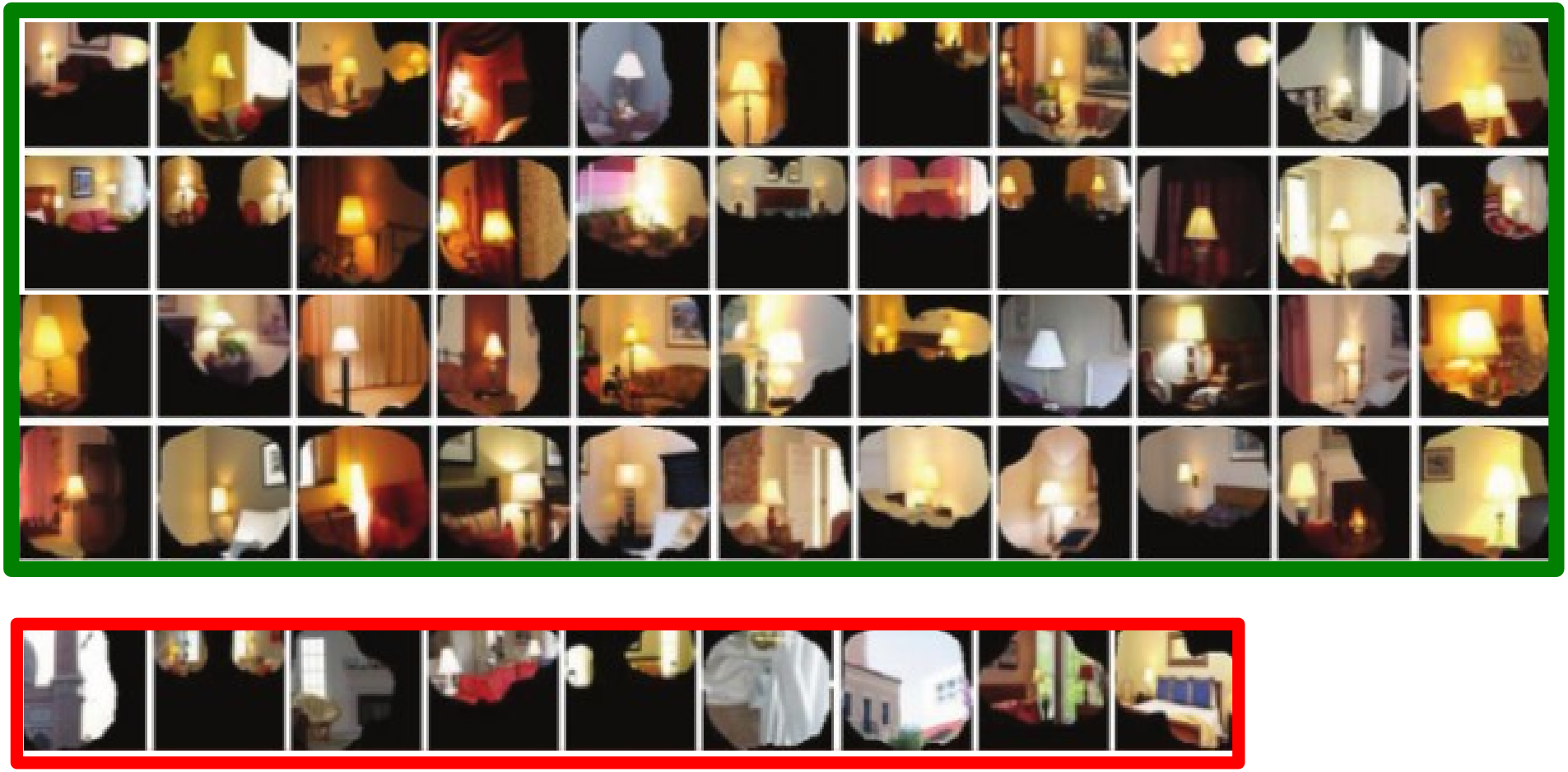
Pool5, unit 76; Label: ocean; Type: scene; Precision: 93%





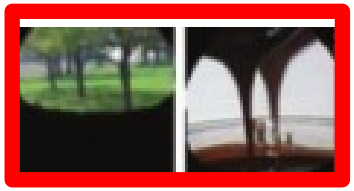
Annotating the Semantics of Units

Pool5, unit 13; Label: Lamps; Type: object; Precision: 84%



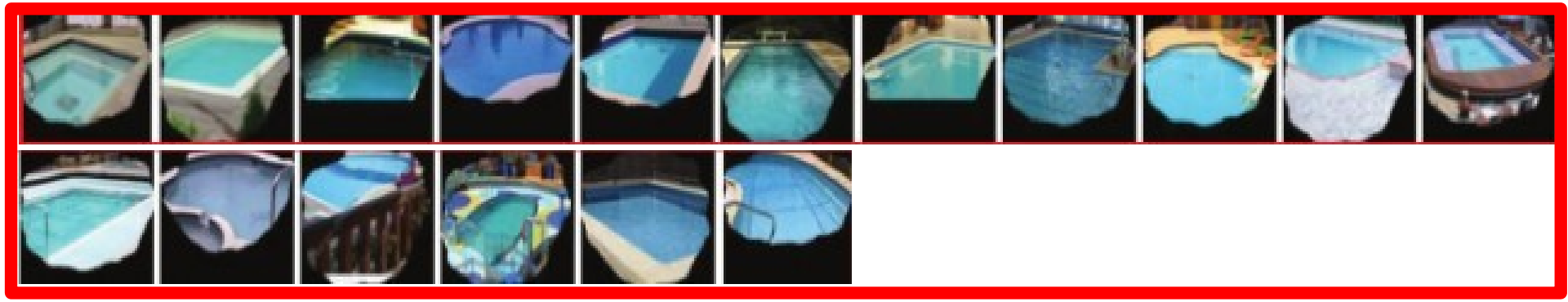
Annotating the Semantics of Units

Pool5, unit 77; Label:legs; Type: object part; Precision: 96%



Annotating the Semantics of Units

Pool5, unit 112; Label: pool table; Type: object; Precision: 70%



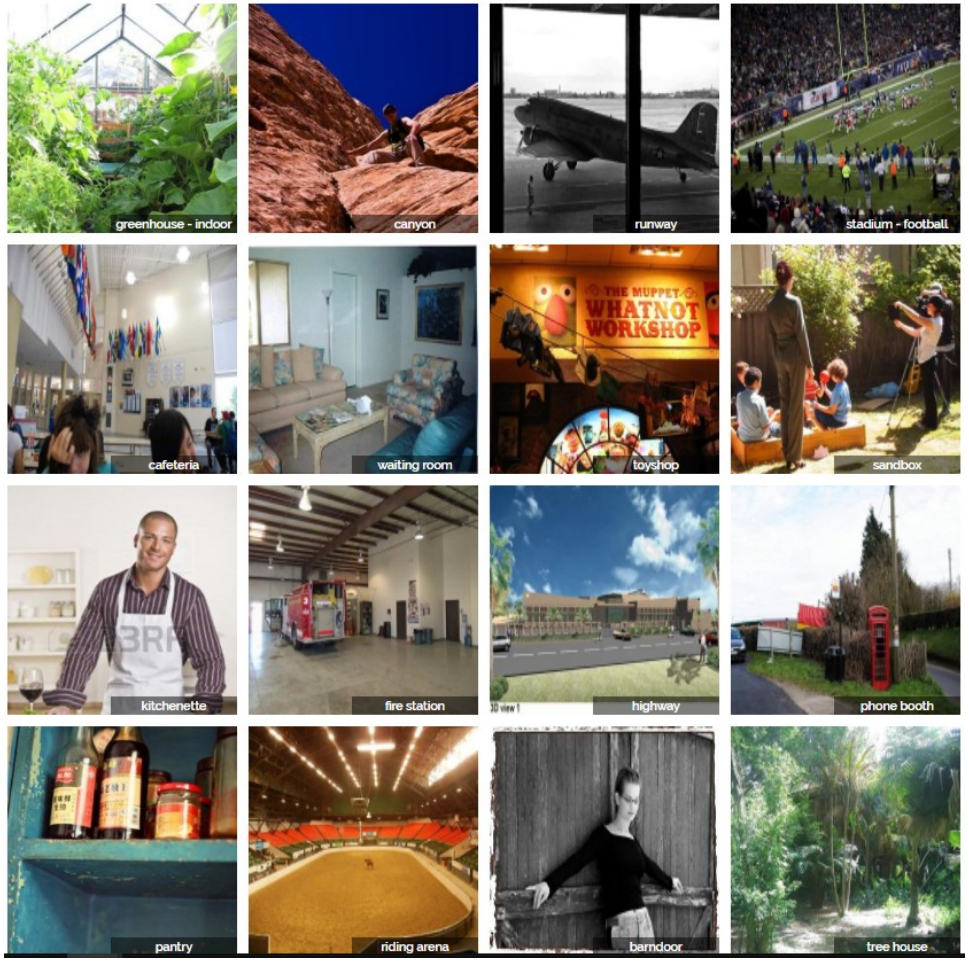
ImageNet vs. PlacesNet

ImageNet

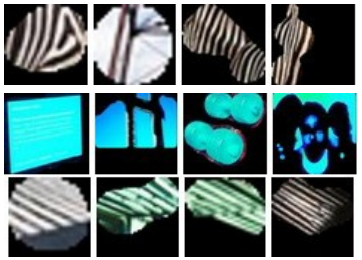
- ~1 mil object-level images over 1000 classes

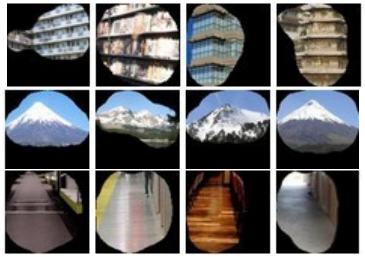
PlacesNet

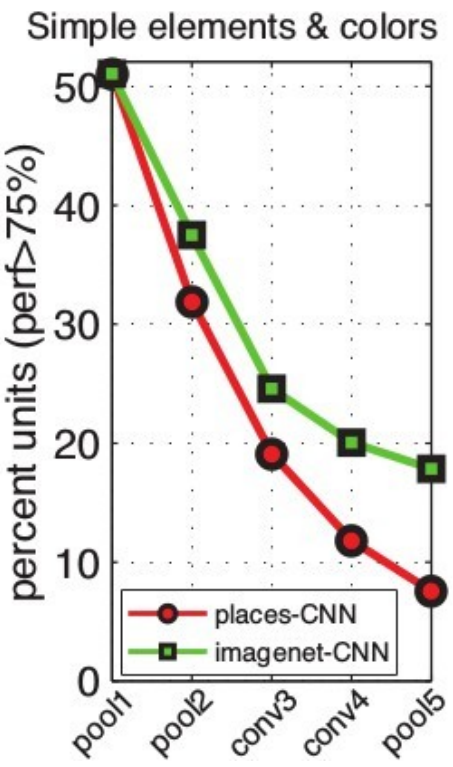
- ~1.8 million images from 365 scene categories
(at most 5000 images per category).

