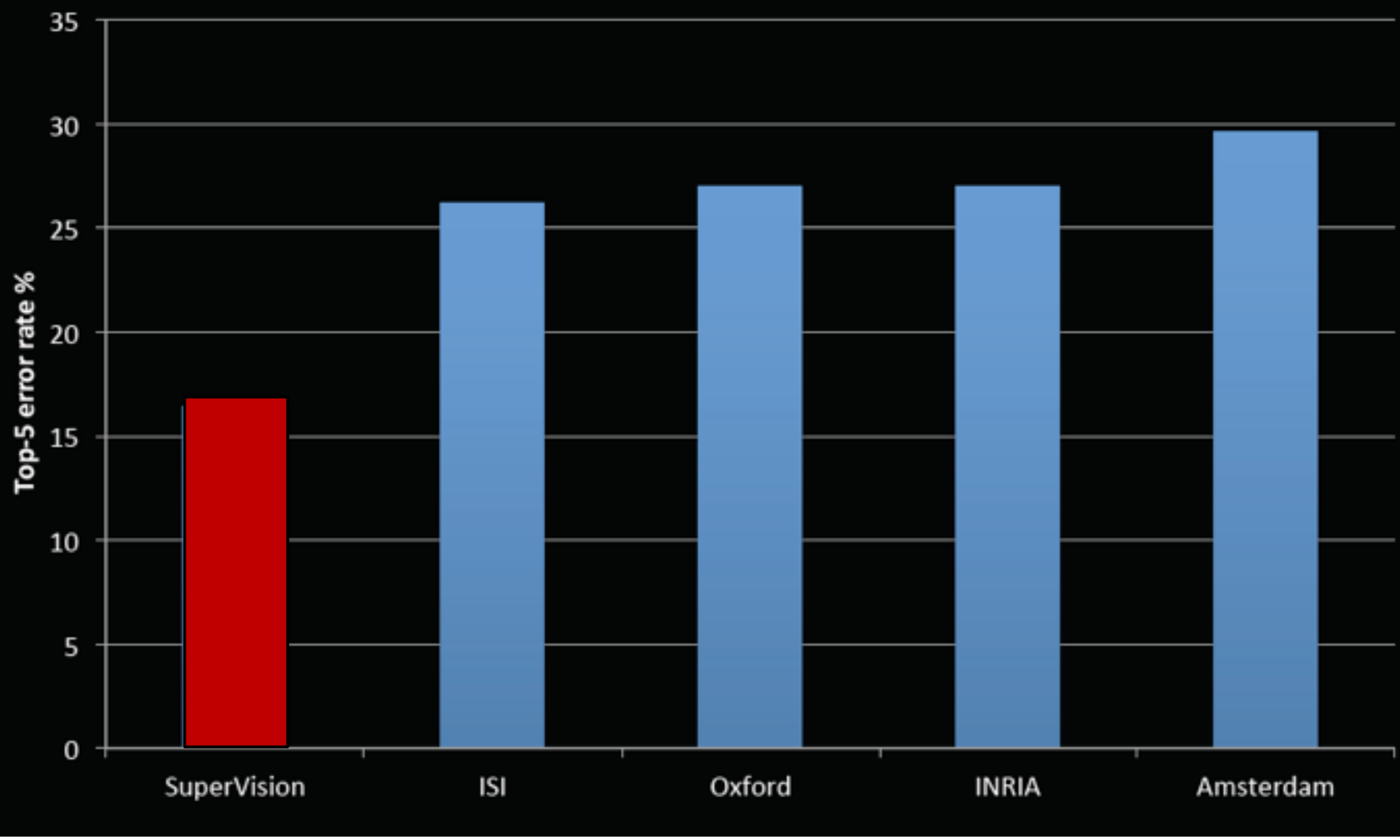
Learning Based Vision II

Computer Vision
Fall 2019
Columbia University

Slide from Rob Fergus, NYU

ImageNet Classification 2012

- Krizhevsky et al. -- 16.4% error (top-5)
- Next best (non-convnet) – 26.2% error



Breaking Neural Nets

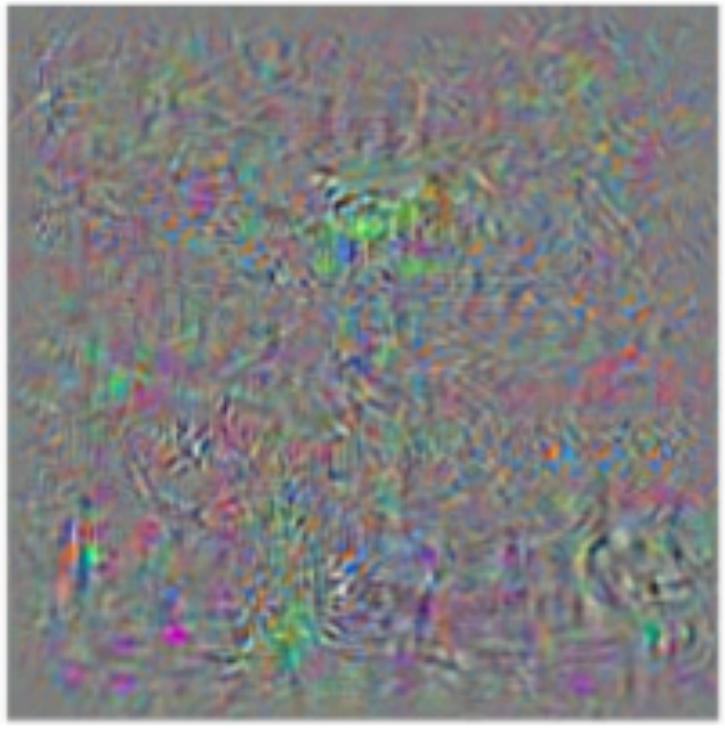
“school bus”

“school bus”

“ostrich”

“school bus”

+

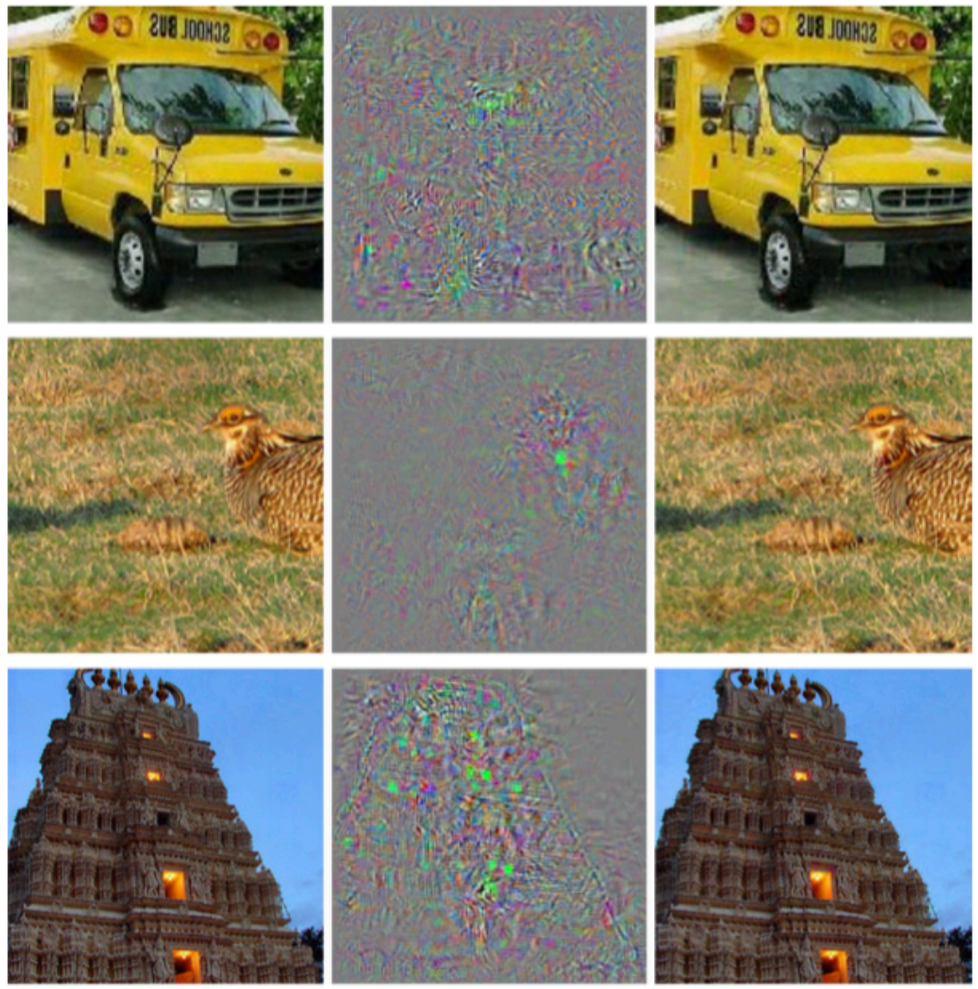


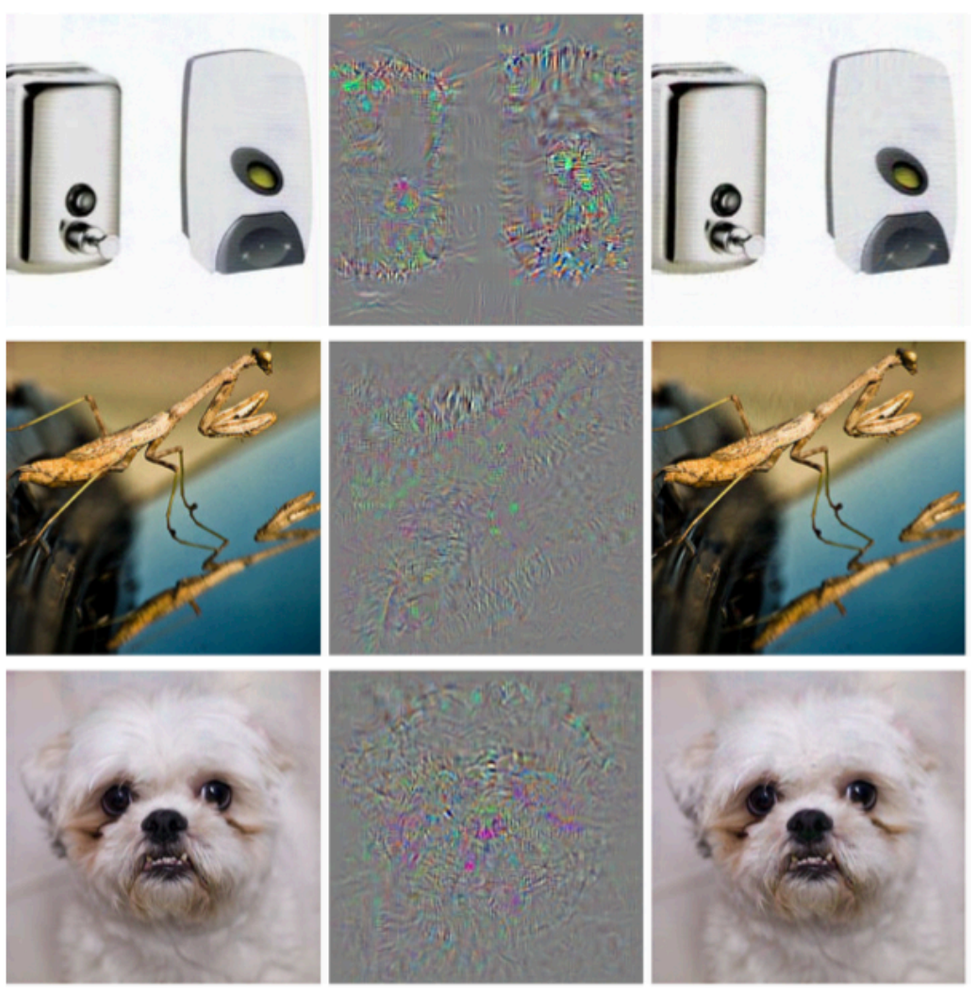
(scaled for visualization)

=

“ostrich”

Images on left are correctly classified
Images on the right are incorrectly classified as ostrich





Intriguing properties of neural networks

Christian Szegedy
Google Inc.

Wojciech Zaremba
New York University

Ilya Sutskever
Google Inc.

Joan Bruna
New York University

Dumitru Erhan
Google Inc.

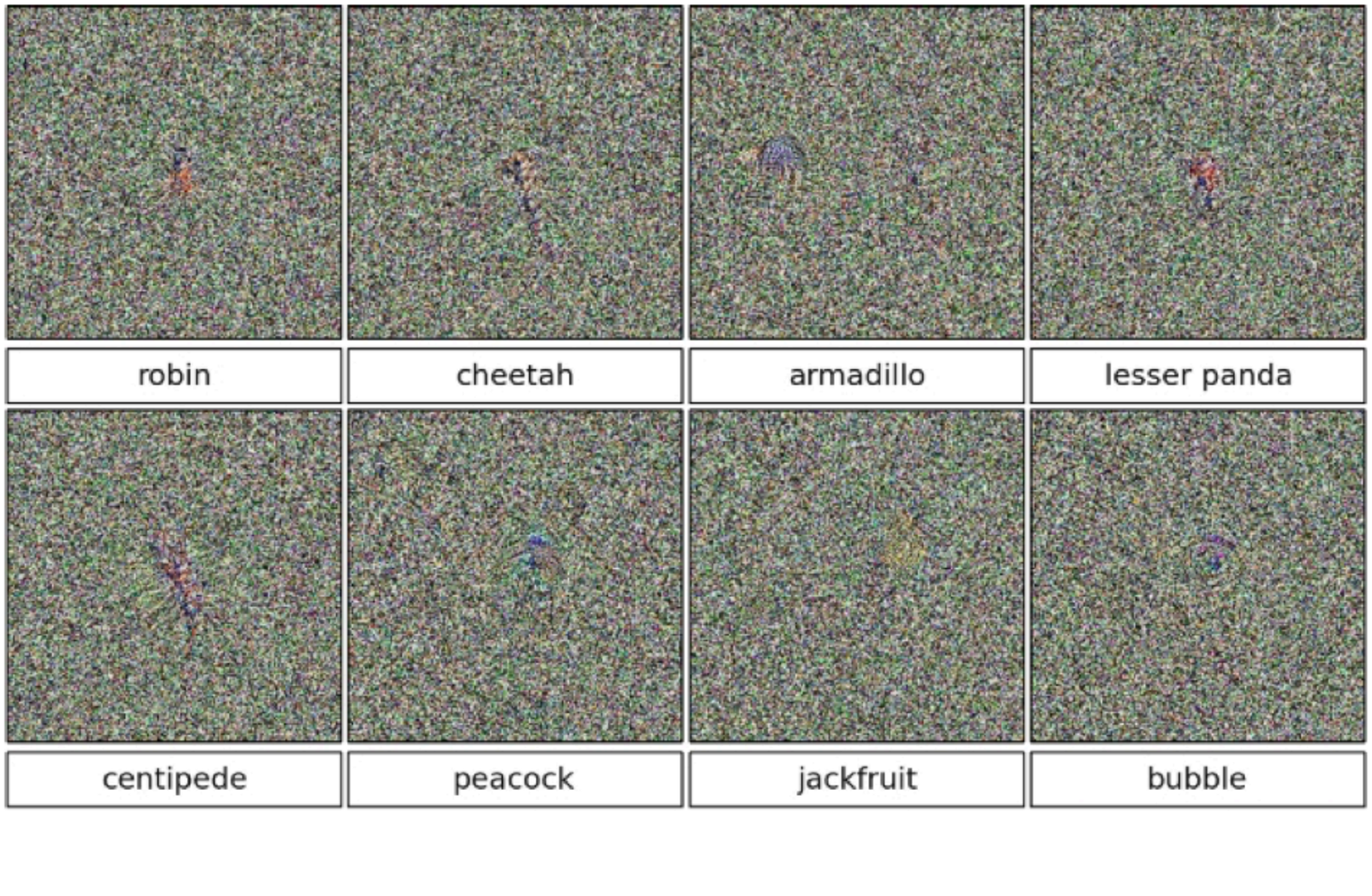
Ian Goodfellow
University of Montreal

Rob Fergus
New York University
Facebook Inc.

How can we find these?

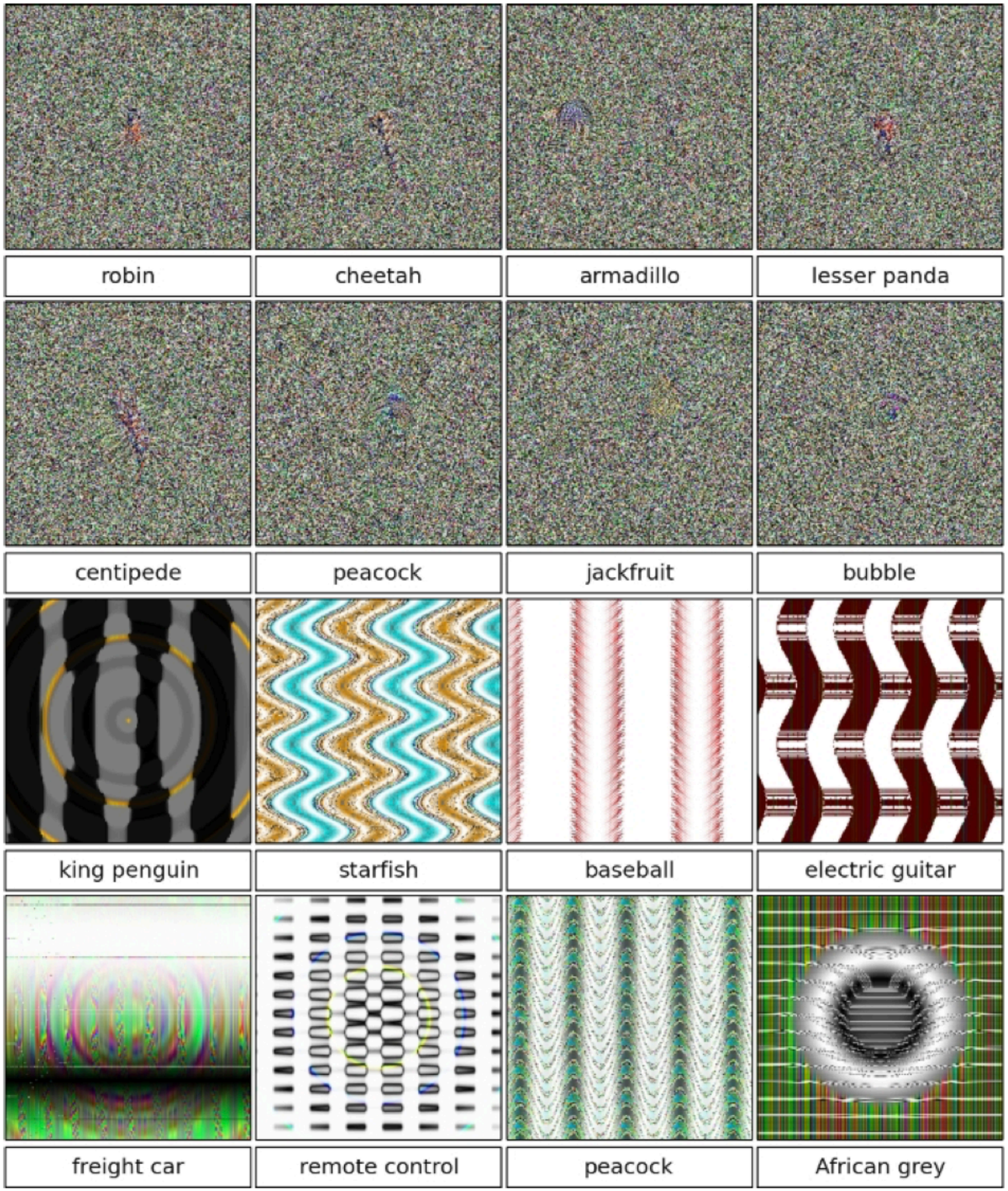
Solve optimization problem to find minimal change that maximizes the loss

$$\max_{\Delta} \mathscr{L}\left(f(x+\Delta), y\right) - \lambda \|\Delta\|_2^2$$



99% confidence!

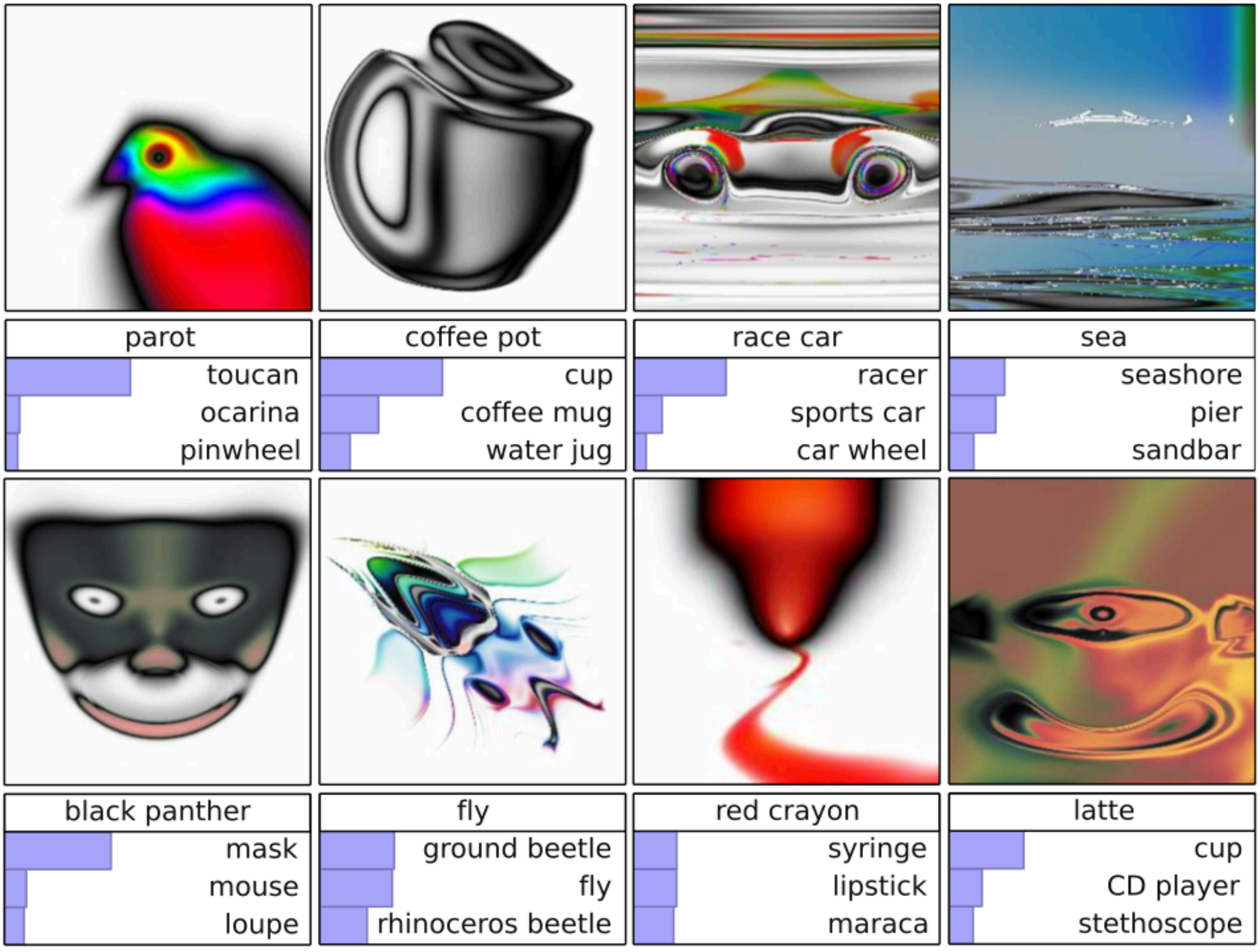
Nguyen, Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images



99% confidence!

Also 99% confidence!

Nguyen, Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images



Nguyen, Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images

Universal attacks

(a) CaffeNet (b) VGG-F (c) VGG-16

(d) VGG-19 (e) GoogLeNet (f) ResNet-152

Moosave-Dezfooli et al. arXiv 1610.08401

Universal attacks

Attack is agnostic to the image content

wool

Indian elephant

Indian elephant

African grey

tabby

African grey

common newt

carousel

grey fox

macaw

three-toed sloth

macaw

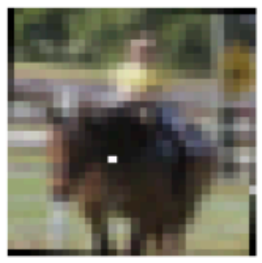
Moosave-Dezfooli et al. arXiv 1610.08401

Change just one pixel

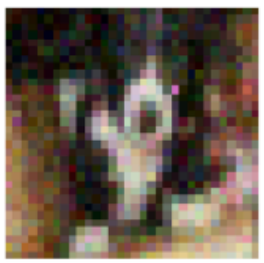
SHIP
CAR(99.7%)

HORSE
FROG(99.9%)

DEER
AIRPLANE(85.3%



HORSE
DOG(70.7%)

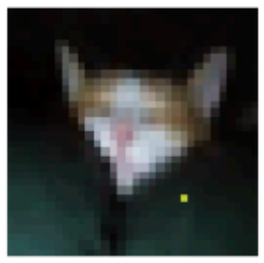


DOG
CAT(75.5%)

BIRD
FROG(86.5%)

CAR
AIRPLANE(82.4%)

DEER
DOG(86.4%)



CAT
BIRD(66.2%)

Su et al, “One pixel attack for fooling deep neural networks”

How to fix this?

