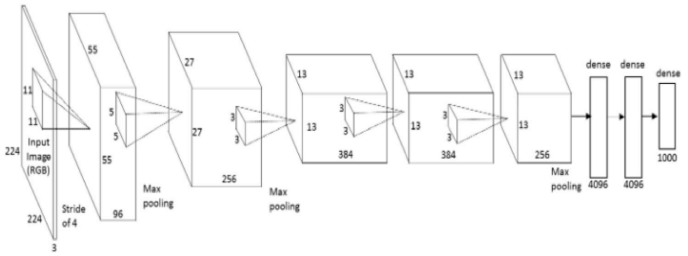
Understanding What CNNs Learn

Credit: CS143 by James Tompkin

Training Neural Networks



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

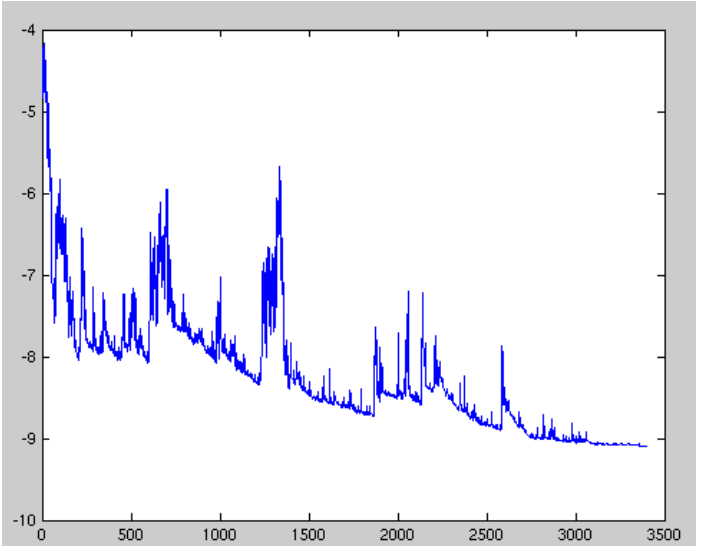
- While loss still decreasing
  - Shuffle training data
  - For each data point *i=1…n (maybe as mini-batch)*
    - *Gradient descent*
  - Check validation set loss

"Epoch"

Stochastic Gradient Descent

Try to speed up processing with random training subsets

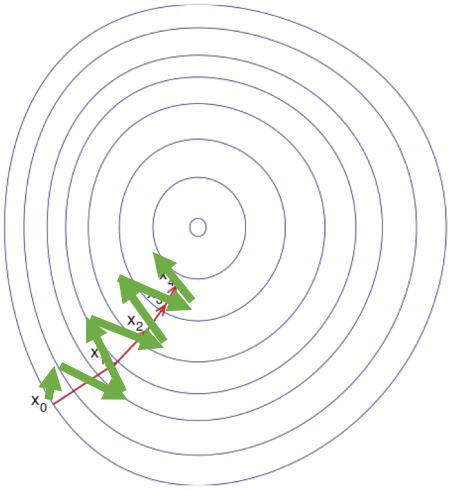
Loss will not always decrease (locally) as training data point is random, but converges over time.



Momentum

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

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



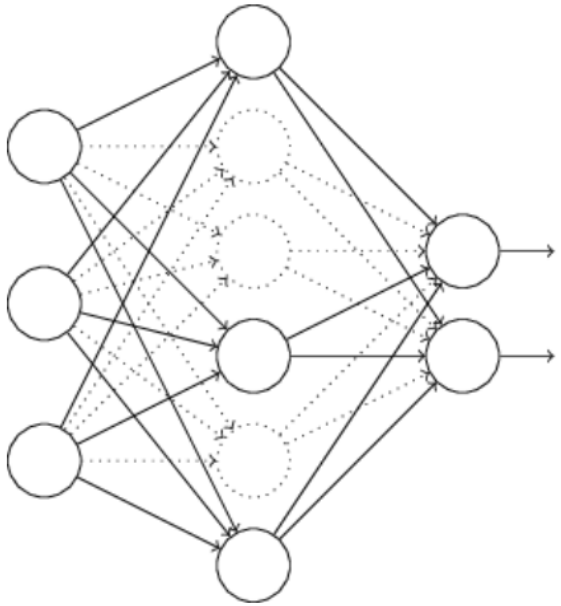
Wikipedia

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

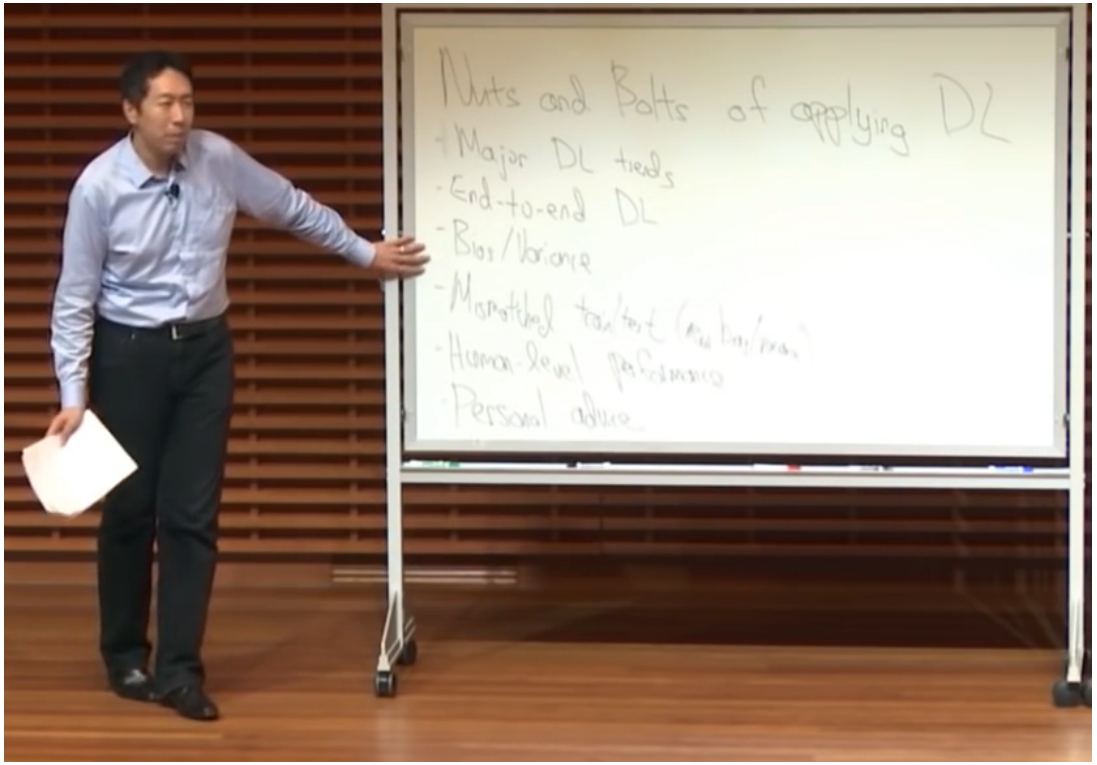


When something is not working…

…how do I know what to do next?

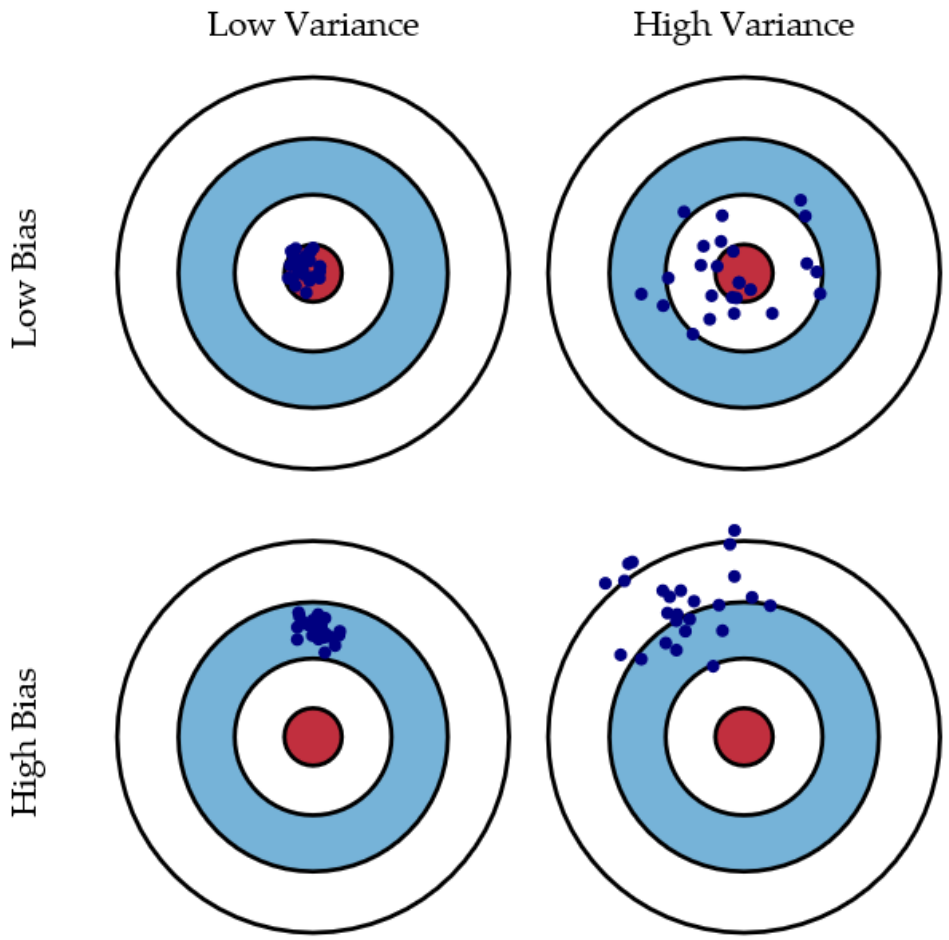
The Nuts and Bolts of Building Applications using Deep Learning