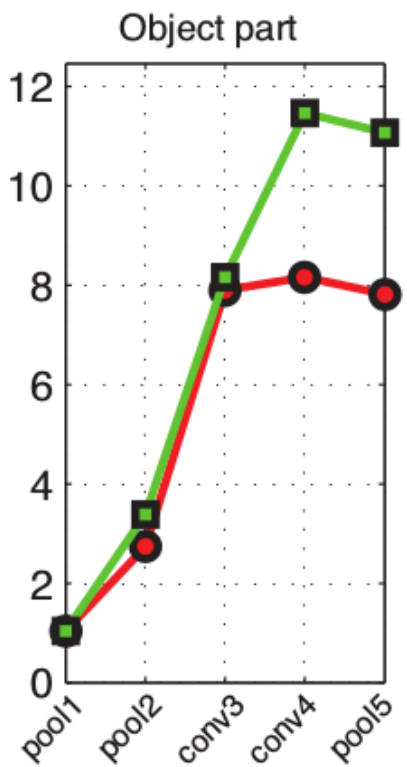


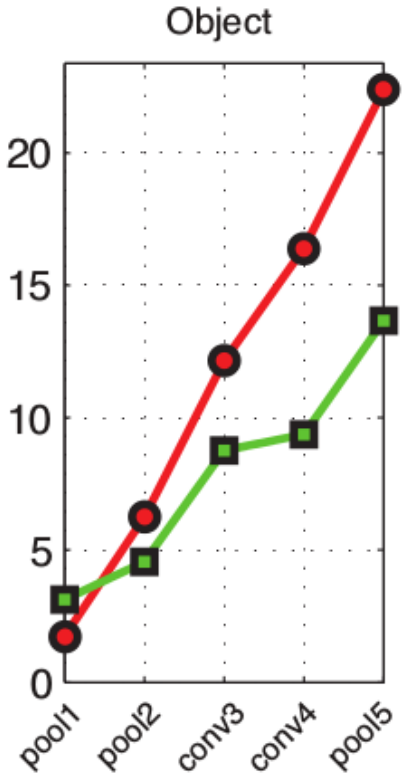
Distribution of Semantic Types at Each Layer

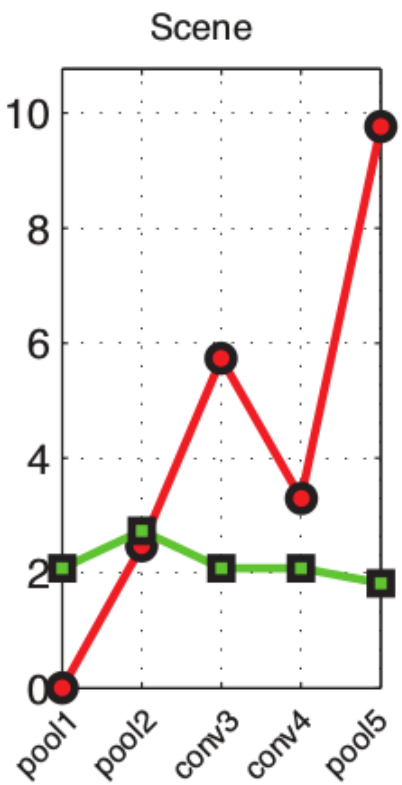




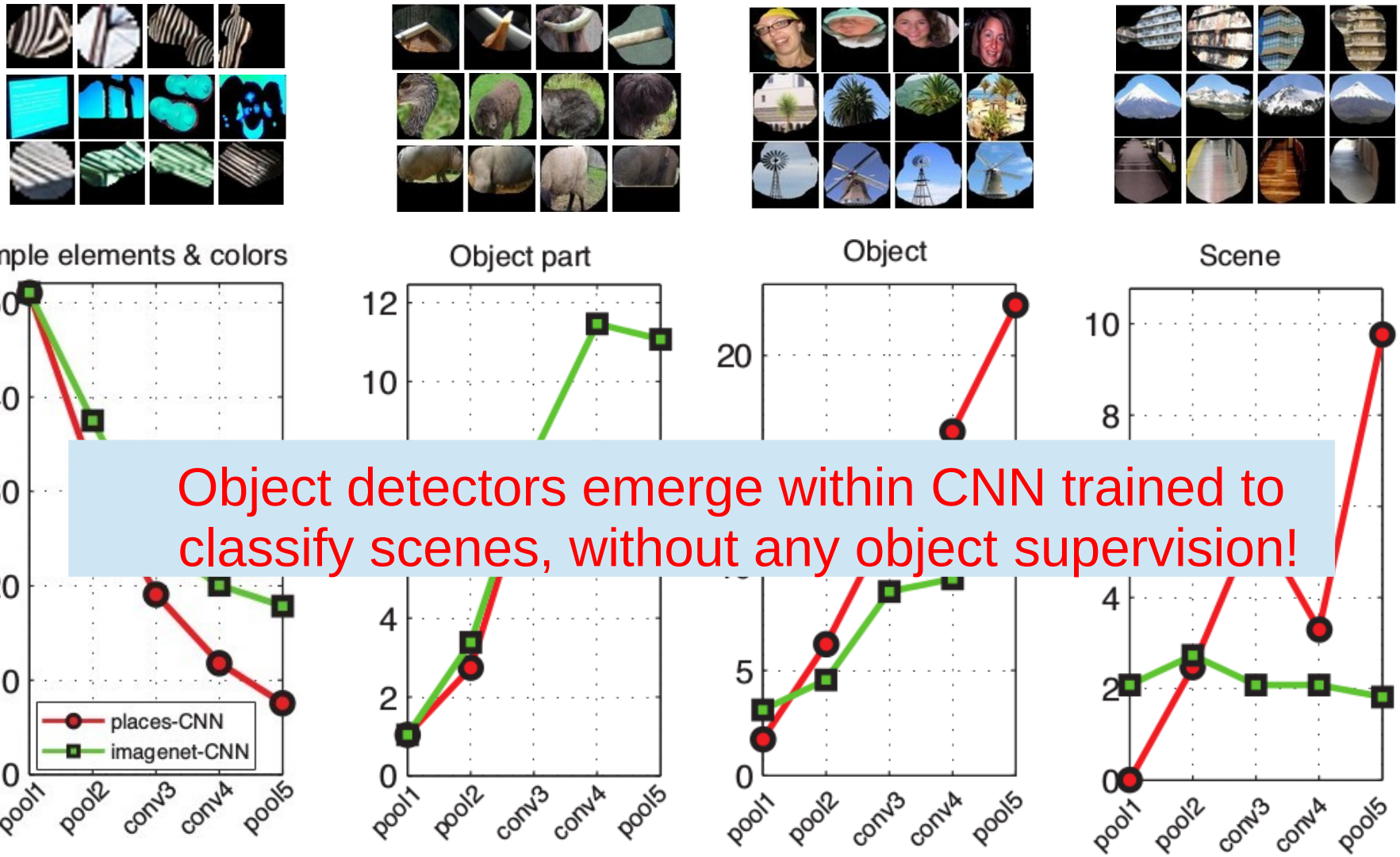








Distribution of Semantic Types at Each Layer

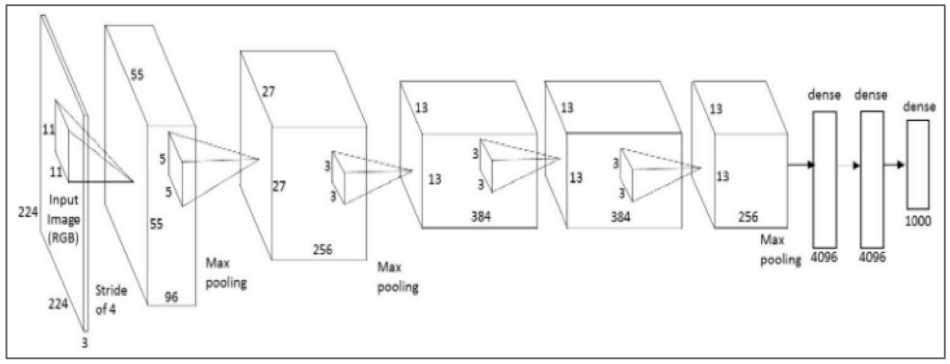


Interesting CNN properties

… other ways to measure reception

http://yosinski.com/deepvis

What input to a neuron maximizes a class score?



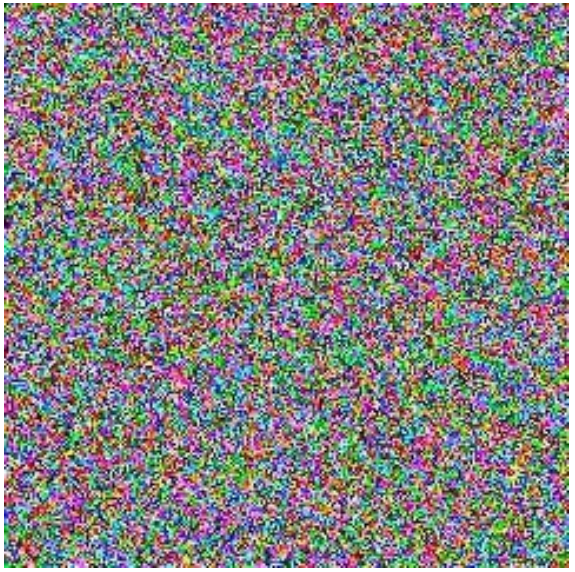
To visualize the function of a specific unit in a neural network, we synthesize an input to that unit which causes high activation.

