Generative Adversarial Networks

ML@B Deep Learning Workshop
Fall 2017

Who we are

Phillip Kuznetsov
EECS
philkuz [at] ml.berkeley.edu

Phillip Kravtsov
EECS
phillipk [at] ml.berkeley.edu

Deep Learning Introduction

Survey

1. Who knows what a fully-connected layer is?
    a. What about a Convolutional Layer?
2. Who knows what a gradient is?
3. Who knows how to use gradients to update the parameters in a neural network?

Disclaimer:
This is **not** a full overview of Deep Learning

But these are our favorite resources

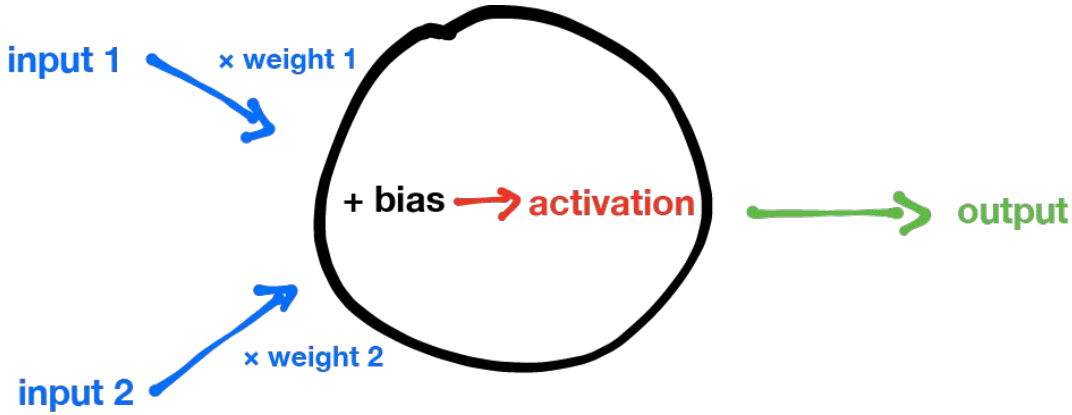
ML@B's Machine Learning Crash Course: Part 3

Stanford's CS231n Convolutional Networks for Visual Recognition

Deep Learning Textbook by Goodfellow et. al.

Coursera's Intro to Deep Learning

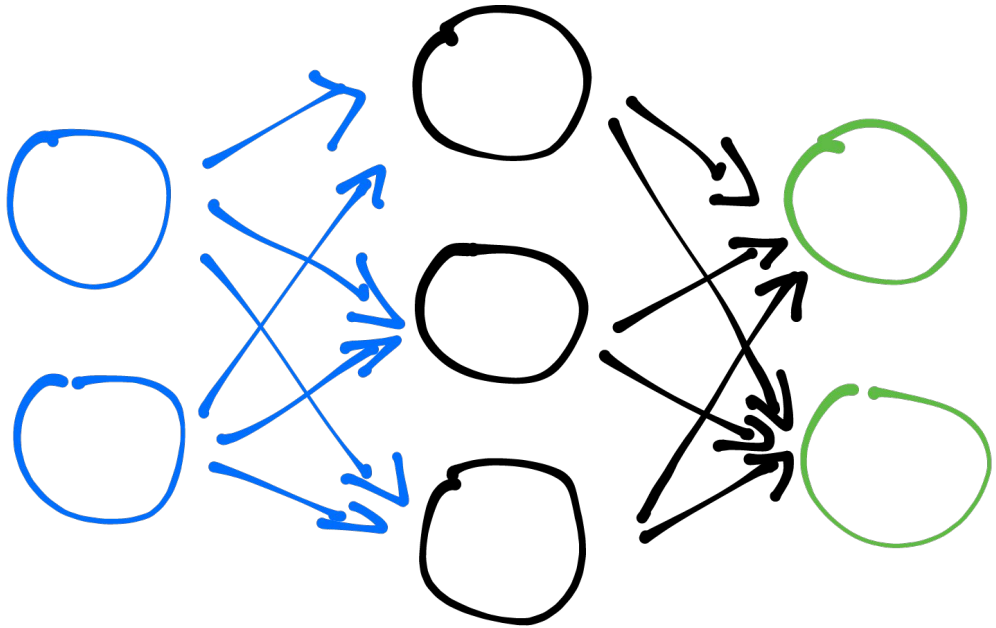
The Neuron



$$\sigma\left(\sum_{i=1}^{n} w_i x_i + b_i\right)$$

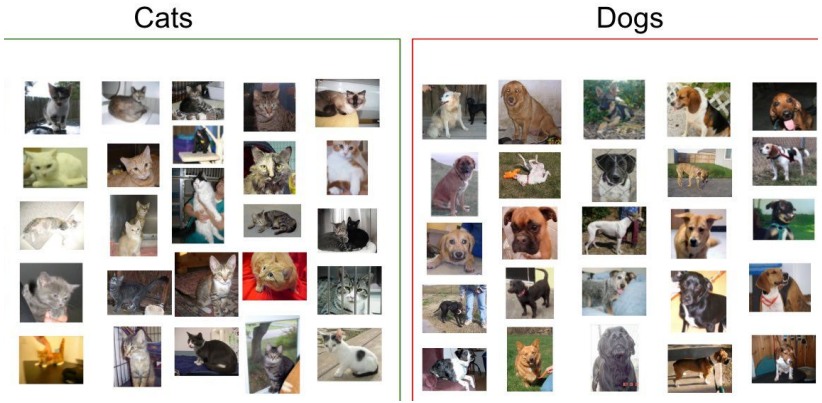
$$\sigma\left(\vec{w} \cdot \vec{x} + \vec{b}\right)$$

Networks



$$\hat{y} = f(\vec{x}) = \sigma\left(W^{(2)}\sigma\left(W^{(1)}\vec{x} + \beta^{(1)}\right) + \beta^{(2)}\right)$$

How Do Neural Networks Learn?



Dataset

$$\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$$

+

$$\mathcal{L}(f(x), y) = \begin{cases} 0 & \text{if } f(x) = y \\ 1 & \text{else} \end{cases}$$

Loss Function

Common Loss Functions

- MSE: Simple, works for regression

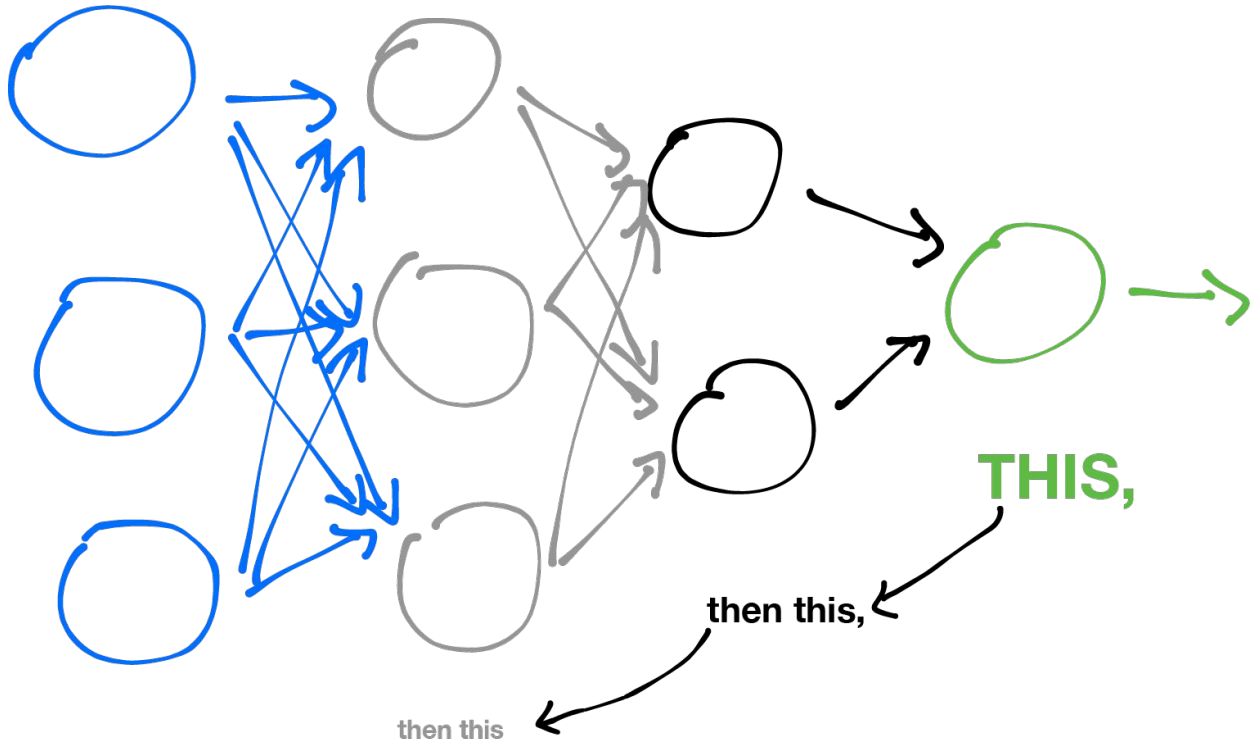
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