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

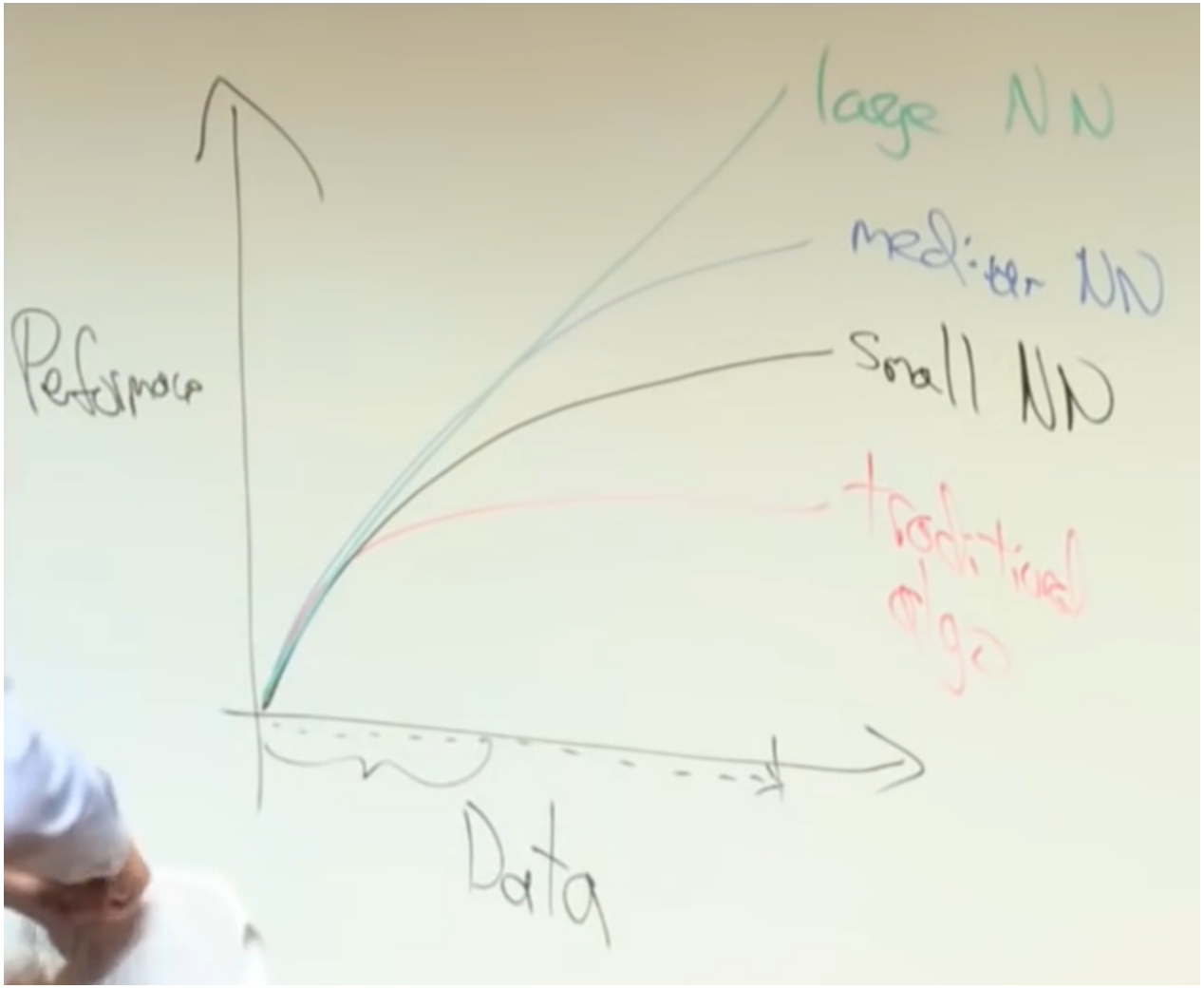


Bias/variance trade-off

"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’)

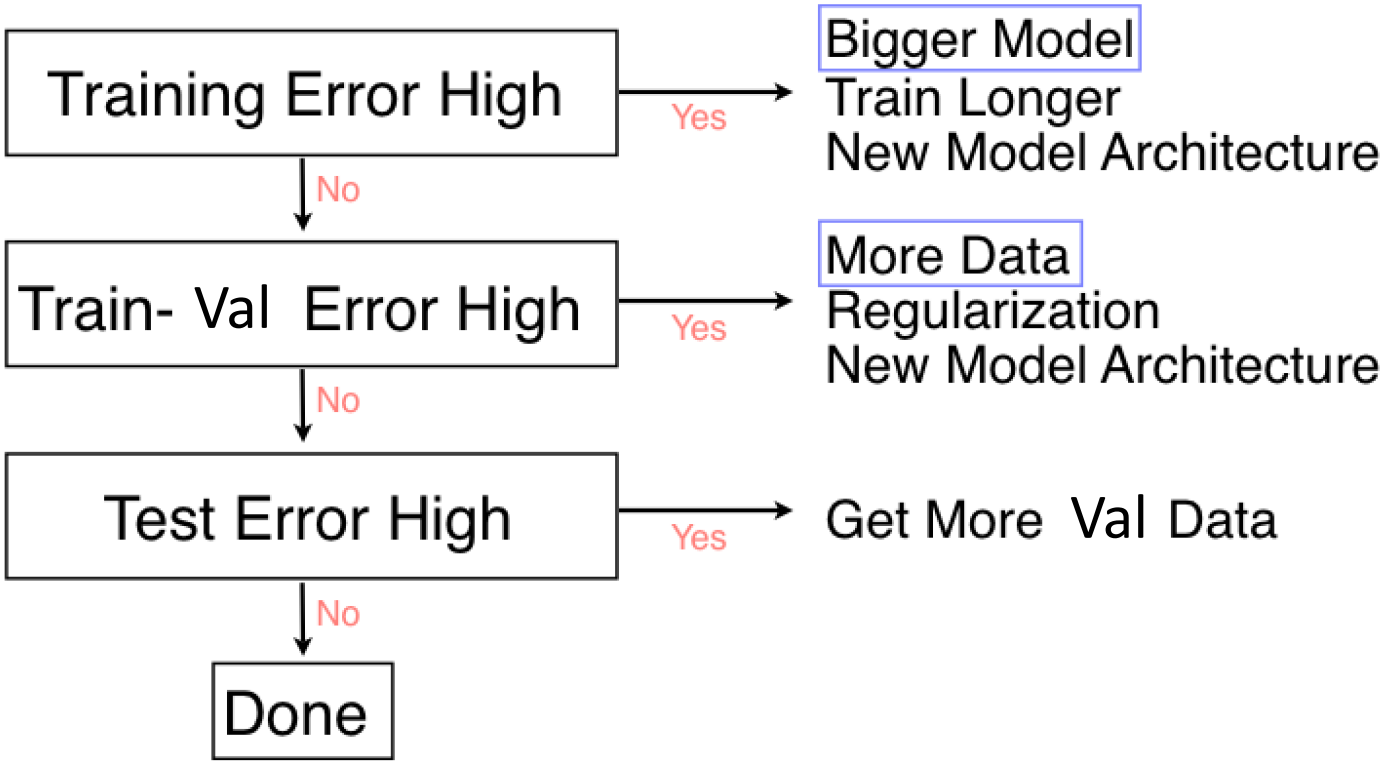
Properties

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

"Bias"

Overfitting to validation

The Nuts and Bolts of Building Applications Using Deep Learning



[Andrew Ng]

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?

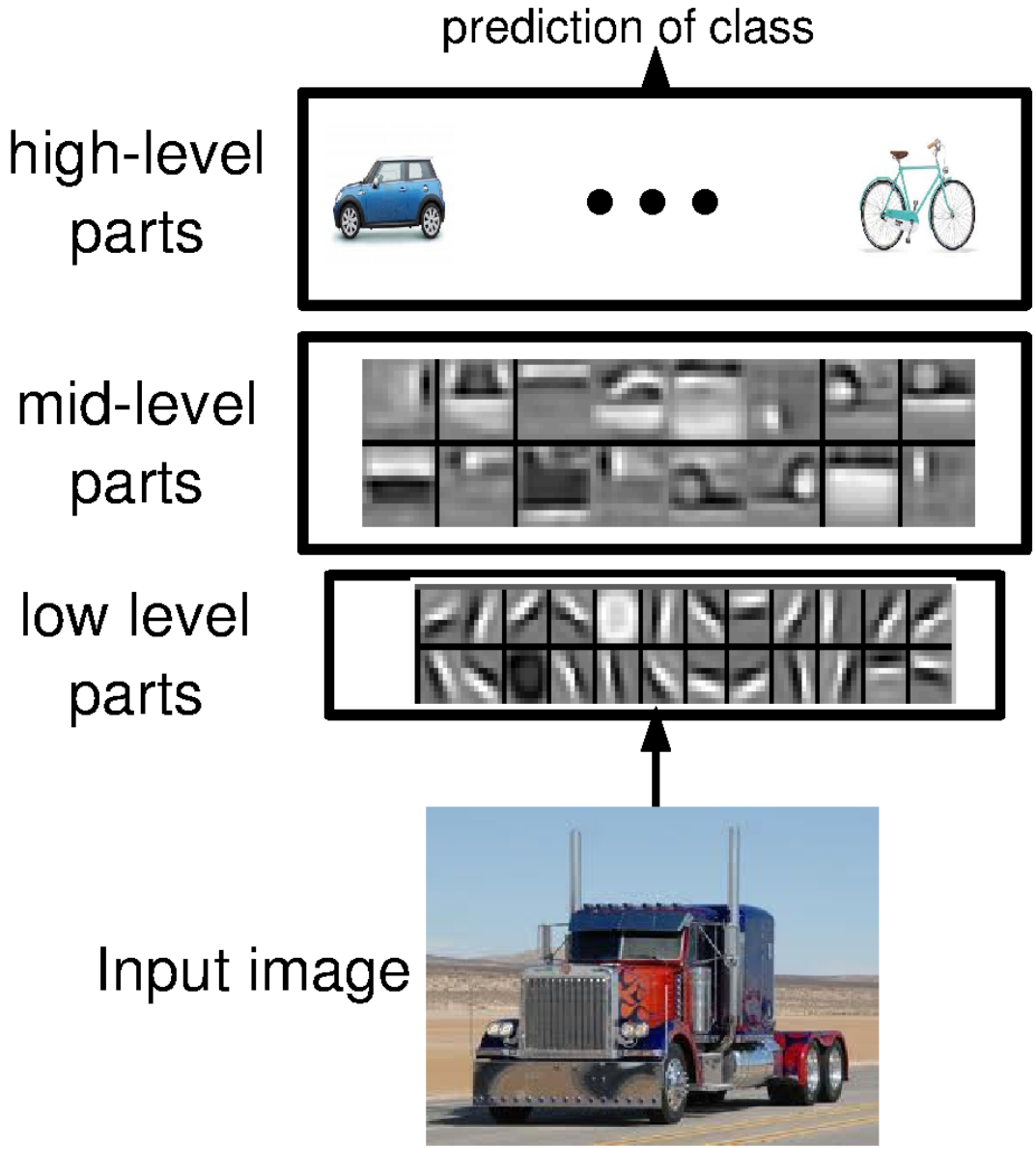
Well luckily for you I'm here with a list of all the things you've probably done wrong and compiled from my own experiences implementing neural networks and supervising other students with their projects:

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

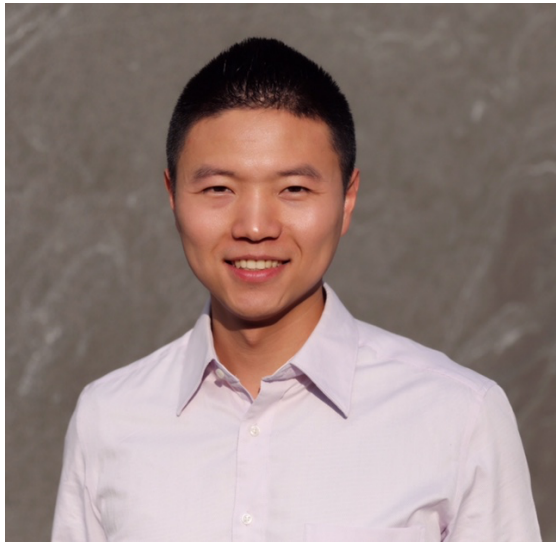
http://theorangeduck.com/page/neural-network-not-working

Interpretation



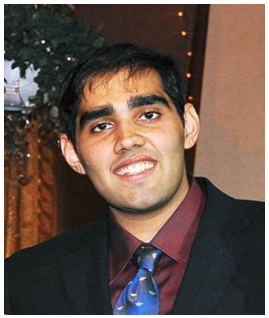
- distributed representations
- feature sharing
- compositionality