Neuron of choice *i*

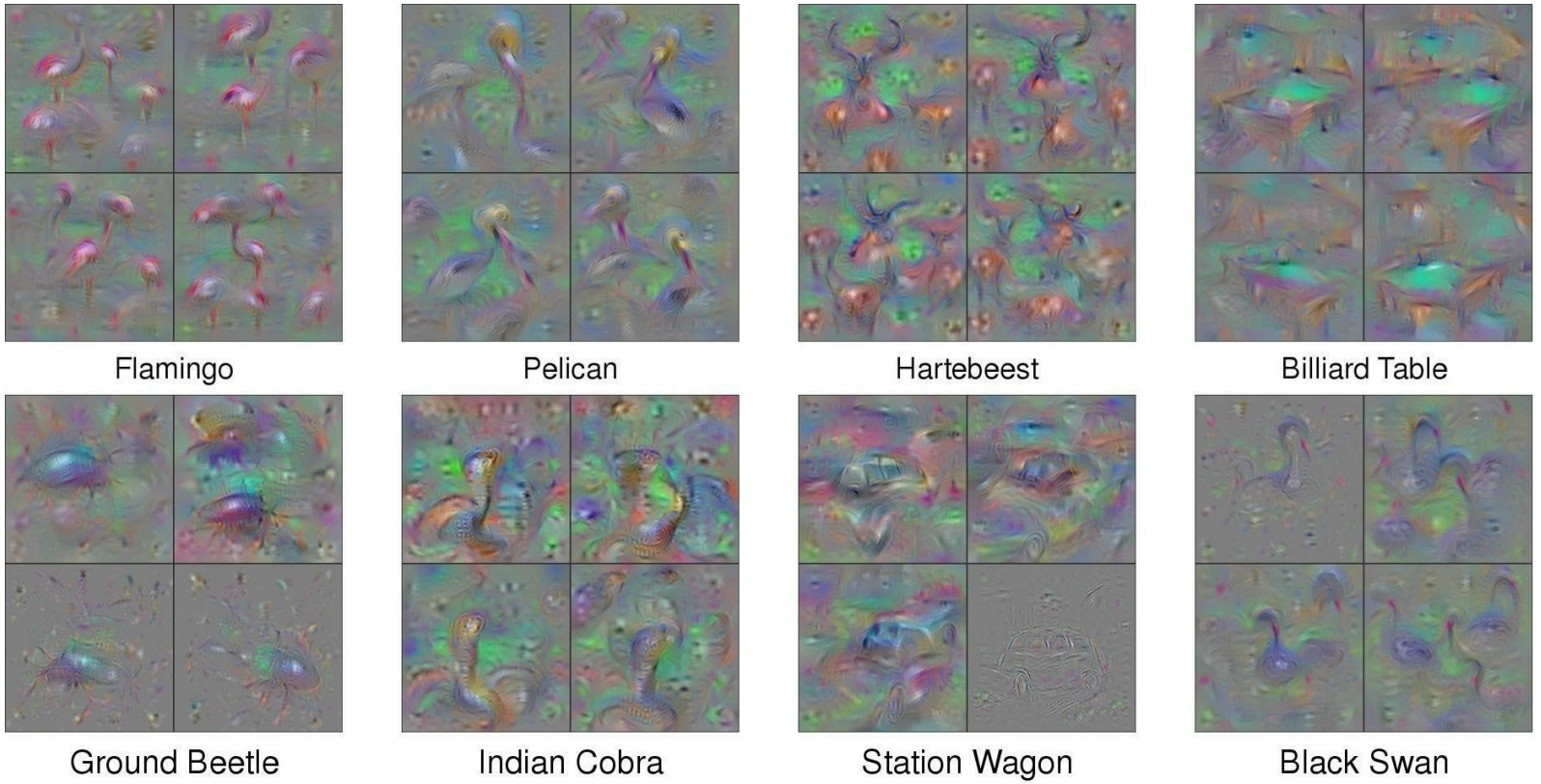
An image of random noise *x*.

Repeat:

1. Forward propagate: compute activation $a_i(x)$
2. Back propagate: compute gradient at neuron $\partial a_i(x) / \partial x$
3. Add small amount of gradient back to noisy image.



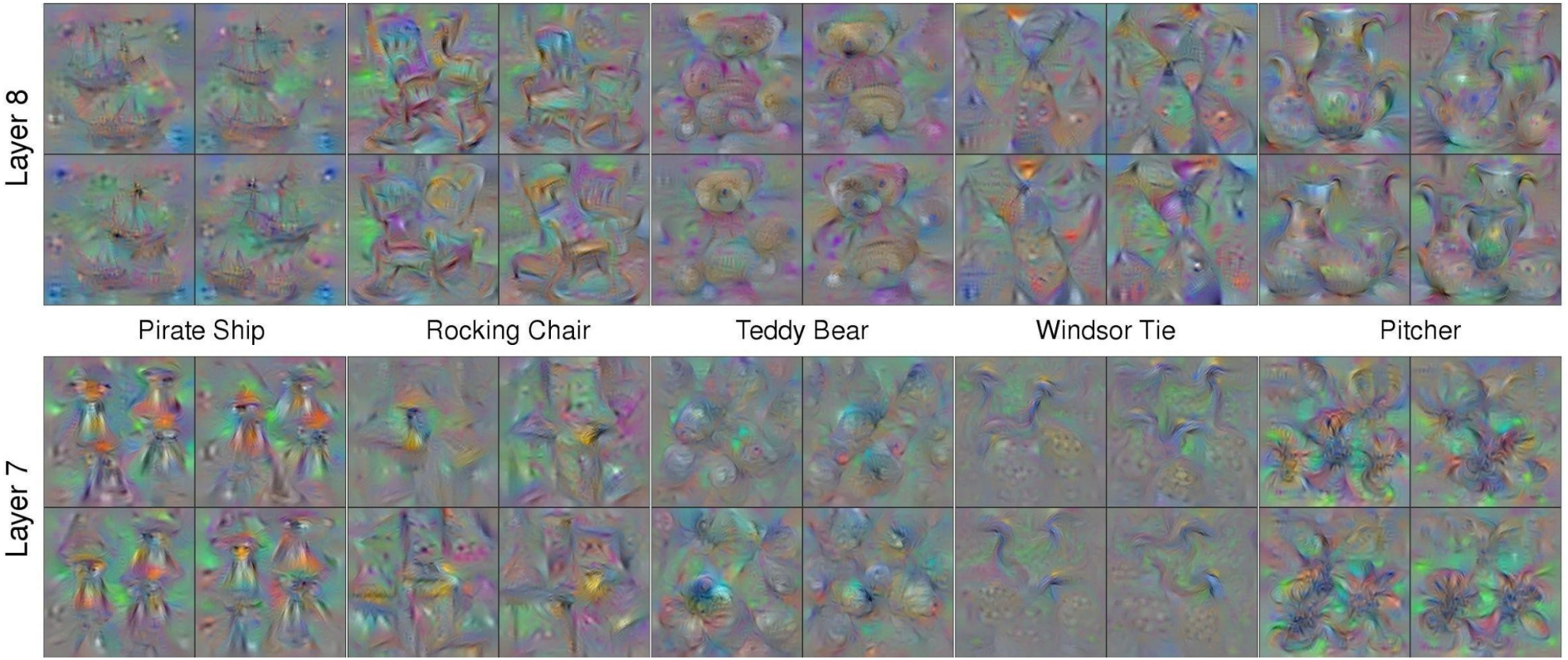
What image maximizes a class score?



[Understanding Neural Networks Through Deep Visualization, Yosinski et al. , 2015]

http://yosinski.com/deepvis

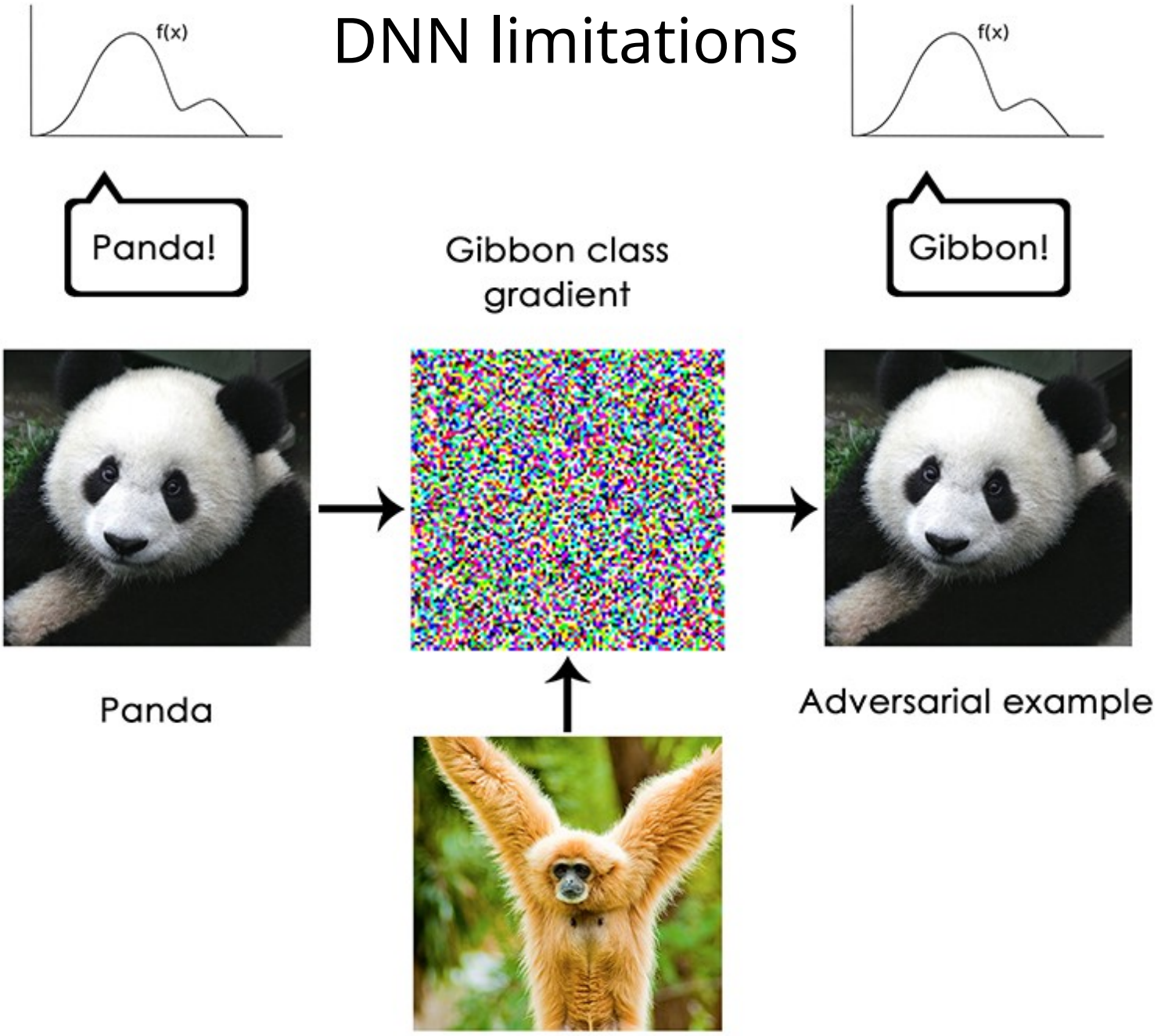
What image maximizes a class score?