- Cross entropy: Very common, best for classification

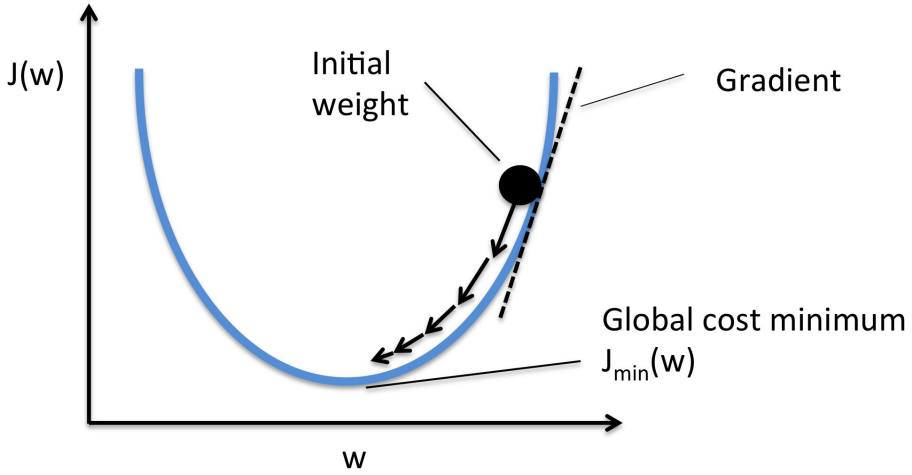
$$C = -\frac{1}{n}\sum_{x}\left[y \ln a + (1-y)\ln(1-a)\right],$$

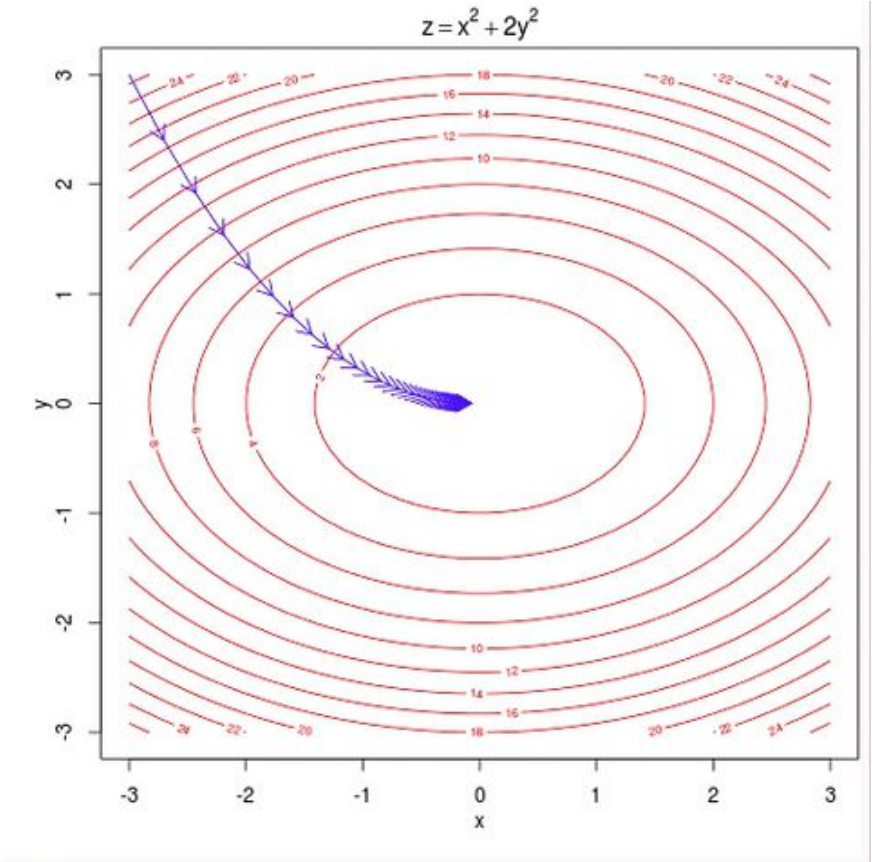
How Do Neural Networks Learn?

To correct the network, you must first fix...