- Generate adversarial template: $\eta = \epsilon \text{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right)$
- Add to the input image x
- Error rate of 87% with convolutional network on MINIST digit classification!
- Still active area of research

Goodfellow et al. Explaning and harnessing adversarial examples

In the physical world

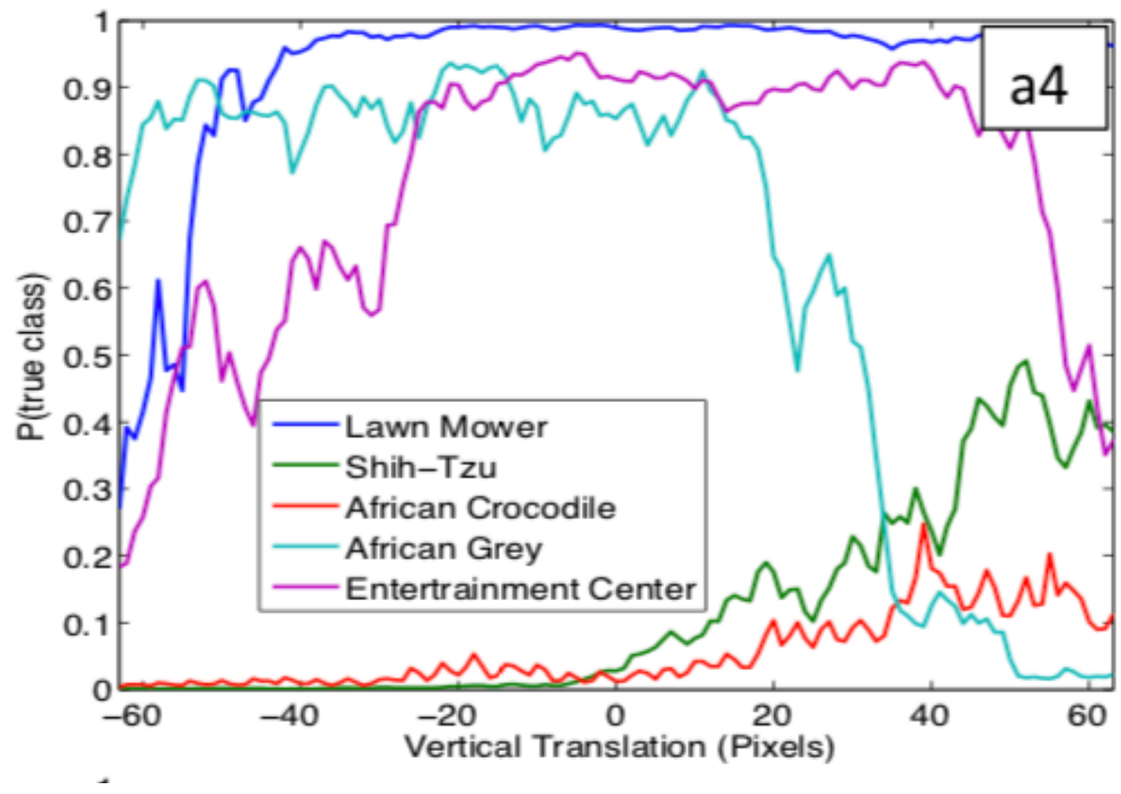
In the 3D physical world

Fooling Image Recognition

Neural network camouflage

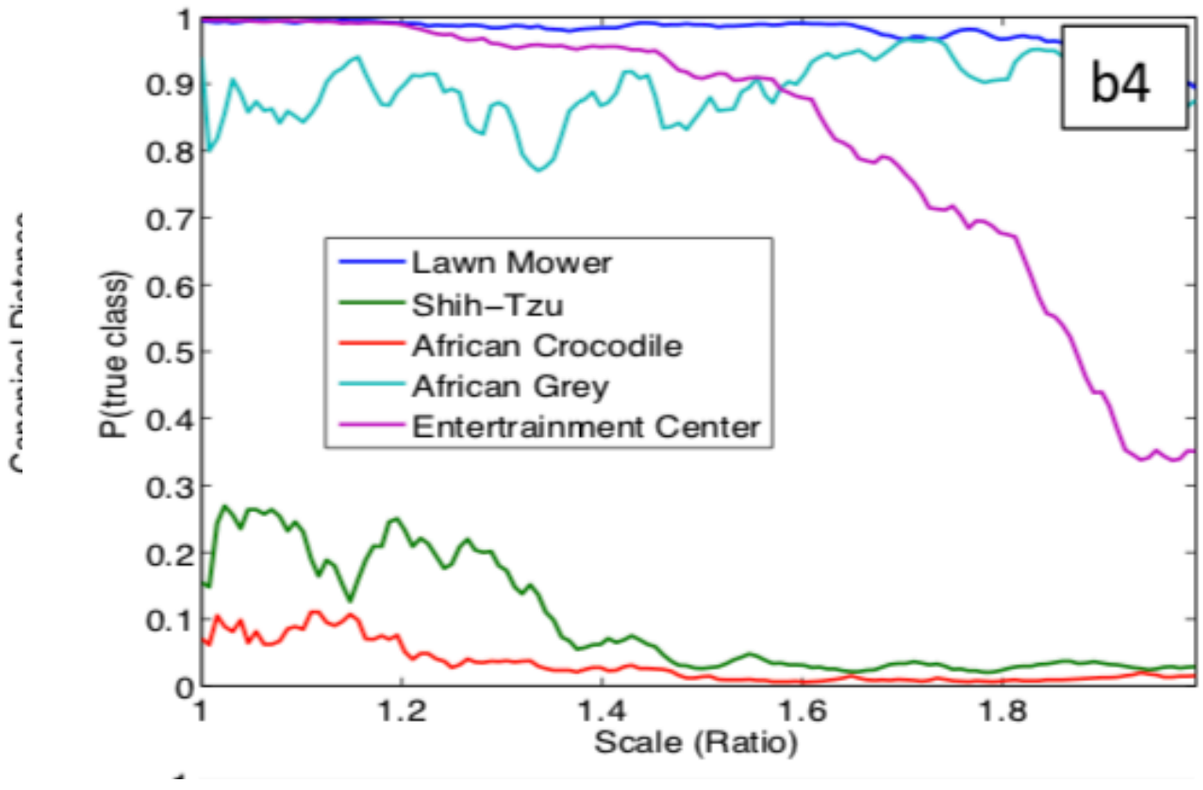
https://cvdazzle.com/

What if we translate?



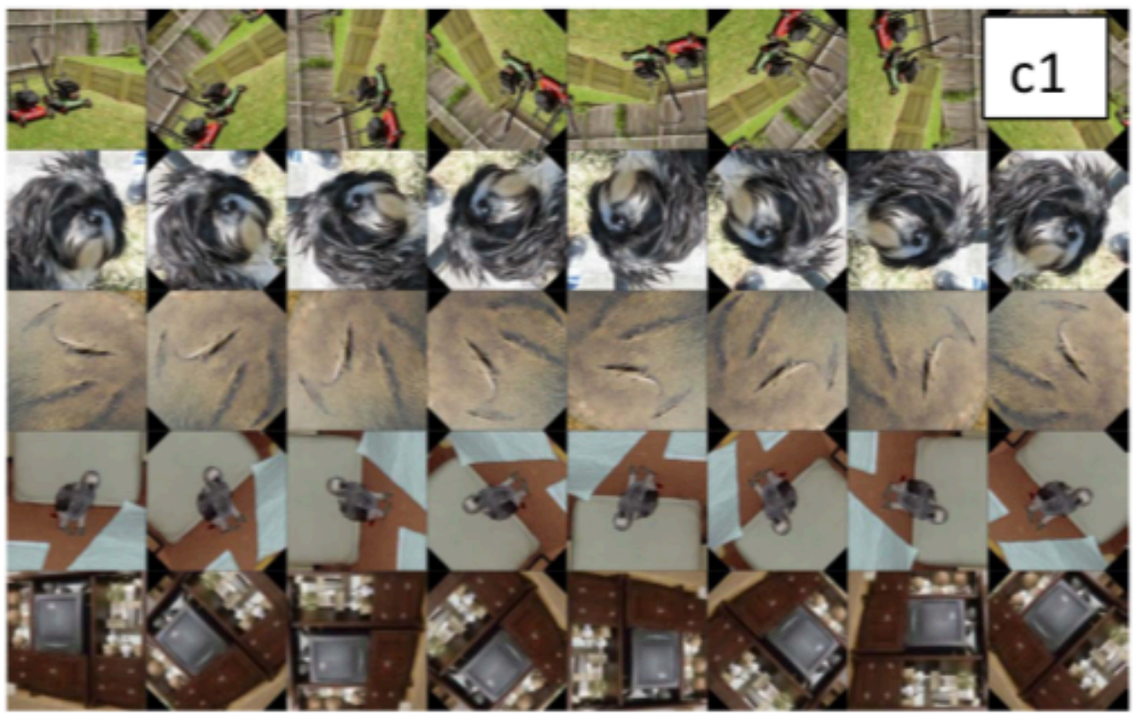
Zeiler and Fergus. Visualizing and Understanding Convolutional Networks

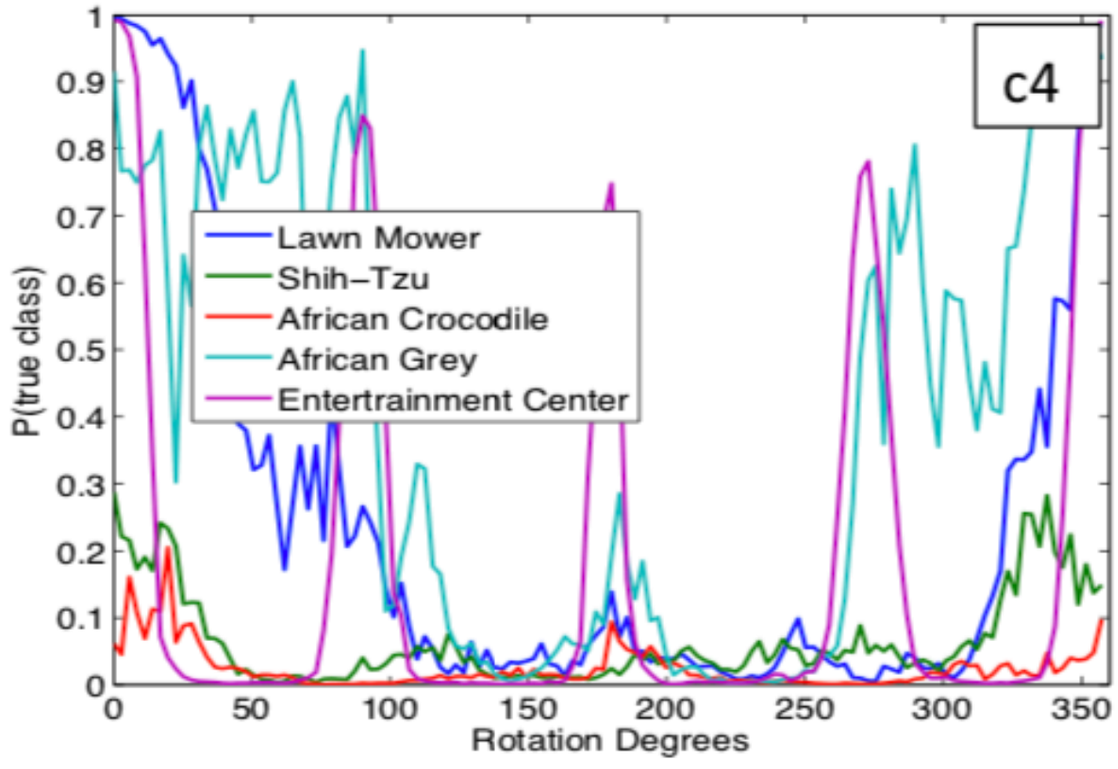
What if we scale?



Zeiler and Fergus. Visualizing and Understanding Convolutional Networks

What if we rotate?



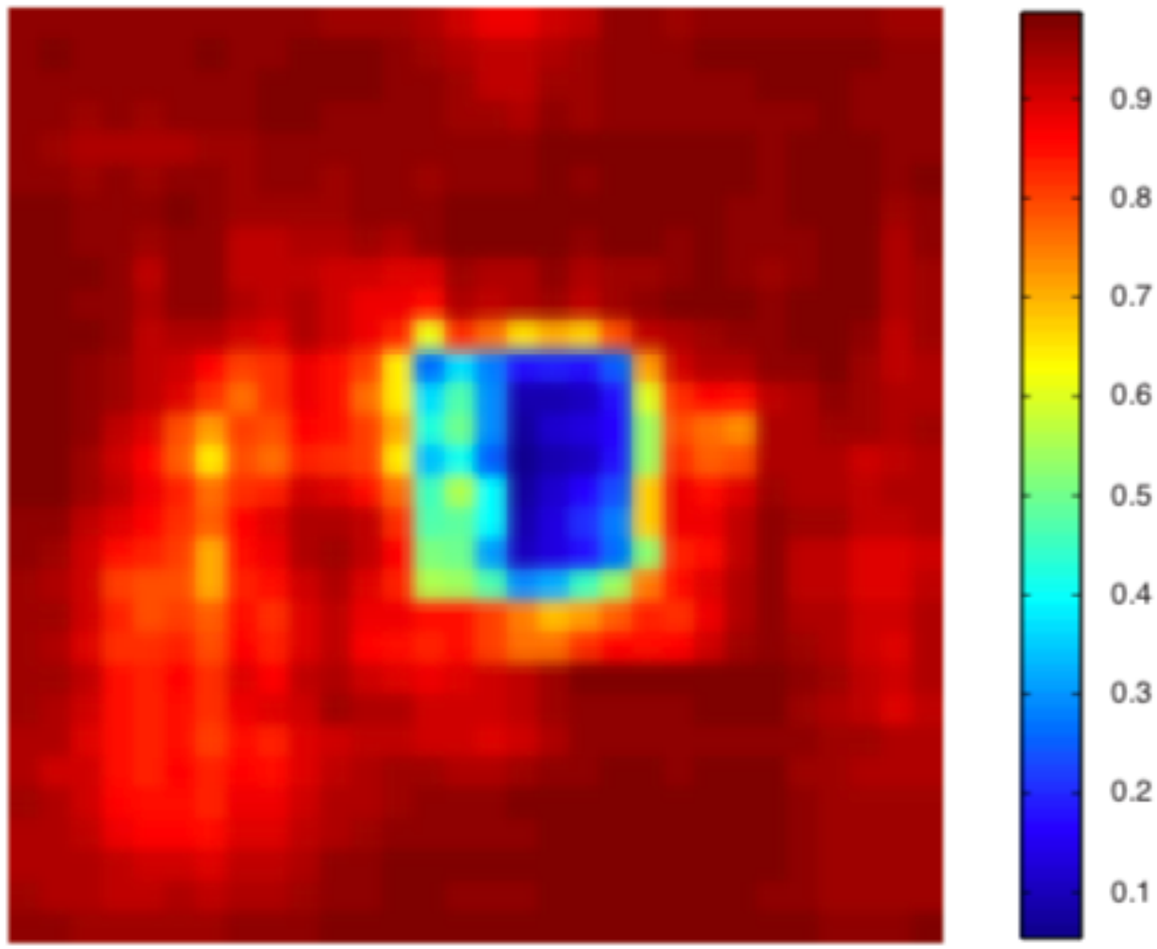


Zeiler and Fergus. Visualizing and Understanding Convolutional Networks

What does the neural network use to classify?

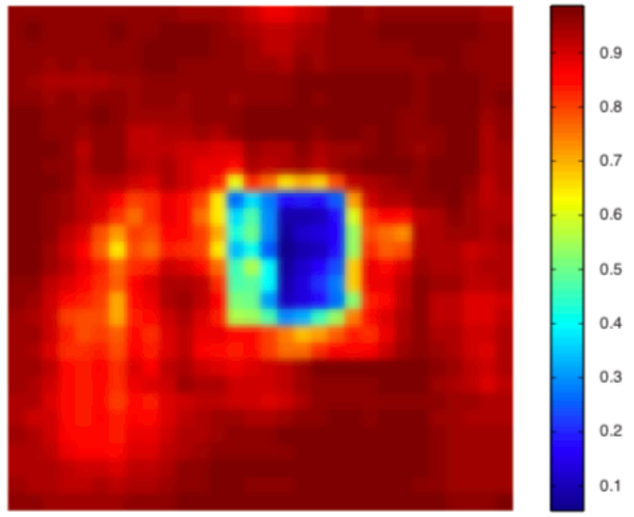
Zeiler and Fergus. Visualizing and Understanding Convolutional Networks

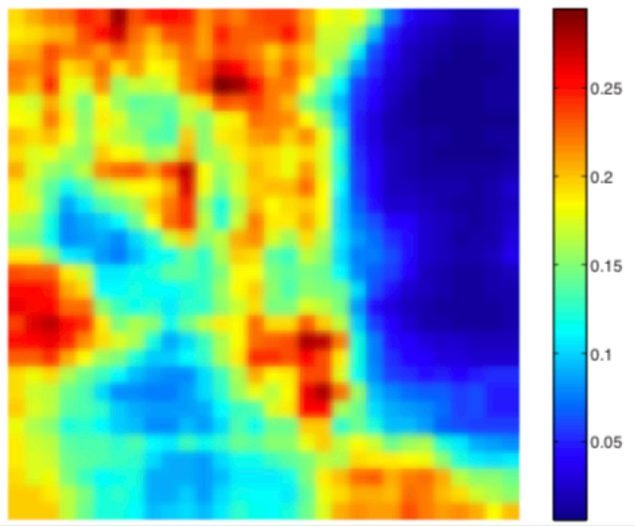
What does the neural network use to classify?

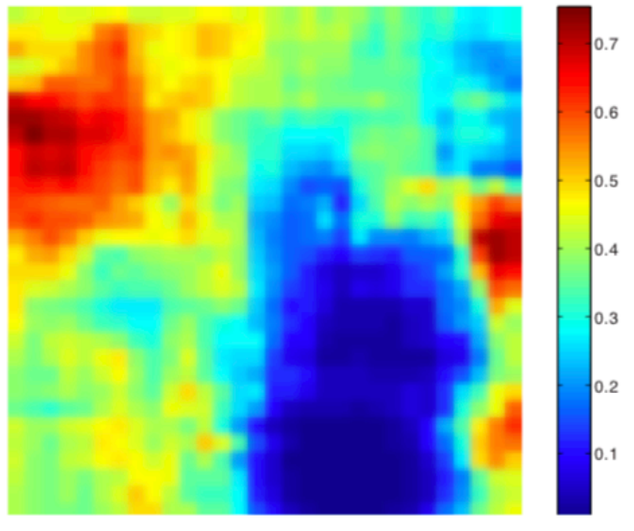


Probability of correct class

Zeiler and Fergus. Visualizing and Understanding Convolutional Networks







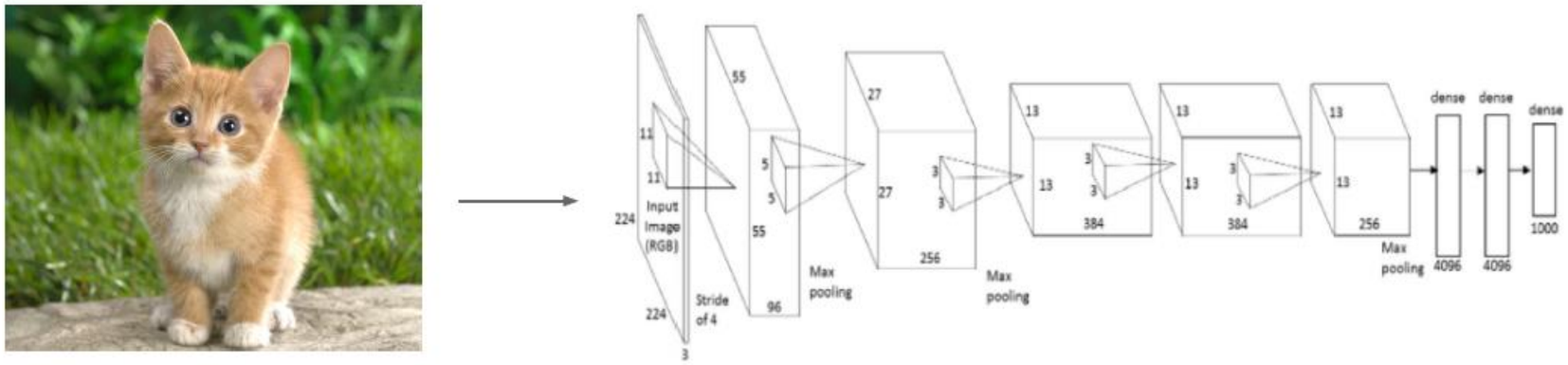
Probability of correct class

Which Pixels in the Input Affect the Neuron the Most?

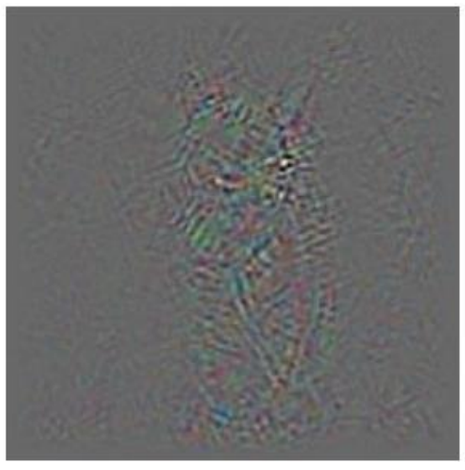
- Rephrased: which pixels would make the neuron not turn on if they had been different?
- In other words, for which inputs is $\frac{\partial neuron}{\partial x_i}$ large?