Reading material:

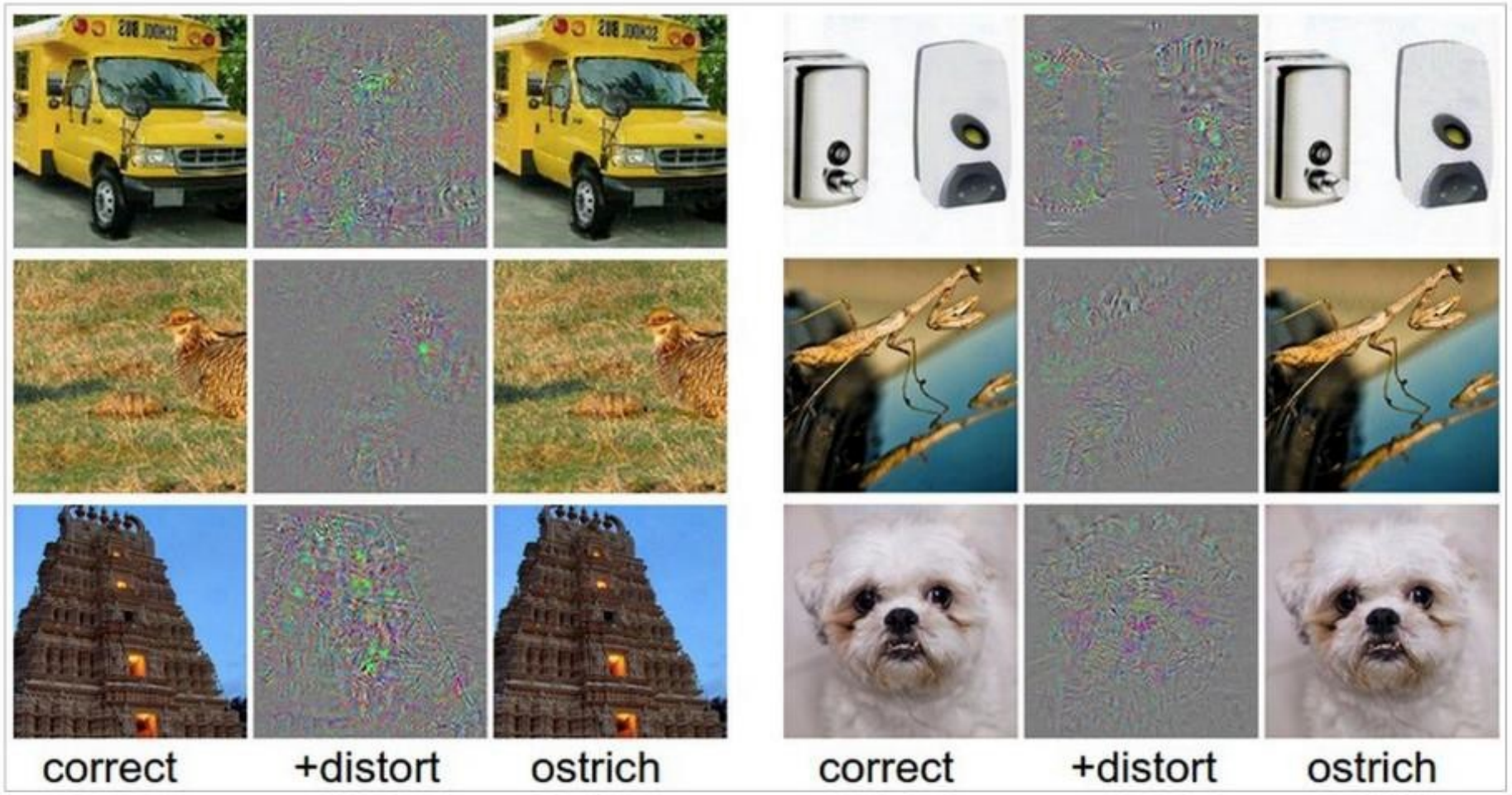
https://blog.keras.io/the-limitations-of-deep-learning.html

DNN limitations



Francois Chollet - https://blog.keras.io/the-limitations-of-deep-learning.html

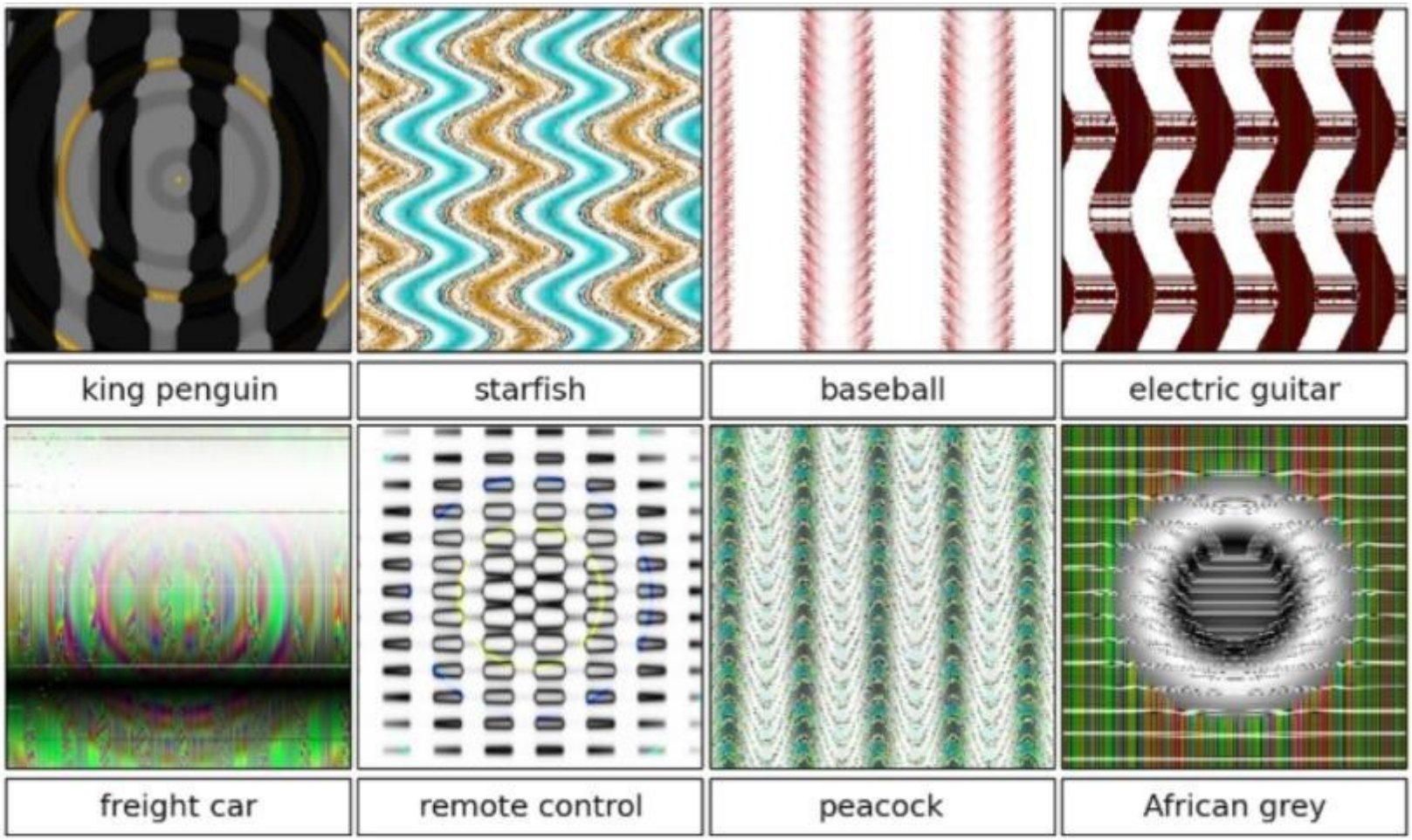
Breaking CNNs



Take a correctly classified image (left image in both columns), and add a tiny distortion (middle) to fool the ConvNet with the resulting image (right).