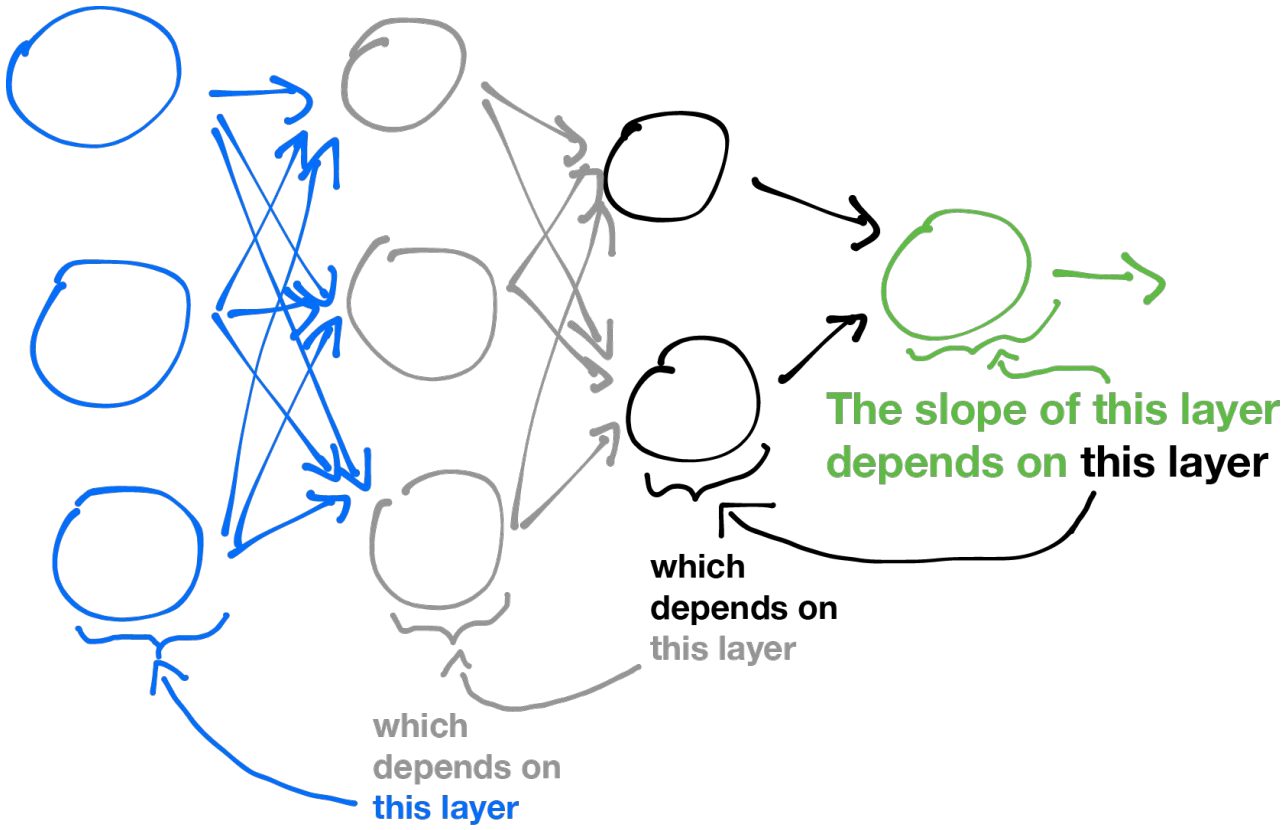


Gradient Descent

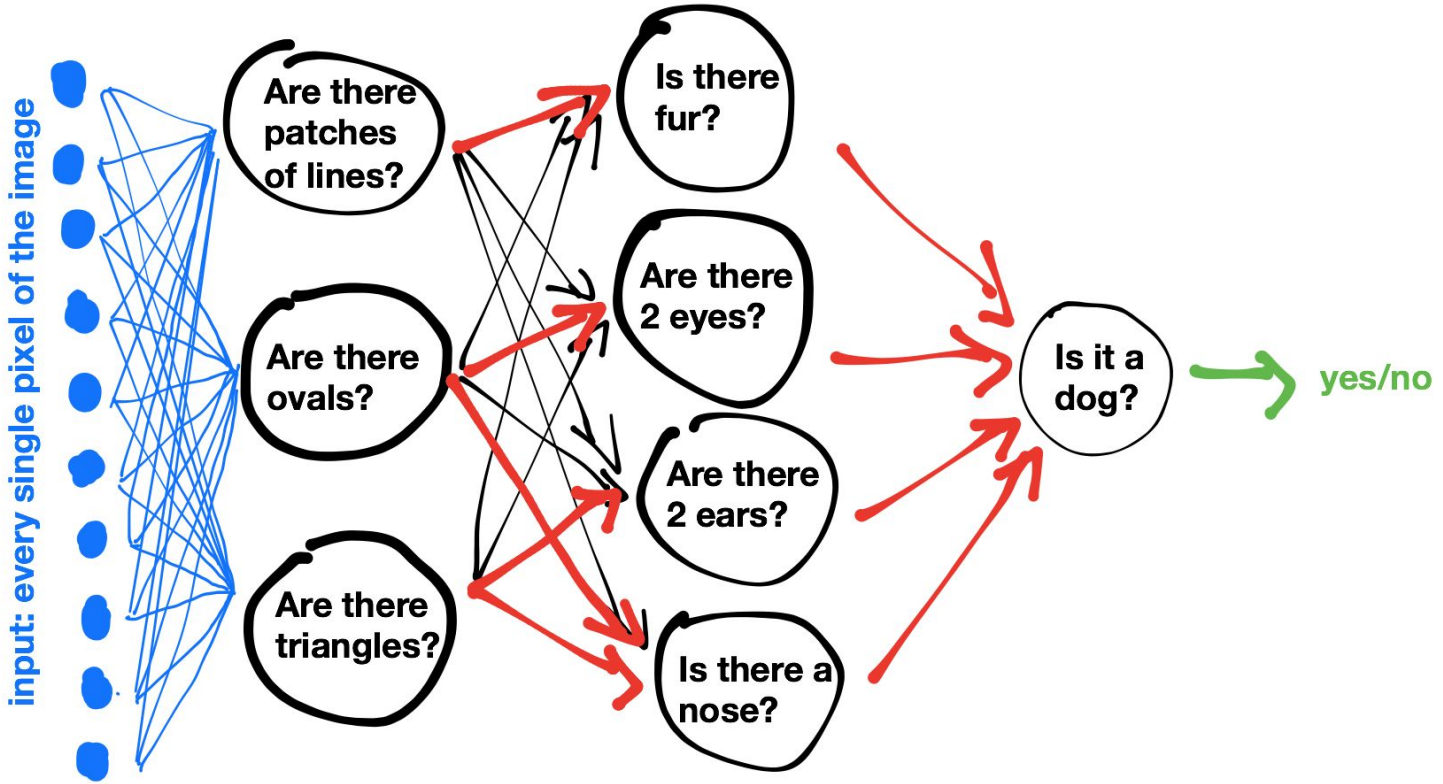




How Do Neural Networks Learn?



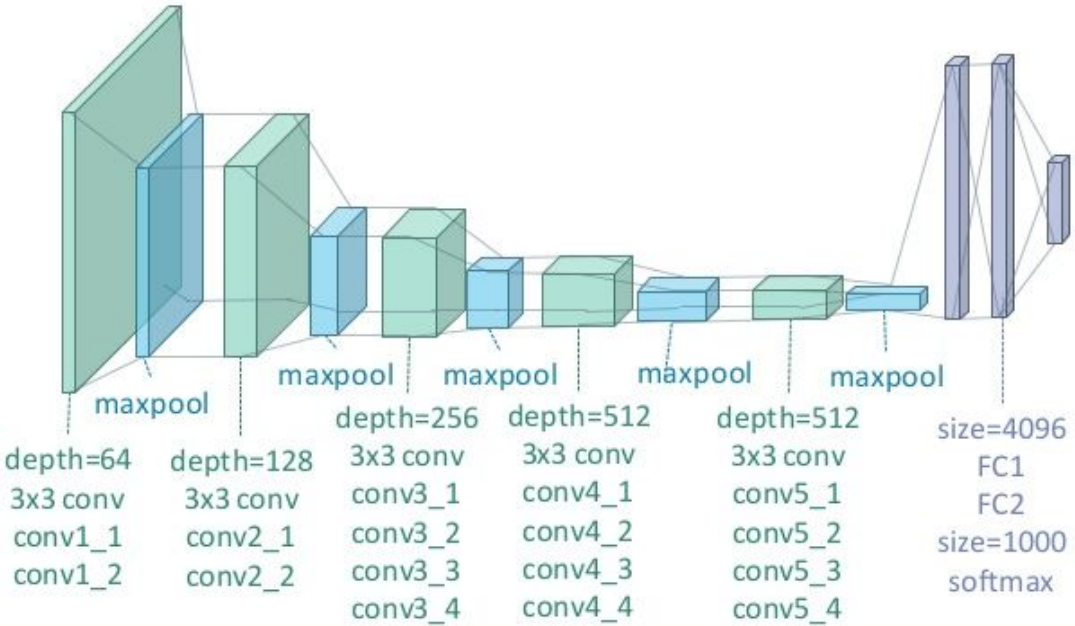
Networks


input: every single pixel of the image
Are there patches of lines?
Are there ovals?
Are there triangles?
Is there fur?
Are there 2 eyes?
Are there 2 ears?
Is there a nose?
Is it a dog?
yes/no

Recap

- Neural Nets are composed of layers of matrix multiplies with a non-linearity
- We need data and a loss function to train a neural network
- Neural Nets learn by calculating the gradient of the loss function, and descend the gradient
- Neural Nets build abstractions on top of abstractions while learning to make complex decisions