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



Object Detectors Emerge in Deep Scene CNNs

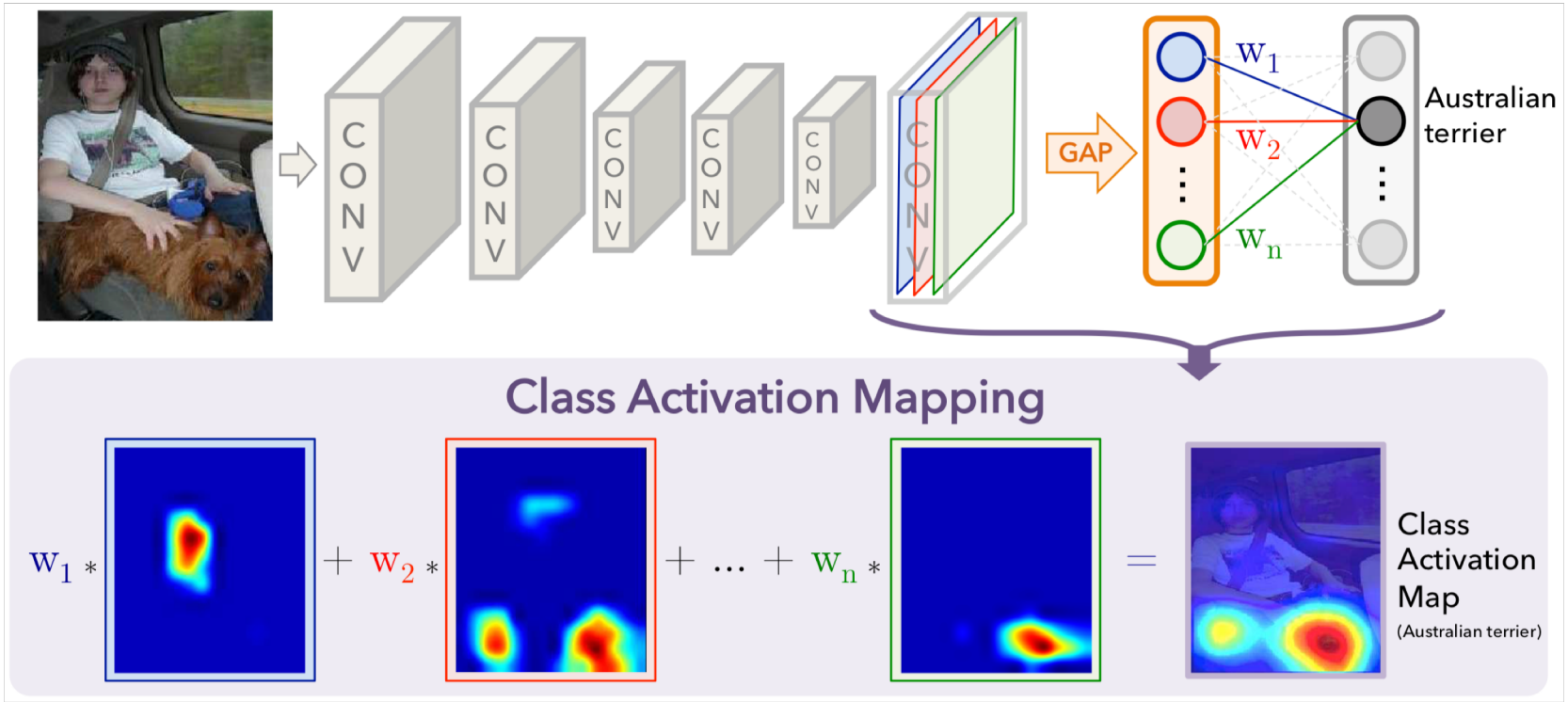
Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, AntonioTorralba




Massachusetts Institute of Technology

ICLR 2015

Class Activation Mapping (MAP)

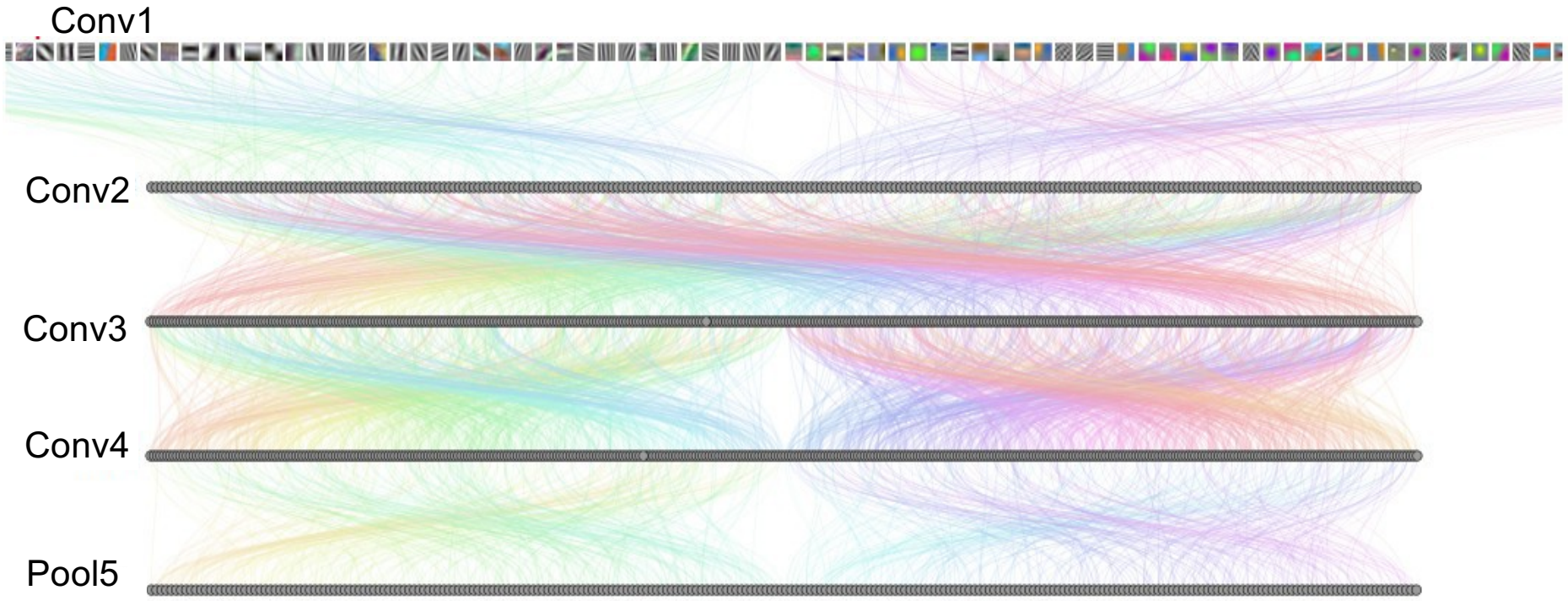


$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c f_k(x,y).$$

http://cnnlocalization.csail.mit.edu

How Objects are Represented in CNN?

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

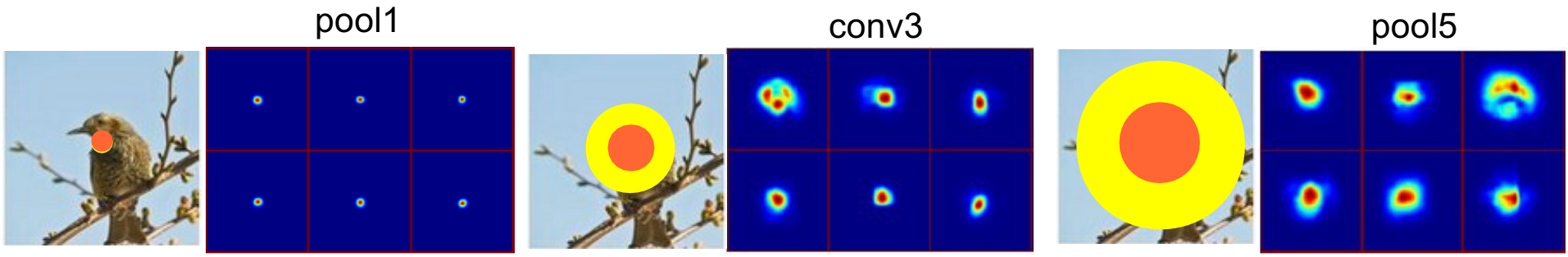


Agrawal, et al. Analyzing the performance of multilayer neural networks for object recognition. ECCV, 2014
Szegedy, et al. Intriguing properties of neural networks.arXiv preprint arXiv:1312.6199, 2013.
Zeiler, M. et al. Visualizing and Understanding Convolutional Networks, ECCV 2014.

