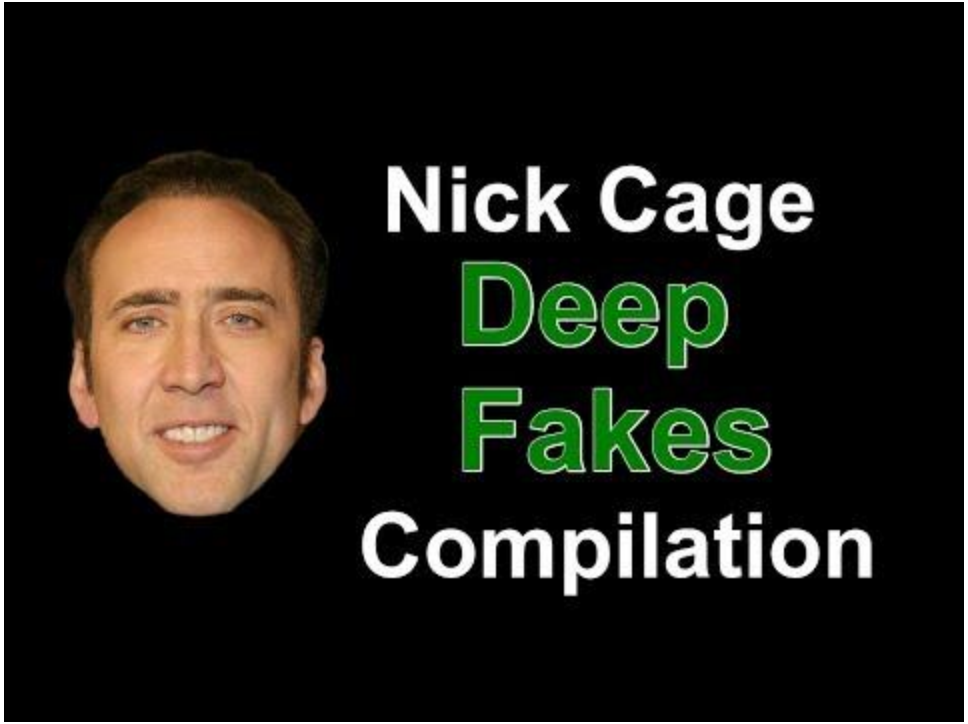
Deepfakes



CIS 522: Lecture 10

GANs
02/18/19

Penn Engineering

GANs in the News

Artificial Intelligence / Machine Learning

The GANfather: The man who's given machines the gift of imagination

By pitting neural networks against one another, Ian Goodfellow has created a powerful AI tool. Now he, and the rest of us, must face the consequences.

by **Martin Giles**

MIT Technology Review

Internet Companies Prepare to Fight the 'Deepfake' Future

Researchers are creating tools to find A.I.-generated fake videos before they become impossible to detect. Some experts fear it is a losing battle.

The New York Times

Fake Obama created using AI tool to make phoney speeches

Researchers at the **University of Washington** have produced a photorealistic former US President Barack Obama.

Artificial intelligence was used to precisely model how Mr Obama moves his mouth when he speaks.

Their technique allows them to put any words into their synthetic Barack Obama's mouth.

BBC Click finds out more.

See more at **Click's website** and **@BBCClick.**

17 Jul 2017

Share

Today's Agenda

- What is the problem solved by GANs
- Success stories
- What are GANs?
- Why are they hard to train?
- Some of the ideas behind cool GANs

- Some attention systems in CV

PollEV: what is the problem solved by a GAN?

Modeling probability distributions

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)})$$

Equivalent to minimizing the

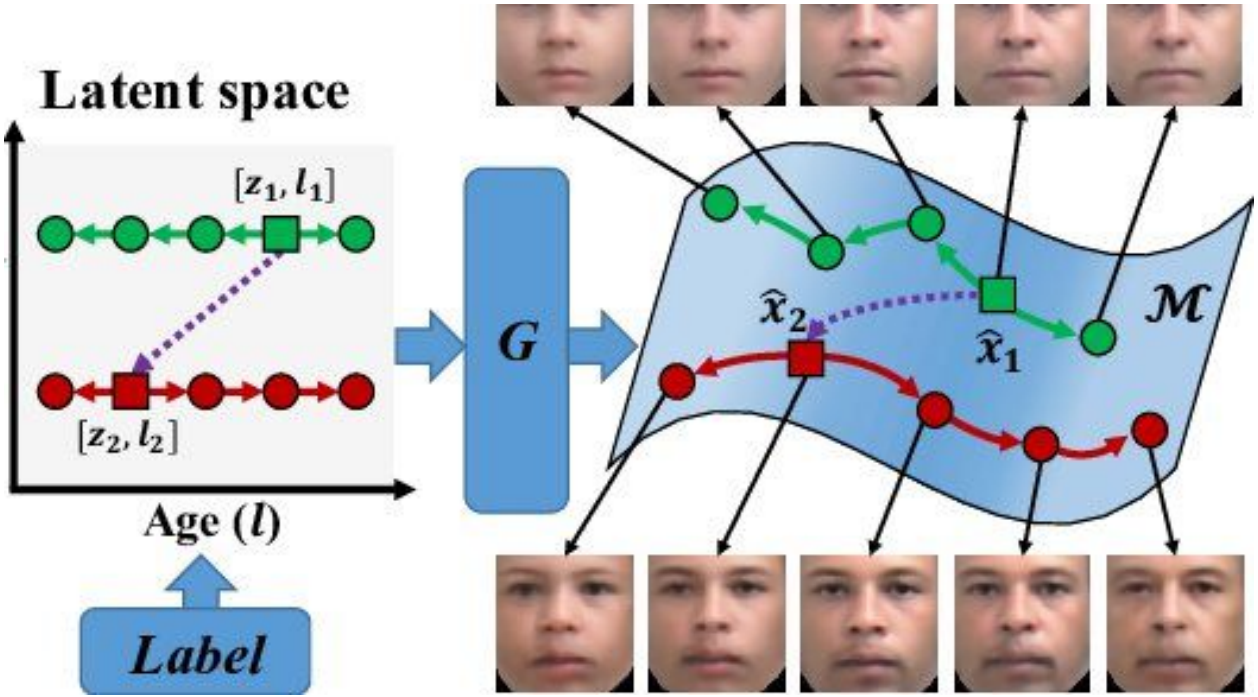
$$KL(\mathbb{P}_r || \mathbb{P}_\theta)$$