Typical Gradient of a Neuron

- Visualize the gradient of a particular neuron with respect to the input *x*
- Do a forward pass:



- Compute the gradient of a particular neuron using backprop:



“Guided Backpropagation”

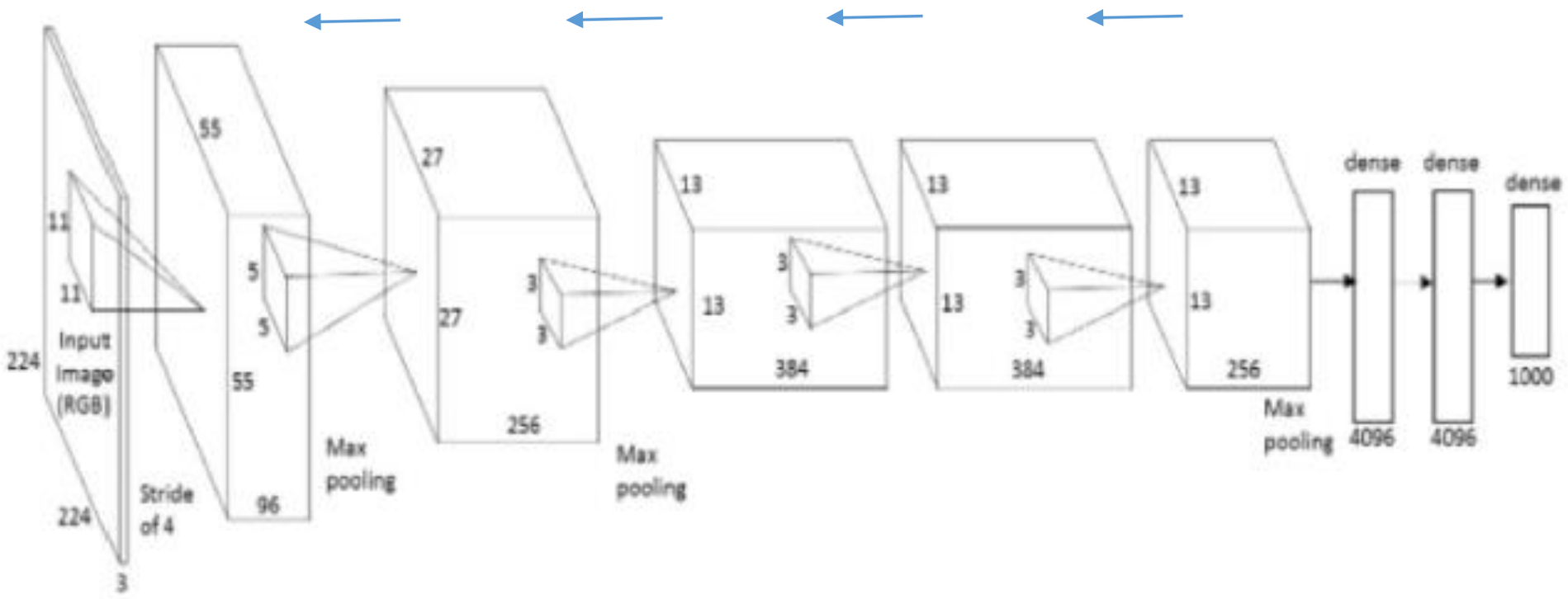
- Idea: neurons act like detectors of particular image features
- We are only interested in what image features the neuron detects, not in what kind of stuff it *doesn’t* detect
- So when propagating the gradient, we set all the negative gradients to 0
  - We don’t care if a pixel “suppresses” a neuron somewhere along the part to our neuron

Guided Backpropagation

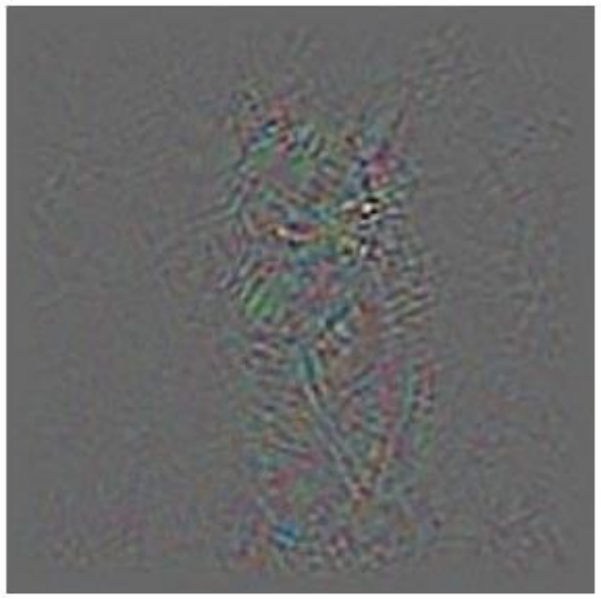
Compute gradient, zero out negatives, backpropagate

Compute gradient, zero out negatives, backpropagate

Compute gradient, zero out negatives, backpropagate



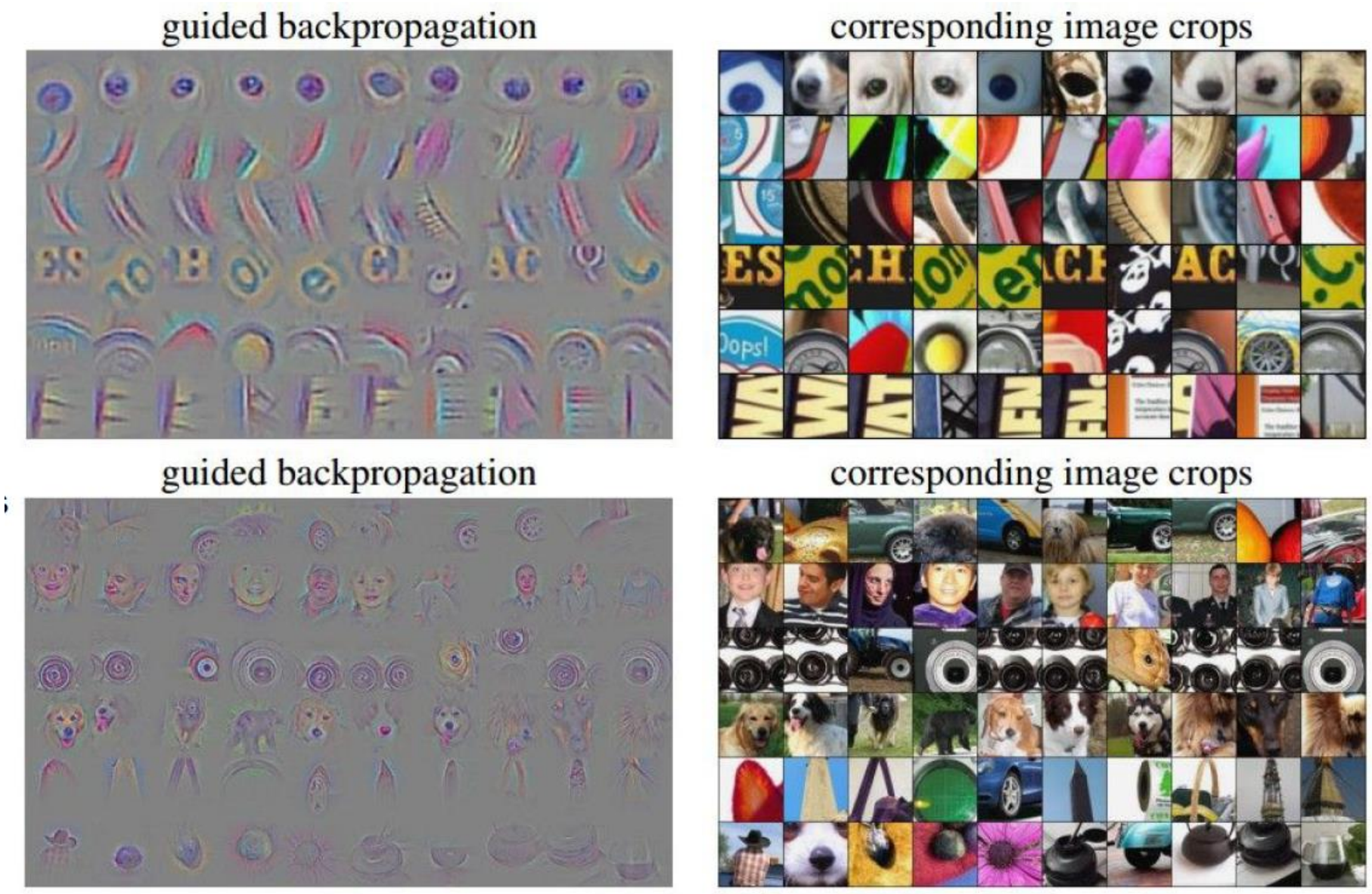
Guided Backpropagation



Backprop

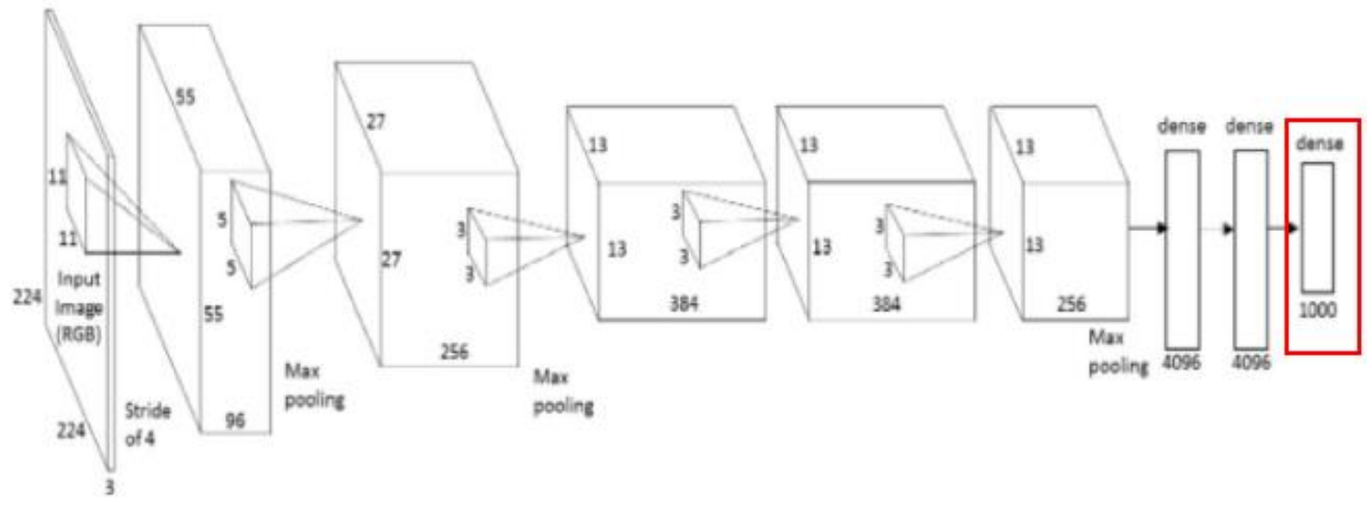
Guided Backprop

Guided Backpropagation

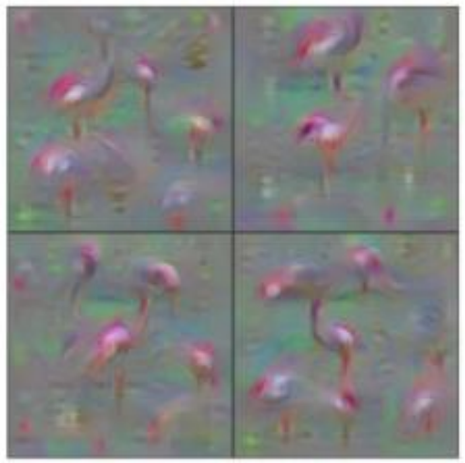


Springerberg et al, Striving for Simplicity: The All Convolutional Net (ICLR 2015 workshops)

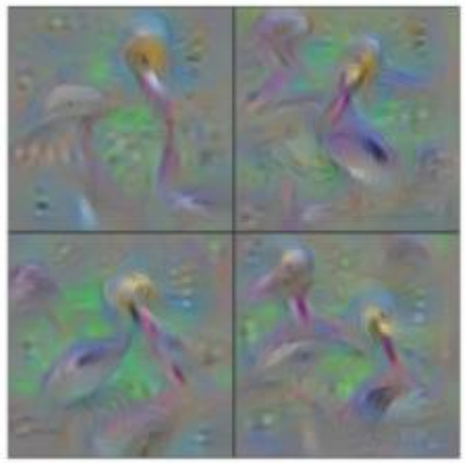
What About Doing Gradient Descent?



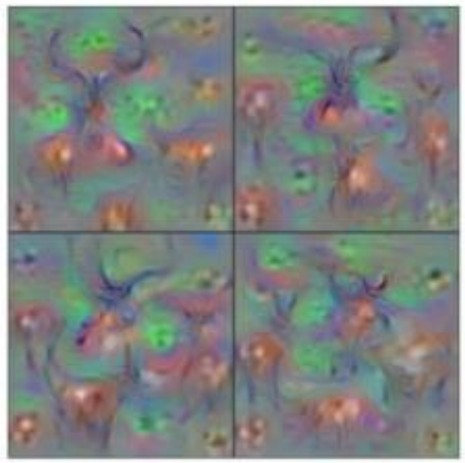
- What to maximize the i-th output of the softmax
- Can compute the gradient of the i-th output of the softmax with respect to the *input x* (the W's and b's are fixed to make classification as good as possible)
- Perform gradient descent on the *input*



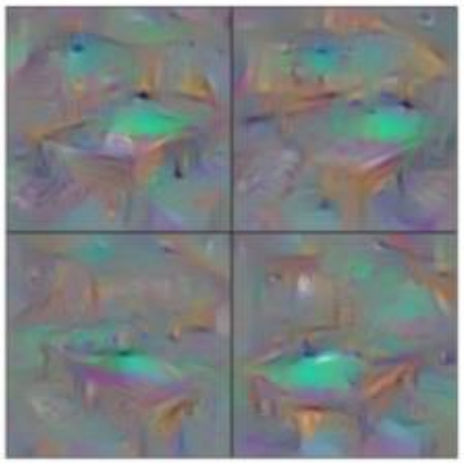
Flamingo



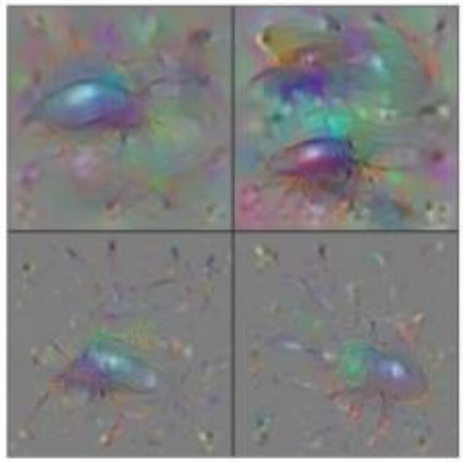
Pelican



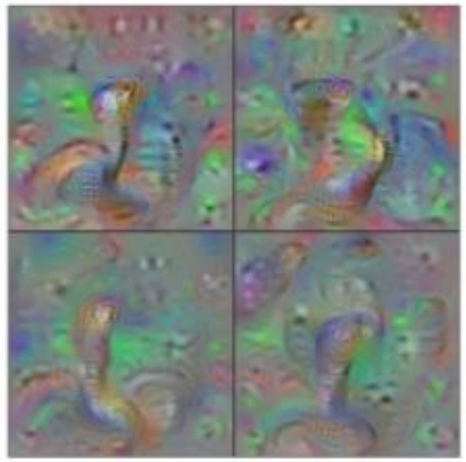
Hartebeest



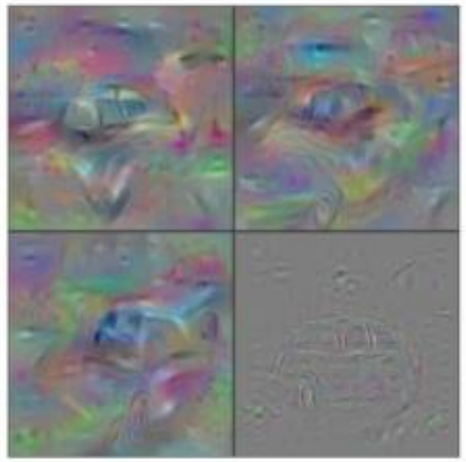
Billiard Table



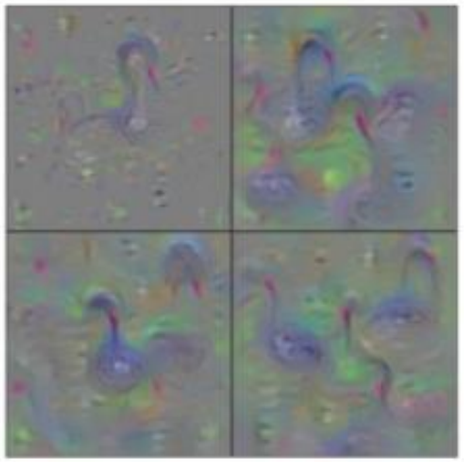
Ground Beetle



Indian Cobra



Station Wagon



Black Swan

Yosinski et al, Understanding Neural Networks Through Deep Visualization (ICML 2015)

Layer 6

Layer 5

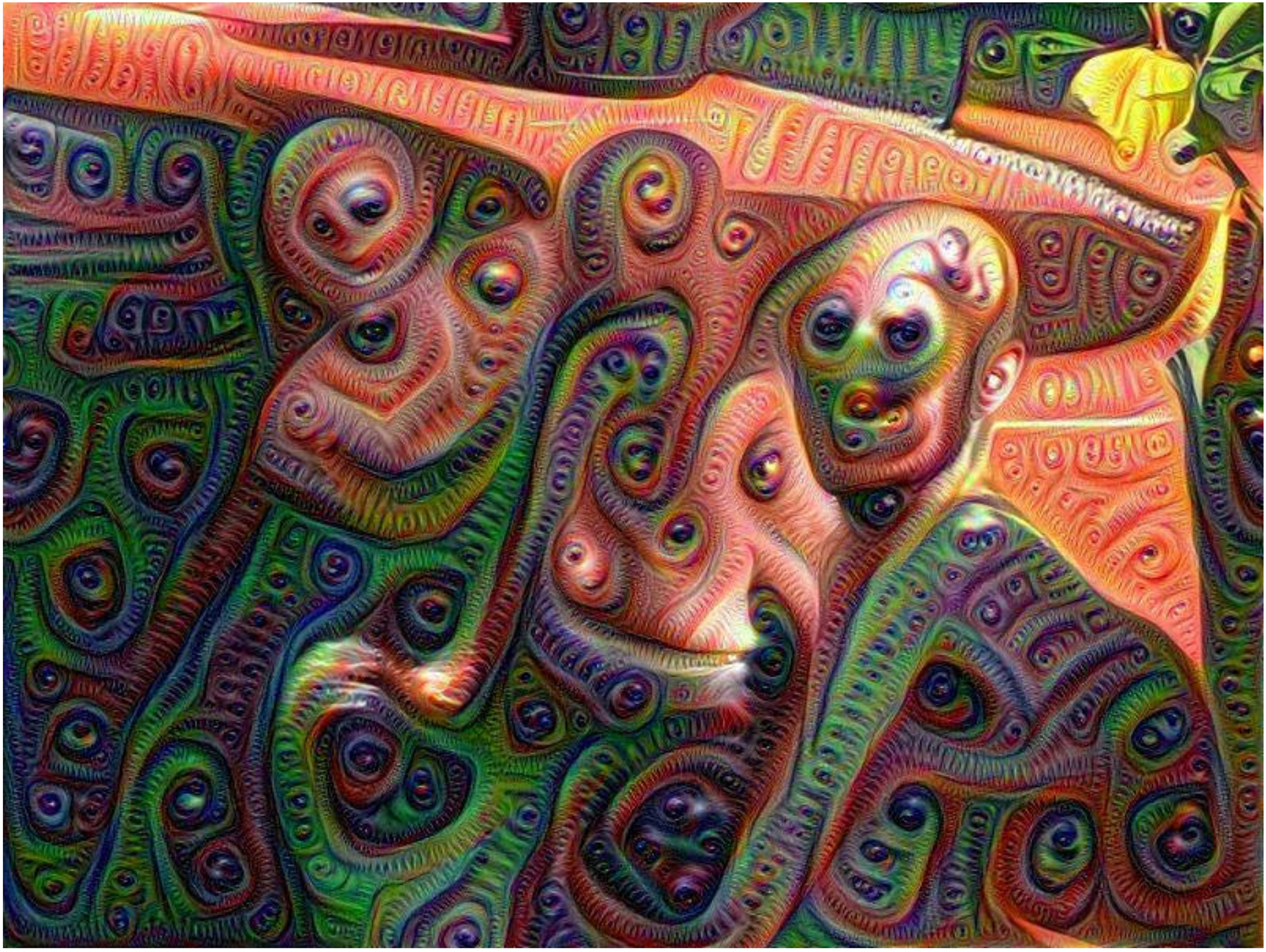
Layer 4

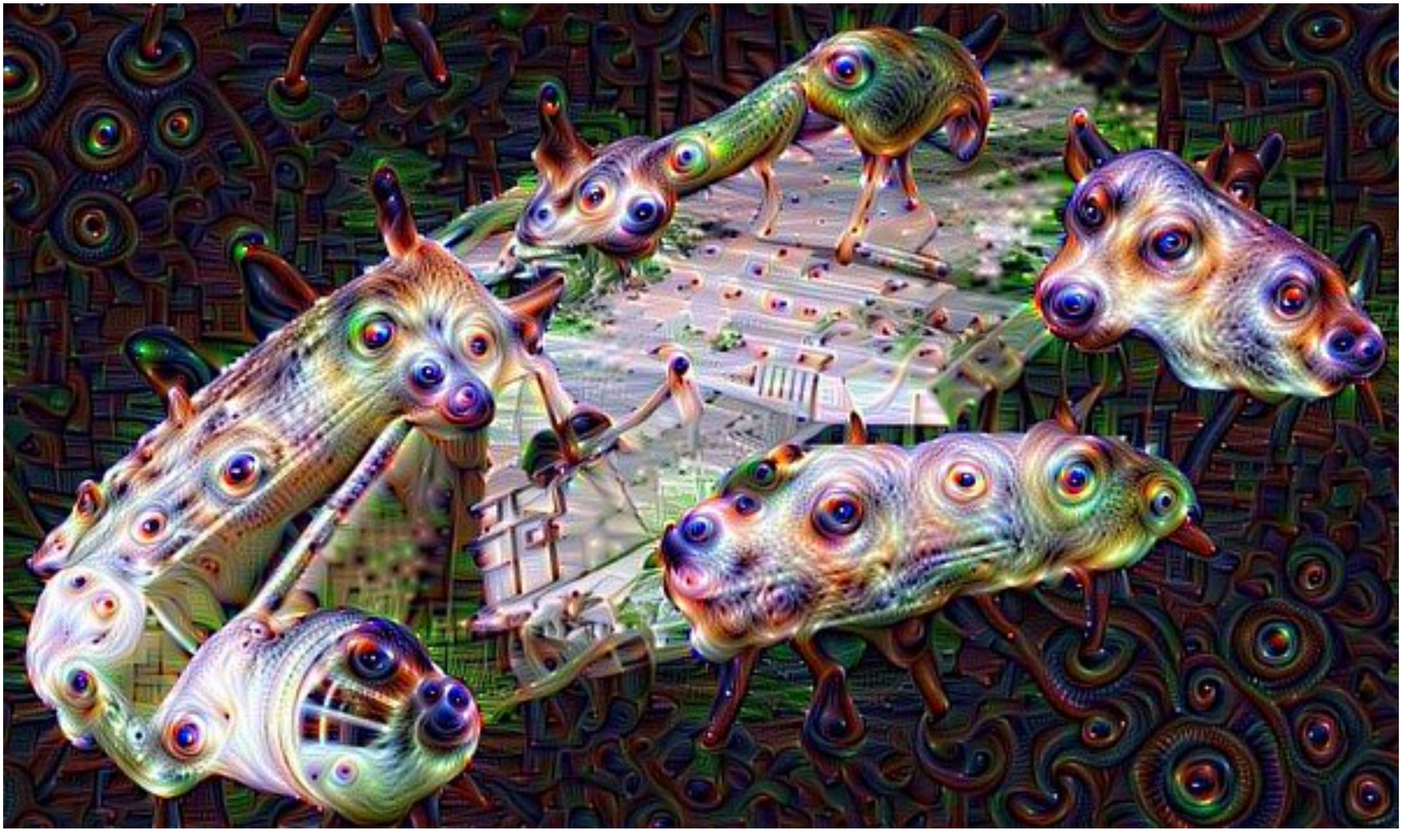
Layer 4

Layer 3

Layer 2

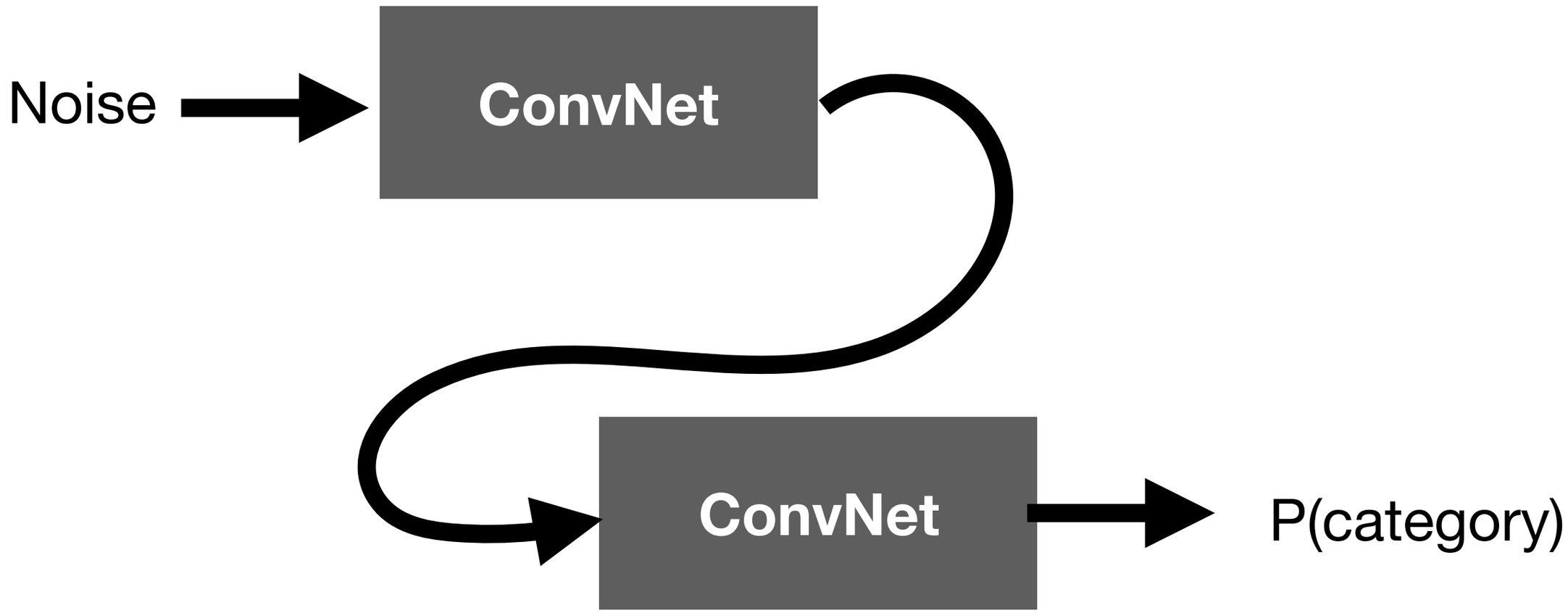
Layer 1






ConvNet
P(category)
Image

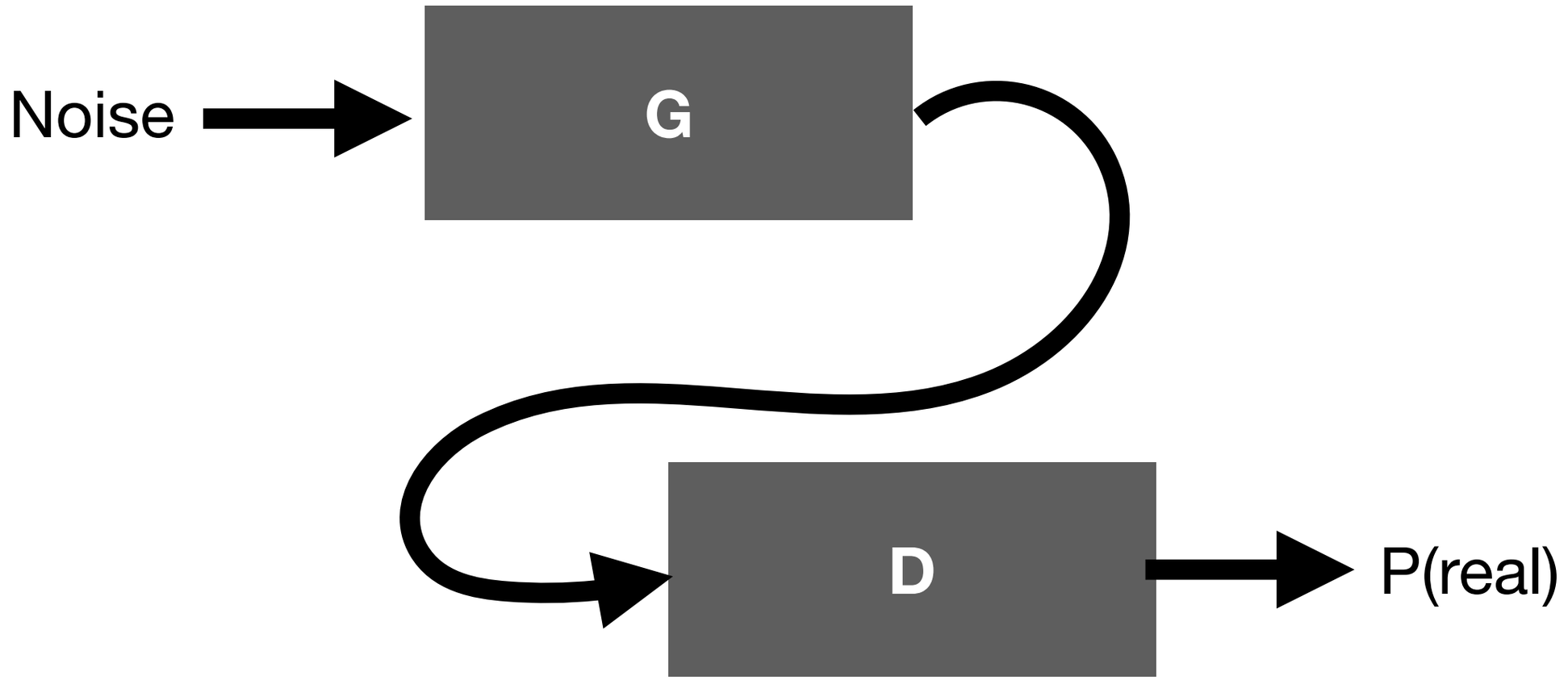
What if we learn to generate adversarial examples?