Intriguing properties of neural networks [Szegedy ICLR 2014]

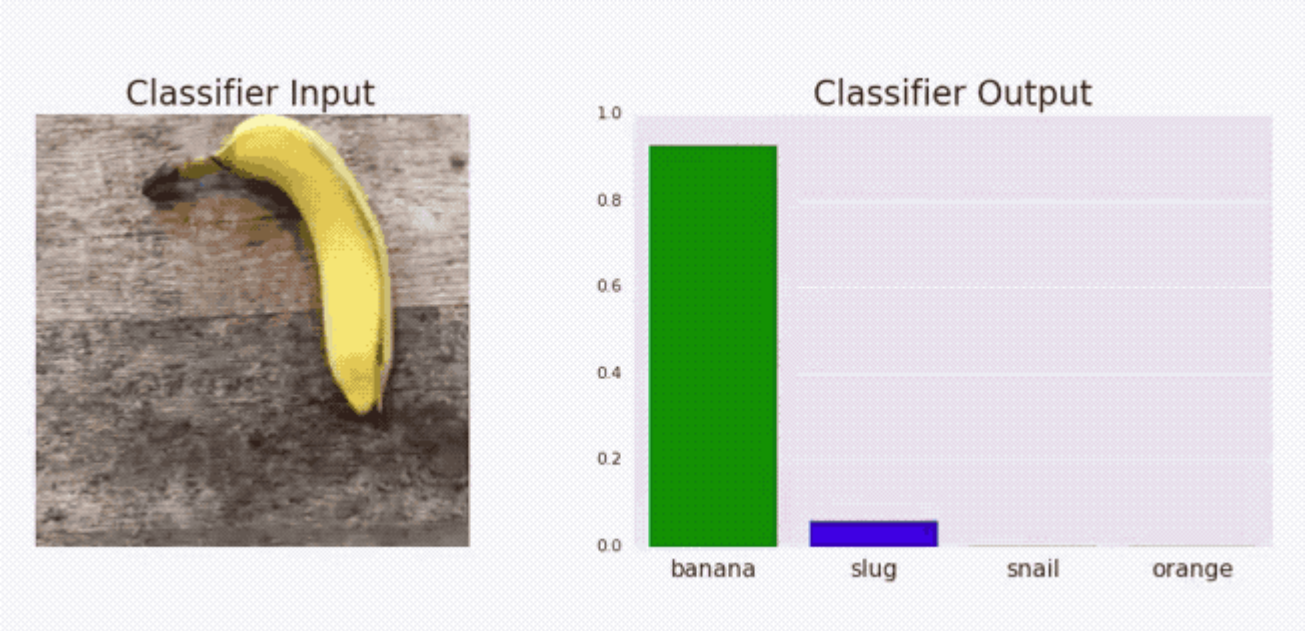
Breaking CNNs



Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images [Nguyen et al. CVPR 2015]

Adversarial Patches

- https://www.theverge.com/2018/1/3/16844842/ai-computer-vision-trick-adversarial-patches-google
- https://arxiv.org/pdf/1712.09665.pdf

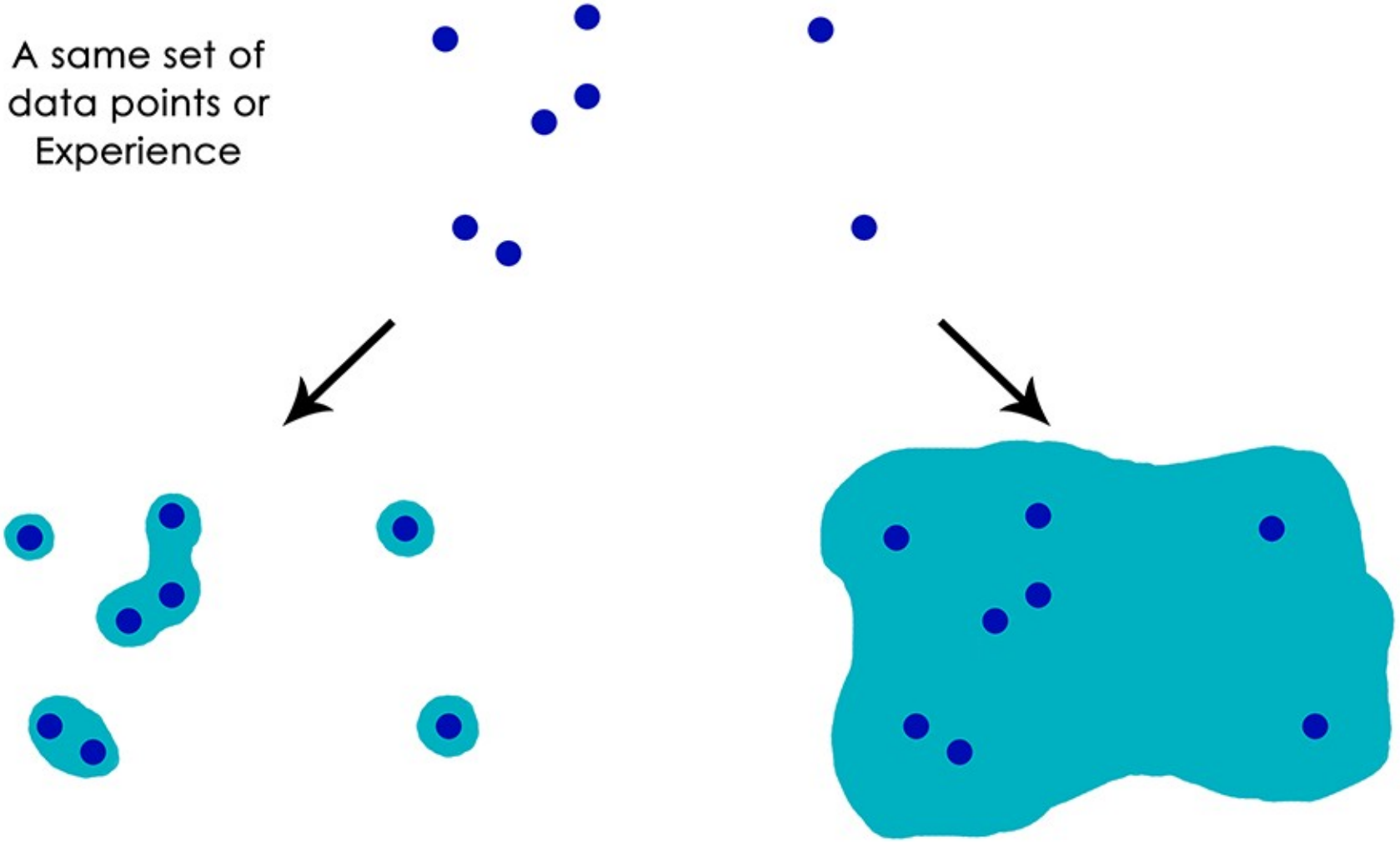


[Brown et al., Google, 2018]

Adversarial Patches

- https://www.theverge.com/2018/1/3/16844842/ai-computer-vision-trick-adversarial-patches-google
- https://arxiv.org/pdf/1712.09665.pdf

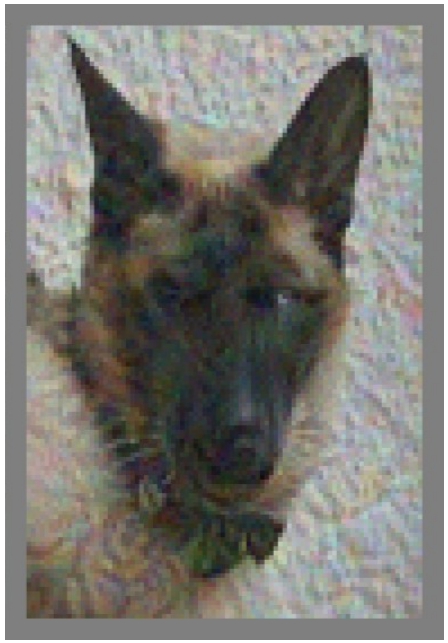
[Brown et al., Google, 2018]



Francois Chollet - https://blog.keras.io/the-limitations-of-deep-learning.html

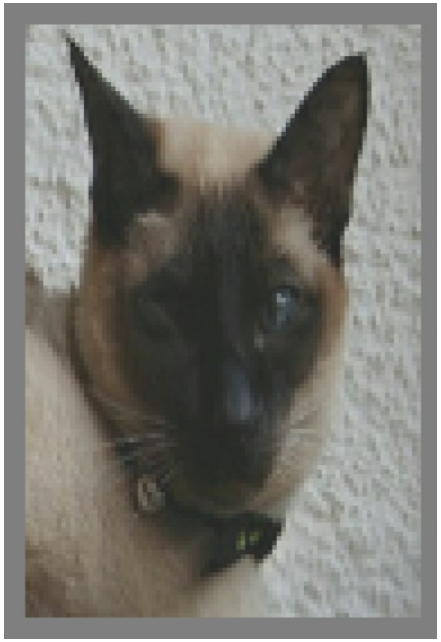
The boy is holding a baseball bat.