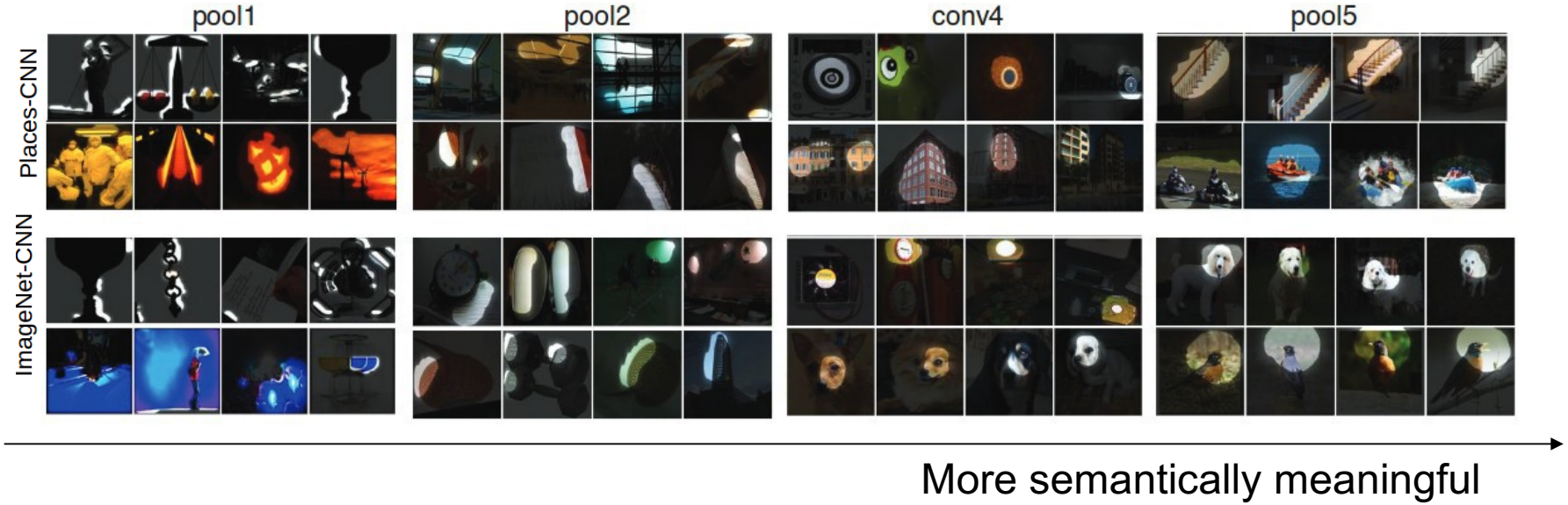
Estimating the Receptive Fields

Estimated receptive fields

Actual size of RF is much smaller than the theoretic size

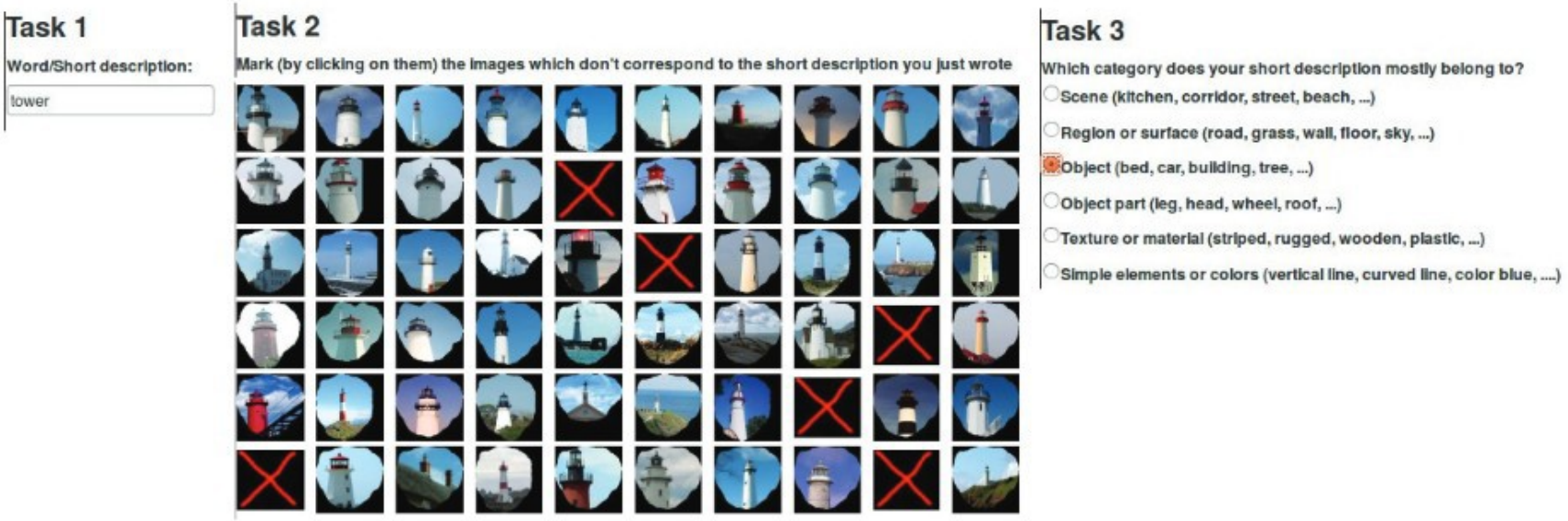


Segmentation using the RF of Units



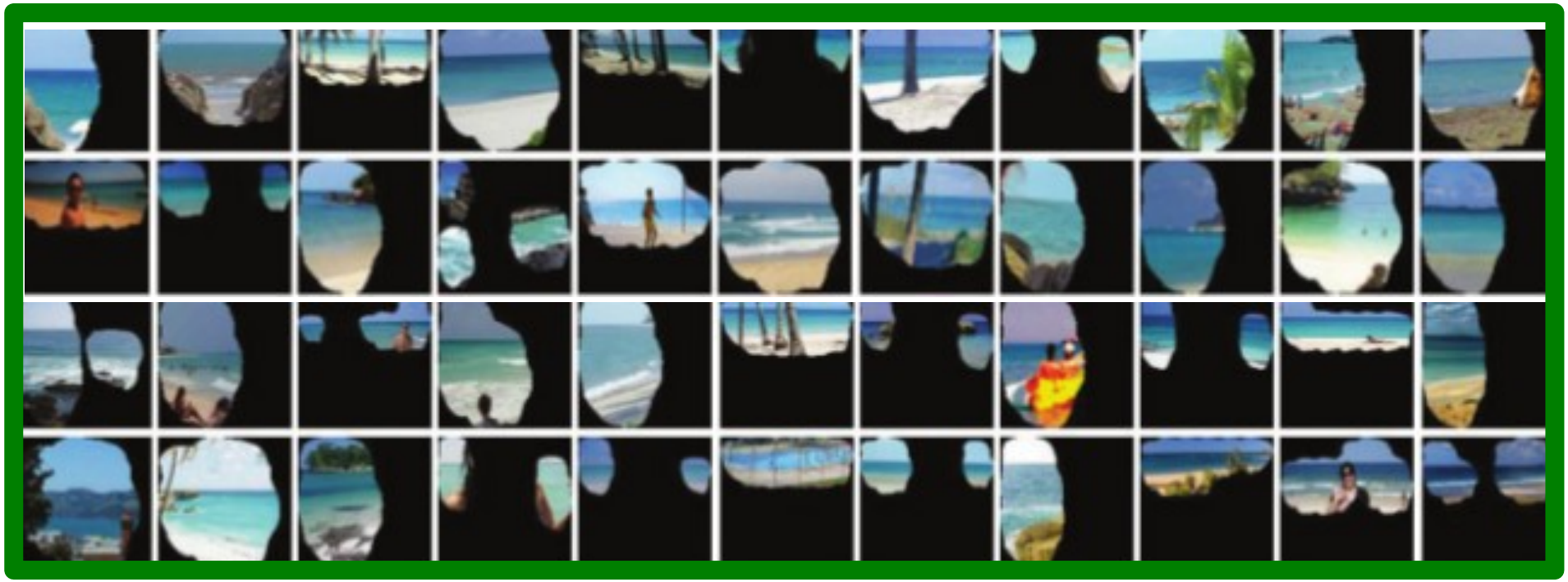
Annotating the Semantics of Units

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



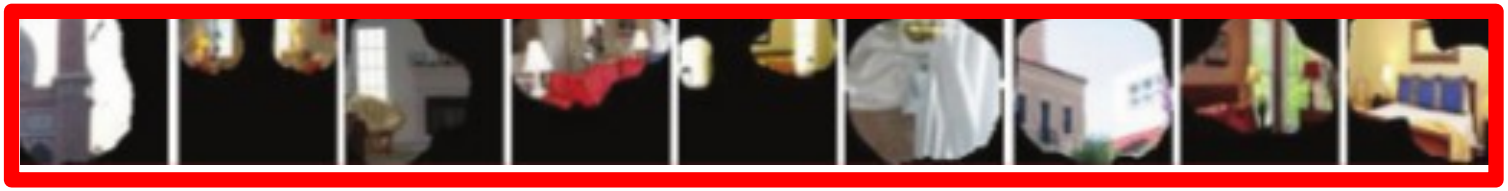
Annotating the Semantics of Units

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



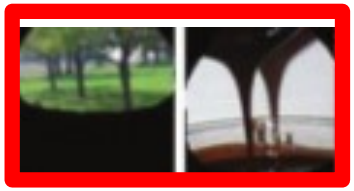
Annotating the Semantics of Units

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



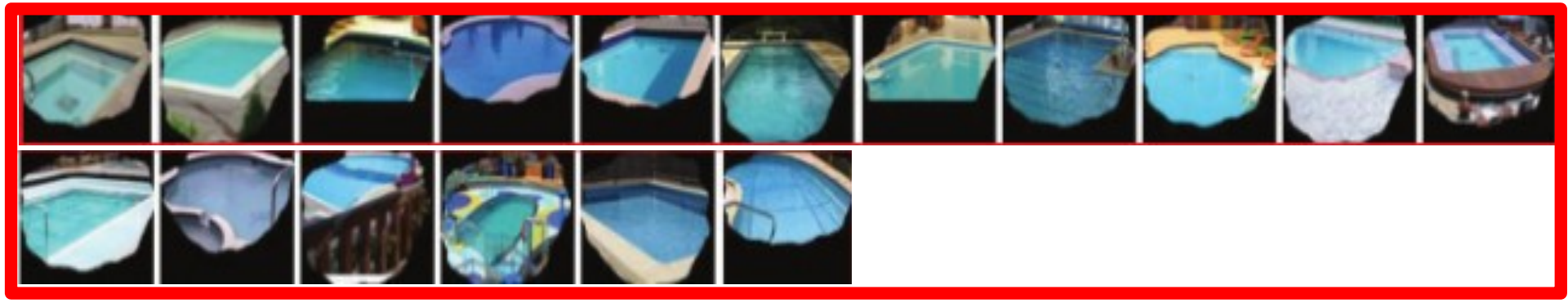
Annotating the Semantics of Units

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%



Annotating the Semantics of Units

Pool5, unit 22; Label: dinner table; Type: scene; Precision: 60%

ImageNet vs. PlacesNet

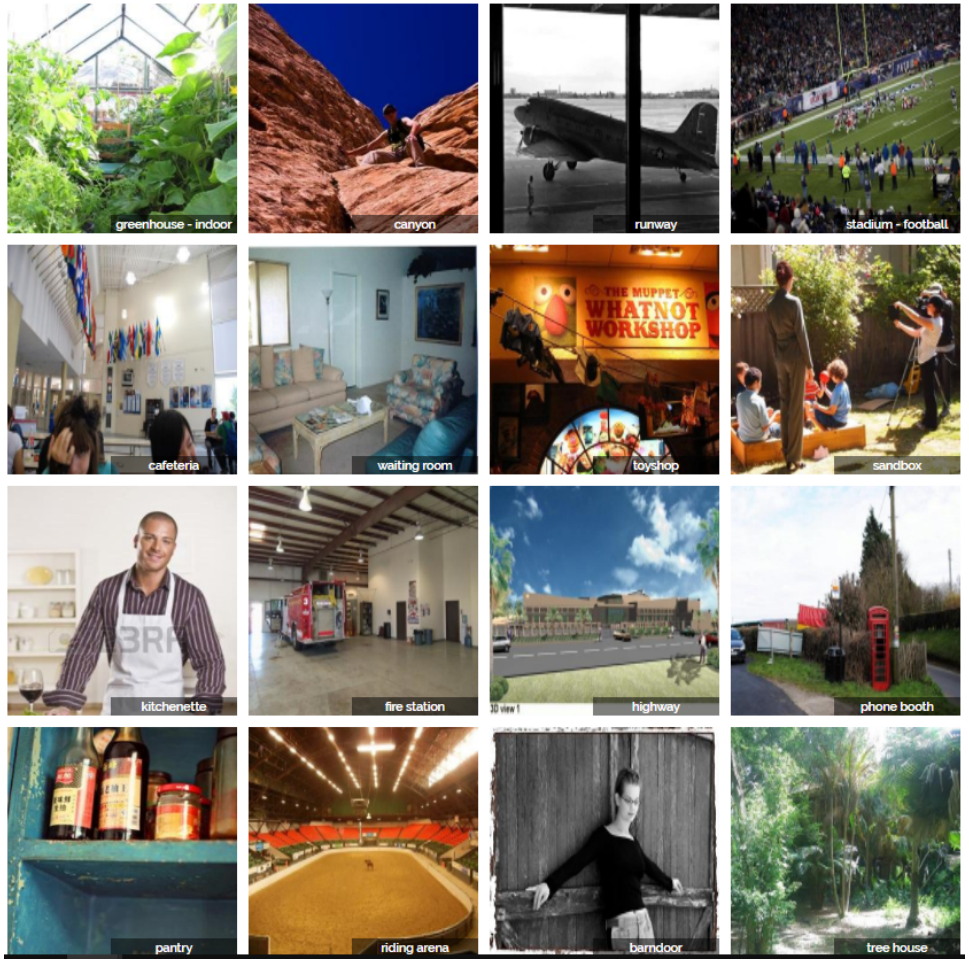
http://places2.csail.mit.edu/demo.html

ImageNet

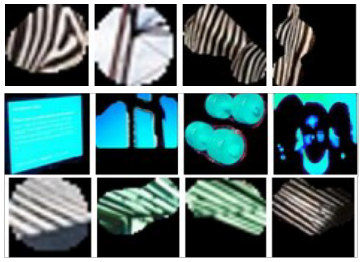
- ~1 mil object-level images over 1000 classes

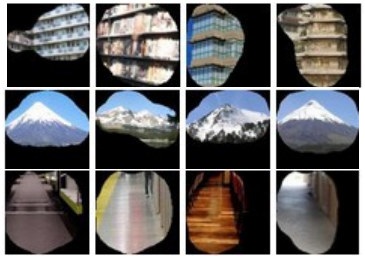
PlacesNet

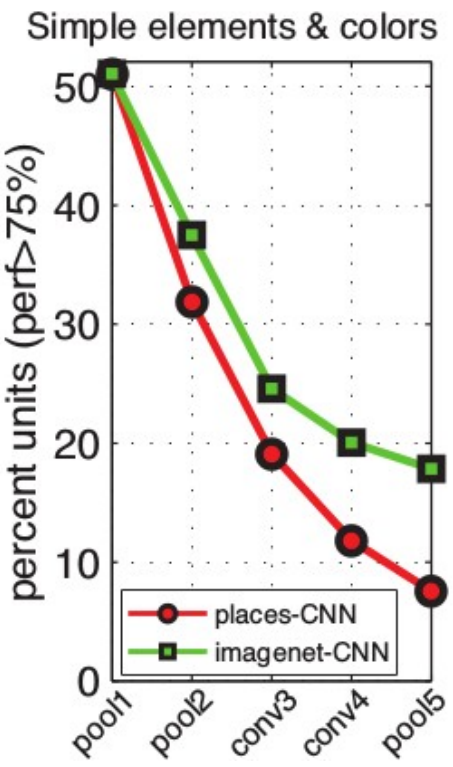
- ~1.8 million images from 365 scene categories (at most 5000 images per category).