What if we learn to generate adversarial examples?

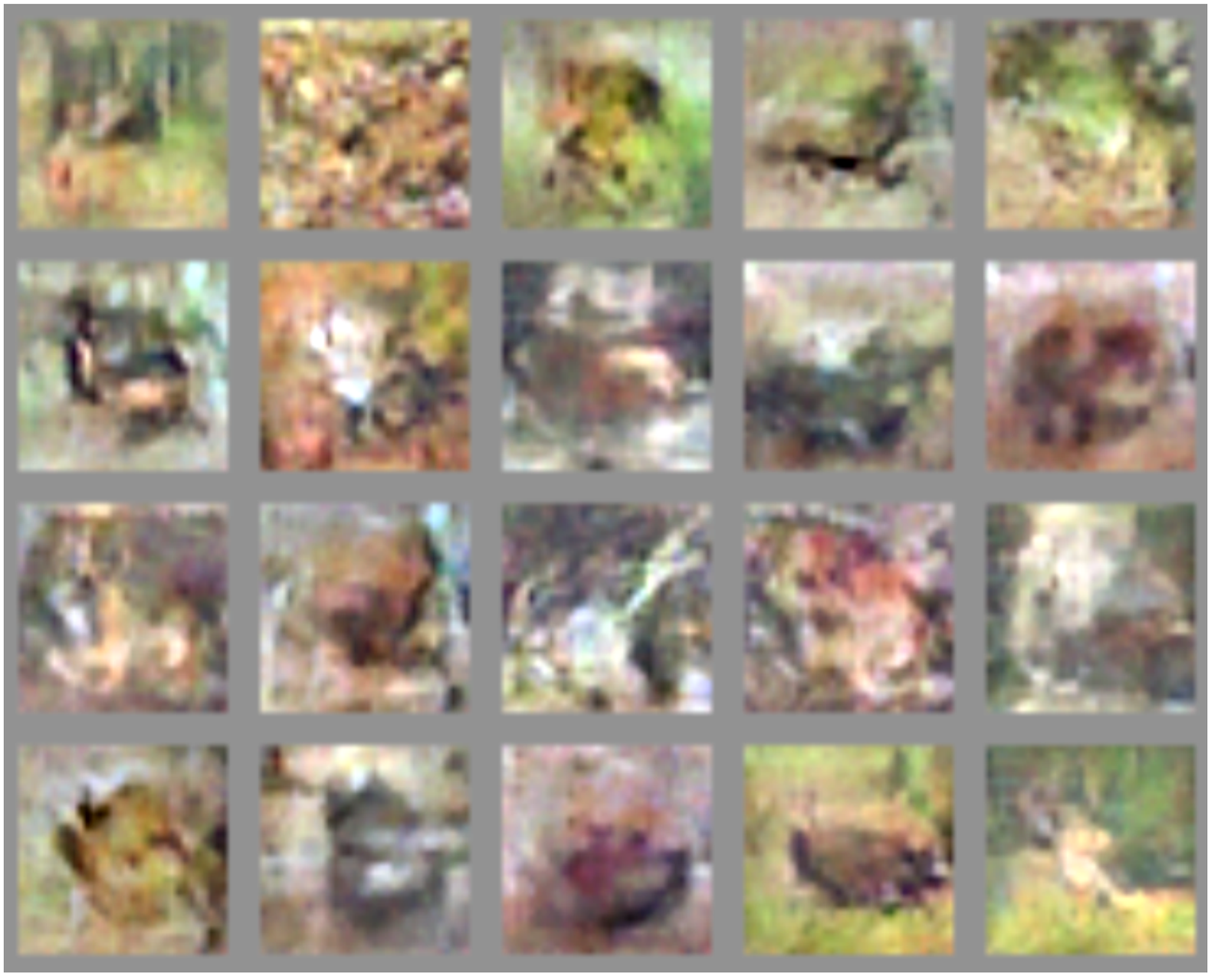
Generative Adversarial Networks

Goodfellow et al



$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Generated images



Trained with CIFAR-10

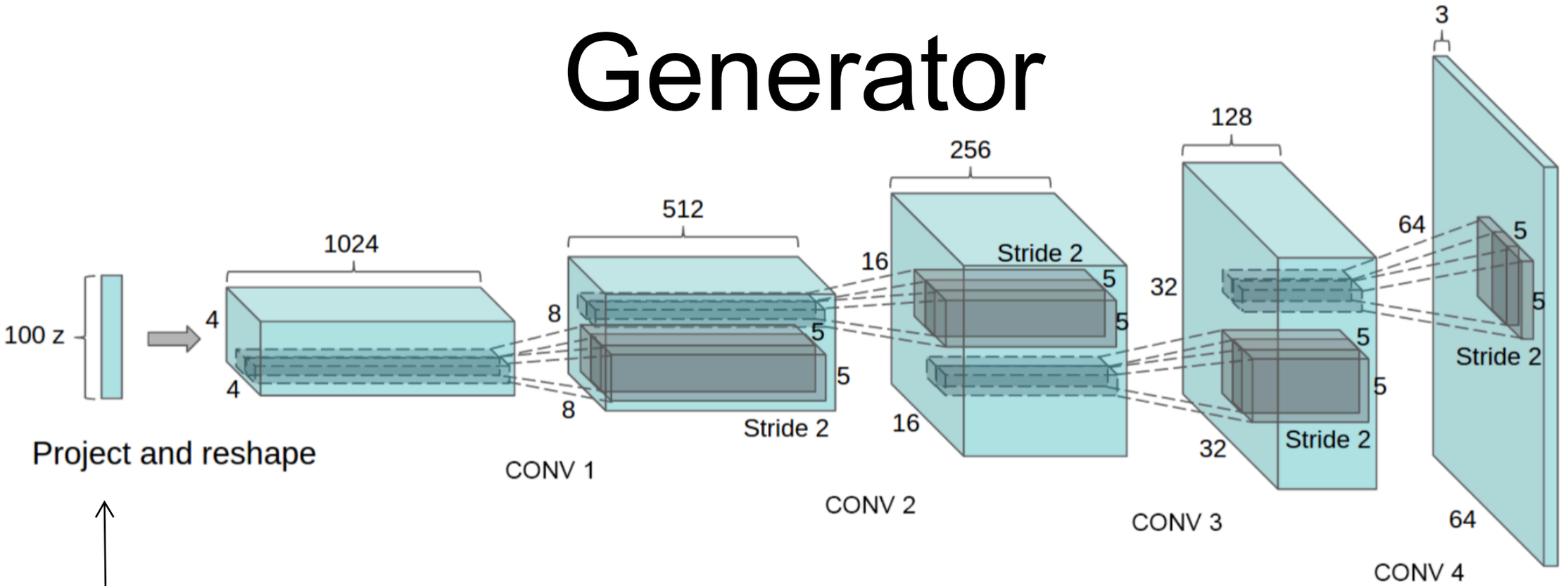
UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Alec Radford & Luke Metz
indico Research
Boston, MA
{alec,luke}@indico.io

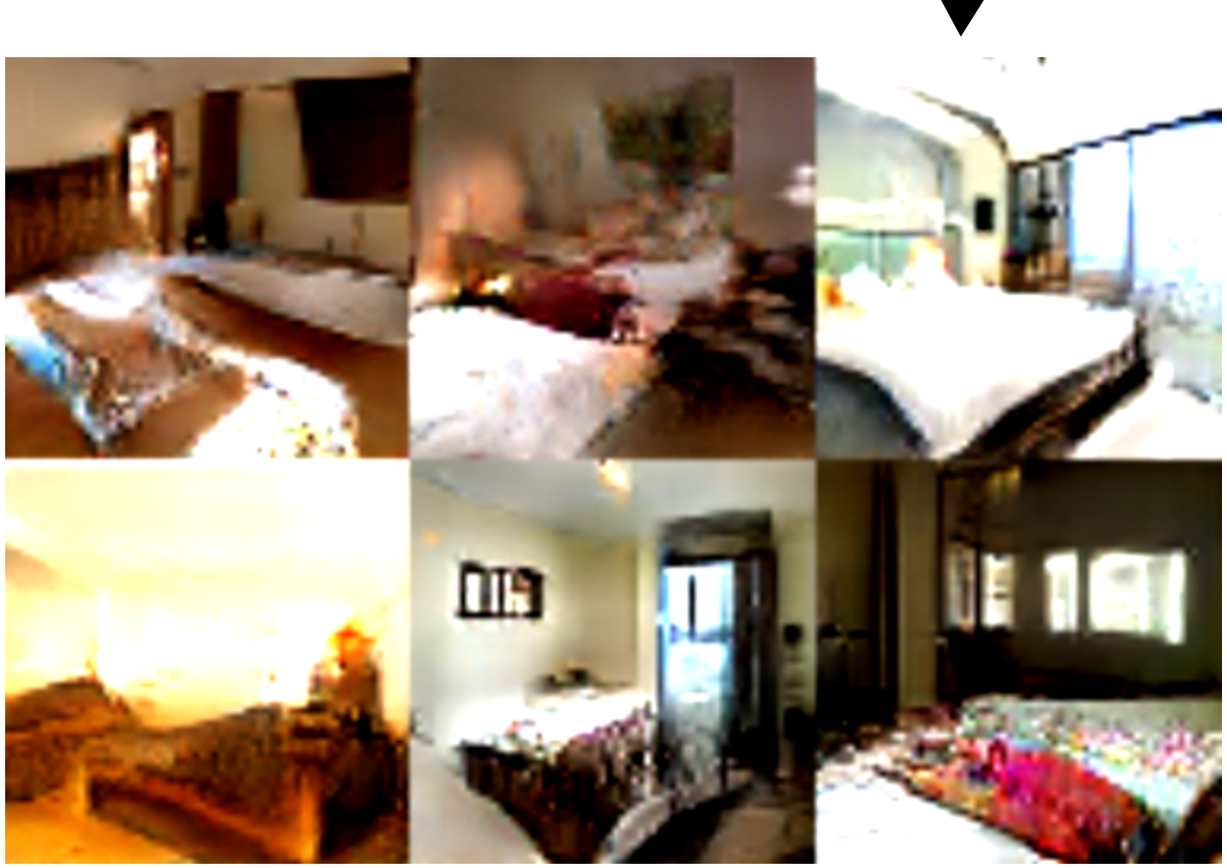
Soumith Chintala
Facebook AI Research
New York, NY
soumith@fb.com

Introduced a form of ConvNet more stable under adversarial training than previous attempts.

Generator

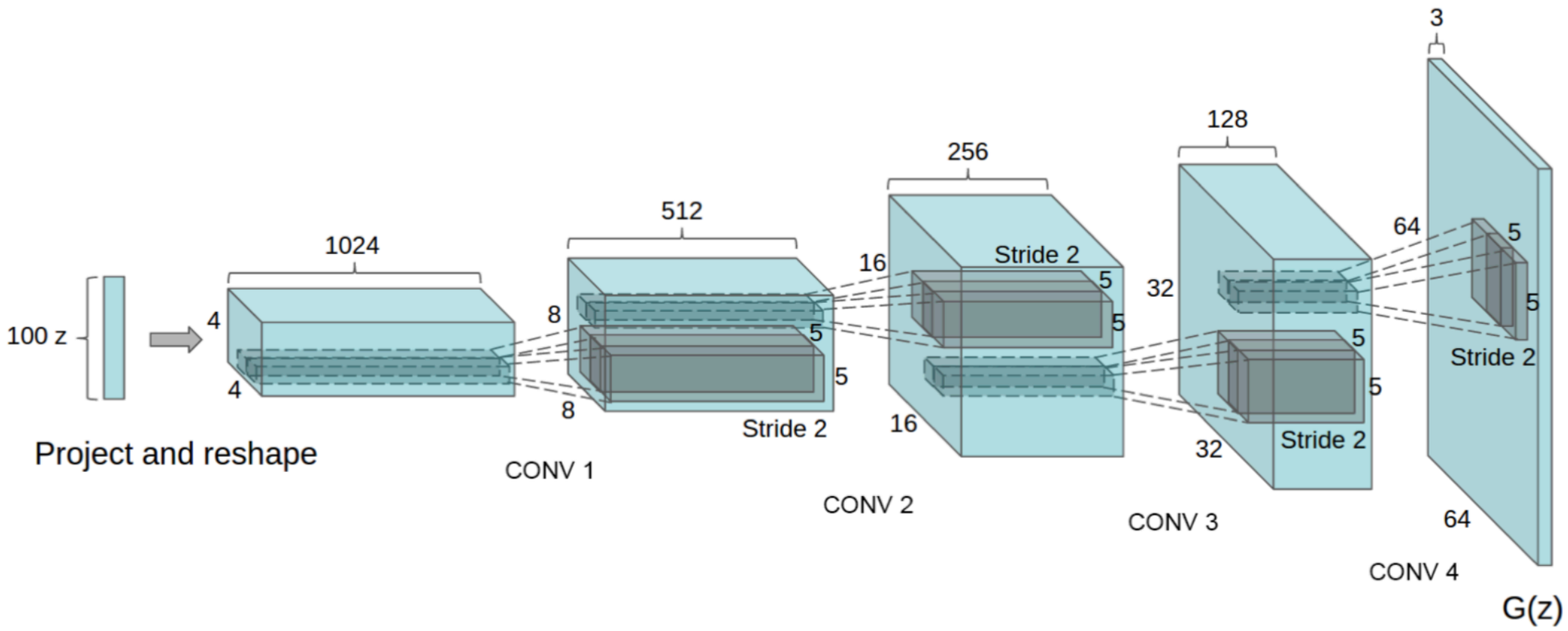


Random uniform vector (100 numbers)

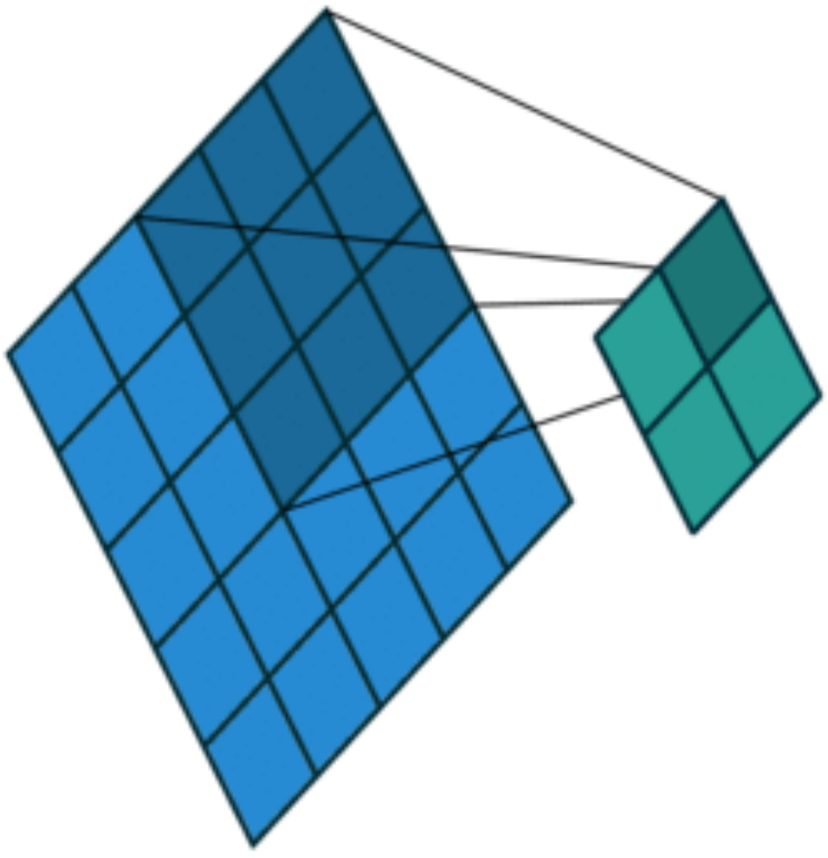


Synthesized images

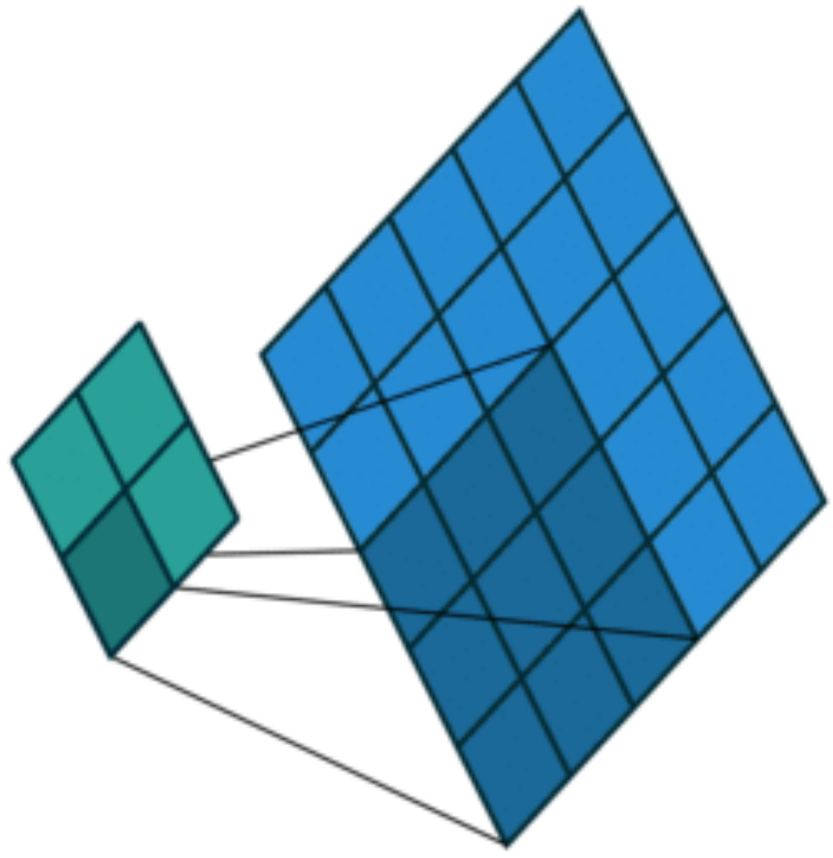
Transposed-convolution



Transposed-convolution



Convolution



Transposed-convolution

Generated Images

Brock et al. Large scale GAN training for high fidelity natural image synthesis

Image Interpolation

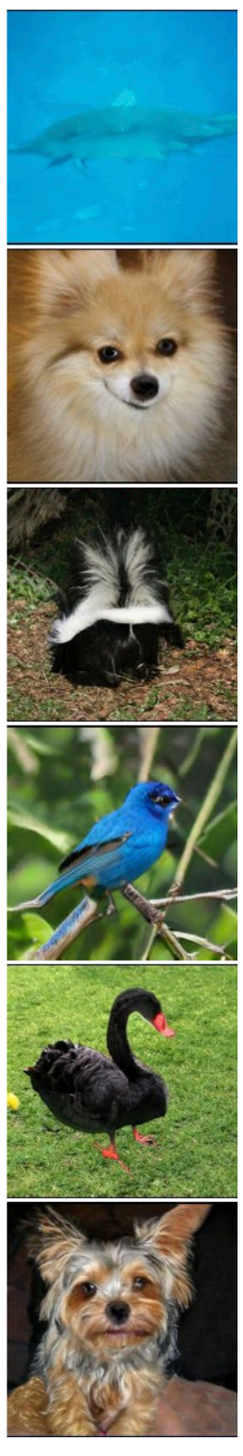
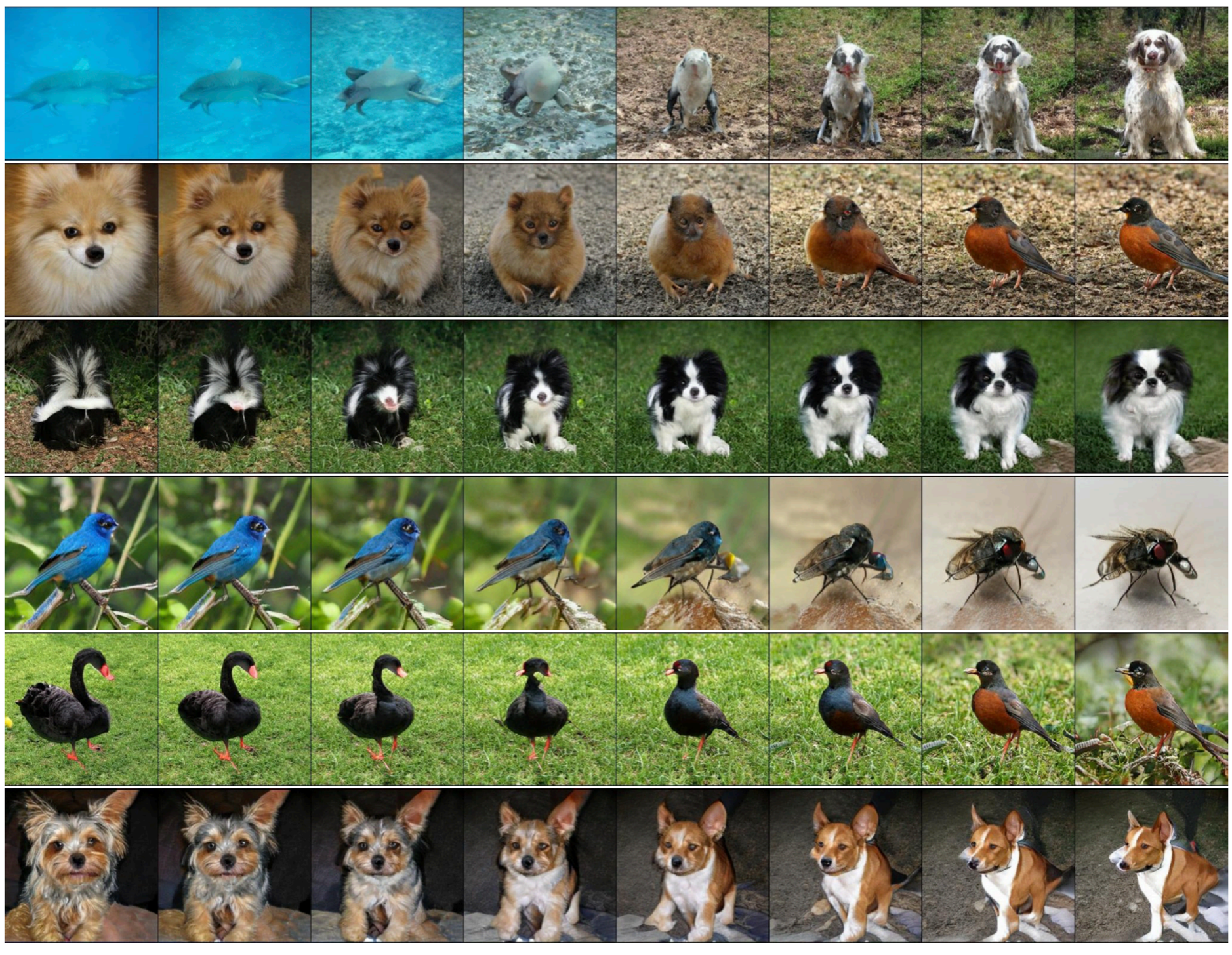