Francois Chollet - https://blog.keras.io/the-limitations-of-deep-



human + CV adversarial examples

- https://arxiv.org/abs/1802.08195

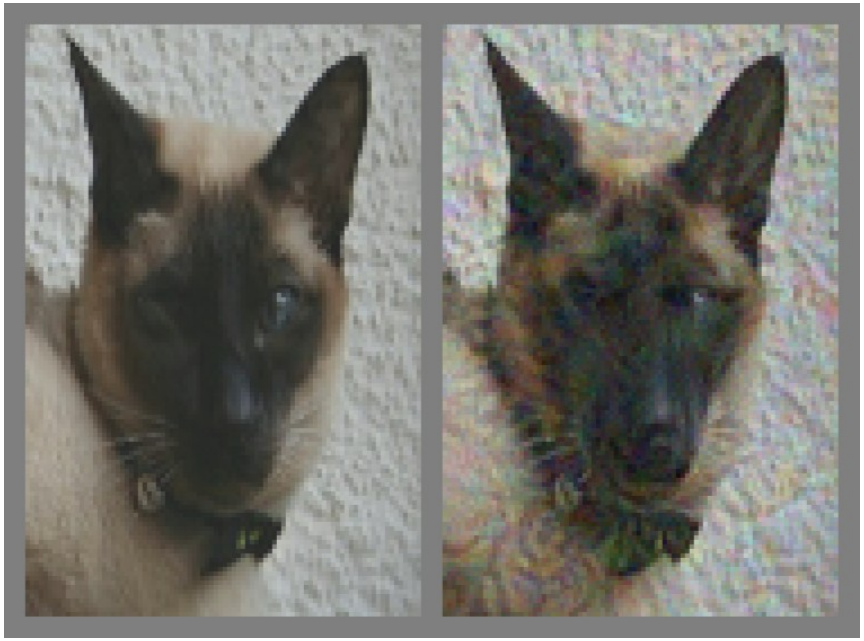


human + CV adversarial examples

- https://arxiv.org/abs/1802.08195

human + CV adversarial examples

- https://arxiv.org/abs/1802.08195



3rd November 2017

Technology

Single pixel change fools AI programs

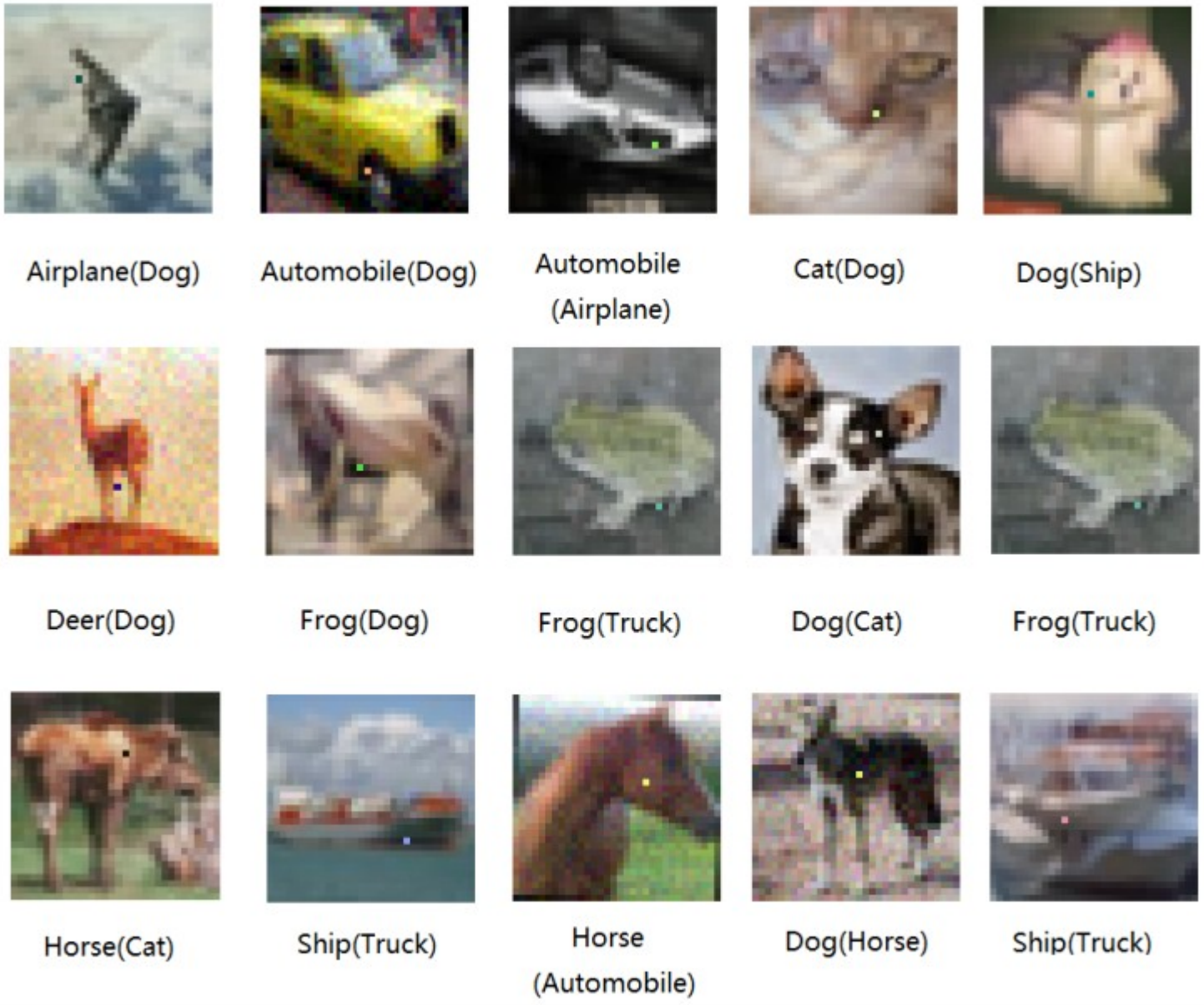
Tiny changes can make image recognition systems think a school bus is an ostrich, find scientists.

3 hours ago | Technology

Algorithm learns to recognise natural beauty

Artificial intelligence fools security

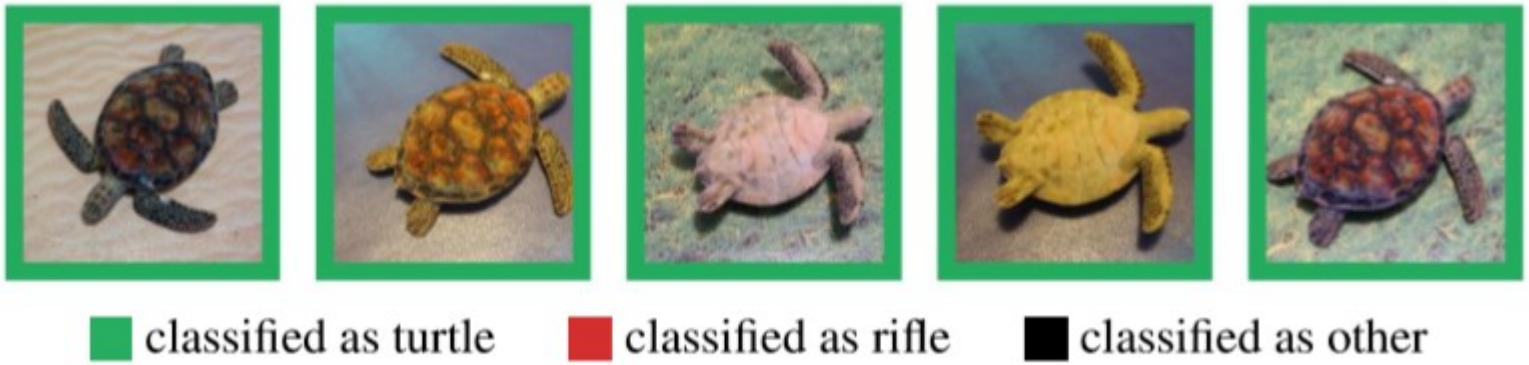
AI used to detect breast cancer

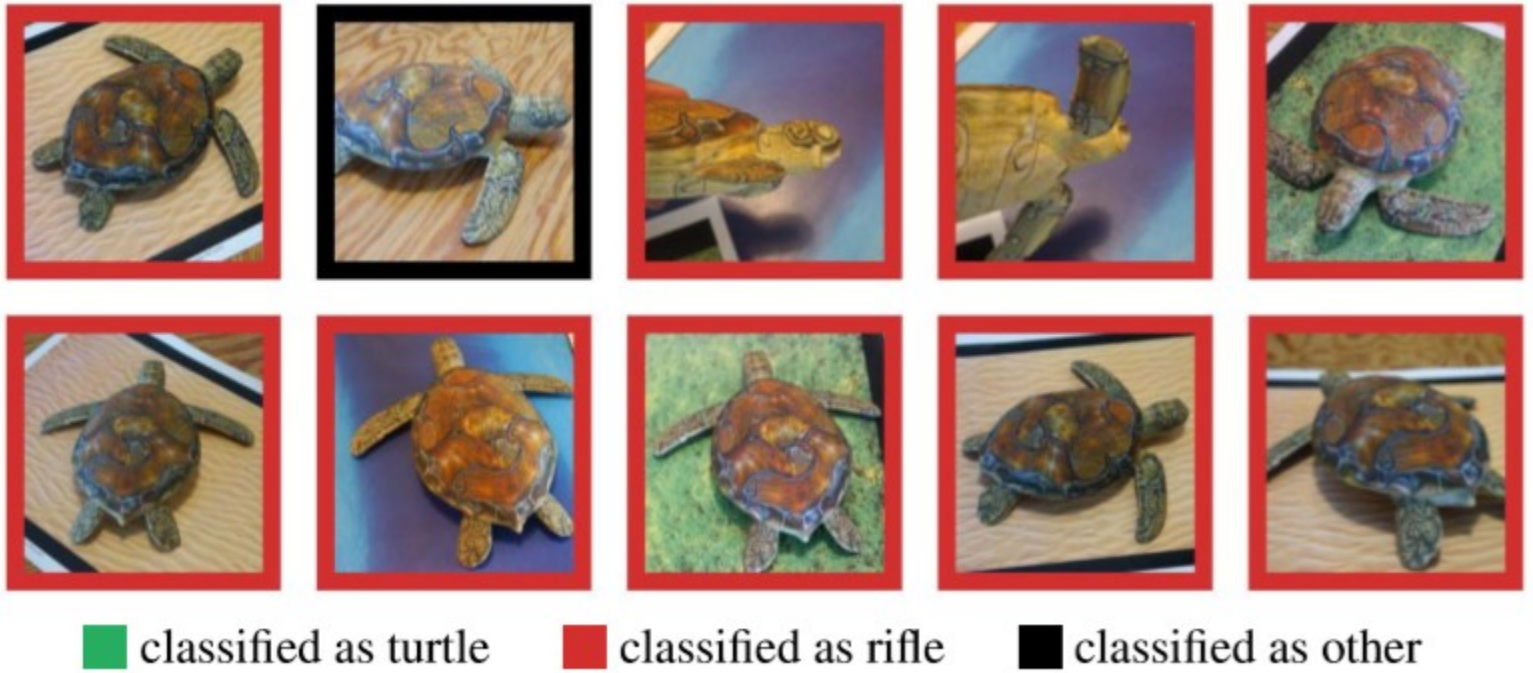


Su et al., One pixel attack for fooling deep neural networks
https://arxiv.org/abs/1710.08864

SYNTHESIZING ROBUST ADVERSARIAL EXAMPLES

Anish Athalye*[1,2], Logan Engstrom*[1,2], Andrew Ilyas*[1,2], Kevin Kwok[2]
[1]Massachusetts Institute of Technology, [2]LabSix
{aathalye,engstrom,ailyas}@mit.edu, kevin@labsix.org





classified as baseball classified as espresso classified as other

classified as baseball classified as espresso classified as other

The success of obfuscating gradients?

- https://github.com/anishathalye/obfuscated-gradients

In our recent paper, we evaluate the robustness of eight papers accepted to ICLR 2018 as non-certified white-box-secure defenses to adversarial examples. We find that seven of the eight defenses provide a limited increase in robustness and can be broken by improved attack techniques we develop.

The only defense we observe that significantly increases robustness to adversarial examples within the threat model proposed is "Towards Deep Learning Models Resistant to Adversarial Attacks" (Madry et al. 2018), and we were unable to defeat this defense without stepping outside the threat model. Even then, this technique has been shown to be difficult to scale to ImageNet-scale (Kurakin et al. 2016). The remainder of the papers rely either inadvertently or intentionally on what we call *obfuscated gradients*. Standard attacks apply gradient descent to maximize the loss of the network on a given image to generate an adversarial example on a neural network. Such optimization methods require a useful gradient signal to succeed. When a defense obfuscates gradients, it breaks this gradient signal and causes optimization based methods to fail.