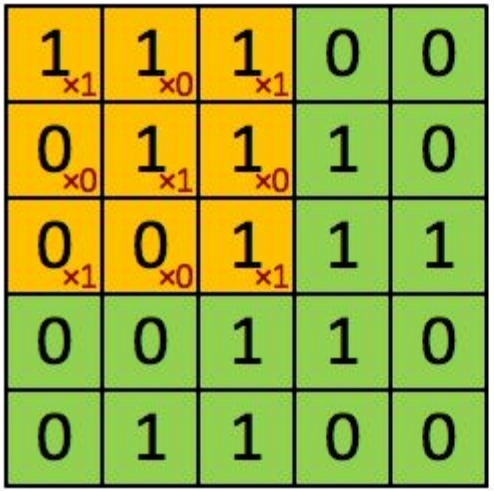
Convolutional Neural Networks



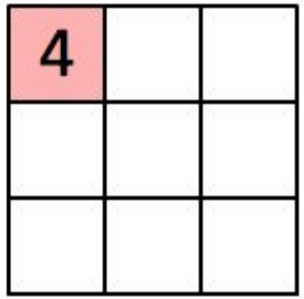
Convolutions

Weight Filter

1	0	1
0	1	0
1	0	1



Image

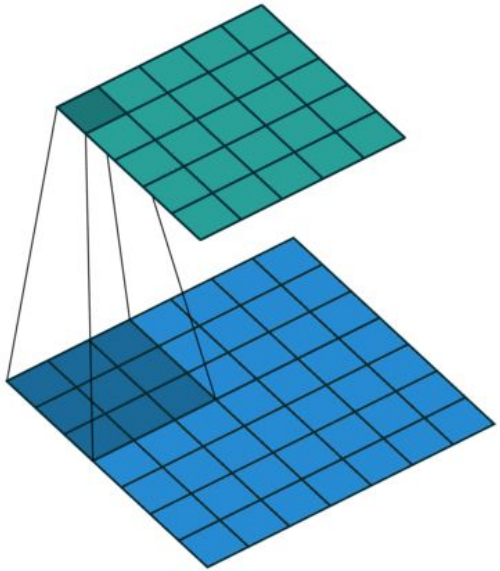


Convolved Feature

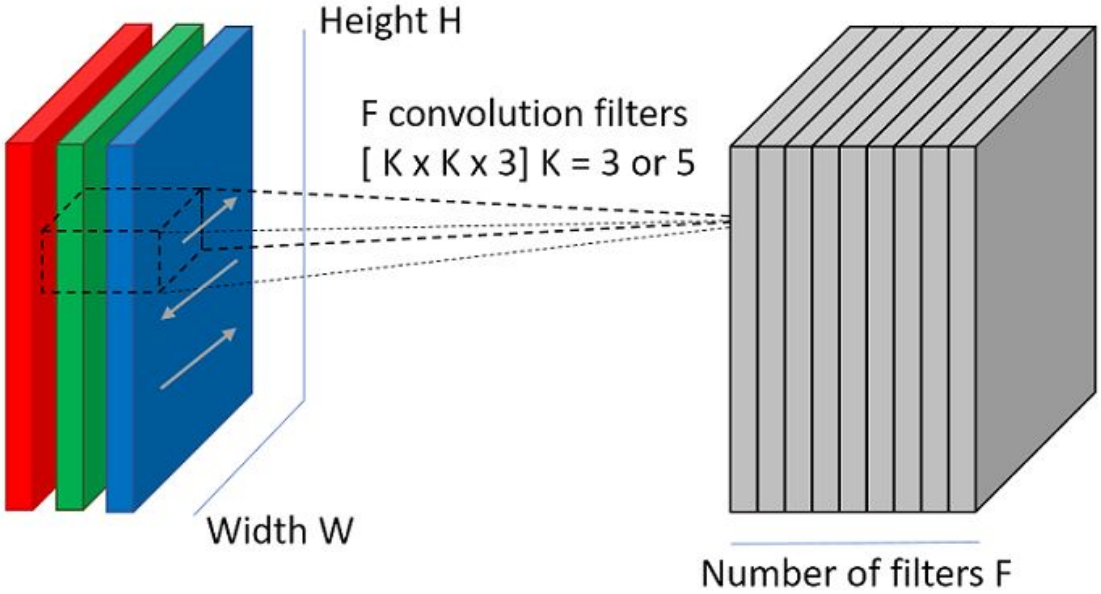
Convolutions

Weight Filter

1	0	1
0	1	0
1	0	1



Convolutional Layers



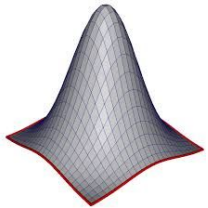
Input Layer (RGB pixels)
[H x W x 3]