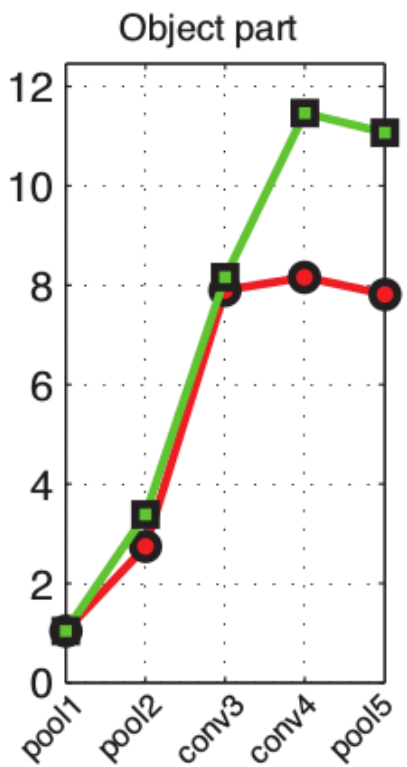


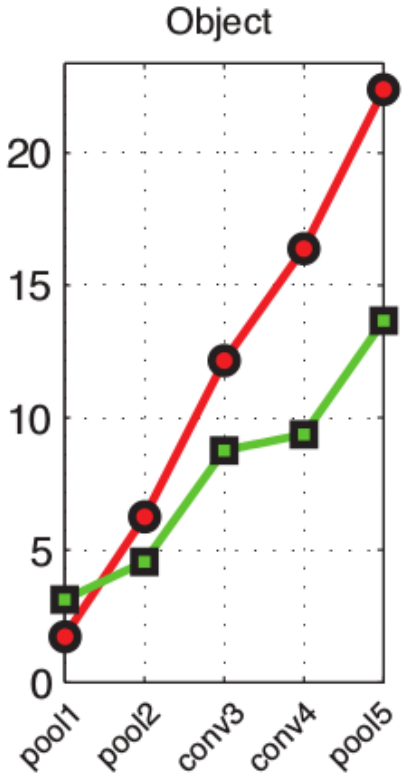
Distribution of Semantic Types at Each Layer

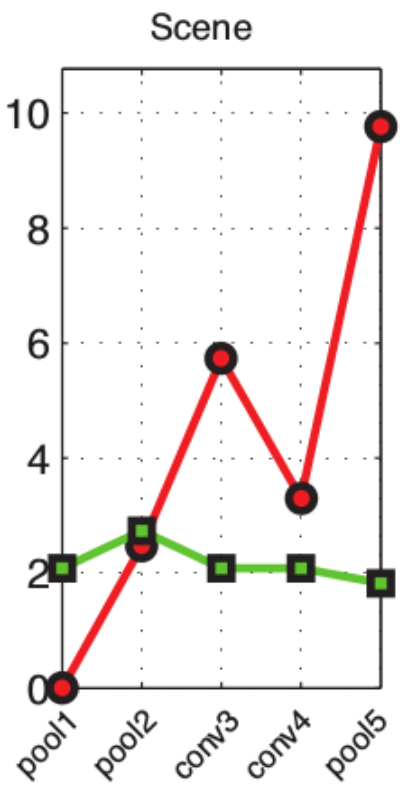




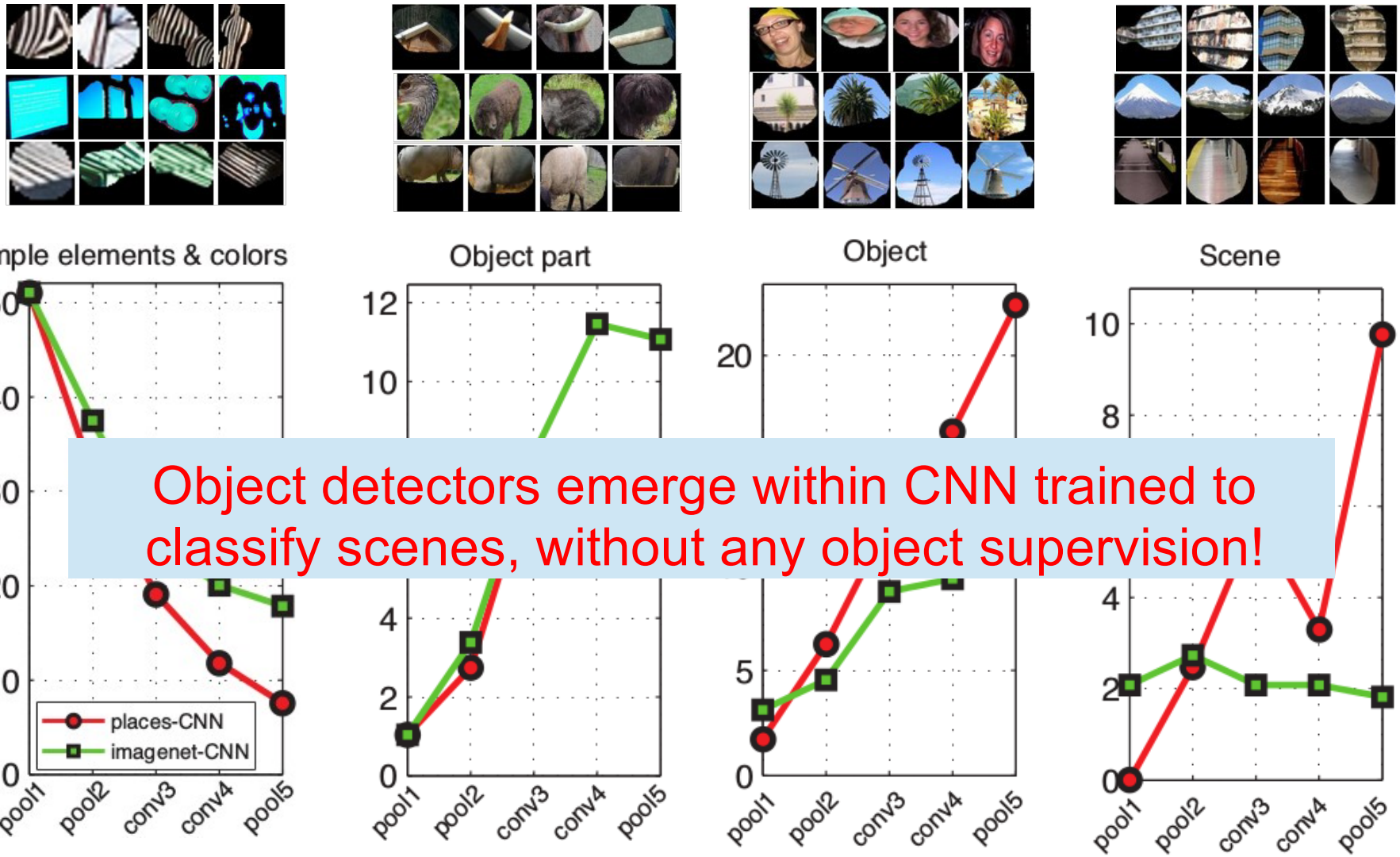








Distribution of Semantic Types at Each Layer



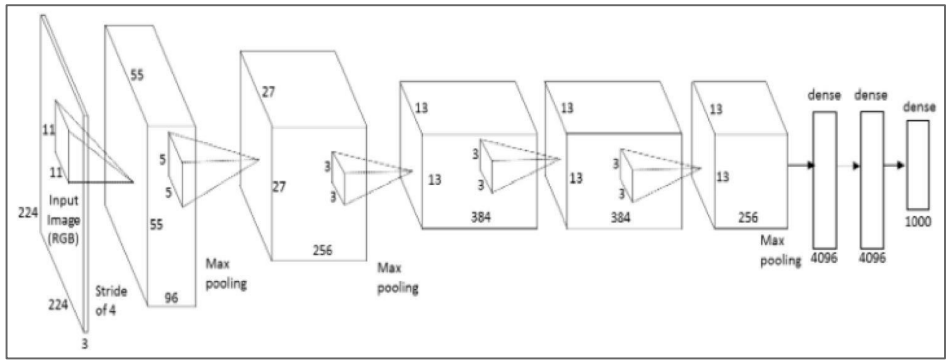
Object detectors emerge within CNN trained to classify scenes, without any object supervision!

Interesting CNN properties

…or other ways to measure reception

http://yosinski.com/deepvis

What input to a neuron maximizes a class score?

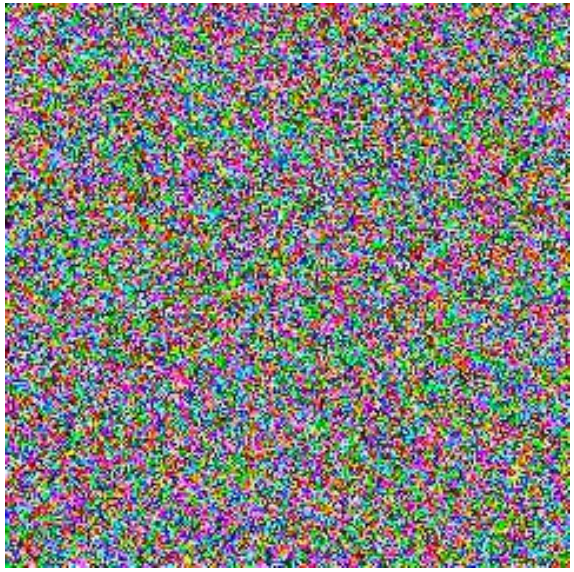


To visualize the function of a specific unit in a neural network, we synthesize an input to that unit which causes high activation.