PollEV KL divergence

So why not go with this?

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)})$$

Image manifolds



Age Progression/Regression by Conditional Adversarial Autoencoder

Advantages of GANs as generators

Backprop makes it fast

No impossible integrals to solve (and no MCMC)

Any differentiable function is possible

Michael Chiu et al https://www.cs.toronto.edu/~duvenaud/courses/csc2541/slides/gan-foundations.pdf

Disadvantages of GANs as generators

Unclear stopping criteria

No explicit form for $p_g(x)$

Hard to train (minimax is poorly understood)

Need to babysit it

Poor evaluation metrics (in comparison to VLB)

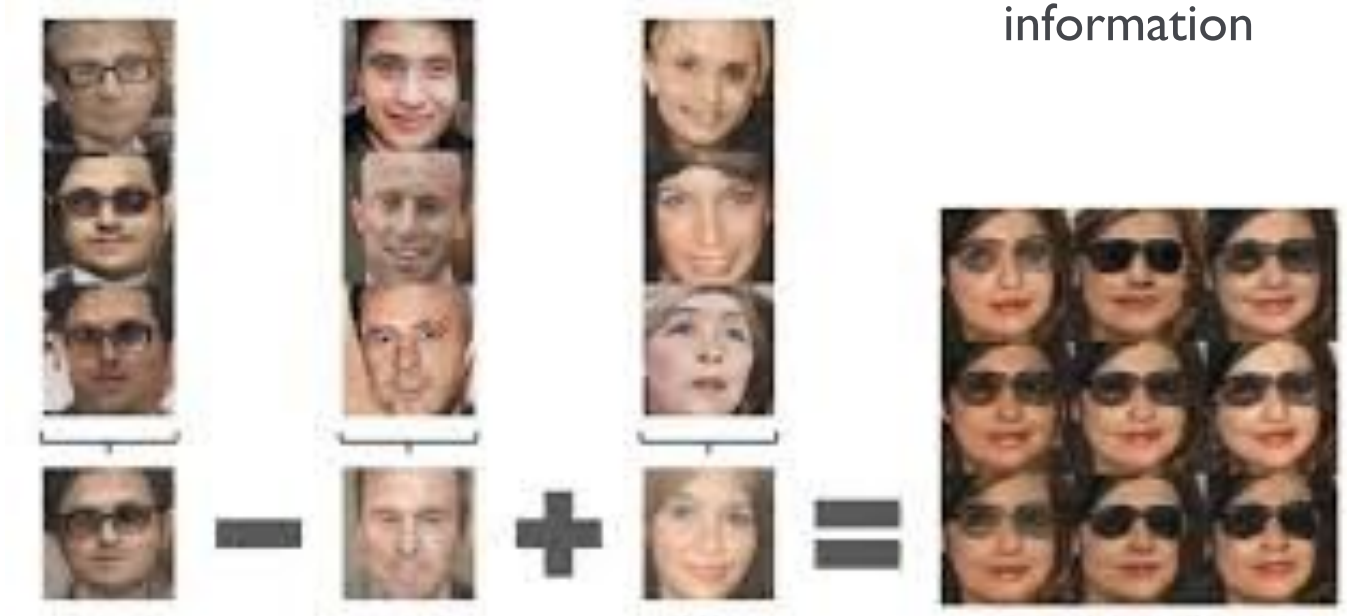
Local optima. Memorize training data

Hard to invert (get latents from training data)