Convolution Layer Output
[H x W x F]
assuming stride=1 and zero padding

Convolution for Blurring

*

1	2	1
2	4	2
1	2	1



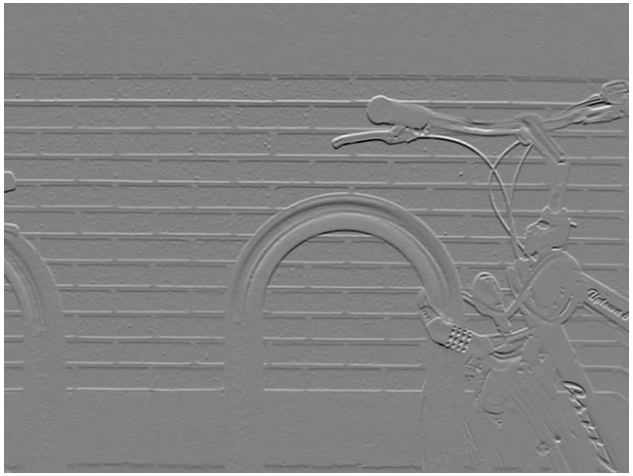
=

Convolution for Edge Detection

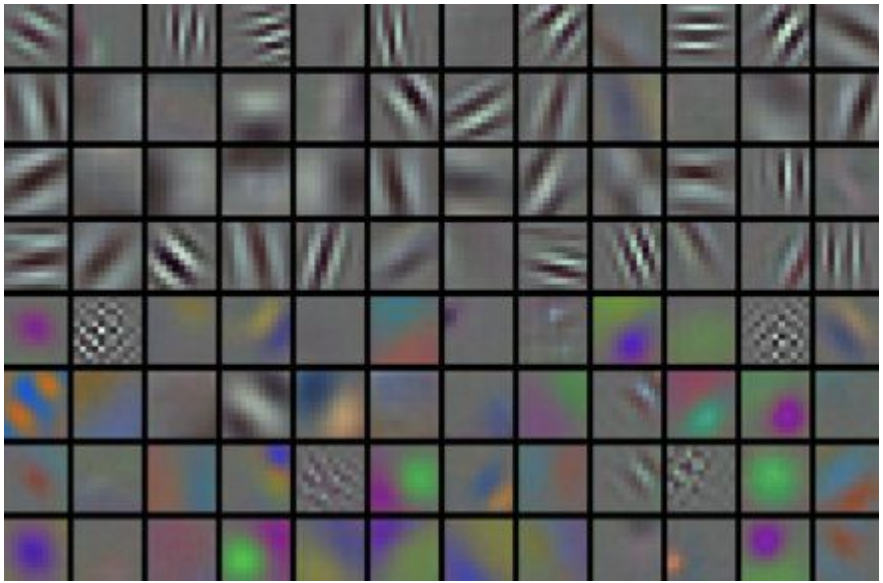
*

1	2	1
0	0	0
-1	-2	-1

=

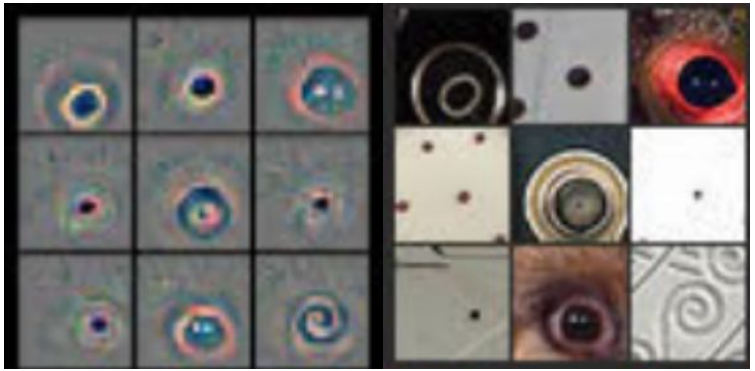


Early Layers Learn Edge Detectors



Later Layers Build Interesting Abstractions

Visualizing and Understanding Convolutional Networks
Zeiler & Fergus, 2013



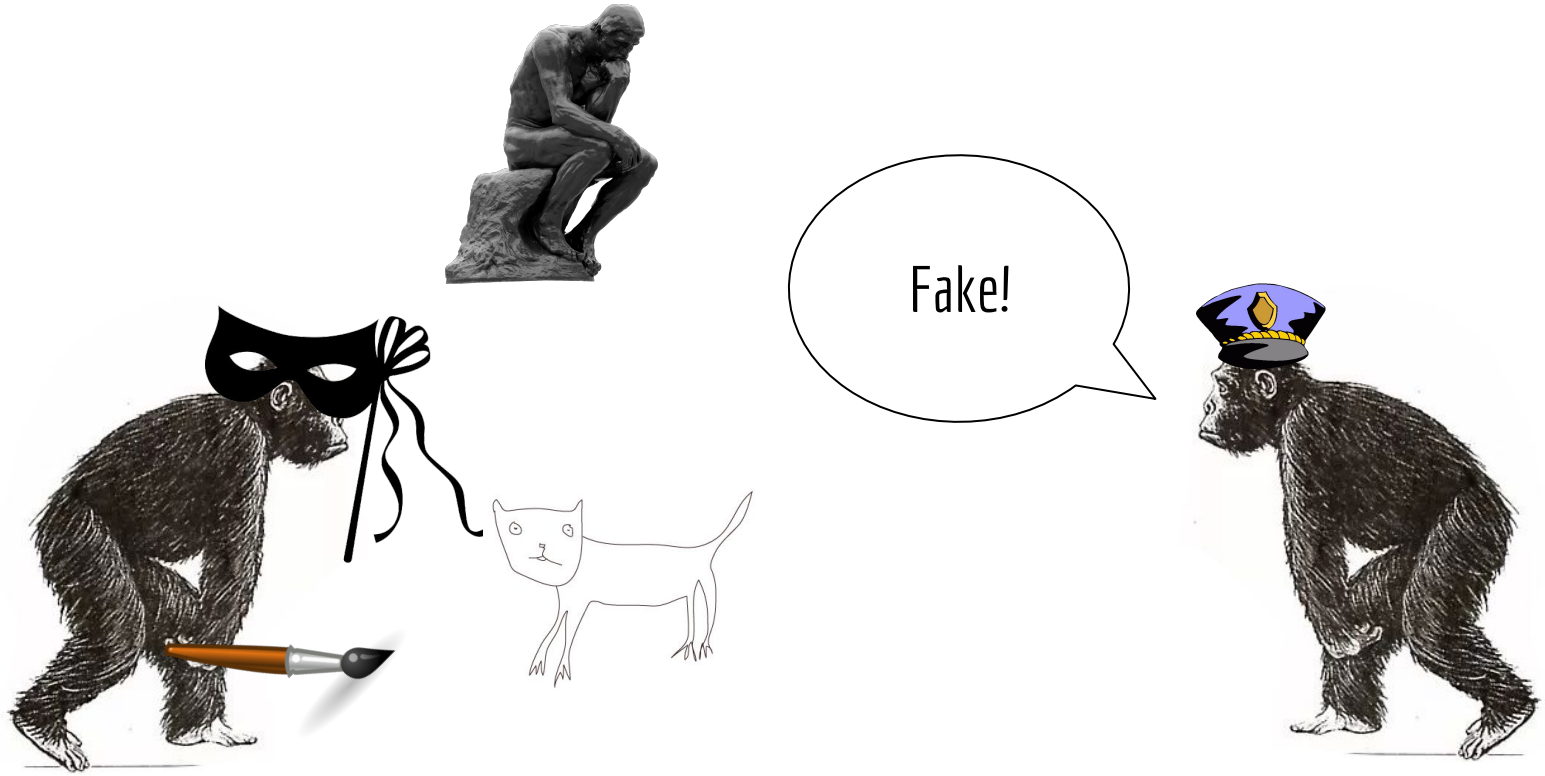
Layer 2 of AlexNet

Layer 4 of AlexNet

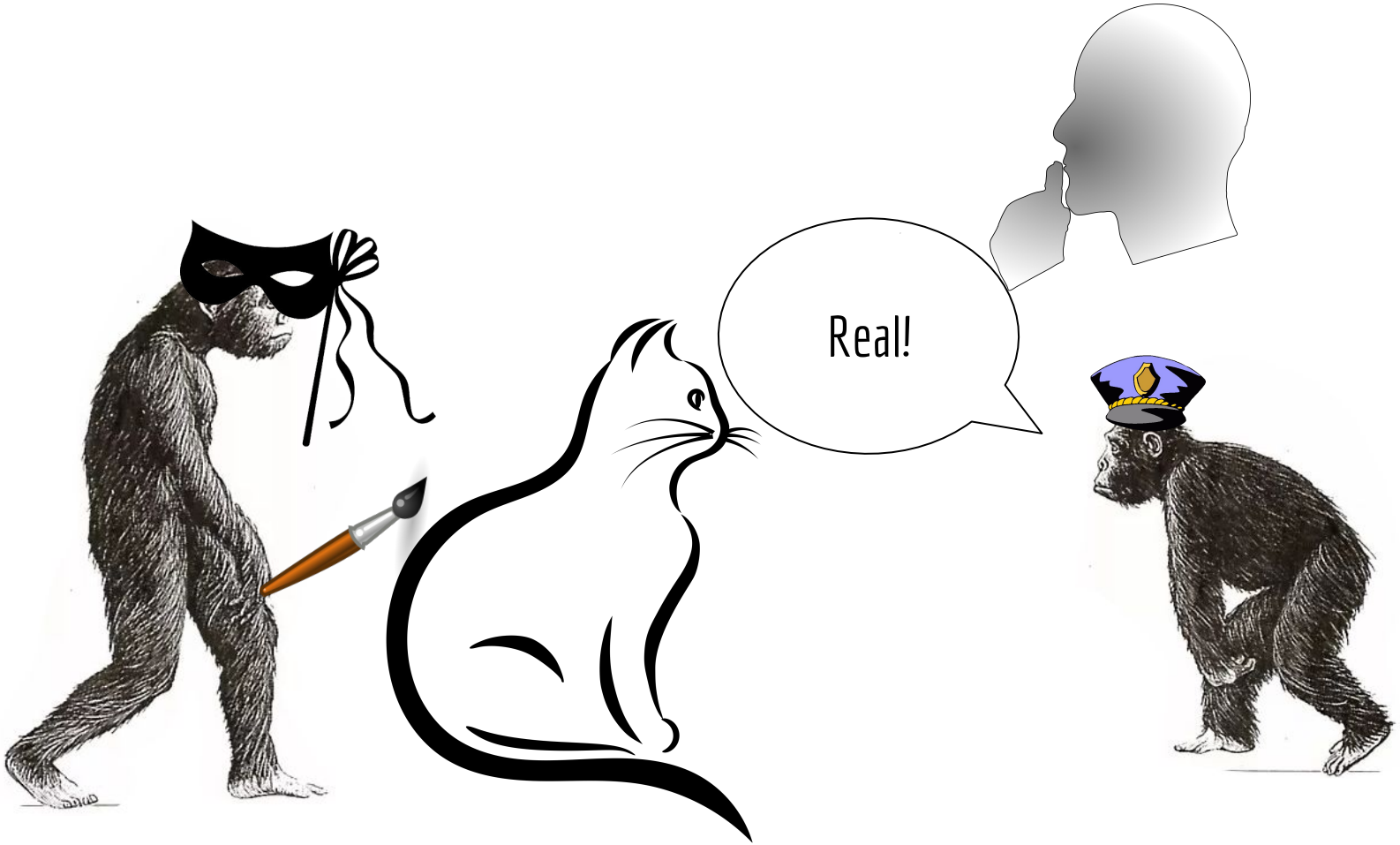
Recap

- CNNs are good for Image Processing and Computer Vision
- Convolutions manipulate the image to build interesting representations
- Deep Layers in CNN pay attention to high level information such as dog faces