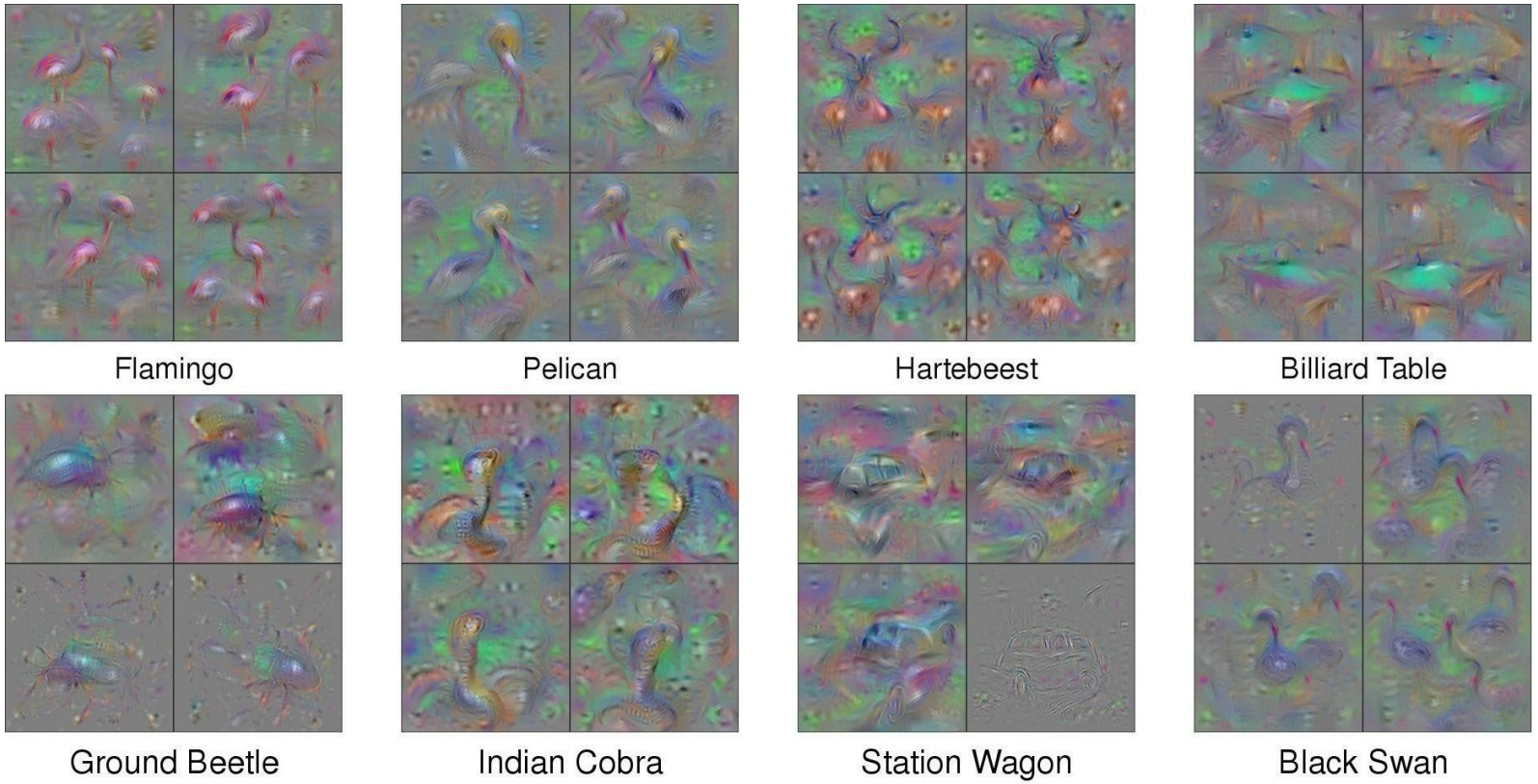
Neuron of choice *i*
An image of random noise *x*.

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.



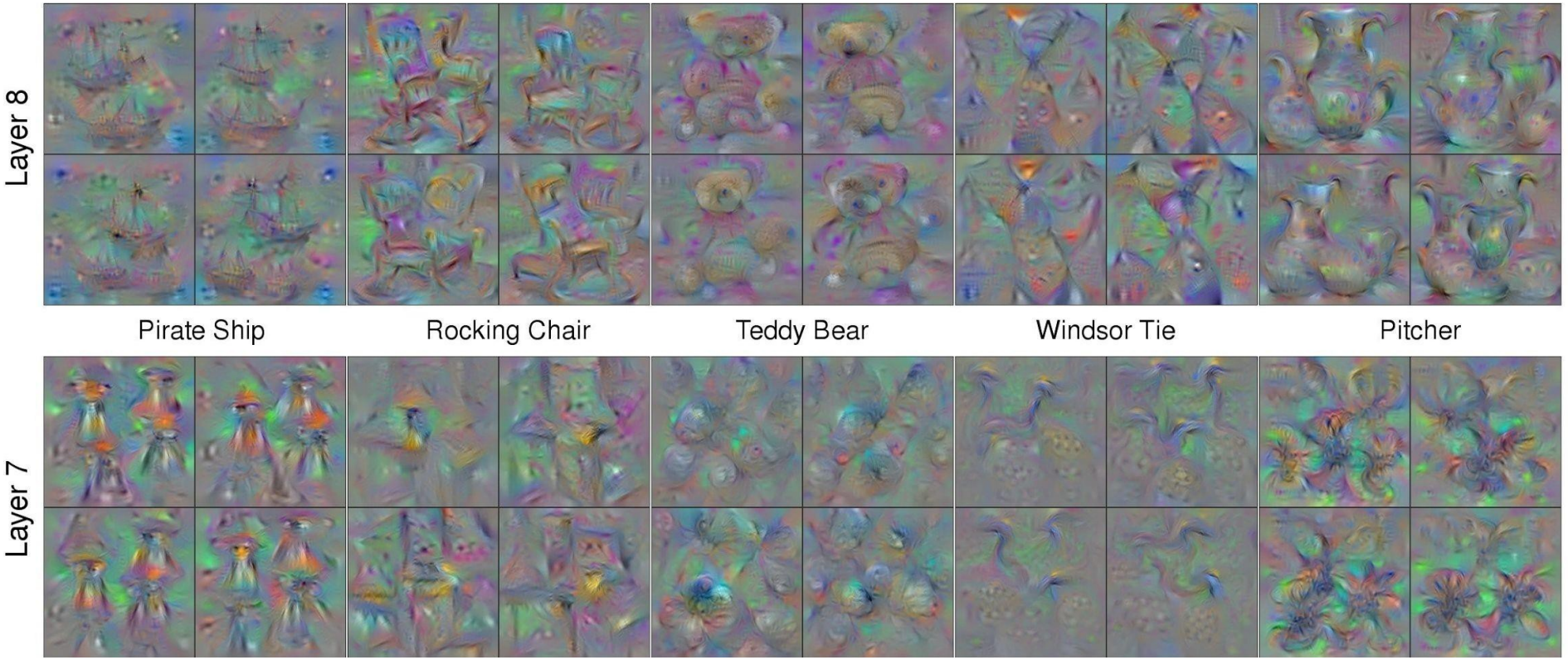
What image maximizes a class score?



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

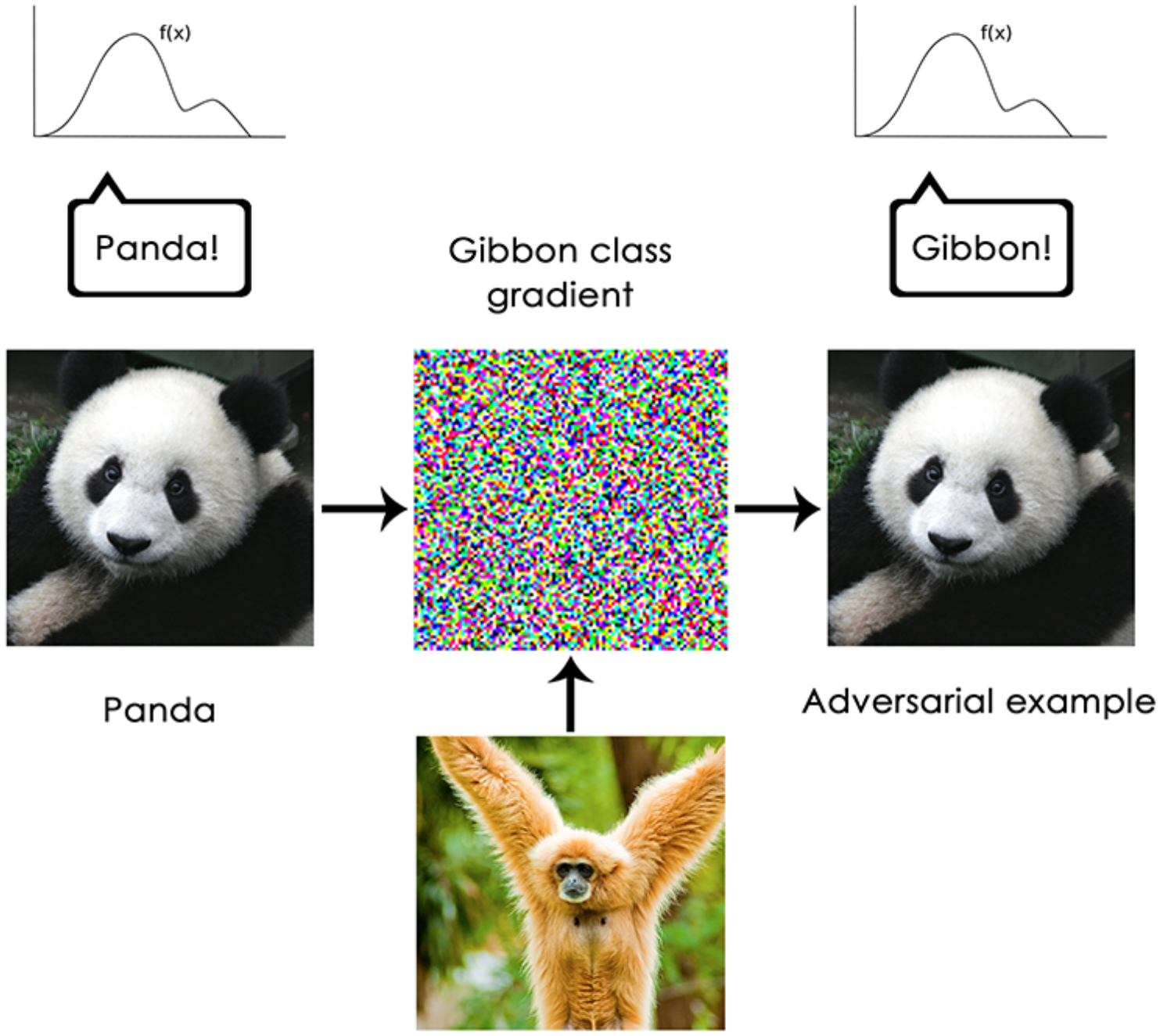
http://yosinski.com/deepvis

What image maximizes a class score?



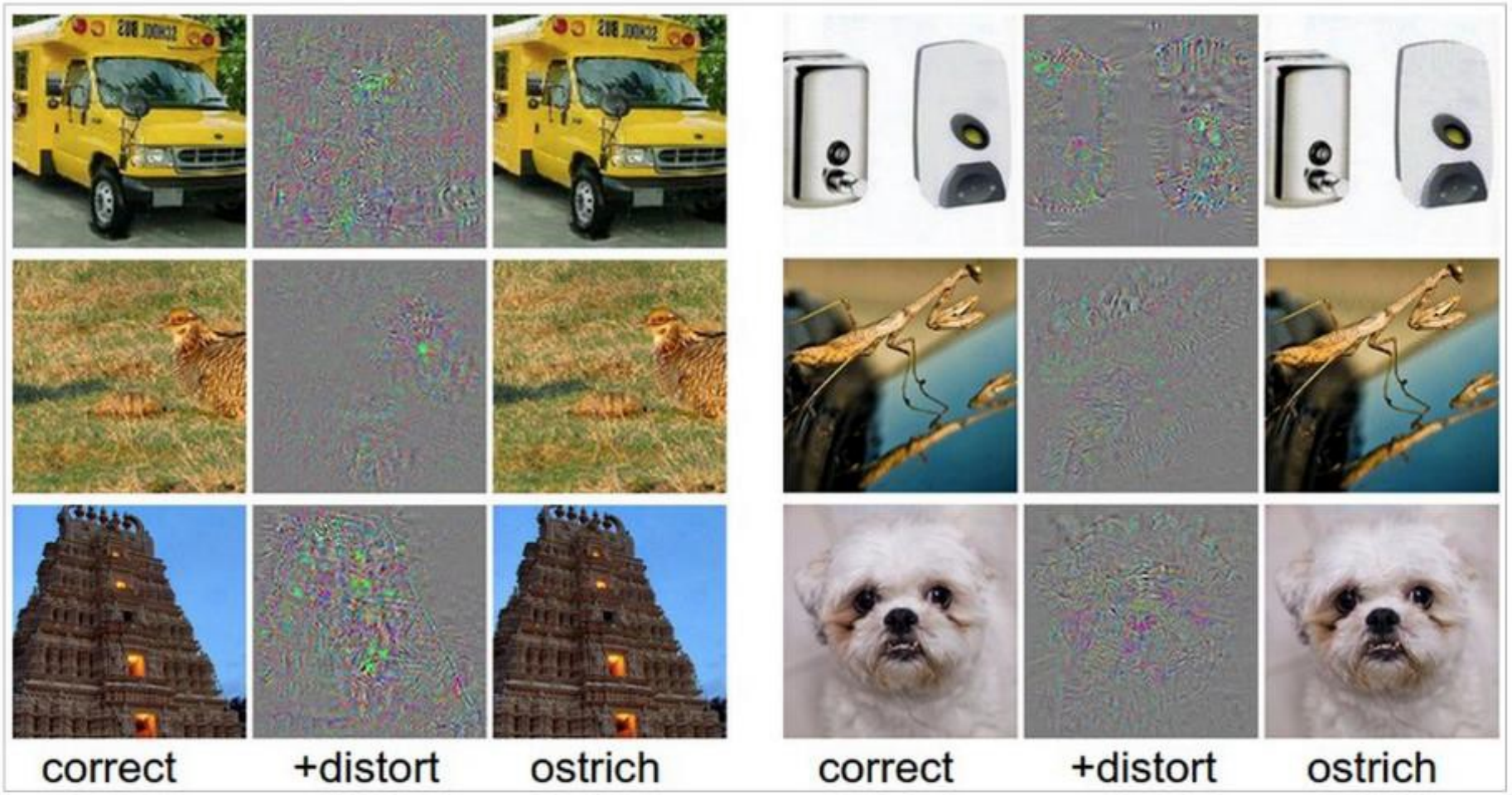
Wow!