Why GANs are wonderful

GAN representations

GAN input vectors can encode semantic information



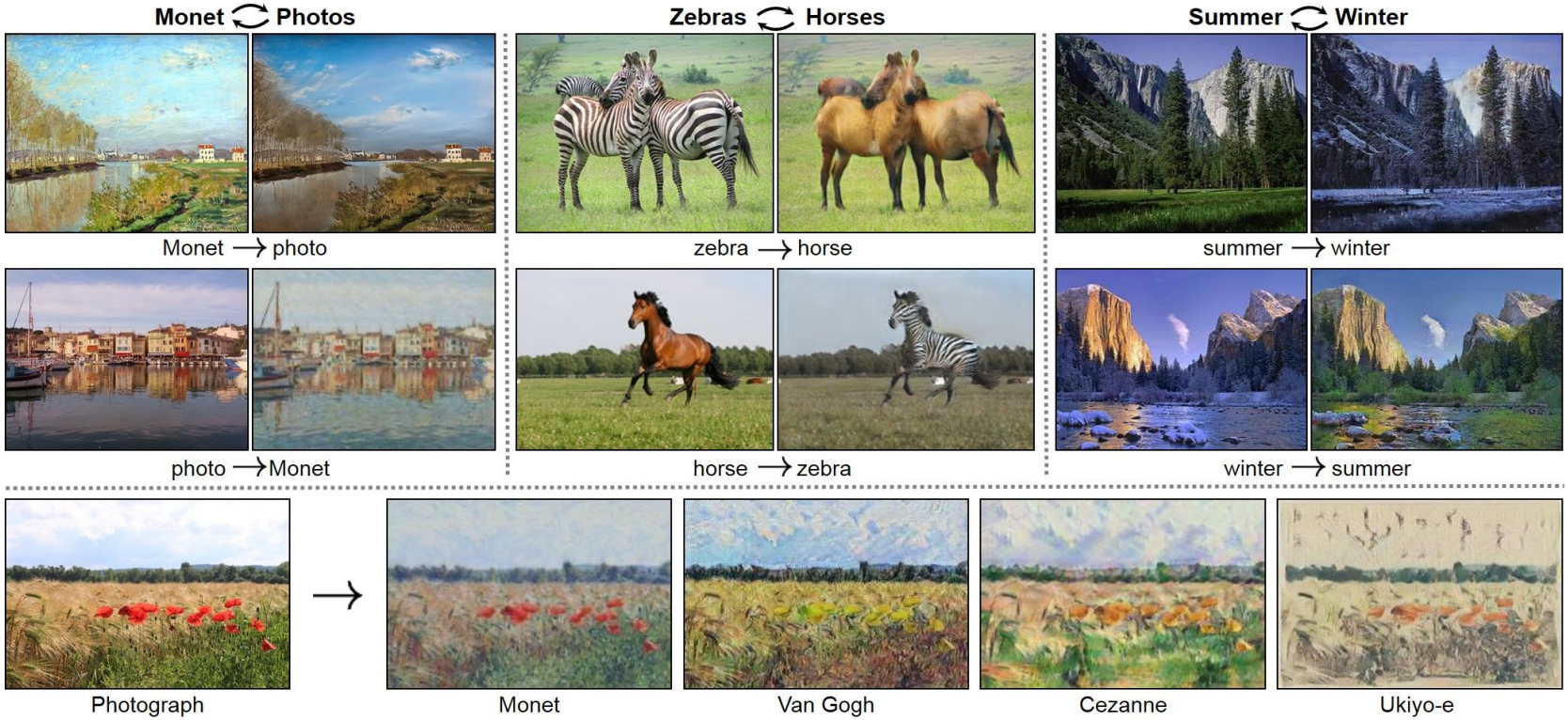
Man with glasses

Man

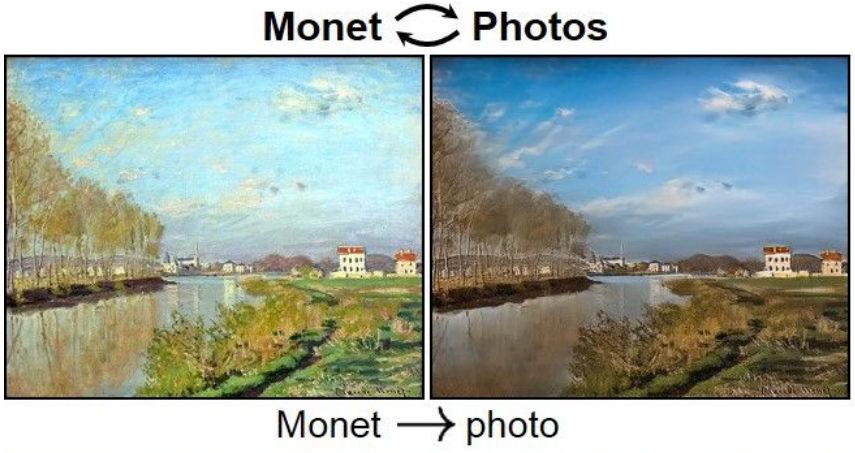
Woman

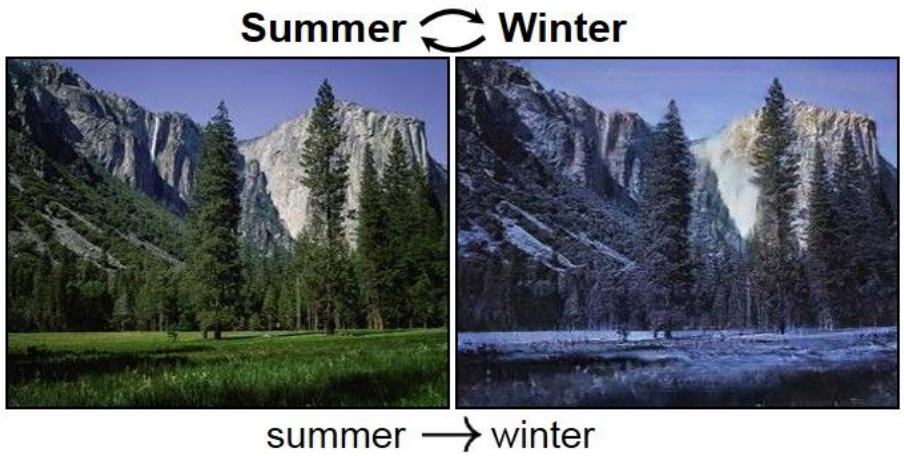
Woman with glasses

CycleGANs

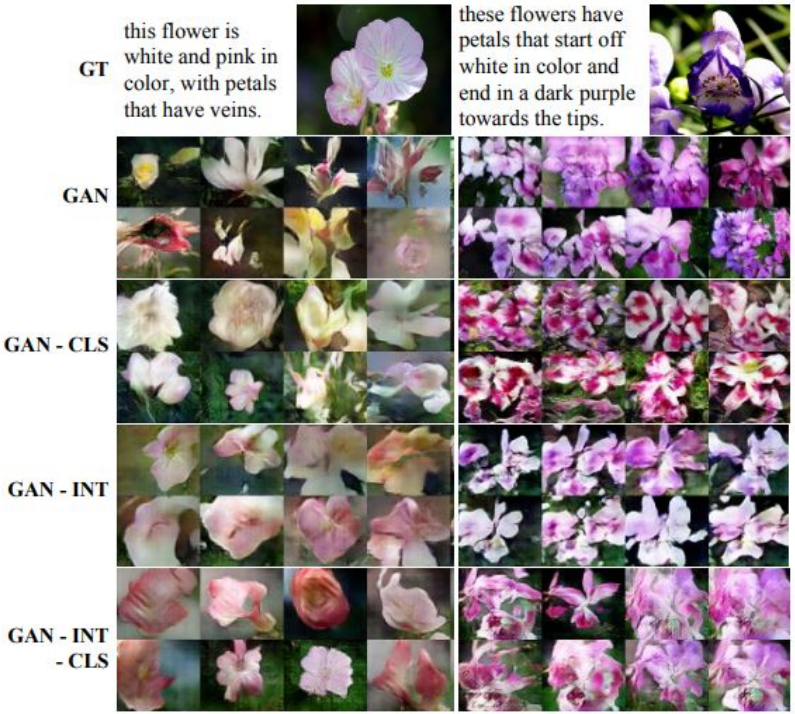

Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