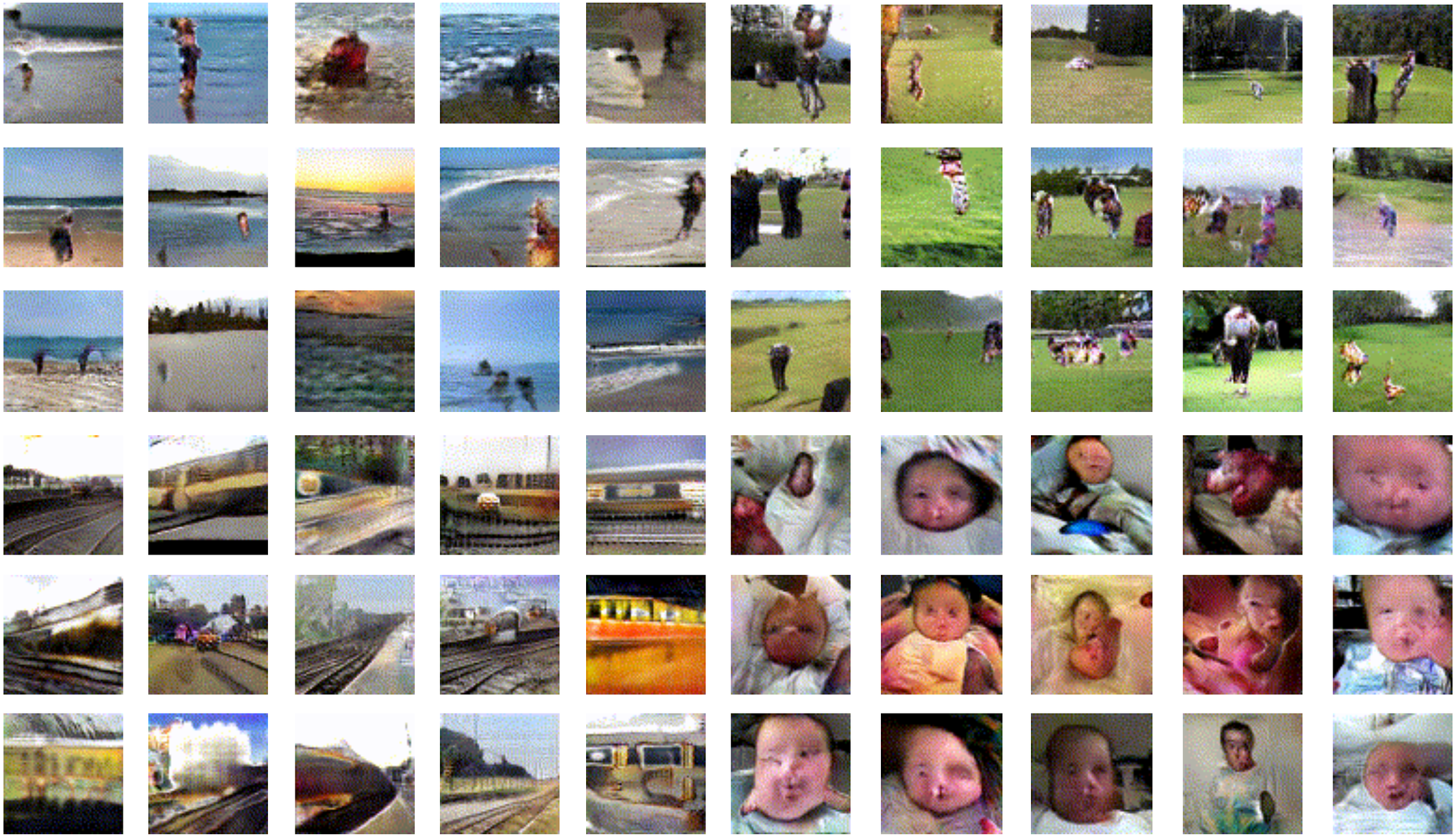
Image Interpolation



Nearest Neighbors

Nearest Neighbors

Generating Dynamics



Synthesizing the preferred inputs for neurons in neural networks via deep generator networks

Anh Nguyen
anguyen8@uwyo.edu

Alexey Dosovitskiy
dosovits@cs.uni-freiburg.de

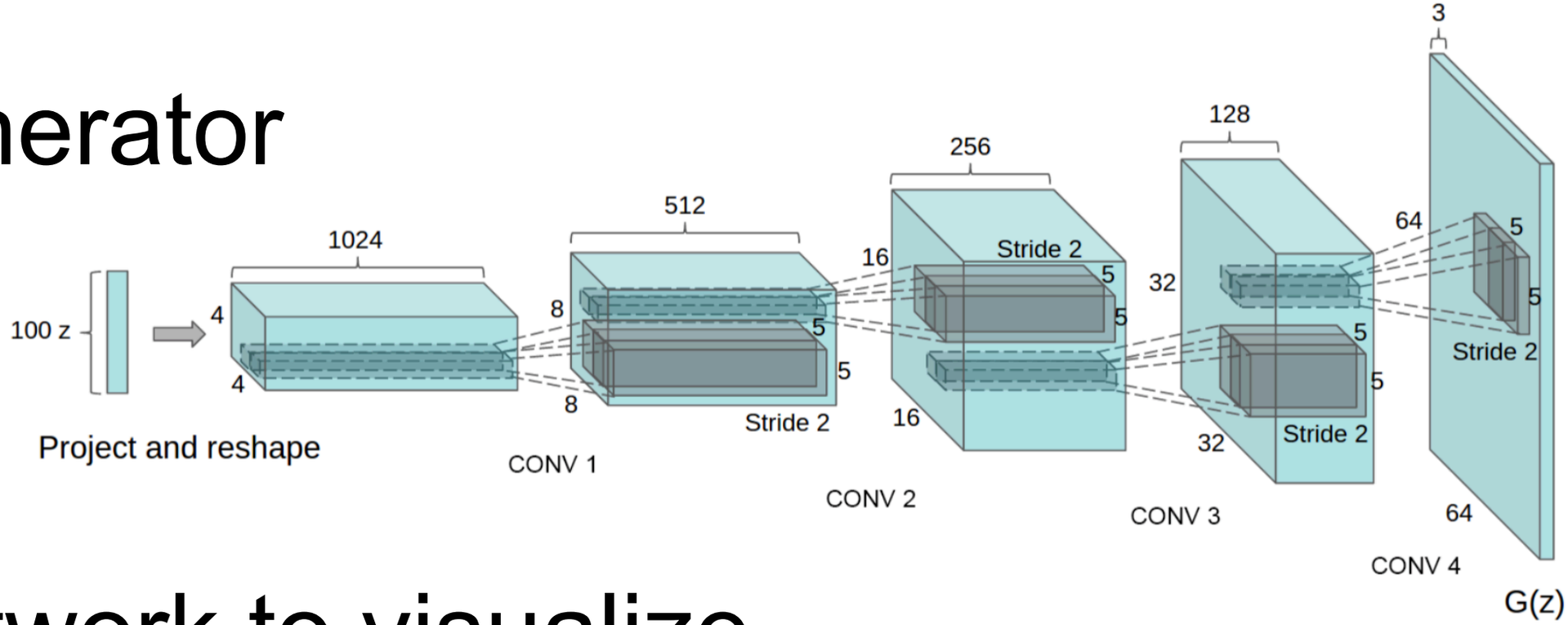
Jason Yosinski
jason@geometricintelligence.com

Thomas Brox
brox@cs.uni-freiburg.de

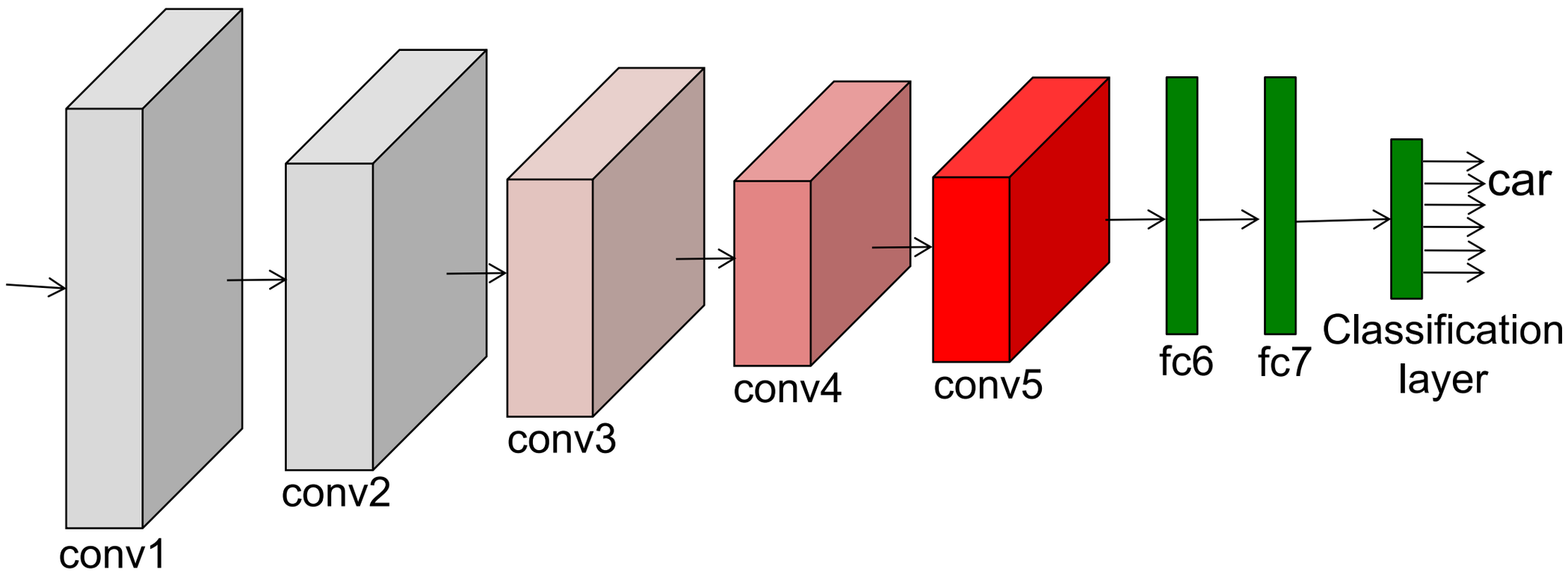
Jeff Clune
jeffclune@uwyo.edu

Two components

Generator

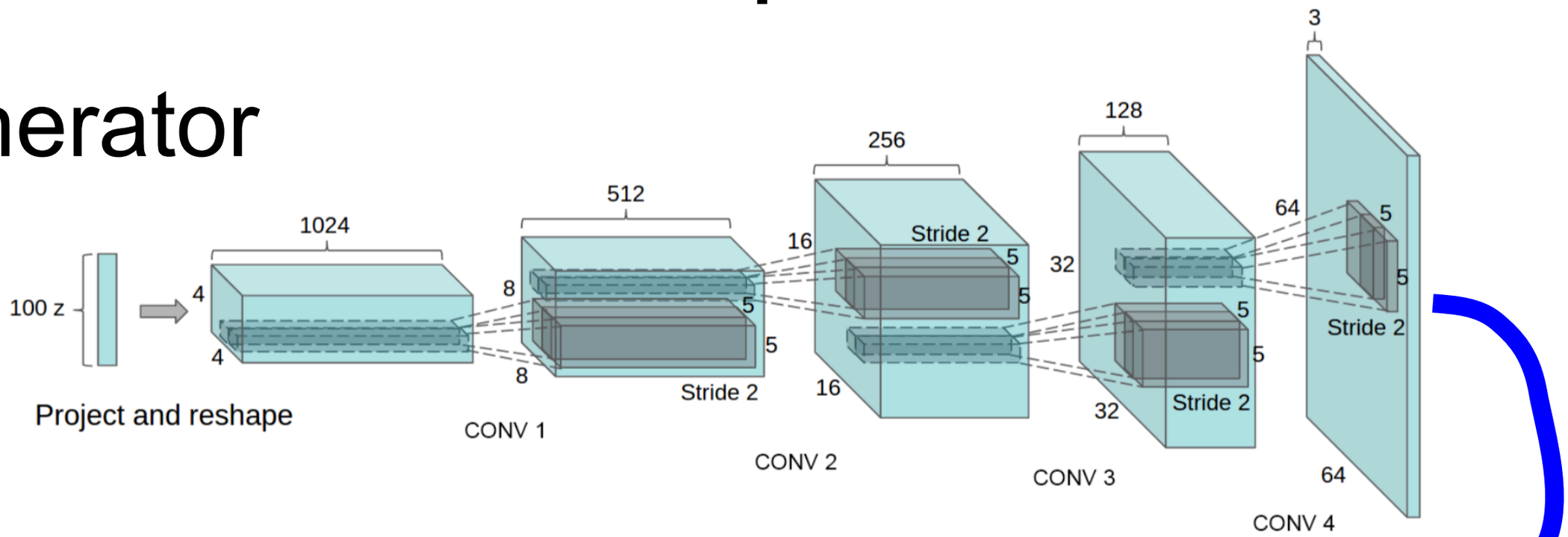


Network to visualize

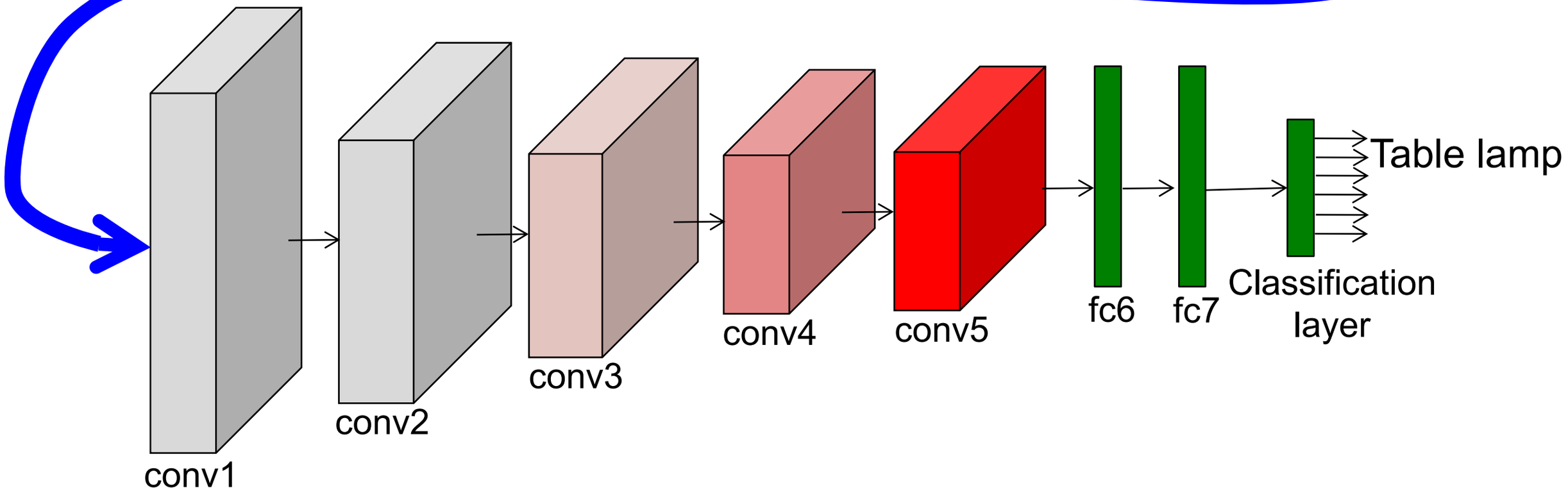


Two components

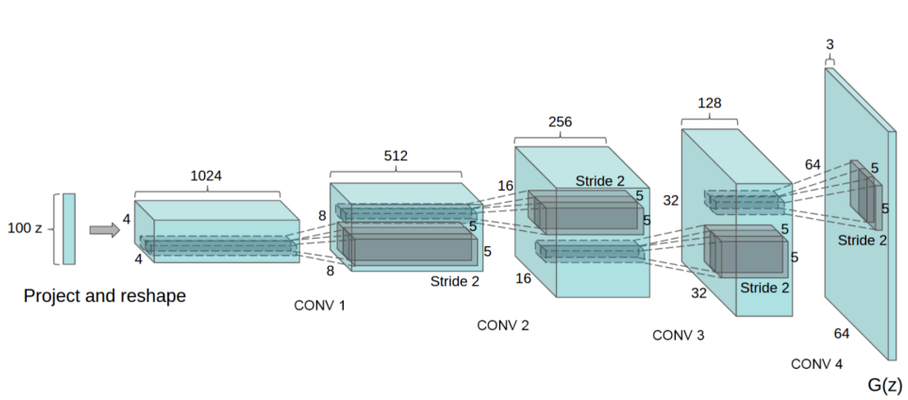
Generator



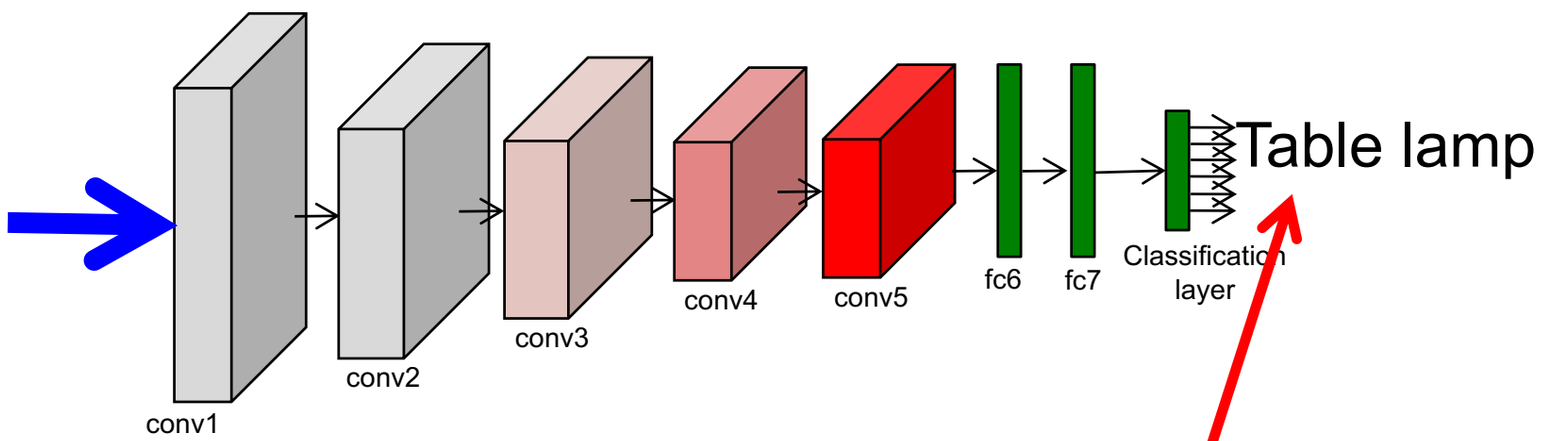
Network to visualize



Two components



Generator



Unit to visualize

Synthesizing Images Preferred by CNN

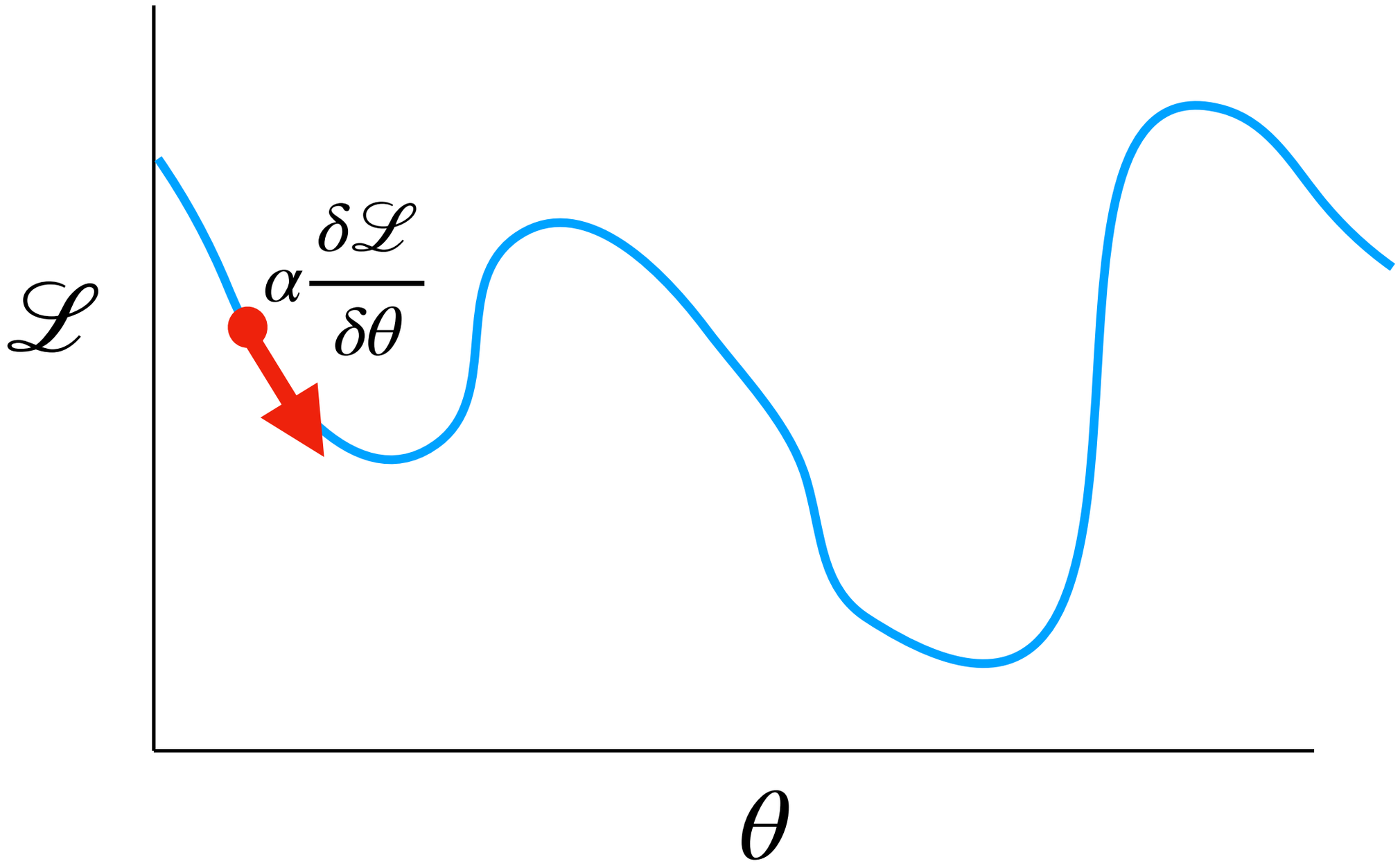
ImageNet-Alexnet-final units (class units)

Nguyen A, Dosovitskiy A, Yosinski J, Brox T, Clune J. (2016). "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks.". arXiv:1605.09304.

Where to start training?