They just ‘fall out’!



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

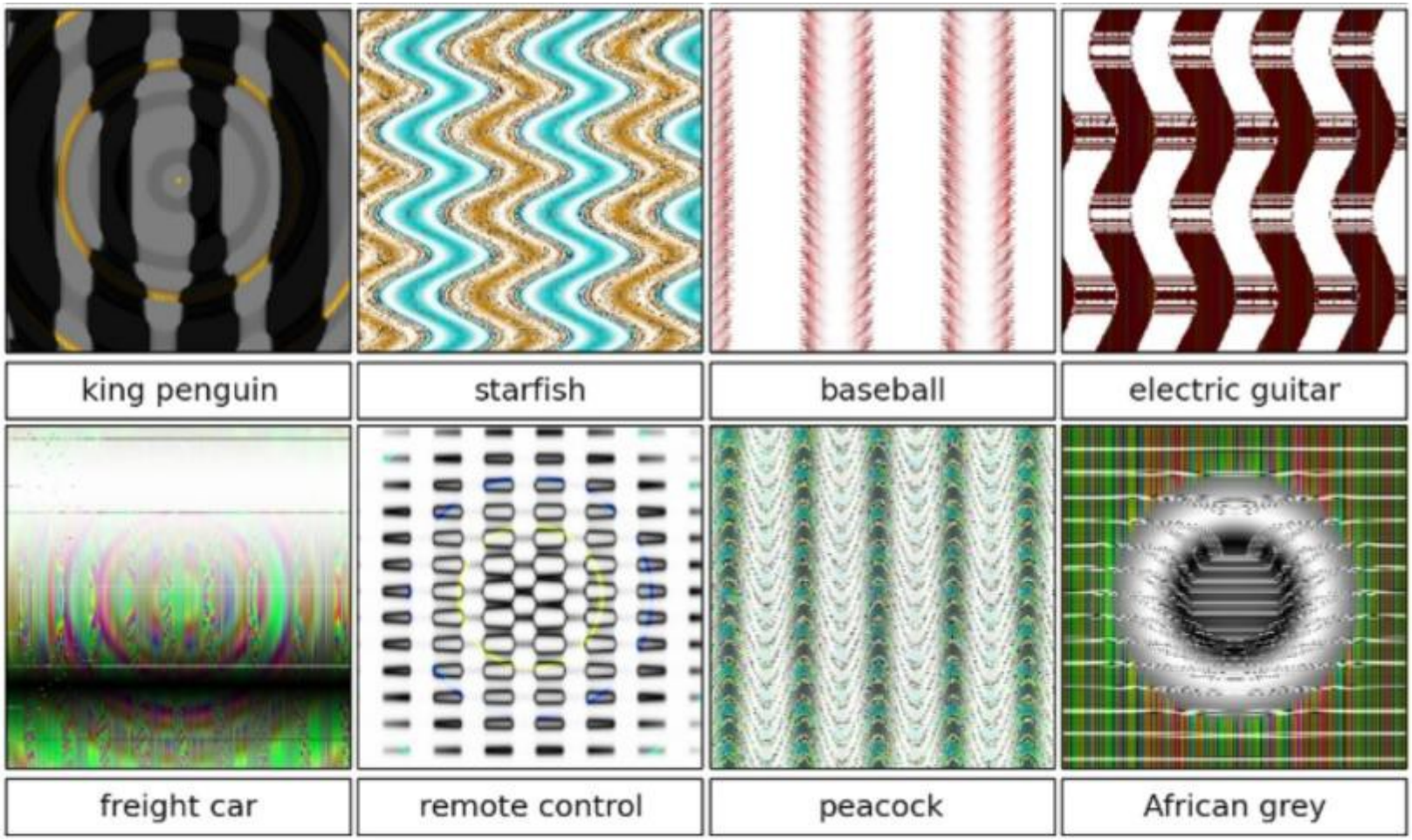
Breaking CNNs



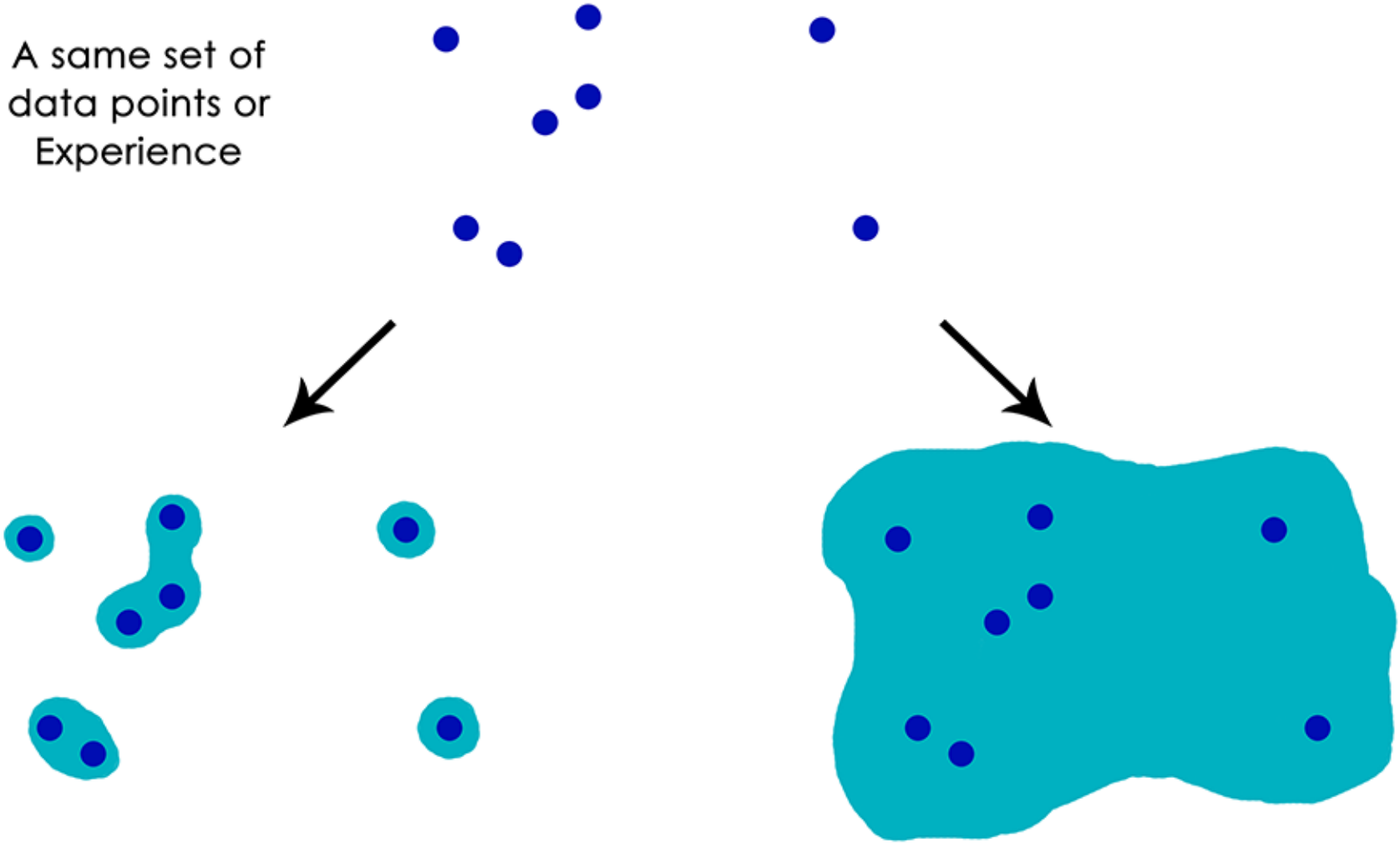
Take a correctly classified image (left image in both columns), and add a tiny distortion (middle) to fool the ConvNet with the resulting image (right).

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]



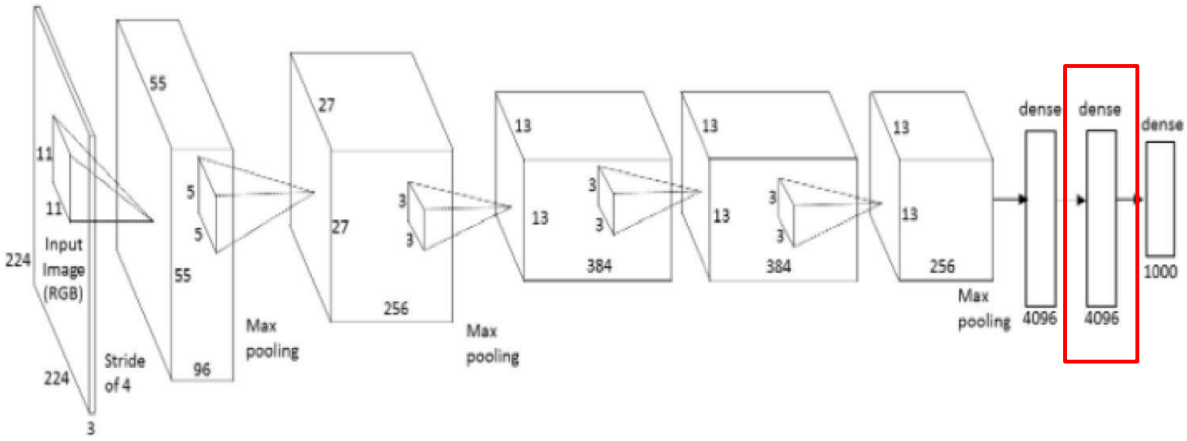
Francois Chollet - https://blog.keras.io/the-limitations-of-deep-learning.html

The boy is holding a baseball bat.