GANs

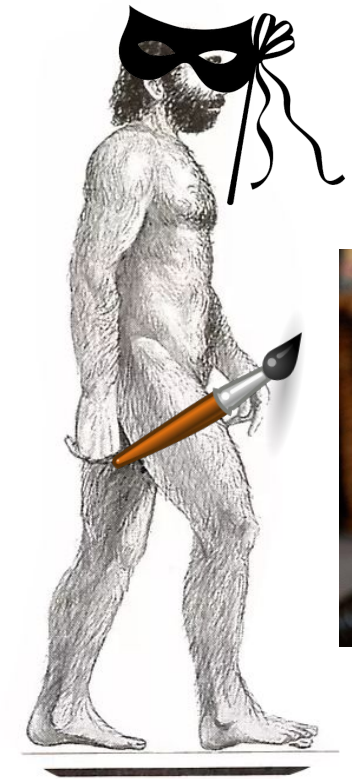


Slides adapted from Lehtinen 2017



Slides adapted from Lehtinen 2017

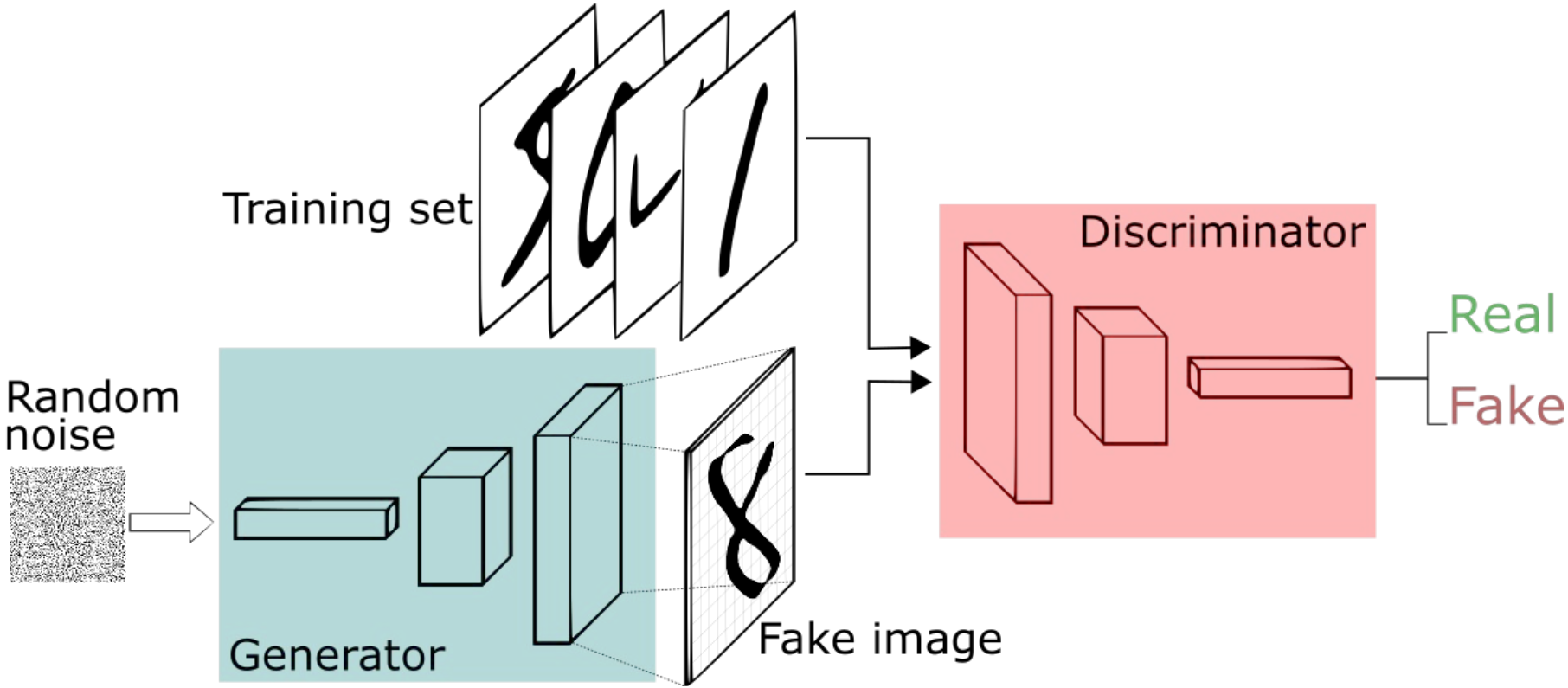
Slides adapted from Lehtinen 2017



Slides adapted from Lehtinen 2017

G

D



From [Thalle Silva's Intro to GANs]