Deep Learning: A Critical Appraisal

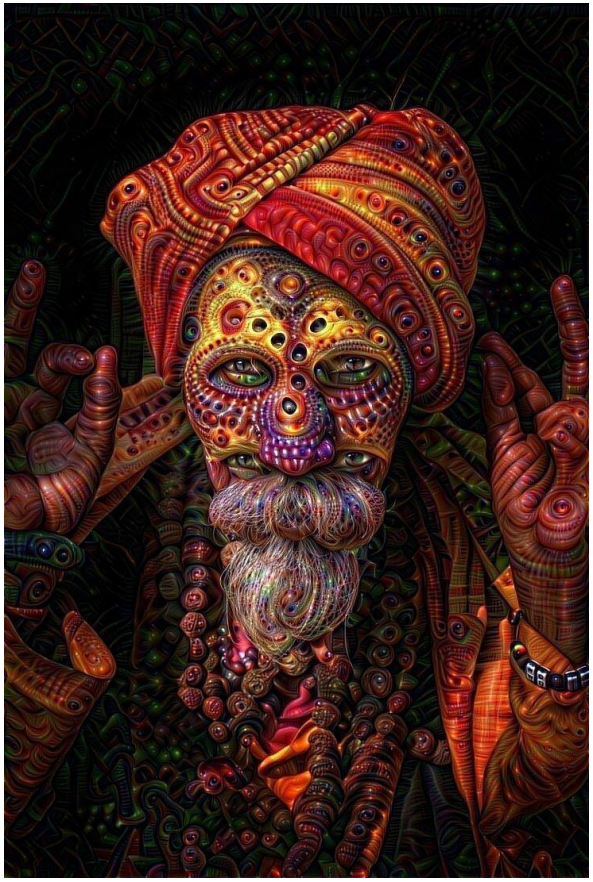
https://arxiv.org/pdf/1801.00631.pdf

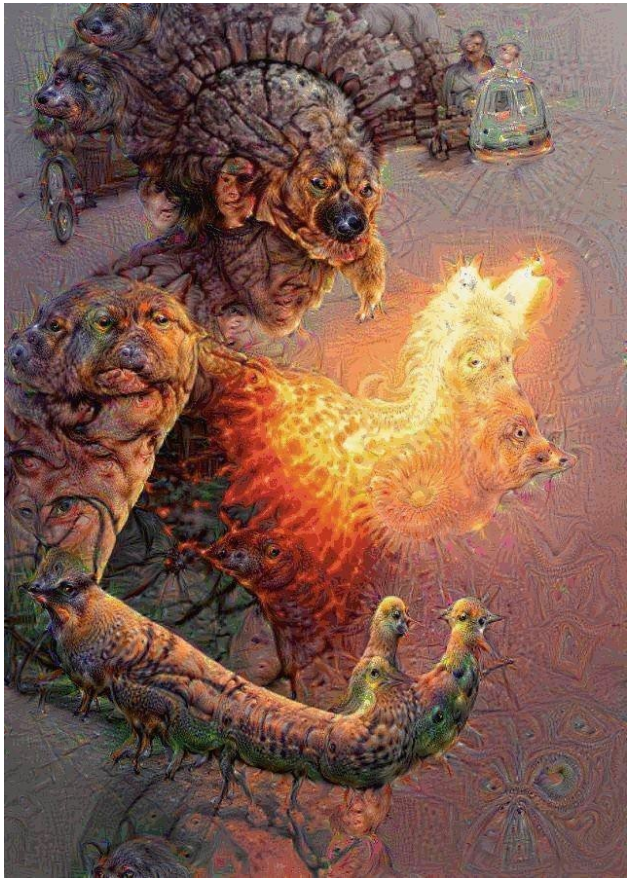
[Marcus, 2018]

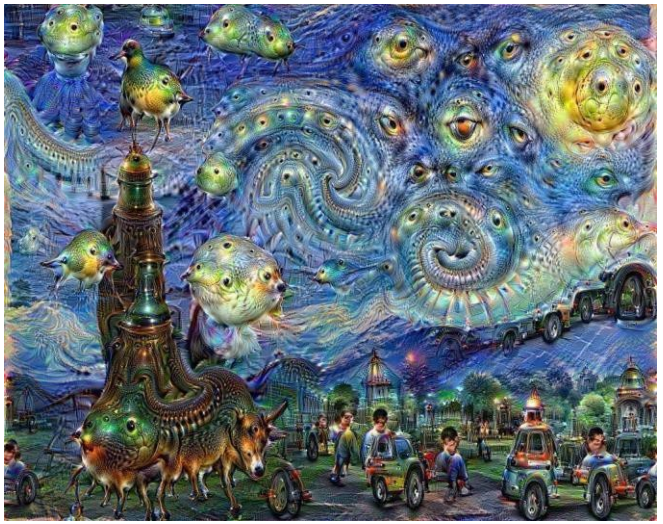
1) Deep learning thus far is data hungry
2) Deep learning thus far is shallow and has limited capacity for transfer
3) Deep learning thus far has no natural way to deal with hierarchical structure
4) Deep learning thus far has struggled with open-ended inference
5) Deep learning thus far is not sufficiently transparent
6) Deep learning thus far has not been well integrated with prior knowledge
7) Deep learning thus far cannot inherently distinguish causation from correlation
8) Deep learning presumes a largely stable world, in ways that may be problematic
9) Deep learning thus far works well as an approximation, but its answers often cannot be fully trusted
10) Deep learning thus far is difficult to engineer with

Other DNN usage examples

DeepDream

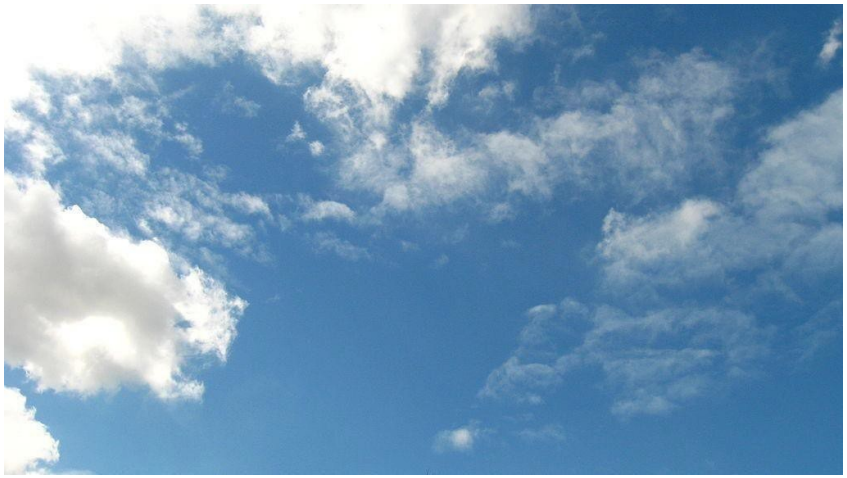






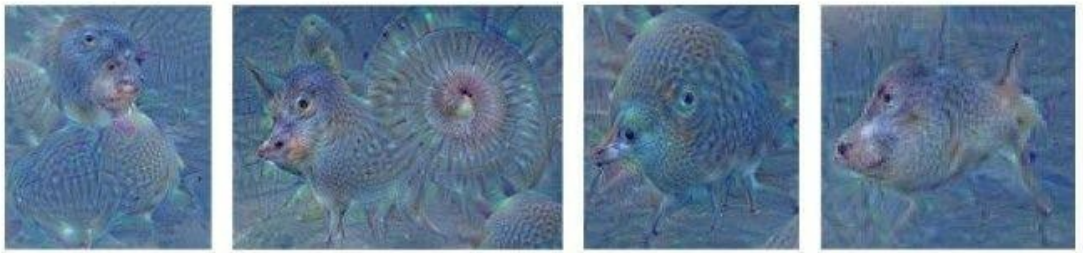
DeepDream https://github.com/google/deepdream

DeepDream



DeepDream modifies the image in a way that “boosts” all activations, at any layer

This creates a feedback loop: e.g., any slightly detected dog face will be made more and more dog-like over time.



DeepDream

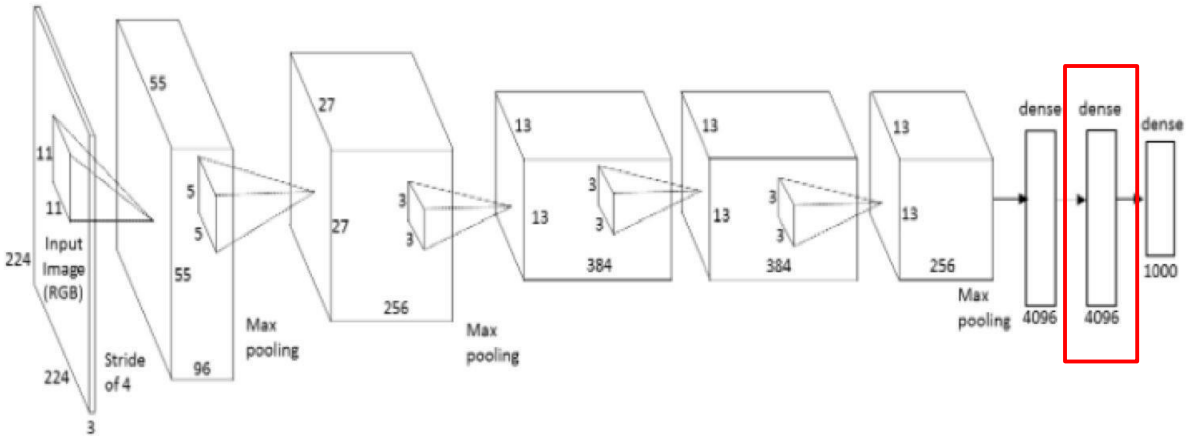
Deep Dream Grocery Trip
https://www.youtube.com/watch?v=DgPaCWJL7XI

Deep Dreaming Fear & Loathing in Las Vegas: the Great San Francisco Acid Wave
https://www.youtube.com/watch?v=oyxSerkkP4o

Viewer discretion advised!

Reconstructing images

Question: Given a CNN code, is it possible to reconstruct the original image?