CycleGANs

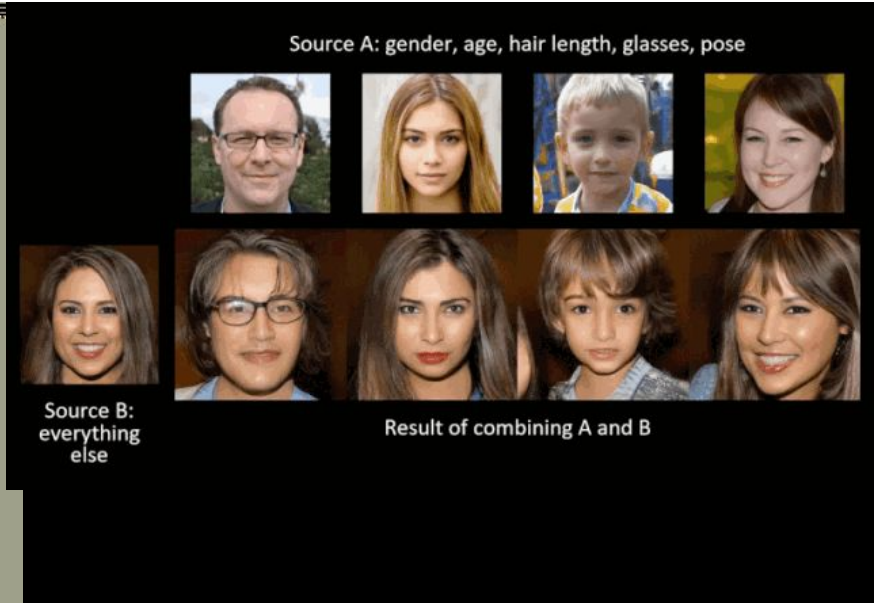




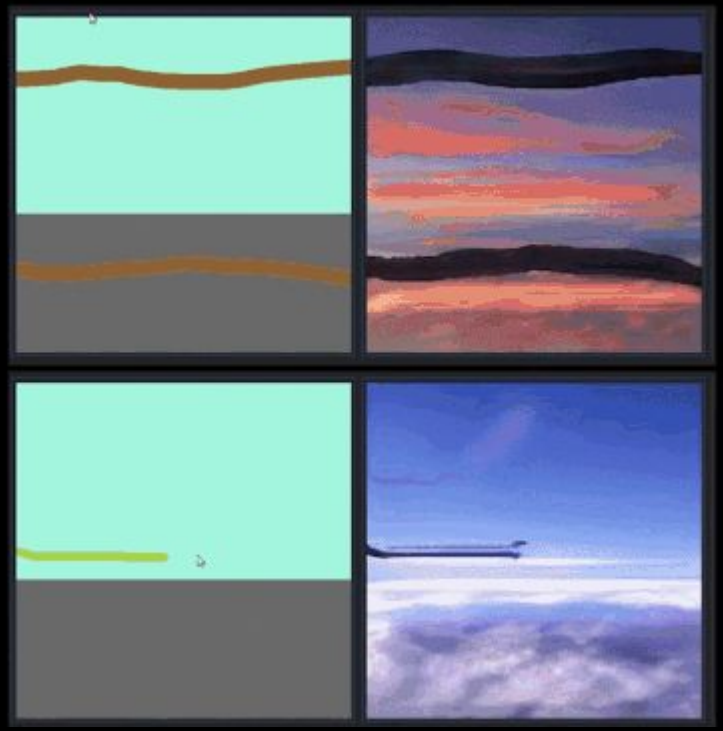
GANs with extra text input



StyleGAN


Source A: gender, age, hair length, glasses, pose
Source B: everything else
Result of combining A and B

GauGAN



So… what are GANs, exactly?

- Generative adversarial networks (GANs) are **generative models** involving