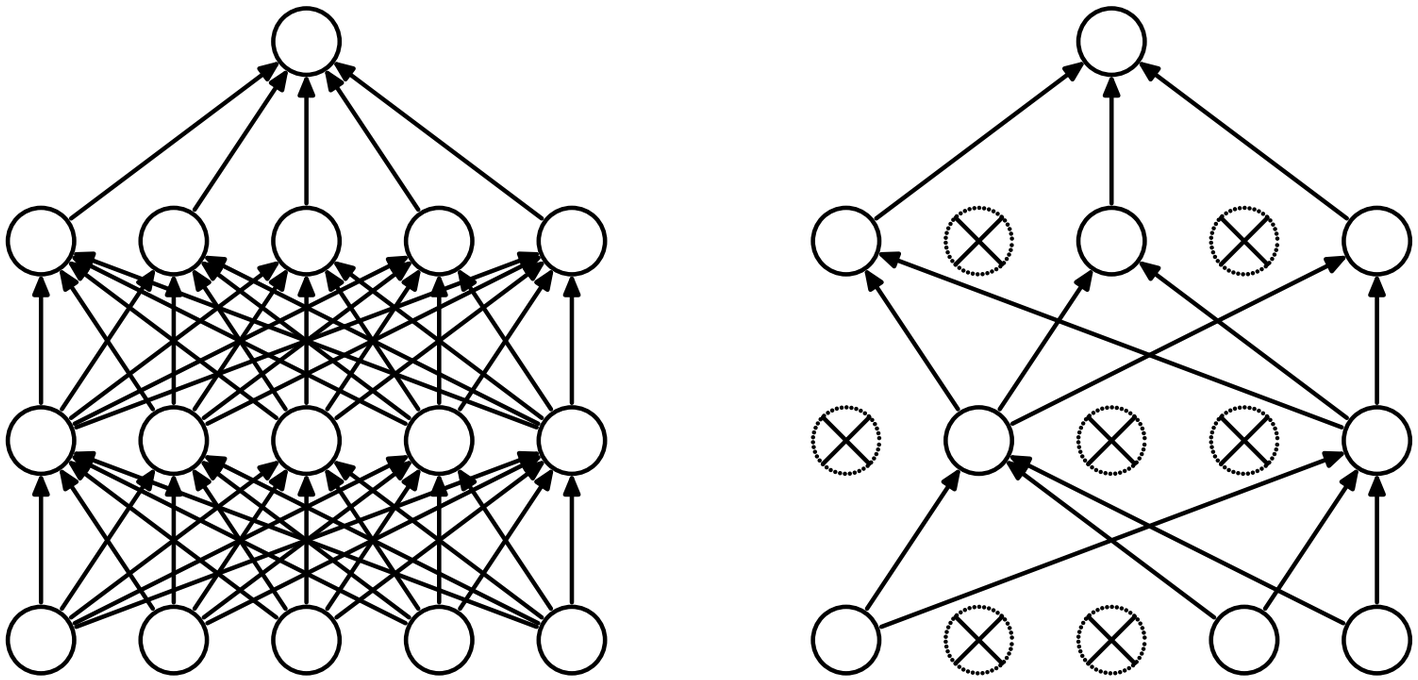
Gradient Descent

How to pick where to start?



Idea 0: Train many models

Drop-out regularization



(a) Standard Neural Net

(b) After applying dropout.

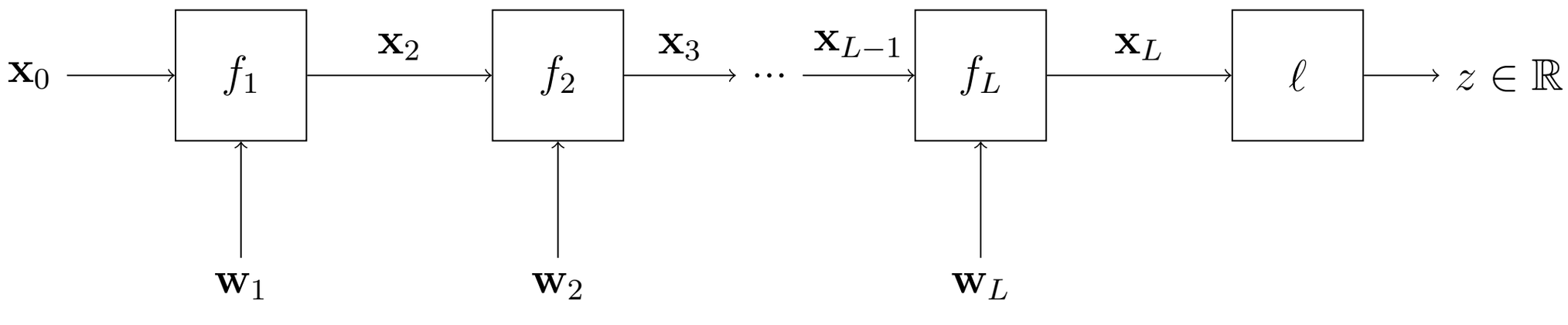
Intuition: we should really train a family of models with different architectures and average their predictions (c.f. model averaging from machine learning)

Practical implementation: learn a single “superset” architecture that randomly removes nodes (by randomly zero’ing out activations) during gradient updates

Slide credit: Deva Ramanan

Idea 1: Carefully pick starting point

Backprop



$$\frac{dz}{d\mathbf{w}_l} = \frac{d}{d\mathbf{w}_l}\left[\ell_{\mathbf{y}} \circ f_L(\cdot; \mathbf{w}_L) \circ ... \circ f_2(\cdot; \mathbf{w}_2) \circ f_1(\mathbf{x}_0; \mathbf{w}_1)\right]$$

$$\frac{dz}{d\mathbf{w}_l} = \frac{dz}{d(\operatorname{vec}\mathbf{x}_L)^\top}\frac{d\operatorname{vec}\mathbf{x}_L}{d(\operatorname{vec}\mathbf{x}_{L-1})^\top}\cdots\frac{d\operatorname{vec}\mathbf{x}_{l+1}}{d(\operatorname{vec}\mathbf{x}_l)^\top}\frac{d\operatorname{vec}\mathbf{x}_l}{d\mathbf{w}_l^\top}$$

Slide credit: Deva Ramanan

Exploding and vanishing gradient

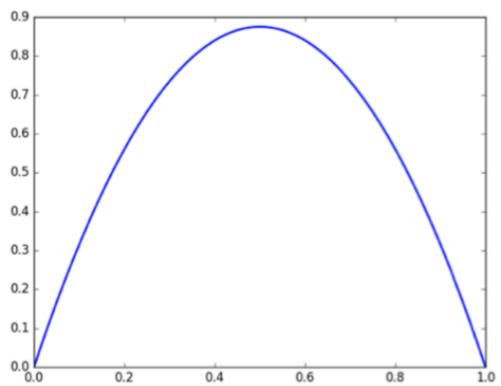
- How does the determinant of the gradients effect the final gradient?

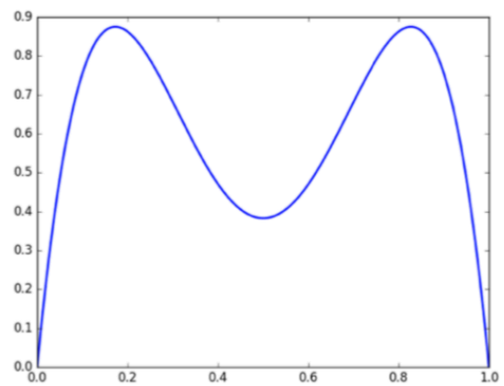
  - What if the determinant is less than one?

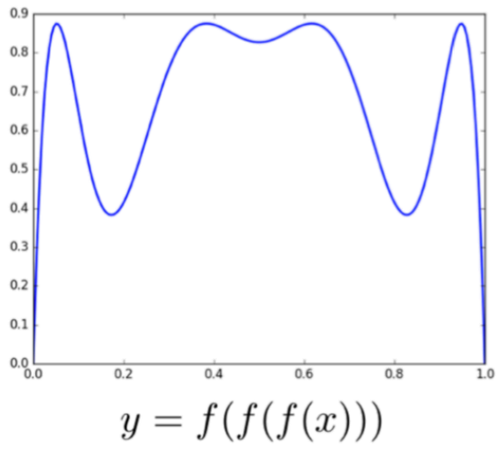
  - What if the determinant is greater than one?

$$\frac{dz}{d\mathbf{w}_l} = \frac{dz}{d(\operatorname{vec}\mathbf{x}_L)^\top}\frac{d\operatorname{vec}\mathbf{x}_L}{d(\operatorname{vec}\mathbf{x}_{L-1})^\top}\cdots\frac{d\operatorname{vec}\mathbf{x}_{l+1}}{d(\operatorname{vec}\mathbf{x}_l)^\top}\frac{d\operatorname{vec}\mathbf{x}_l}{d\mathbf{w}_l^\top}$$

Exploding and vanishing gradient







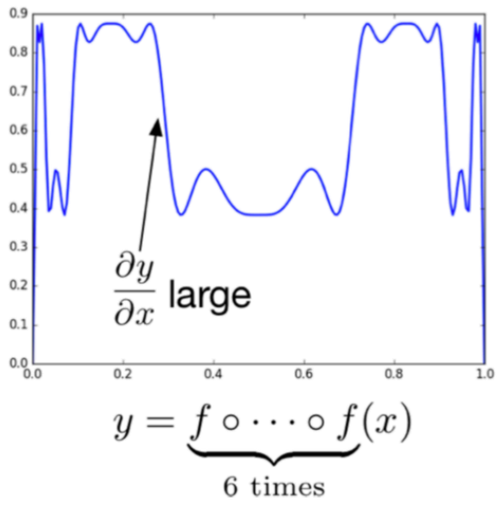


Figure 2: Iterations of the function $f(x) = 3.5\,x\,(1-x)$.

Source: Roger Grosse

Initialization

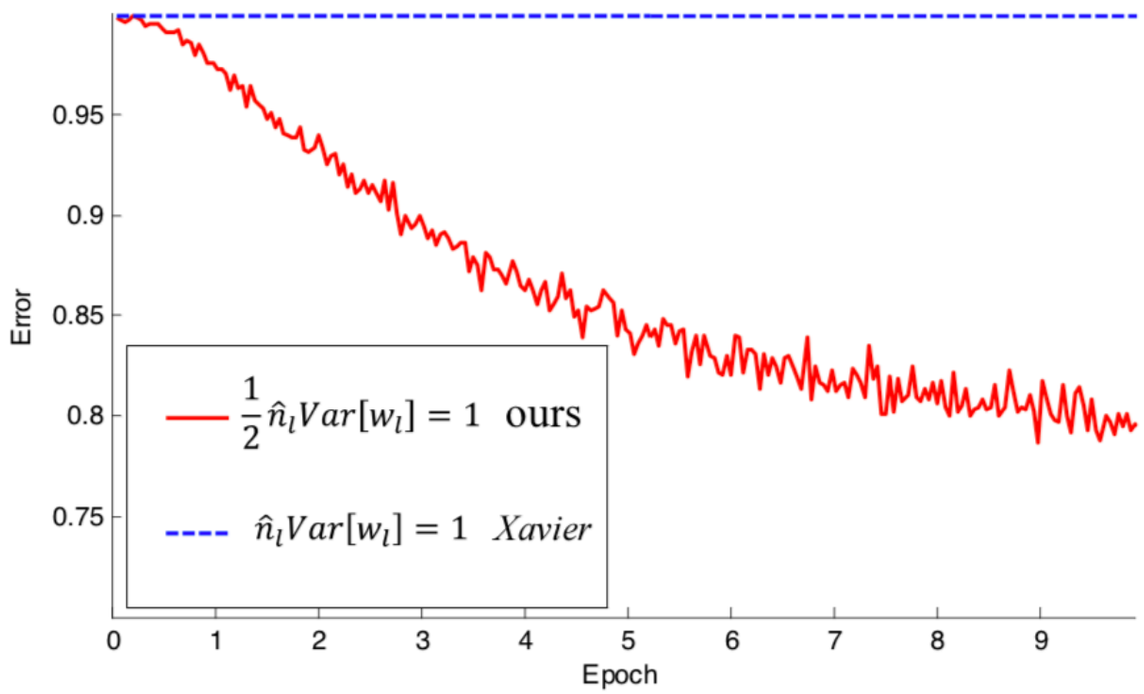


Figure 3. The convergence of a **30-layer** small model (see the main text). We use ReLU as the activation for both cases. Our initialization (red) is able to make it converge. But "*Xavier*" (blue) [7] completely stalls - we also verify that its gradients are all diminishing. It does not converge even given more epochs.

He et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

Initialization

- Key idea: initialization weights so that the variance of activations is one at each layer
- You can derive what this should be for different layers and nonlinearities
- For ReLU: $w_i \sim \mathcal{N}\left(0, \frac{2}{k}\right)$ $b_i = 0$

He et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

Idea 2: How to maintain this throughout training?

Batch Normalization

$$x \rightarrow \hat{x} = \frac{x - \mu}{\sigma} \rightarrow y = \gamma\hat{x} + \beta$$

- μ: mean of x in mini-batch
- σ: std of x in mini-batch
- γ: scale
- β: shift

- μ, σ: functions of x, analogous to responses
- γ, β: parameters to be learned, analogous to weights

Ioffe & Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ICML 2015

Batch Normalization

$$x \rightarrow \hat{x} = \frac{x - \mu}{\sigma} \rightarrow y = \gamma\hat{x} + \beta$$

2 modes of BN:

- Train mode:
  - μ, σ are functions of a batch of x
- Test mode:
  - μ, σ are pre-computed on training set

Caution: make sure your BN usage is correct!
(this causes many of my bugs in my research experience!)

Ioffe & Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ICML 2015

Batch Normalization

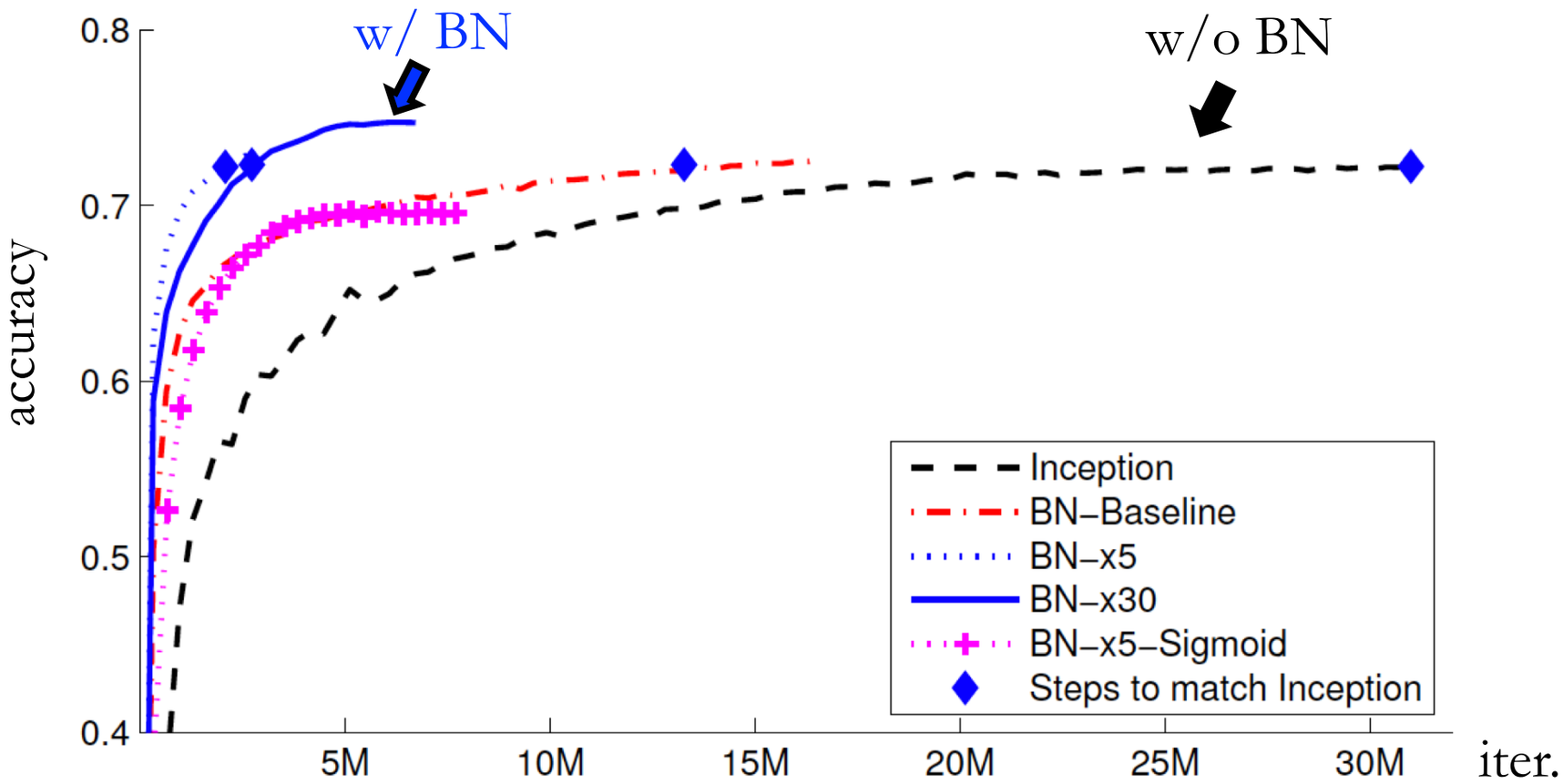


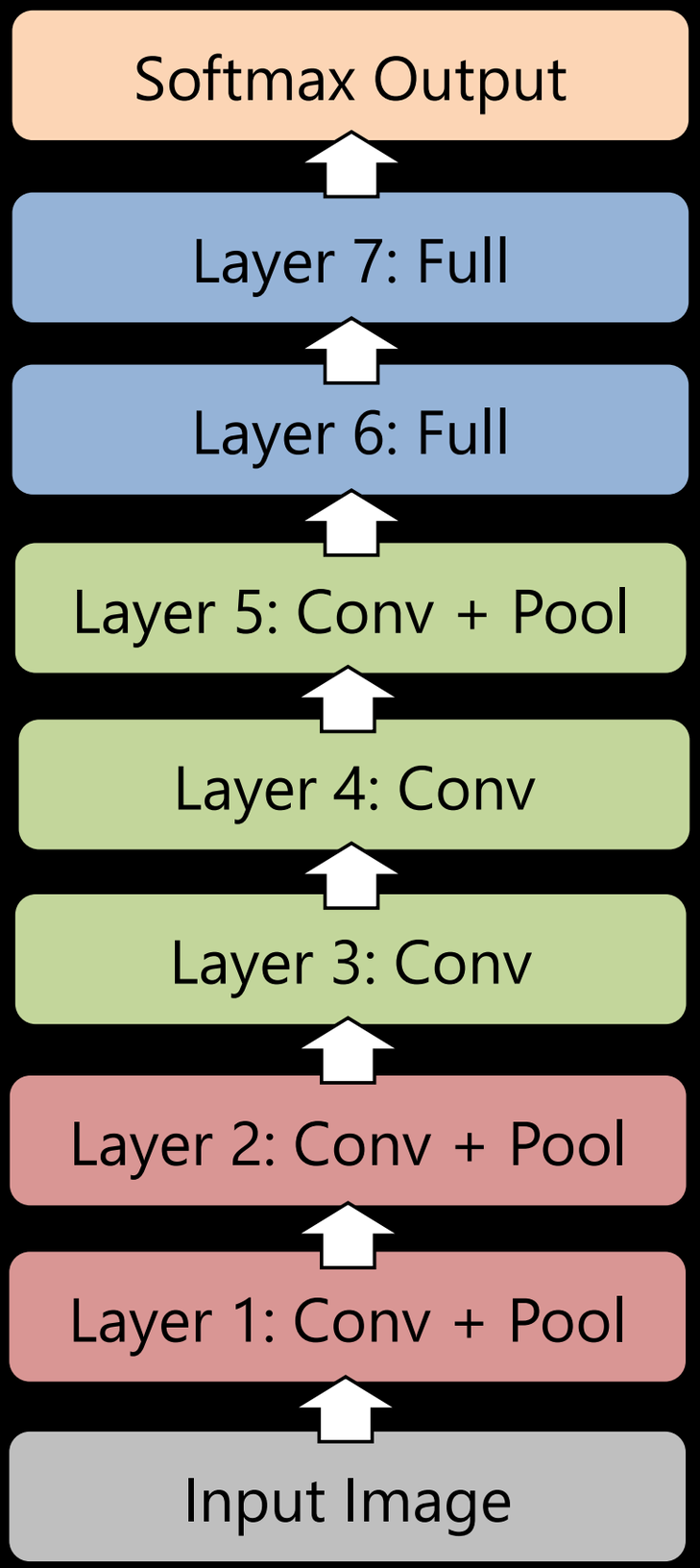
Figure credit: Ioffe & Szegedy

Ioffe & Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ICML 2015

Back to breaking things…

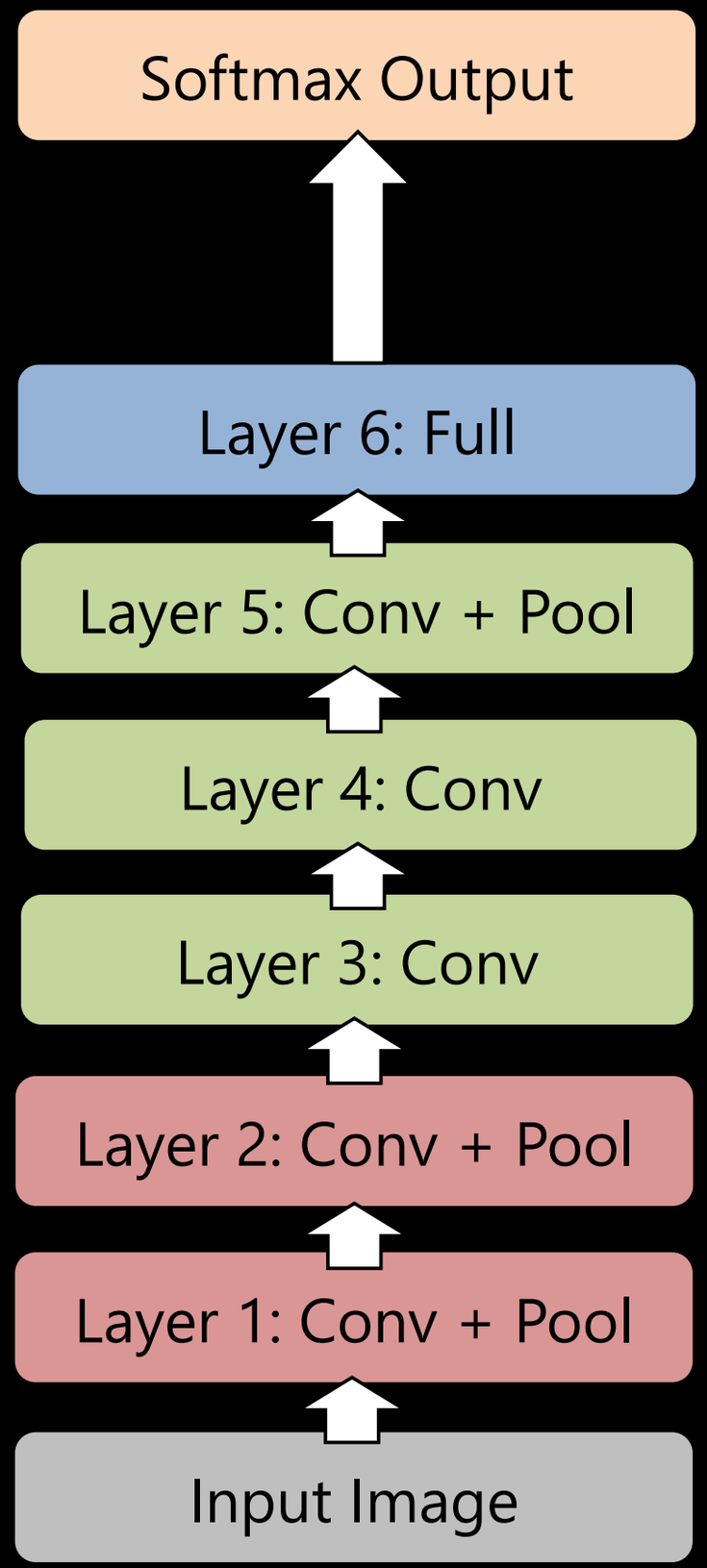
Architecture of Krizhevsky et al.

- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR'09]
- 18.2% top-5 error
- Our reimplementation:
  18.1% top-5 error



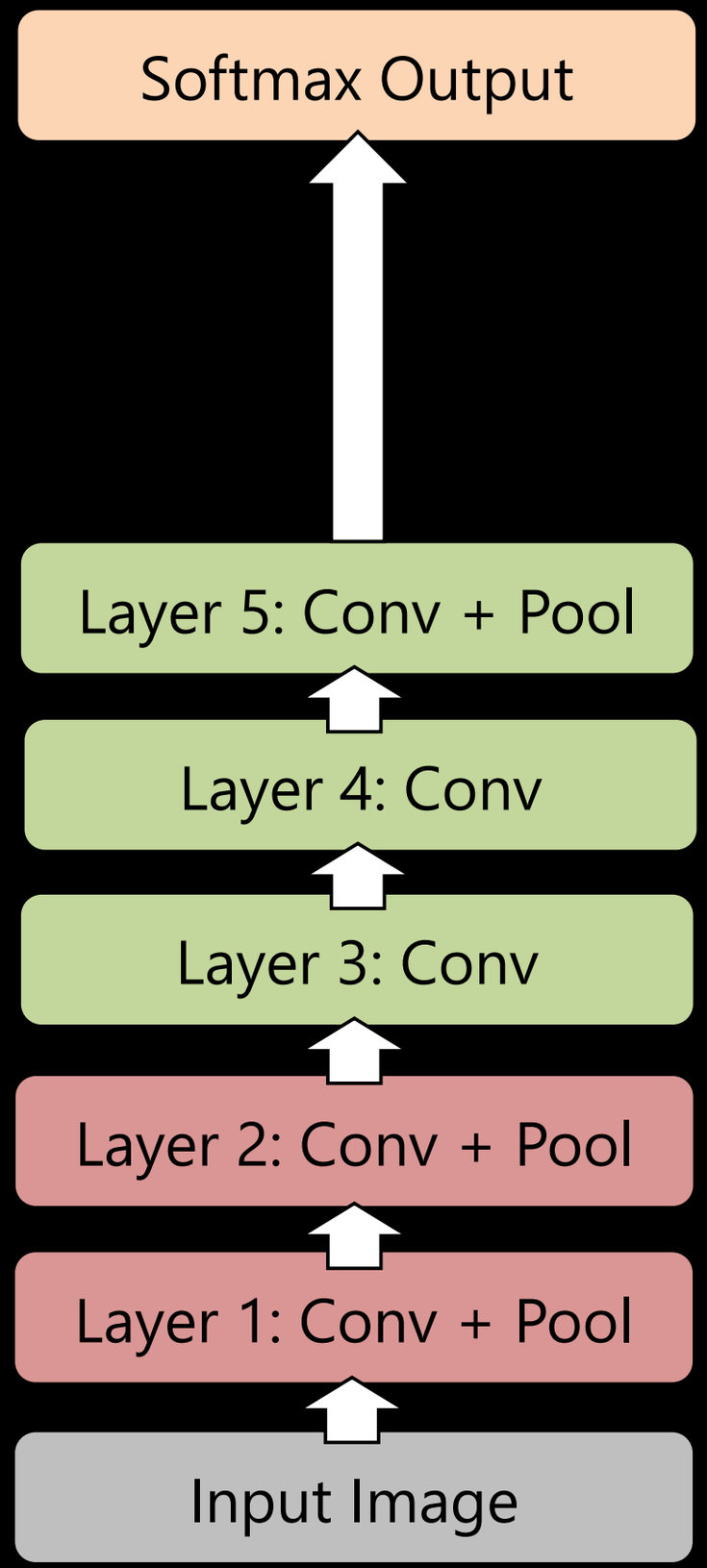
Architecture of Krizhevsky et al.

- Remove top fully connected layer
  - Layer 7
- Drop 16 million parameters
- Only 1.1% drop in performance!



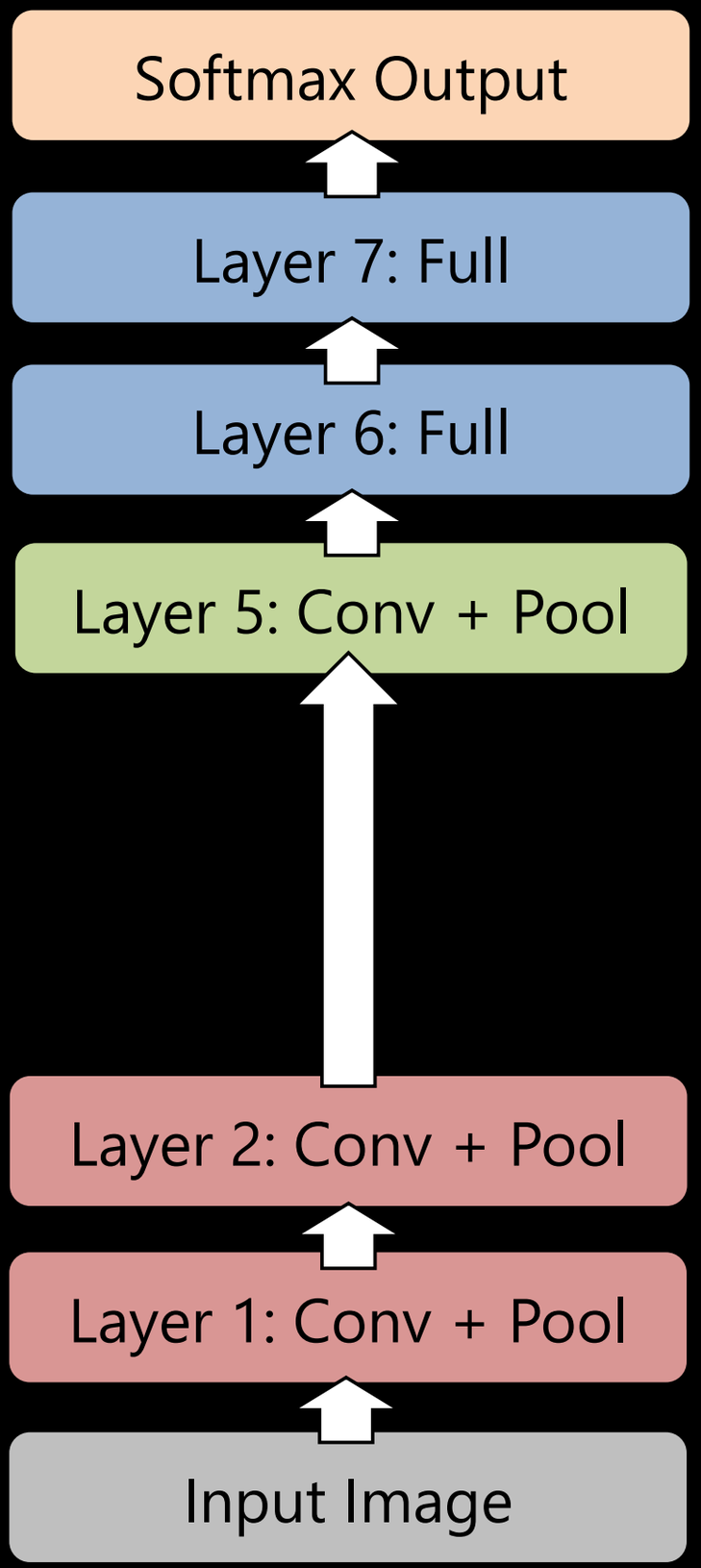
Architecture of Krizhevsky et al.

- Remove both fully connected layers
  - Layer 6 & 7
- Drop ~50 million parameters
- 5.7% drop in performance



Architecture of Krizhevsky et al.

- Now try removing upper feature extractor layers:
  - Layers 3 & 4
- Drop ~1 million parameters
- 3.0% drop in performance



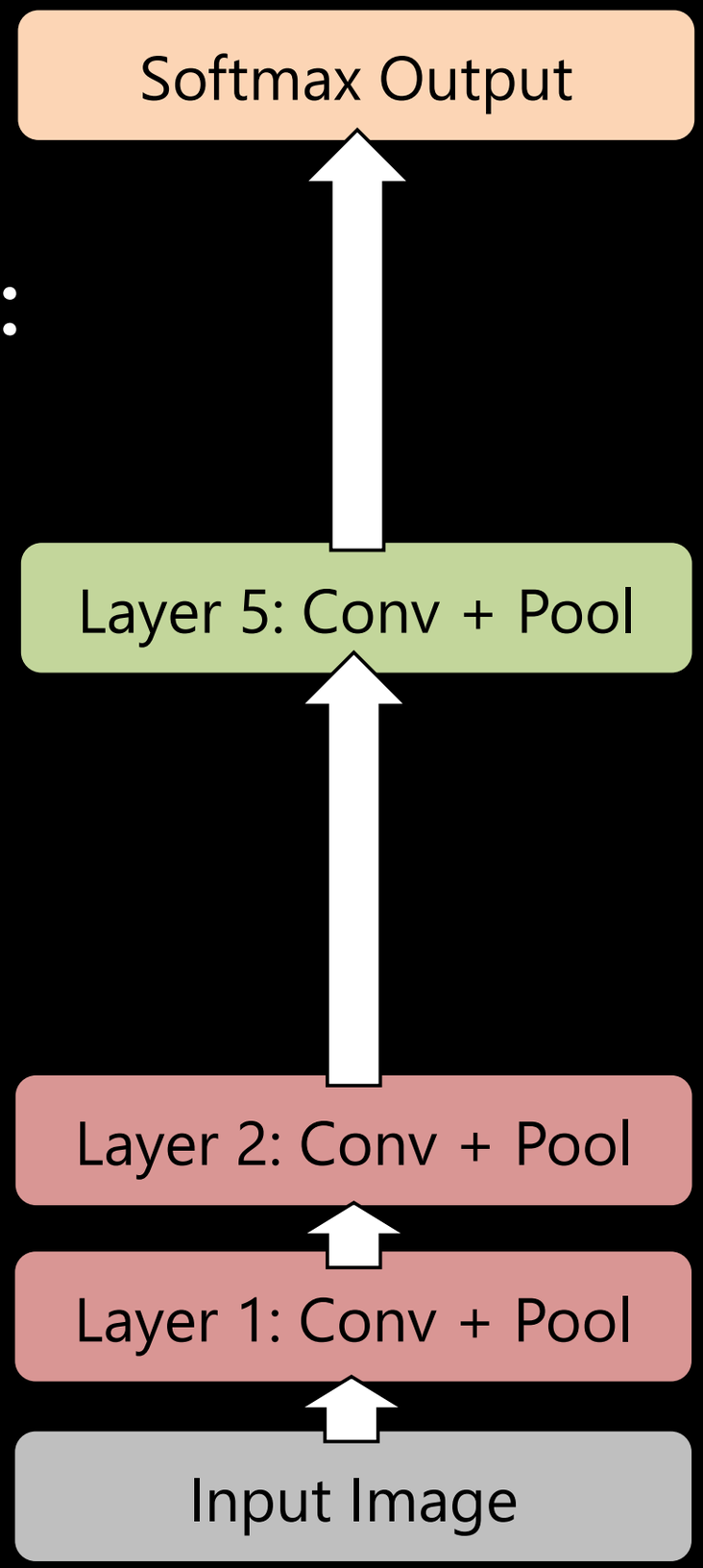
Architecture of Krizhevsky et al.

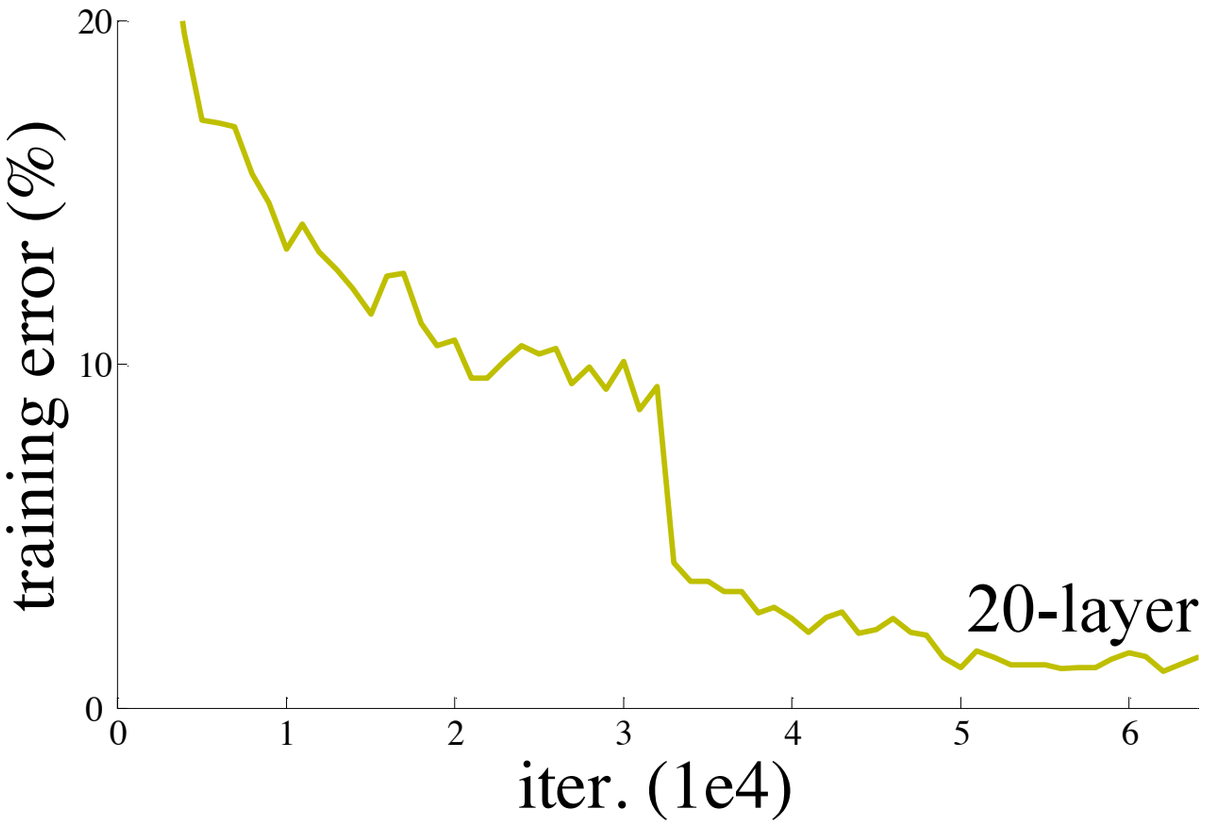
- Now try removing upper feature extractor layers & fully connected:
  - Layers 3, 4, 6 ,7

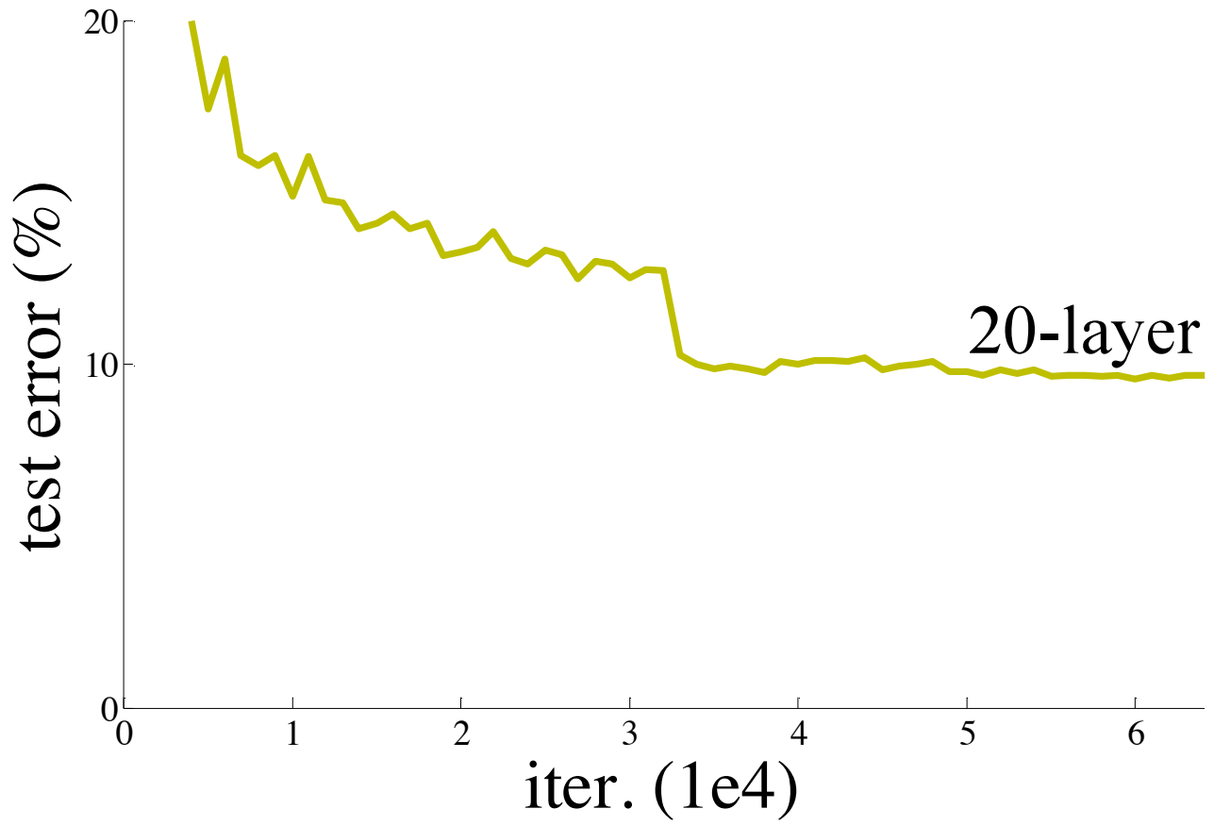
- Now only 4 layers

- 33.5% drop in performance

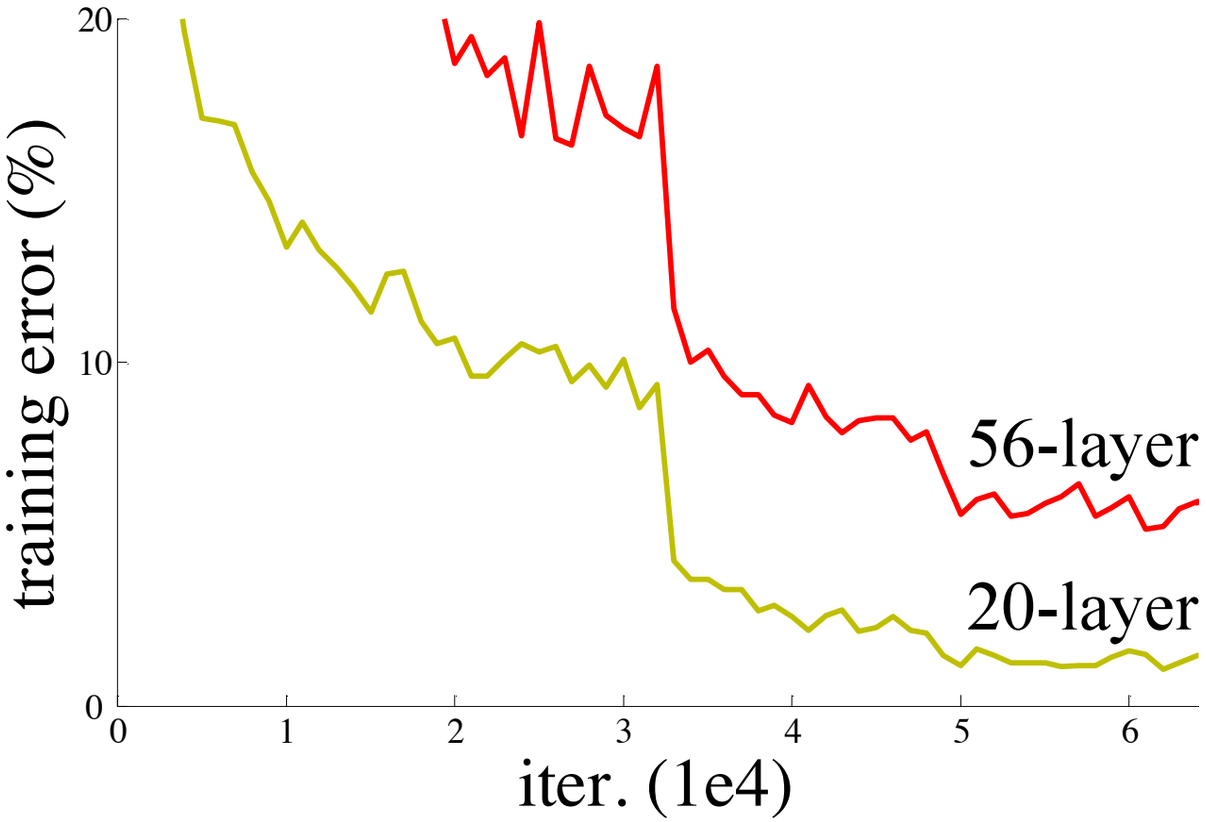
→Depth of network is key

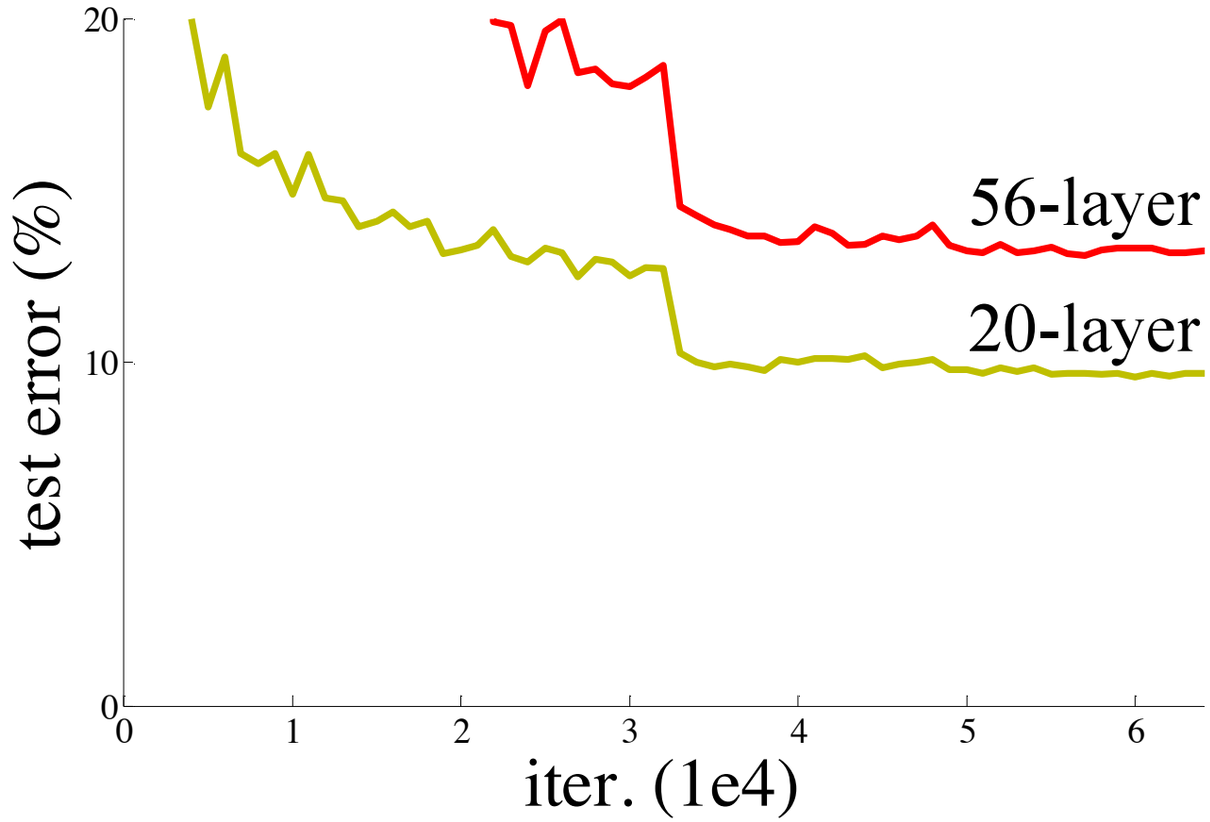






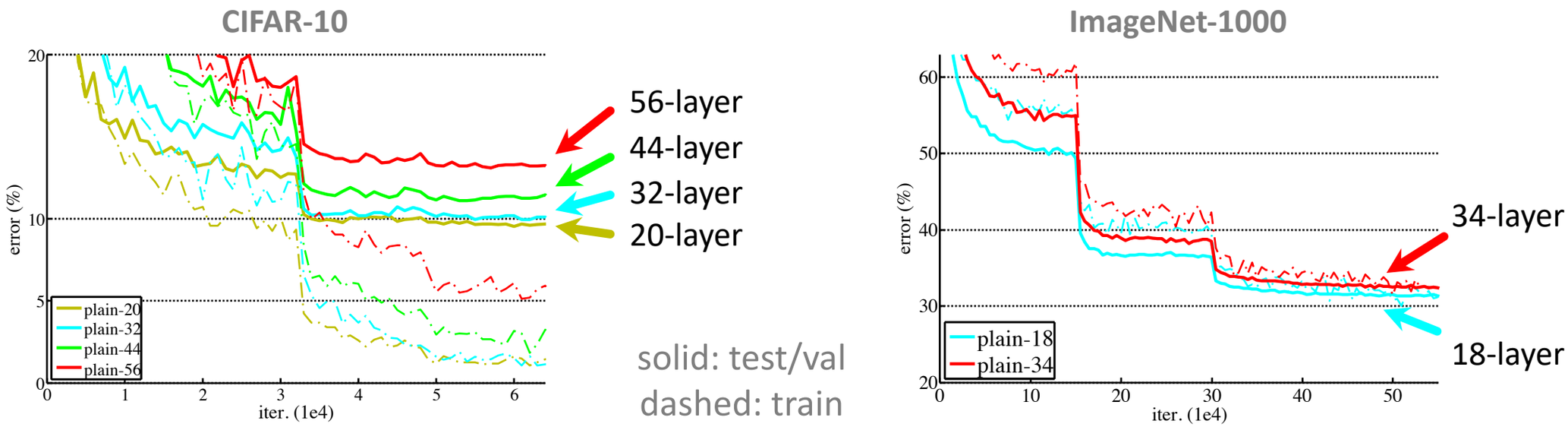
What should happen if I train a deeper network?





What should happen if I train a deeper network?