Reconstructing images

Find an image such that:

- Its code is similar to a given code
- It “looks natural”
  - Neighboring pixels should look similar

$$\text{Image}\mathbf{x}^* = \underset{\mathbf{x}\in\mathbb{R}^{H\times W\times C}}{\operatorname{argmin}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda\mathcal{R}(\mathbf{x})$$

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

Reconstructing images

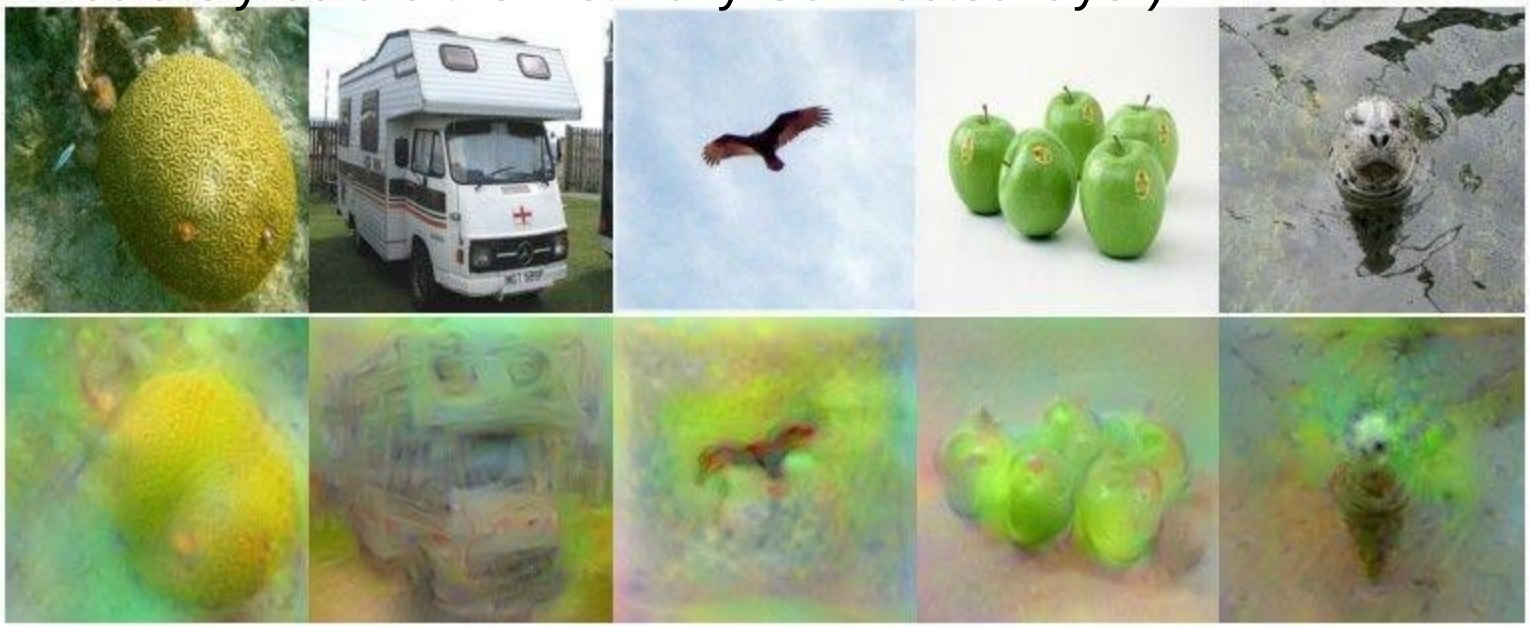
original image

Reconstructions from the 1000 log probabilities for ImageNet (ILSVRC) classes

Understanding Deep Image Representations by Inverting Them
[Mahendran and Vedaldi, 2014]
https://arxiv.org/pdf/1412.0035.pdf

Reconstructing images

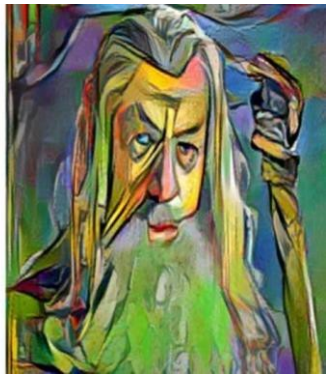
Reconstructions from the representation after last last pooling layer
(immediately before the first Fully Connected layer)



Style transfer

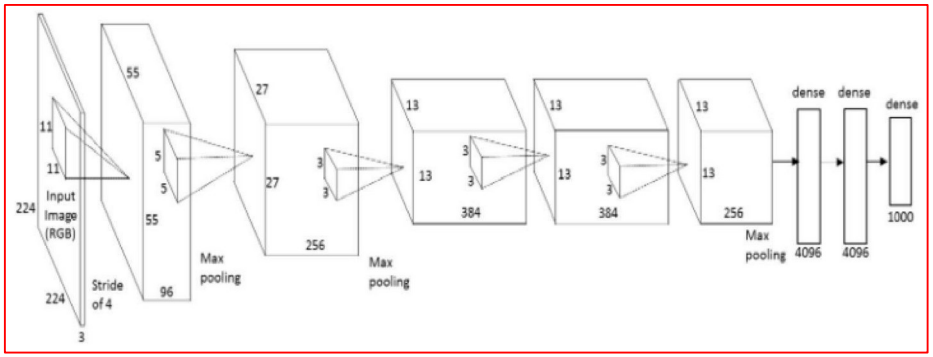
Neural Style

[A Neural Algorithm of Artistic Style by Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, 2015] **good implementation by Justin Johnson in Torch: https://github.com/jcjohnson/neural-style**



Neural Style

Step 1: Extract **content targets** (ConvNet activations of all layers for the given content image)

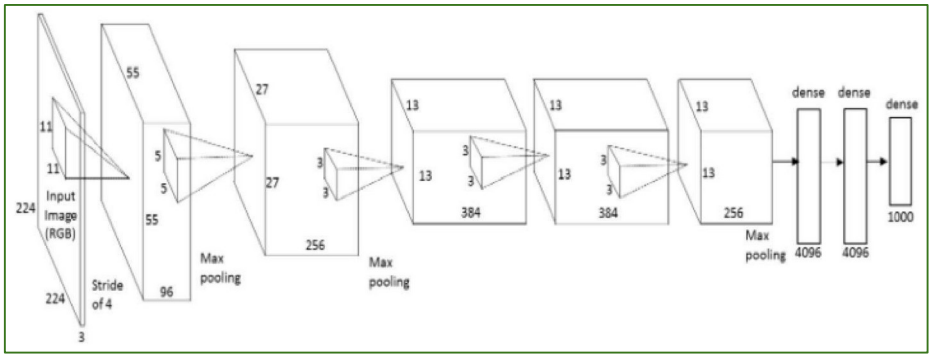


content activations

e.g.
at CONV5_1 layer we would have a [14x14x512] array of target activations

Neural Style

Step 2: Extract **style targets** (Gram matrices of ConvNet activations of all layers for the given style image)



style gram matrices

$$G = V^{\mathrm{T}} V$$

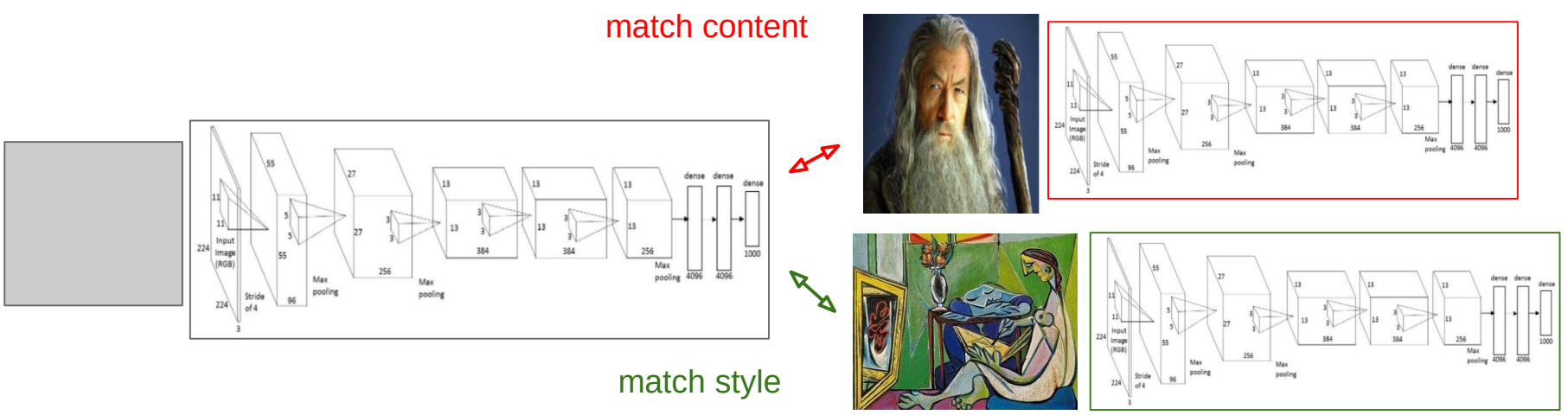
e.g.
at CONV1 layer (with [224x224x64] activations) would give a [64x64] Gram matrix of all pairwise activation covariances (summed across spatial locations)

Neural Style

Step 3: Optimize over image to have:

- The **content** of the content image (activations match content)
- The **style** of the style image (Gram matrices of activations match style)

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$



Adapted from Andrej Karpathy

Neural Style

make your own easily on deepart.io or at https://www.instapainting.com

Dataset Distillation

- https://arxiv.org/pdf/1811.10959.pdf

DATASET DISTILLATION

Tongzhou Wang
Facebook AI Research, MIT CSAIL

Jun-Yan Zhu
MIT CSAIL

Antonio Torralba
MIT CSAIL

Alexei A. Efros
UC Berkeley

ABSTRACT

Model distillation aims to distill the knowledge of a complex model into a simpler one. In this paper, we consider an alternative formulation called *dataset distillation*: we keep the model fixed and instead attempt to distill the knowledge from a large training dataset into a small one. The idea is to *synthesize* a small number of data points that do not need to come from the correct data distribution, but will, when given to the learning algorithm as training data, approximate the model trained on the original data. For example, we show that it is possible to compress $60,000$ MNIST training images into just 10 synthetic *distilled images* (one per class) and achieve close to original performance with only a few gradient descent steps, given a fixed network initialization. We evaluate our method in various initialization settings and with different learning objectives. Experiments on multiple datasets show the advantage of our approach compared to alternative methods.