two networks: a **generator** and a **discriminator**
- These networks "duel" each other and gradually generate more and more realistic creations

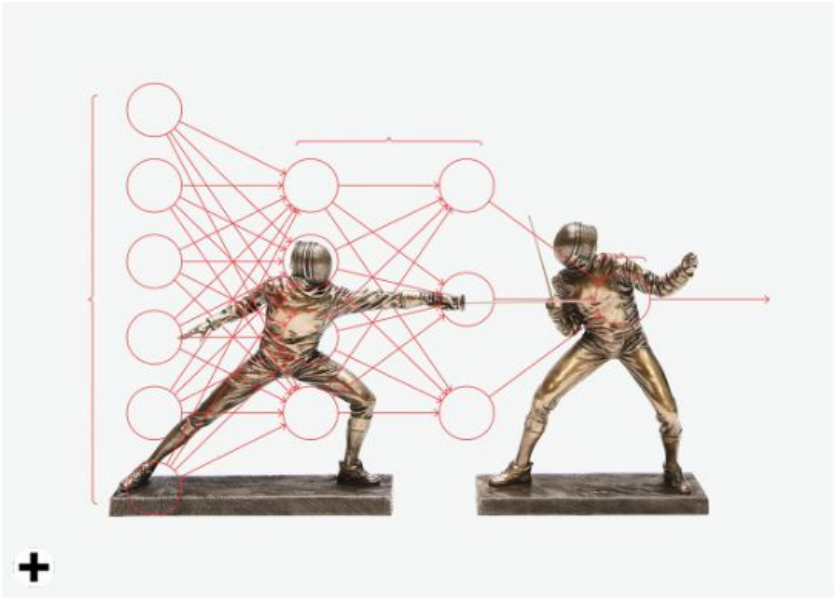


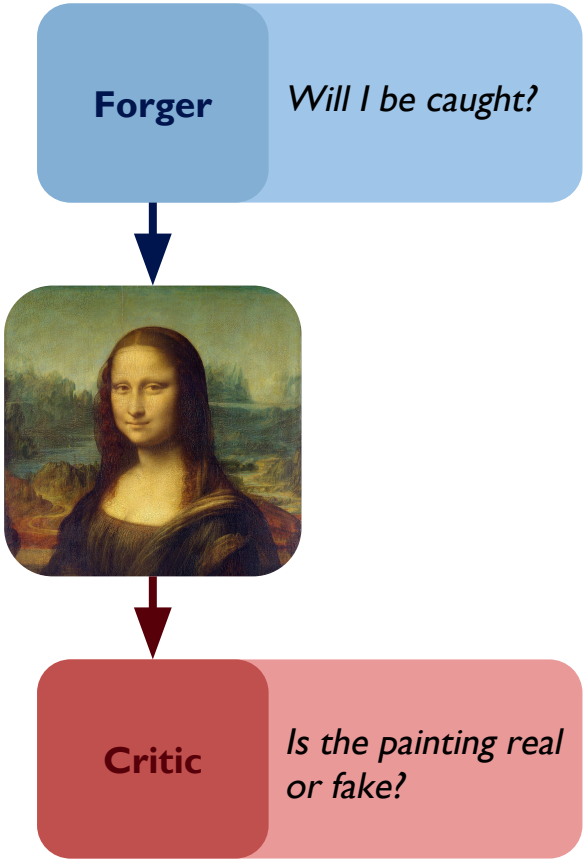
ILLUSTRATION BY DEREK BRAHNEY | DIAGRAM COURTESY OF MICHAEL NIELSEN, "NEURAL NETWORKS AND DEEP LEARNING", DETERMINATION PRESS, 2015