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

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

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

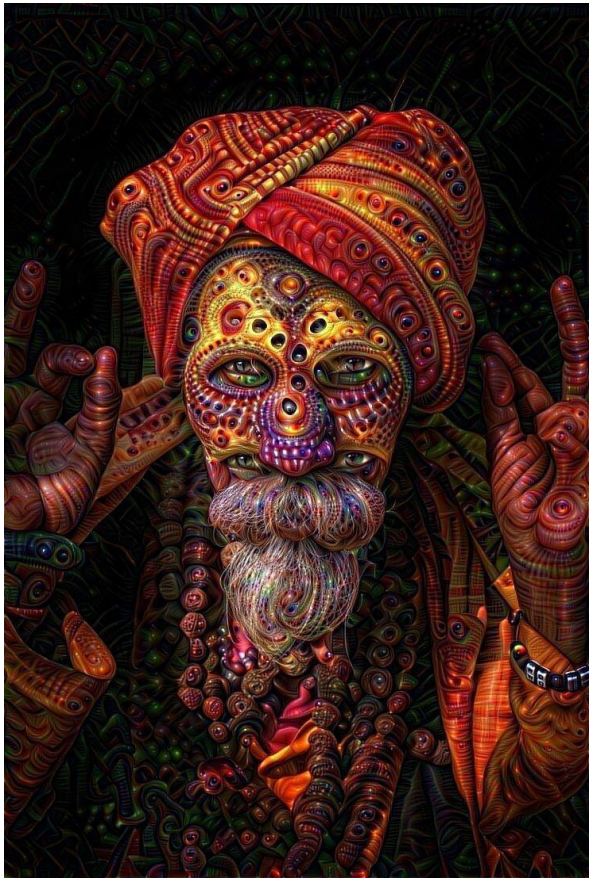
Reconstructions from the 1000 log probabilities for ImageNet (ILSVRC) classes

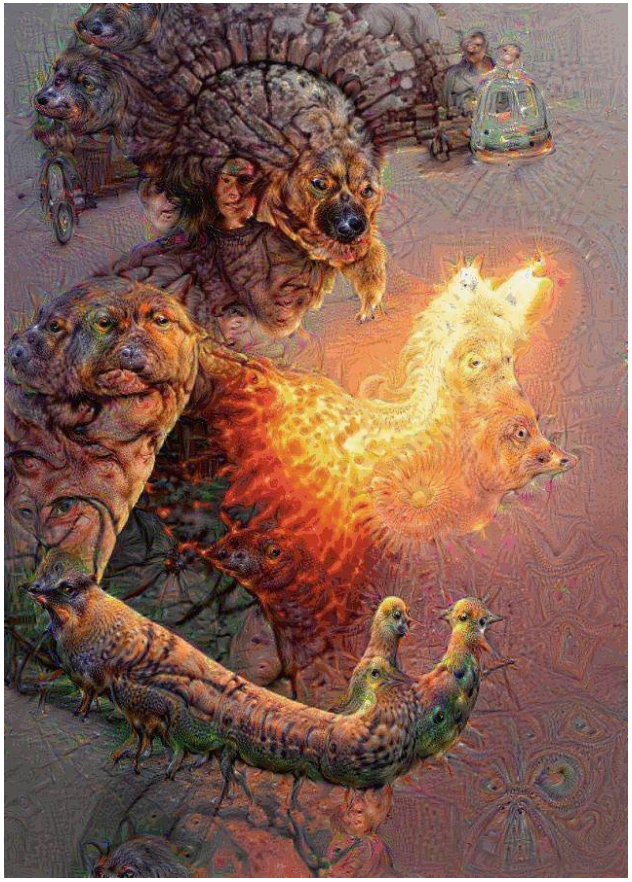
Understanding Deep Image Representations by Inverting Them
[Mahendran and Vedaldi, 2014]

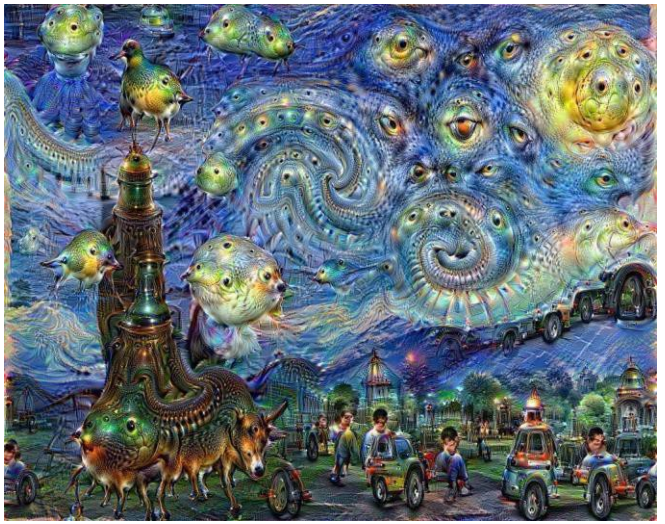
Reconstructing images

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

DeepDream

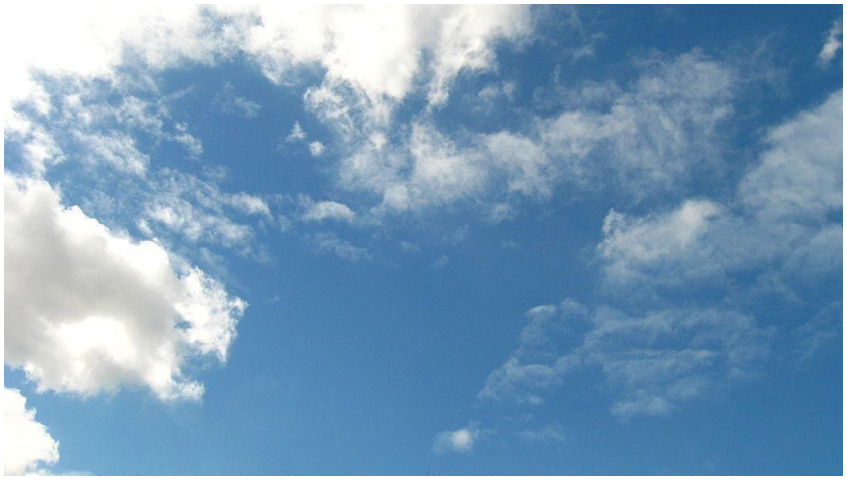






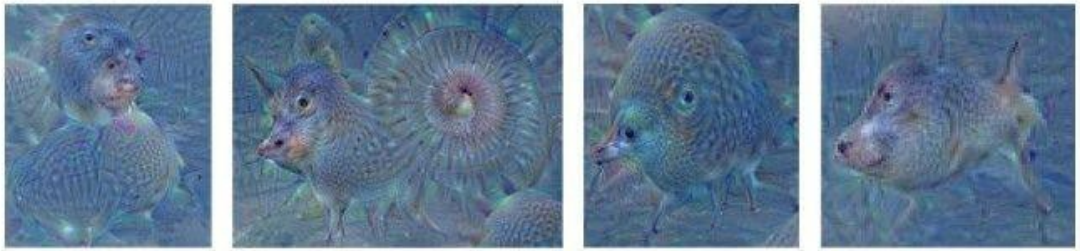
DeepDream https://github.com/google/deepdream

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

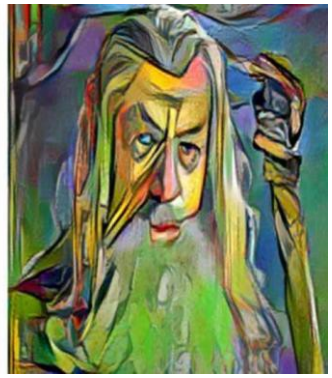
Deep Dream Grocery Trip
https://www.youtube.com/watch?v=DgPaCWJL7XI

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

Style transfer

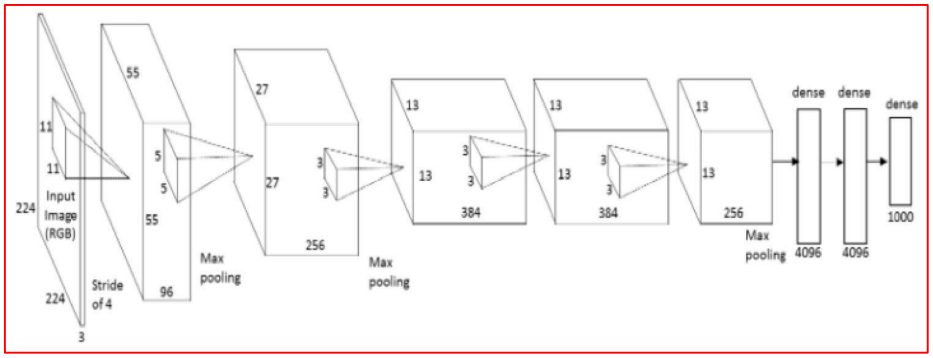
Neural Style

[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)

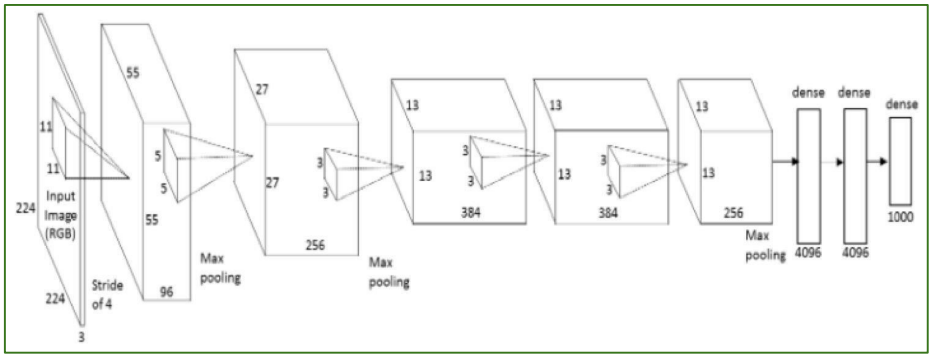


content activations

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

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

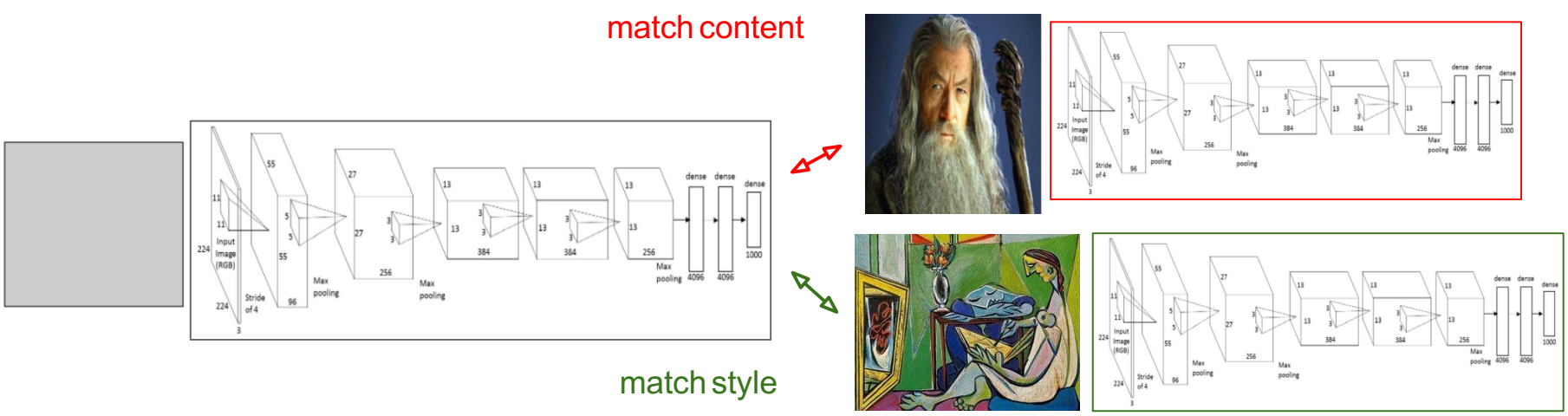
e.g.
at CONV1 layer (with [224x224x64] activations) would give a [64x64] Gram matrix of all pairwise activation covariances (summed across spatial locations)

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