Loss Formalization

$$\max_D \mathbb{E}_{x \sim p_{data}} [\log D(x))]$$

$$\min_G \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z))]$$

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

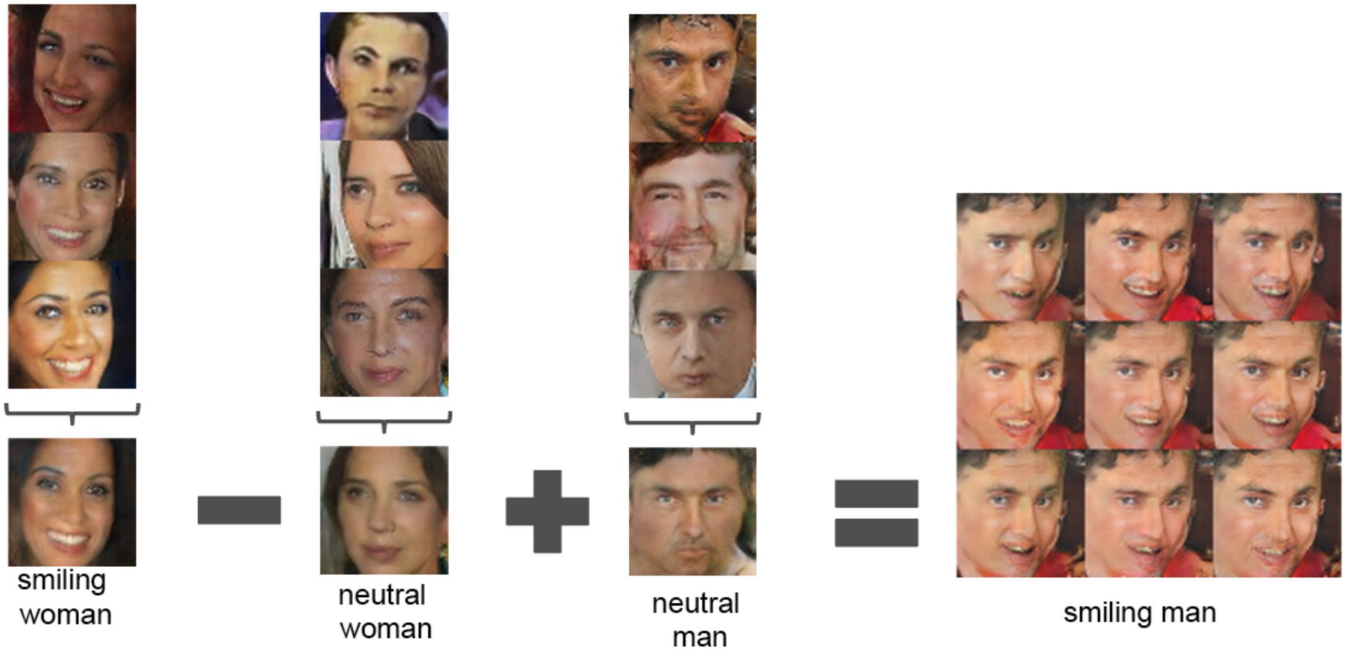
Demo

Cool Properties

Image Interpolation

Input A — Interpolation from A to B — Input B

Input Space Arithmetic



From DCGAN

Conditional GAN

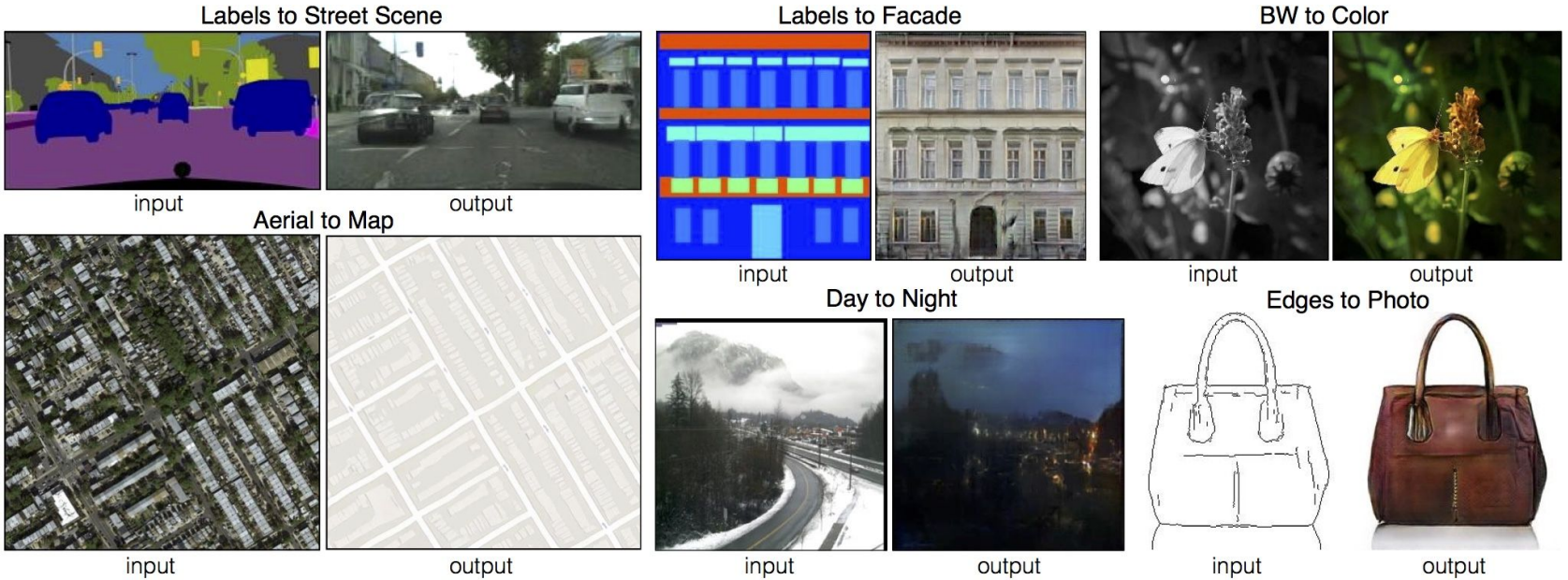
- Add extra info to the generator's input
- Text embeddings, class labels, sketches

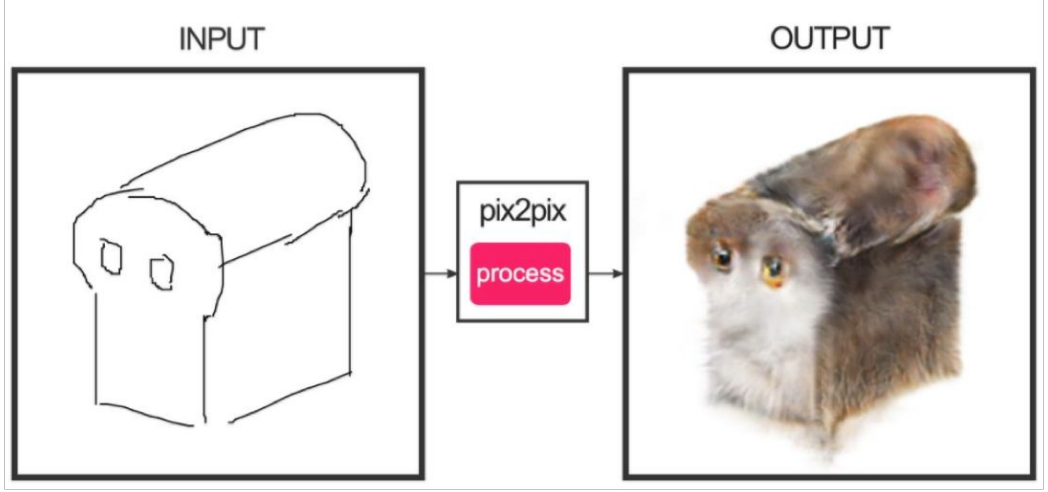
Caption	Image
this vibrant red bird has a pointed black beak	
this bird is yellowish orange with black wings	
the bright blue bird has a white colored belly	

StackGAN

Pix to Pix

Demo





Ivy Tasi @ivymyt

Vitaly Vidmirov @vvid

Issues

Measuring Quality is Difficult

Measuring Quality is Difficult

Attempts to Measure Quality

Inception Score - uses a neural network to measure quality of output

Human Trials - Amazon Mechanical Turk

MS-SSIM

Mode Collapse

- Extremely common
- Many inputs map to same output
- Easy to diagnose, difficult to fix

Poor output structure

Difficult to capture the structural properties of images for many datasets

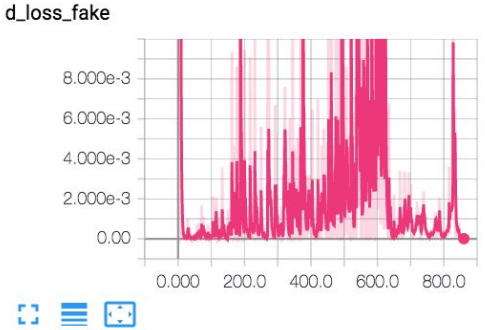
Often work better on datasets with many similar images

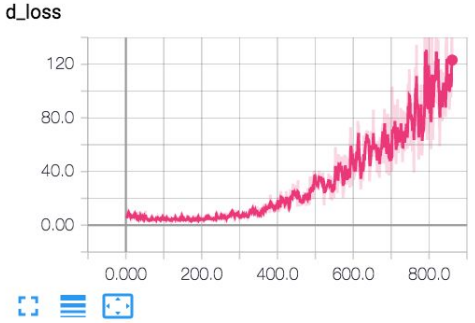


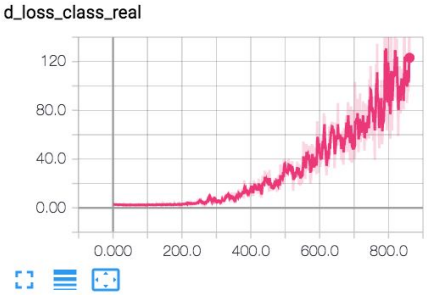


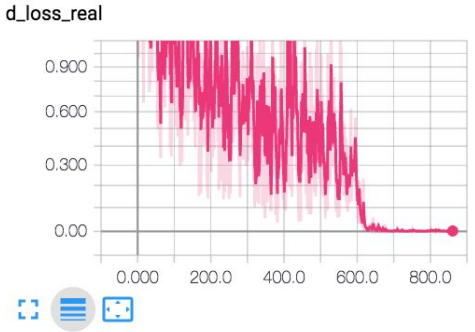

Training Fails Often

- Discriminator can win easily
- Difficult to control
- Look at TensorBoard for info