MIT Technology Review

Intuitions

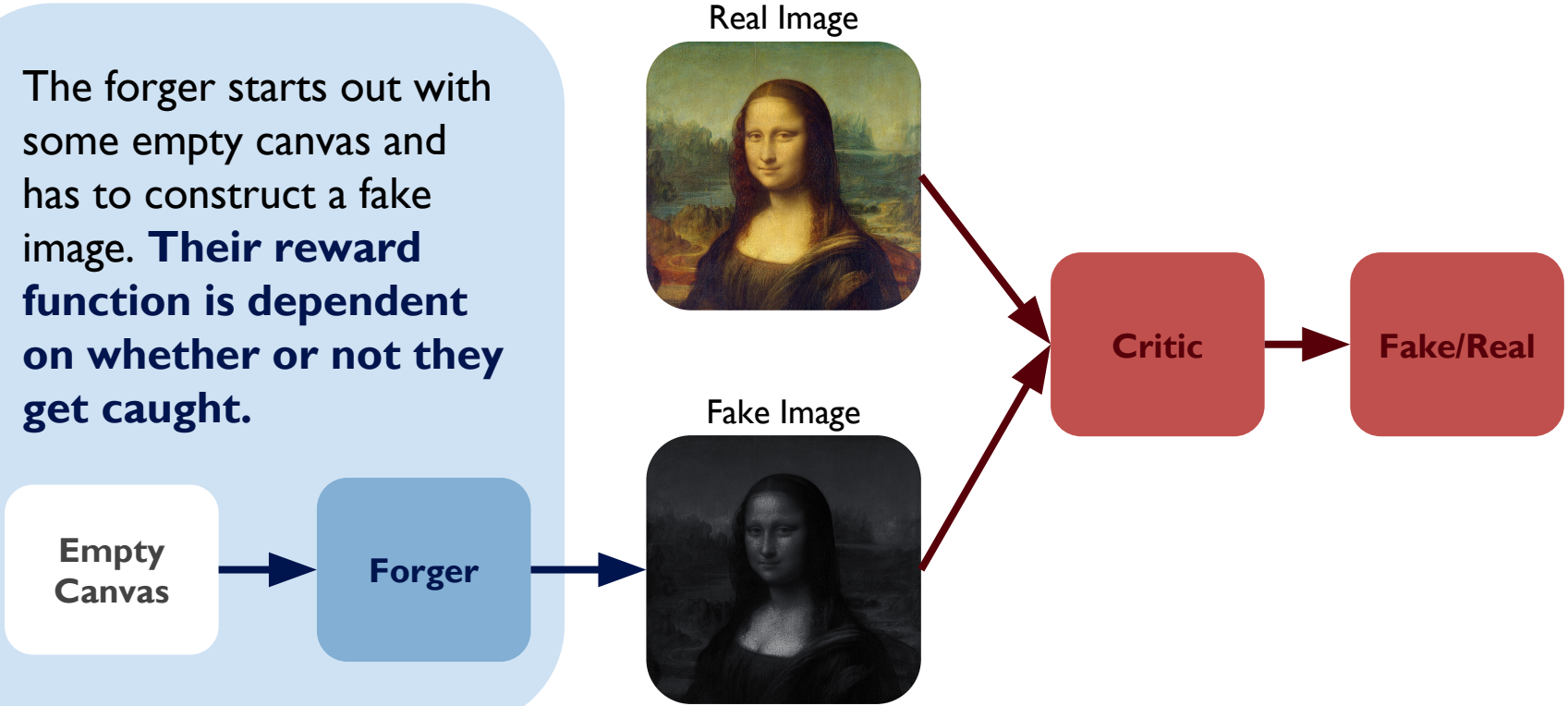
The Forger and the Critic

- Suppose we have two agents, an **art forger** and an **art critic**.
- The forger's job is to make a compelling fake image and the critic's job is to tell whether or not a given painting is a fake
- How might we model this situation?