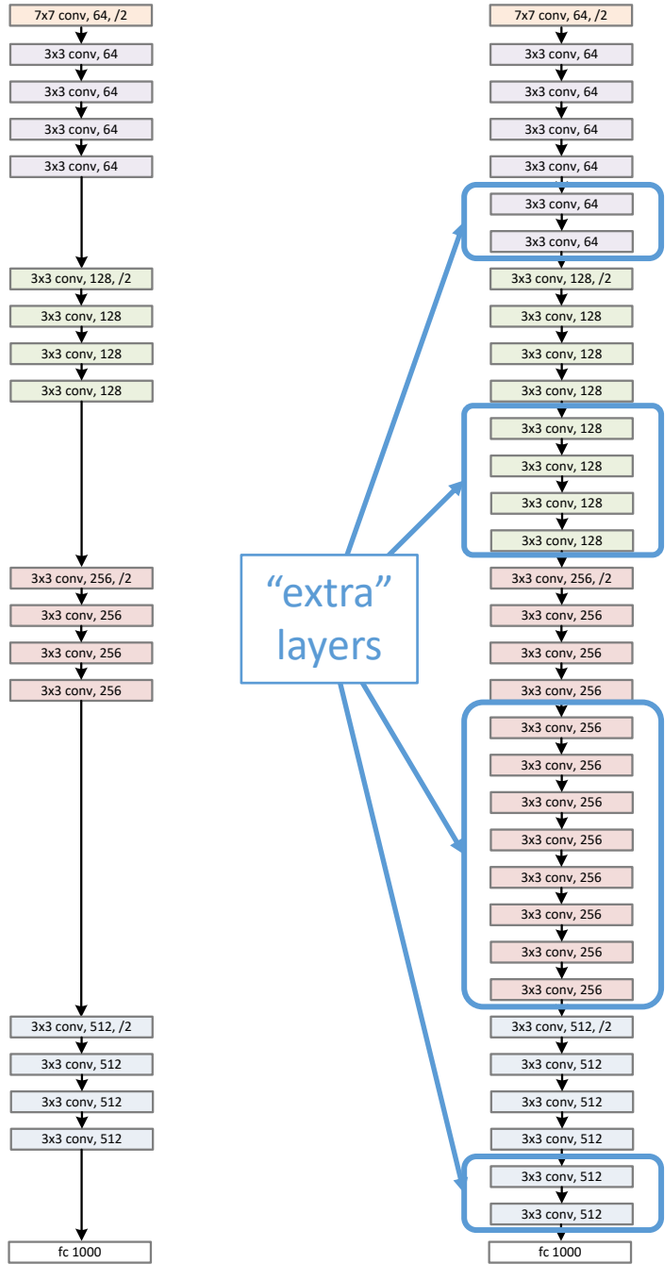
Simply stacking layers?



- "Overly deep" plain nets have **higher training error**
- A general phenomenon, observed in many datasets

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

a shallower model (18 layers)



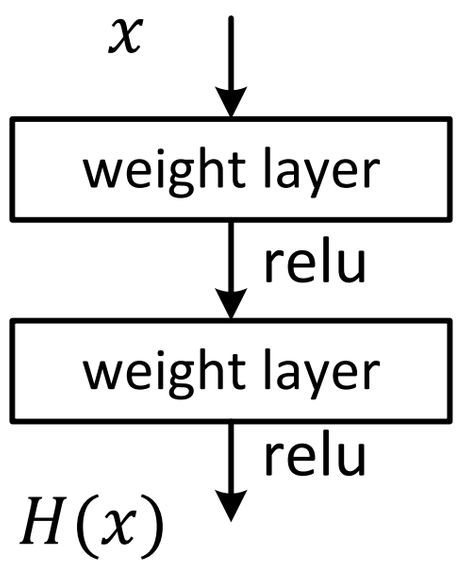
a deeper counterpart (34 layers)

- Richer solution space
- A deeper model should not have **higher training error**
- A solution *by construction*:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
- Optimization difficulties: solvers cannot find the solution when going deeper…

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Deep Residual Learning

- Plaint net



any two stacked layers

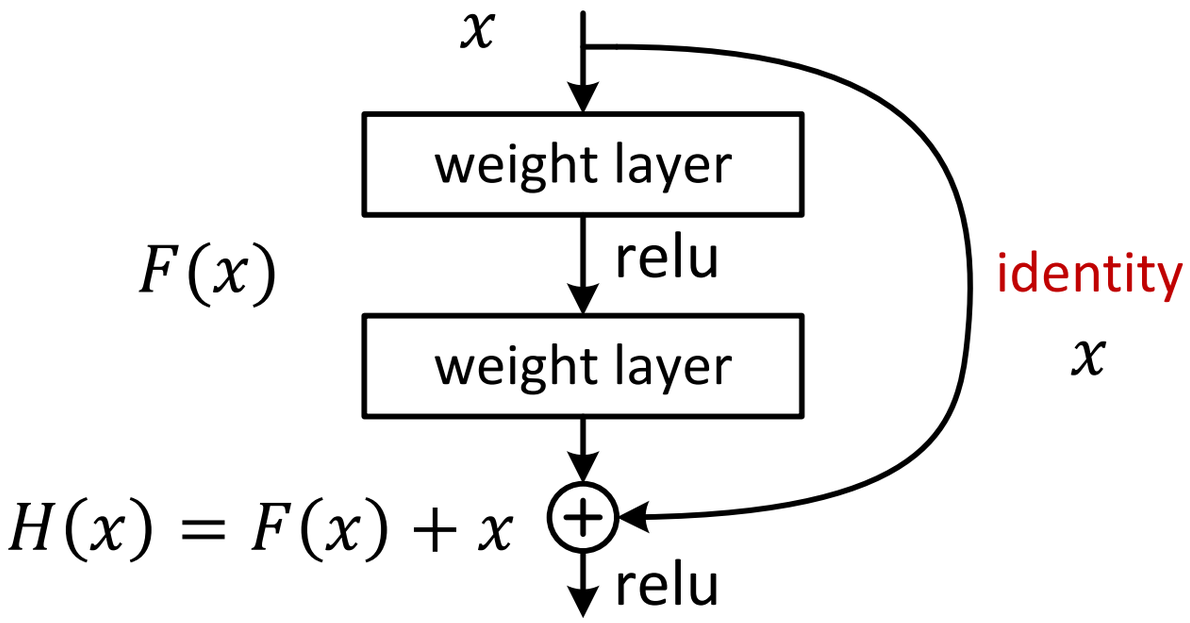
$H(x)$ is any desired mapping,

hope the 2 weight layers fit $H(x)$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Deep Residual Learning

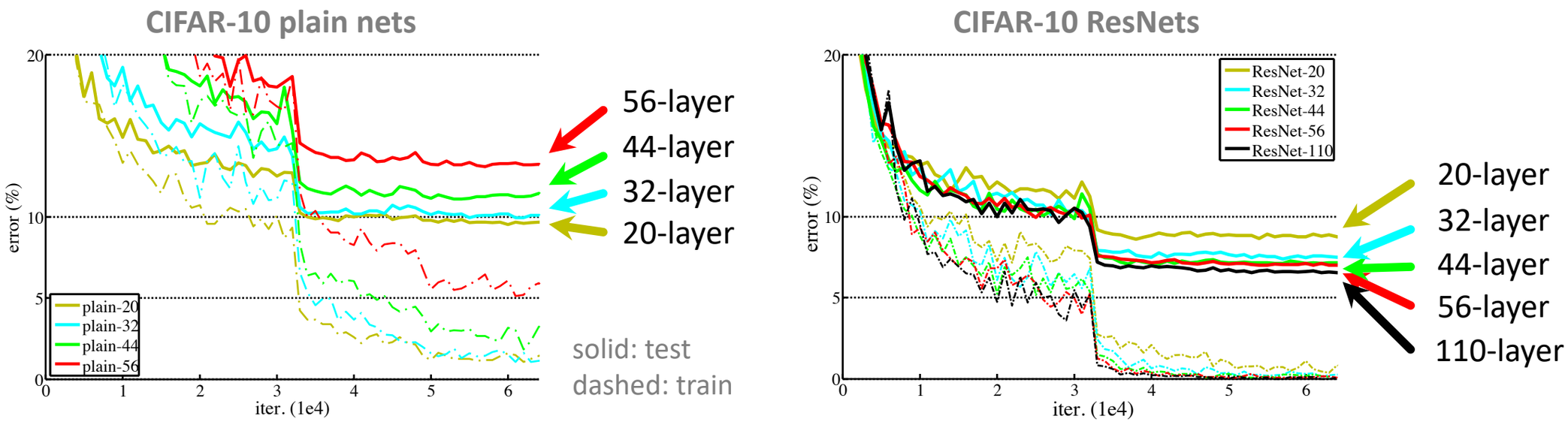
- $F(x)$ is a residual mapping w.r.t. identity



- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

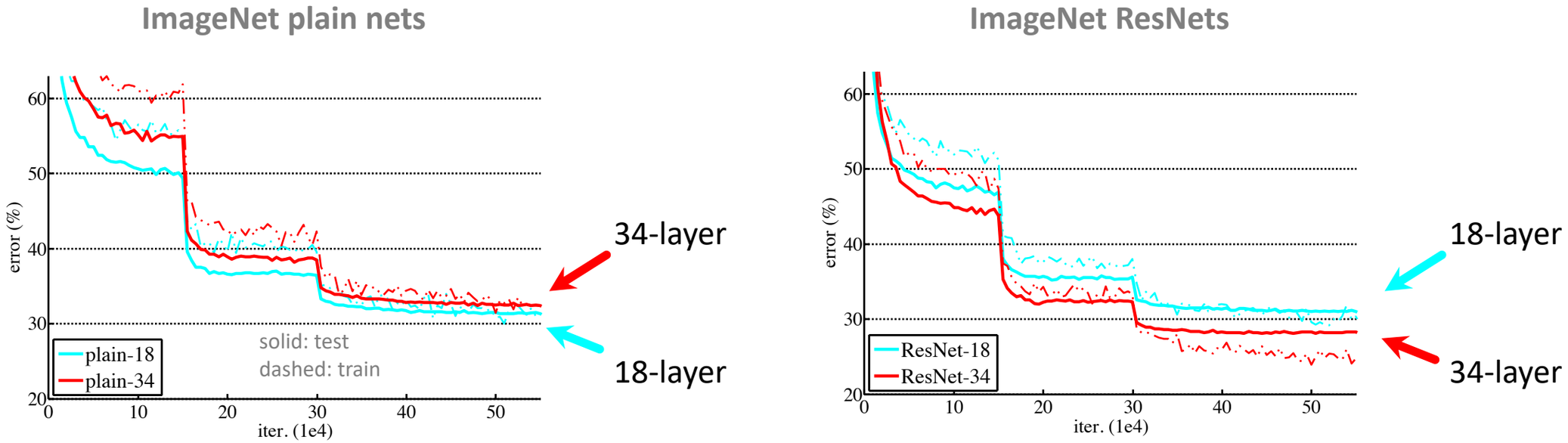
CIFAR-10 experiments



- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

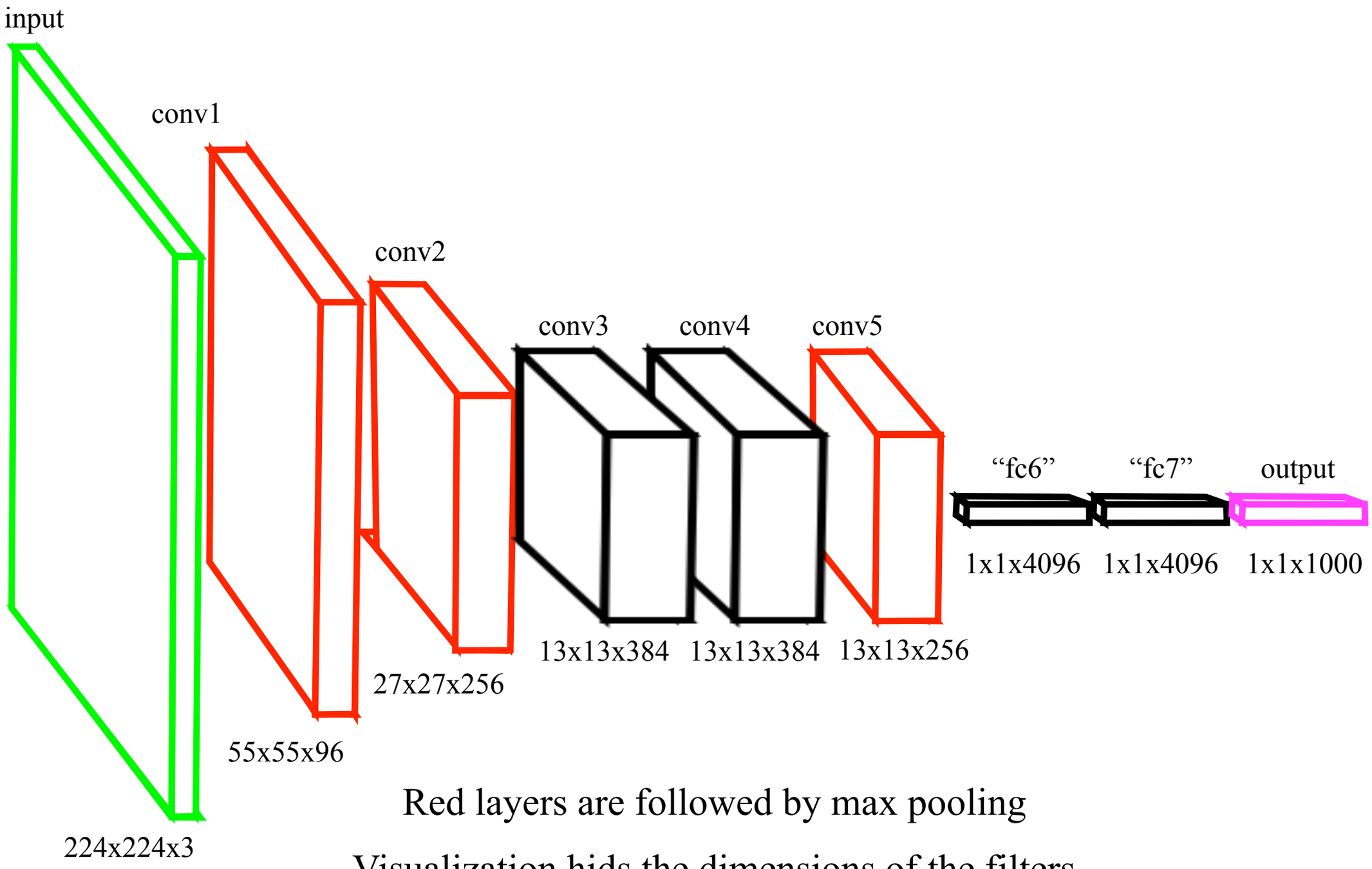
ImageNet experiments



- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Millions of parameters

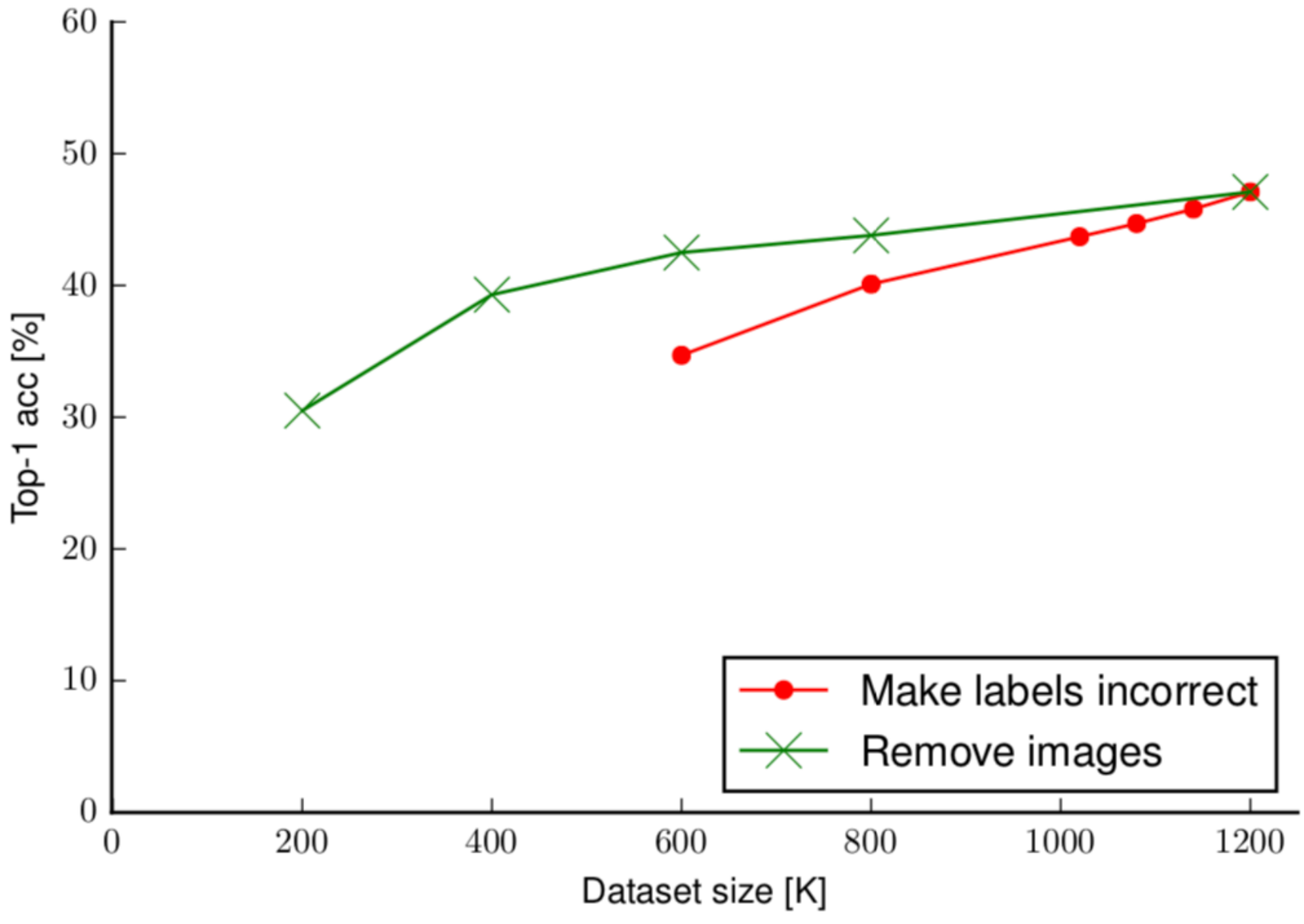


Red layers are followed by max pooling

Visualization hids the dimensions of the filters

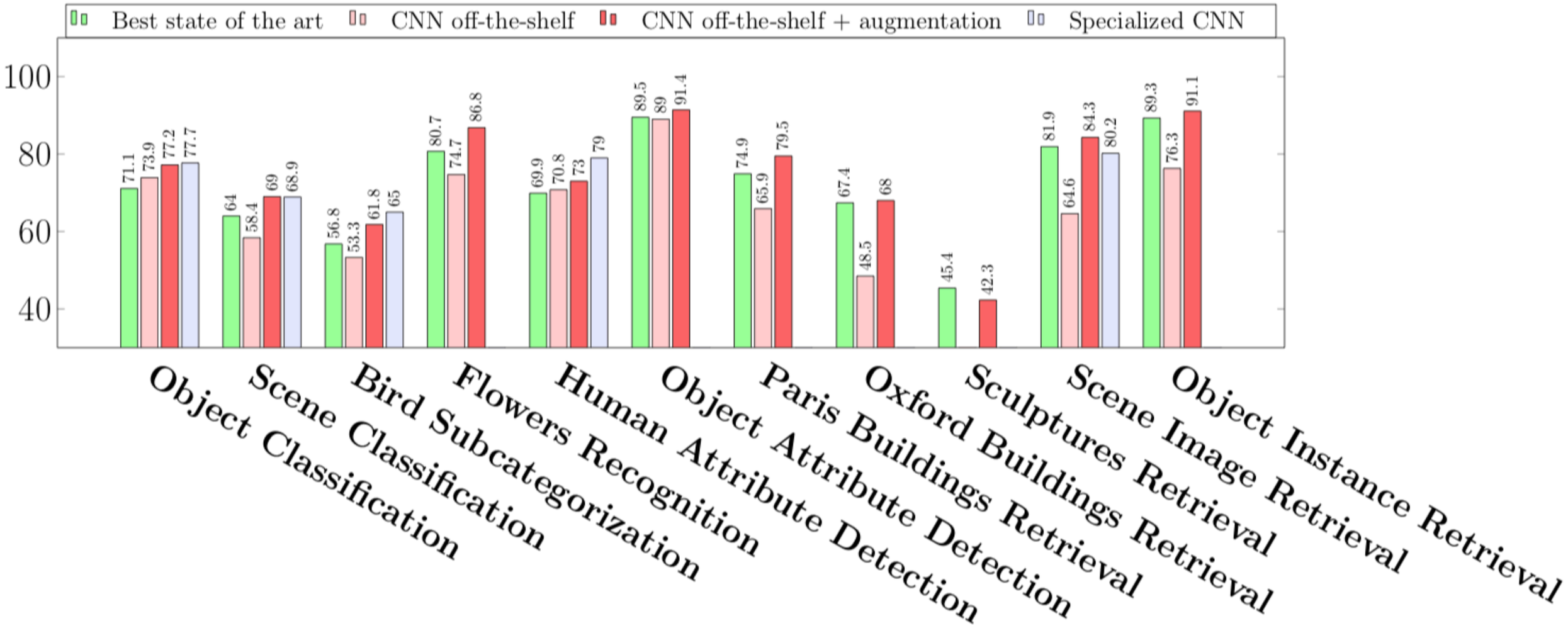
Slide credit: Deva Ramanan

How much data do you need?



Systematic evaluation of CNN advances on the ImageNet

How much data do you need?

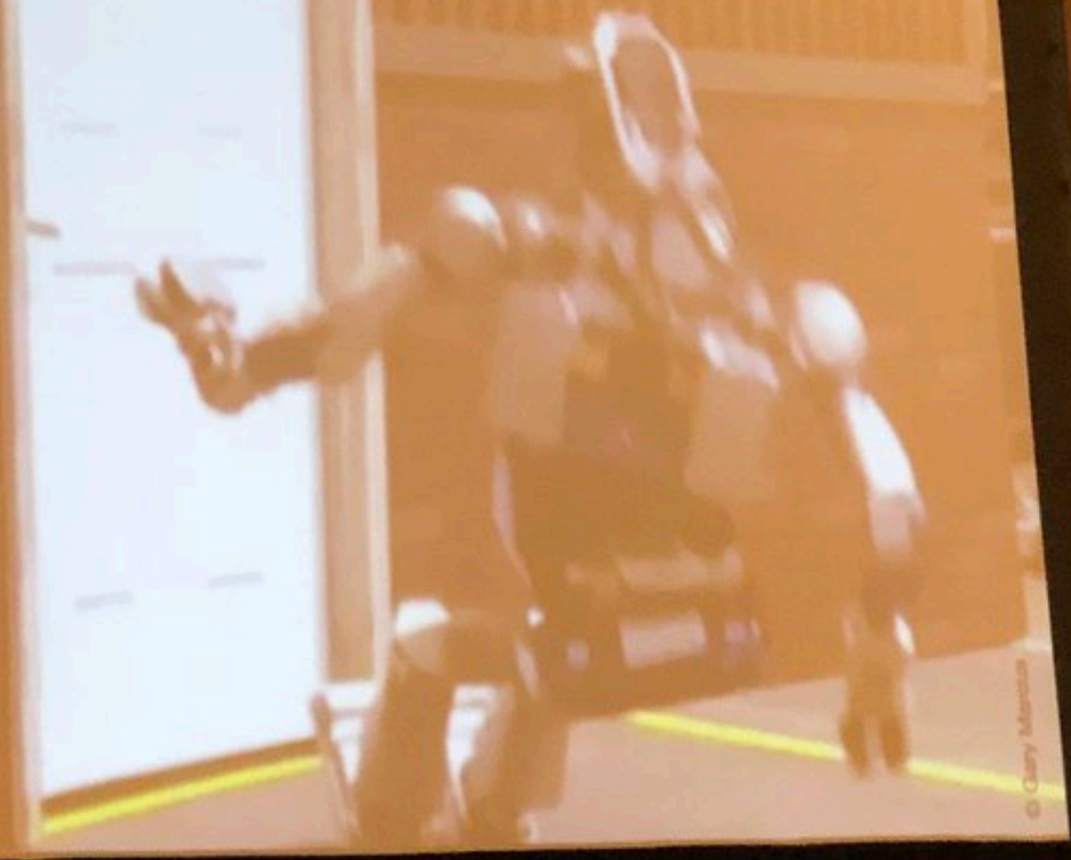


CNN Features off-the-shelf: an Astounding Baseline for Recognition

Slide from Gary Marcus

In the event of robot attack

- Close the door (or if things get really rough, lock it).
- Hide behind a bus, dress like a bus, or hide behind a shiny toaster.
- Keep a pack of psychedelic stickers handy, and a giant fan, have something handy that is slimy or silly (like jacks or banana peels) to throw on the floor.
- Or just talk in a noisy room with a foreign accent.
- In the worst case, climb some stairs, or maybe a tree.



Next Class

Neural networks for visual recognition