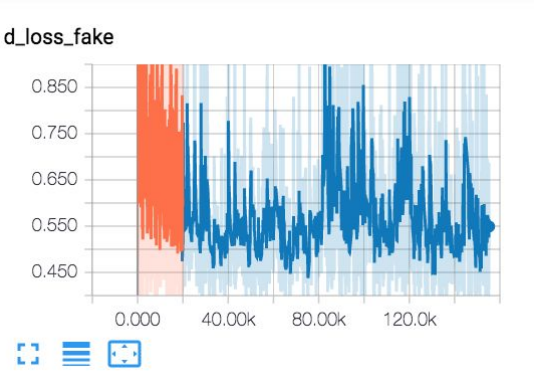


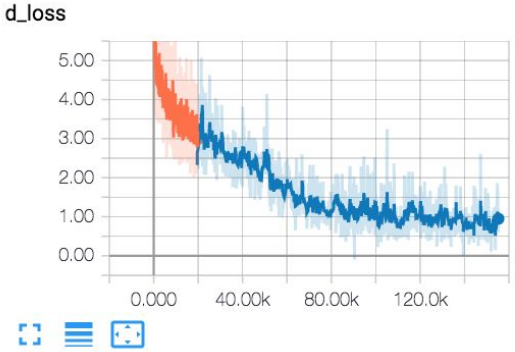


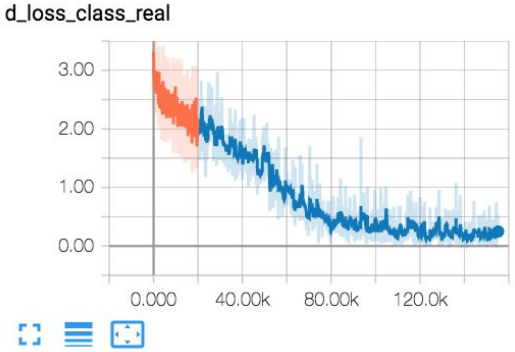


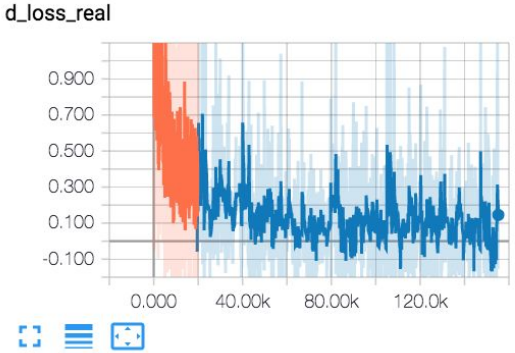
Better Loss Curves

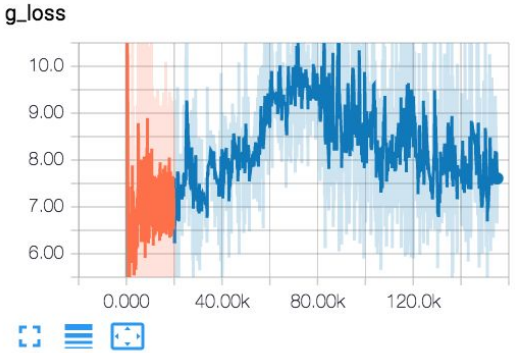
Look something like this:











Long, Difficult Training Cycles

- Even on GPUs, GANs take many days to train
- Even true for 64x64, 128x128 outputs
- Need a lot of memory/time for high-res outputs, which are more prone to issues

GANHacks

- Because of the issues, lots of improvements
- GANhacks repo

1. Normalize the inputs

- normalize the images between -1 and 1
- Tanh as the last layer of the generator output

2: A modified loss function

In GAN papers, the loss function to optimize G is `min (log 1-D)`, but in practice folks practically use `max log D`

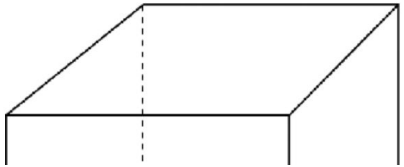
- because the first formulation has vanishing gradients early on
- Goodfellow et. al (2014)

In practice, works well:

- Flip labels when training generator: real = fake, fake = real

3: Use a spherical Z

- Dont sample from a Uniform distribution



Wasserstein GAN (WGAN)

- Modifies loss function
- Value function actually correlates with sample quality!

$$\min_G \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} \left[D(\boldsymbol{x}) \right] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} \left[D(\tilde{\boldsymbol{x}}) \right]$$

- Fancy math, in practice you remove logs and clip weights to constrain forms of D, train D more
- WGAN-GP: Better form, adds gradient penalty

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}}.$$

Cool Results

Alternative Face

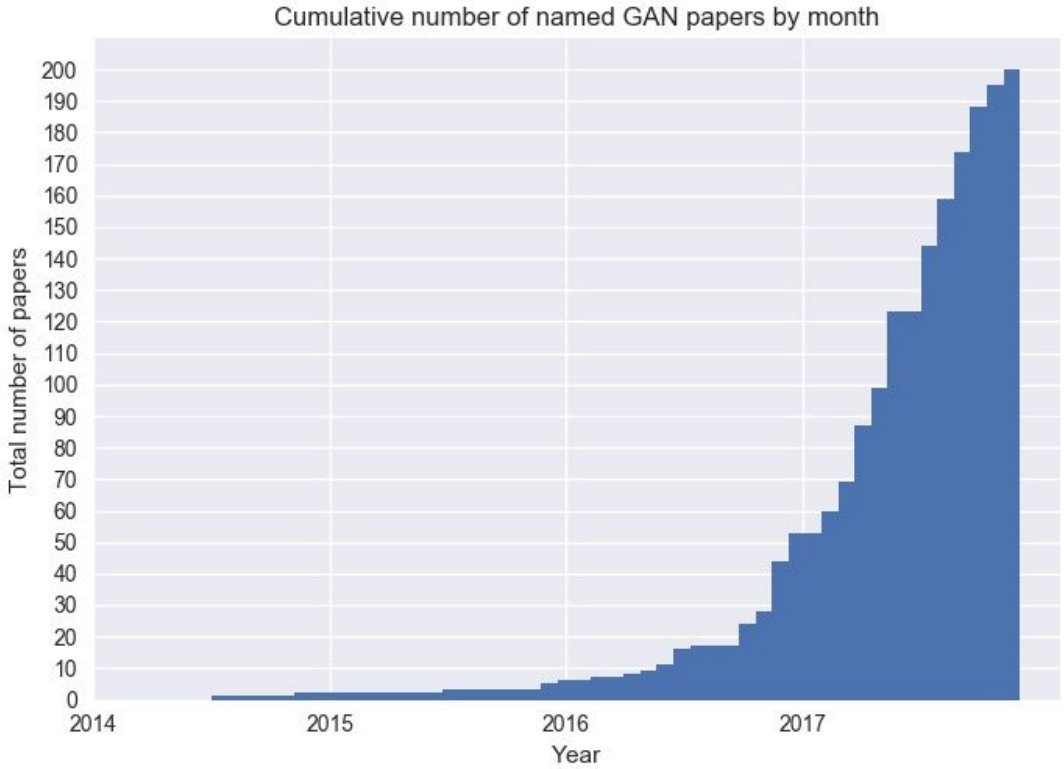
CycleGAN

Creative Adversarial Networks

Repo

Progressive Growing of GANs

What's Next?



Source: [The GAN Zoo]

Papers to Read

Ian Goodfellow's GAN Tutorial

DCGAN

https://arxiv.org/pdf/1606.03498.pdf

Wasserstein GAN

Improved Training of Wasserstein GANs

Progressive Growing of GANs

Interesting GAN papers

InfoGAN

Boundary-Equilibrium GANs

Numerics of GANs

On Convergence and Stability of GANs

Deconvolution and Checkerboard Artifacts -Not GAN specific, but relevant for Image-based GANs

Simple Winter Break Project

1. Get an AWS account and learn how to make instances
2. Get an image dataset (>30k images - check Kaggle)
3. On the server, train DCGAN on the image dataset
4. Observe results!
5. If you get mode collapse, or nothing trains, look at ganhacks