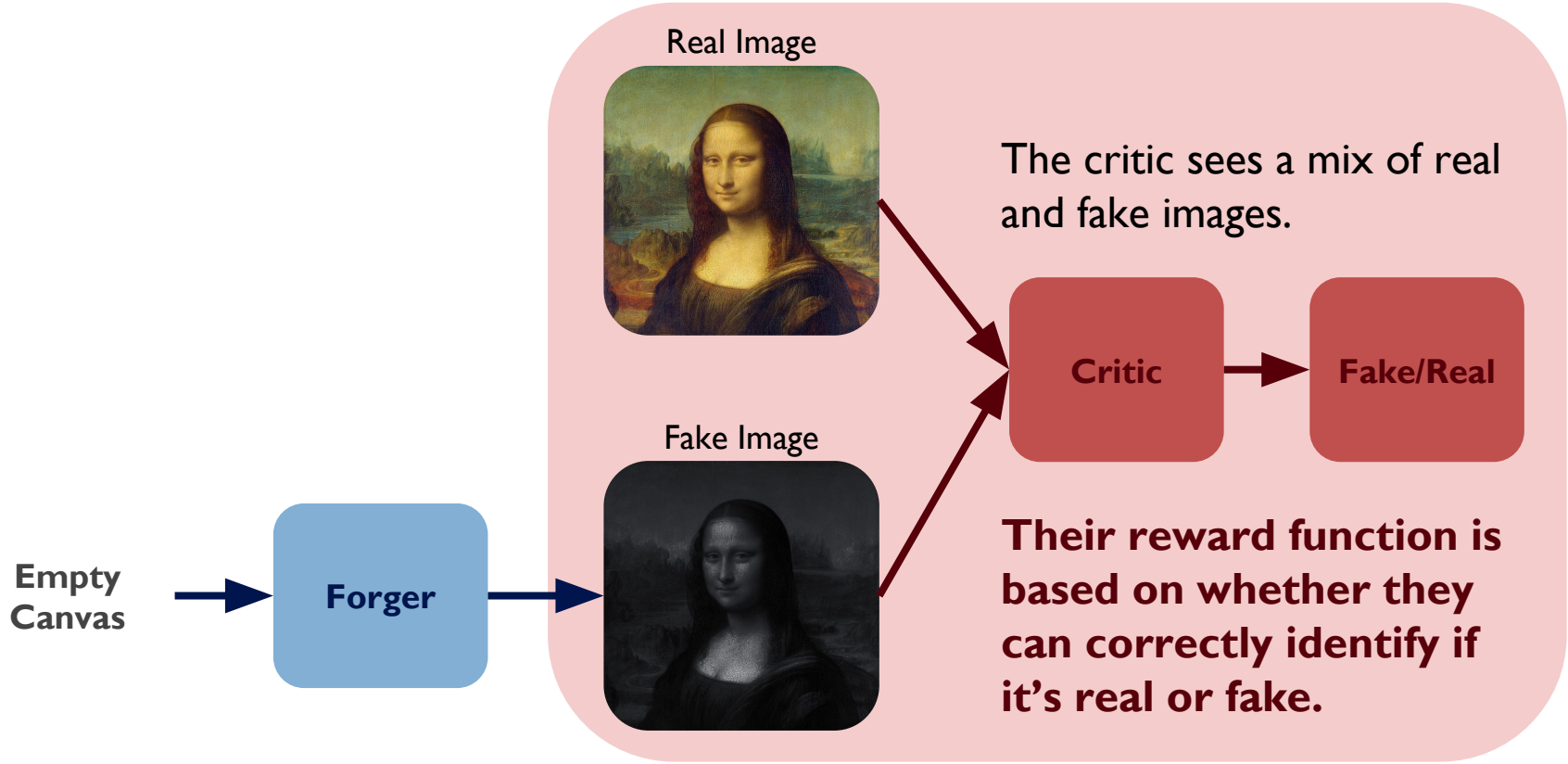


The Forger

The forger starts out with some empty canvas and has to construct a fake image. **Their reward function is dependent on whether or not they get caught.**

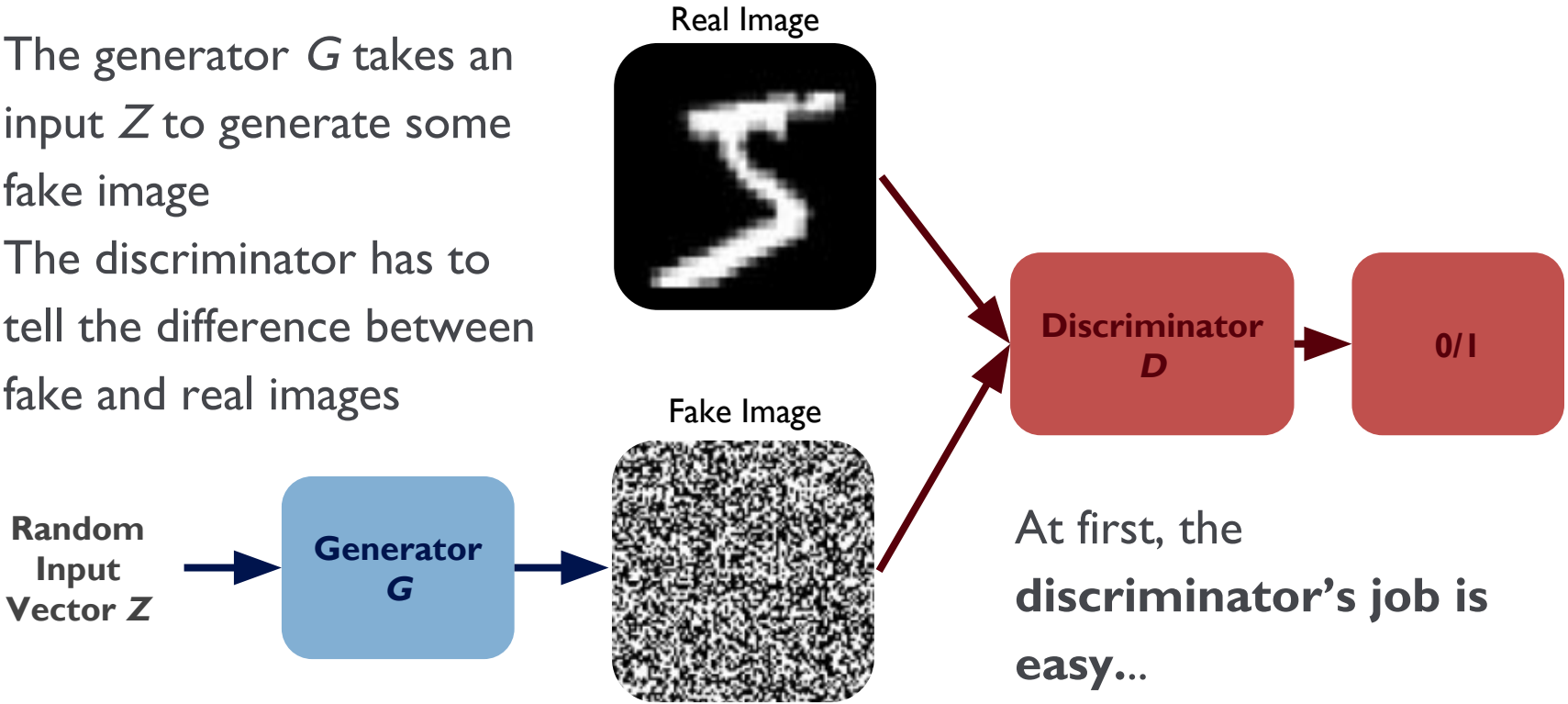


The Critic


Real Image
The critic sees a mix of real and fake images.
Critic
Fake/Real
Fake Image
Empty Canvas
Forger
Their reward function is based on whether they can correctly identify if it's real or fake.

The Model

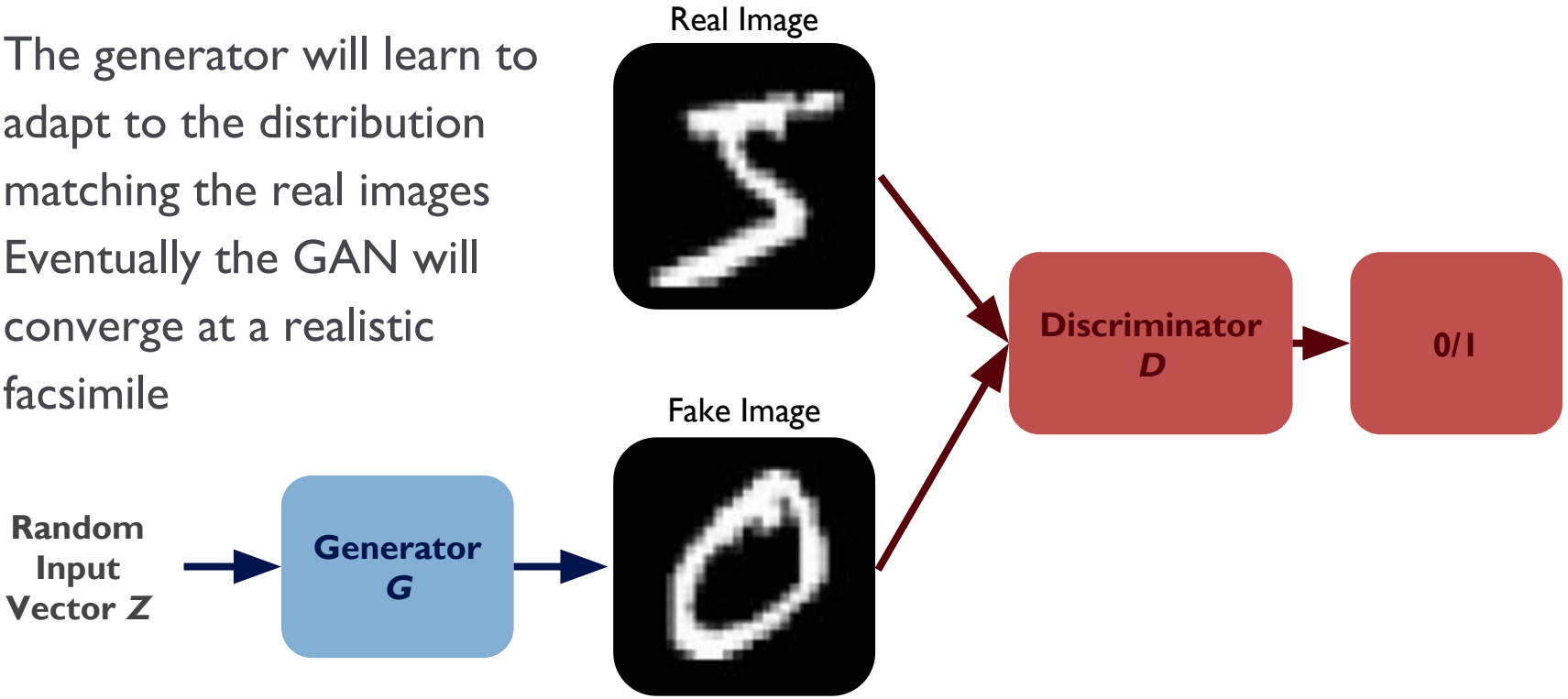
- The generator *G* takes an input *Z* to generate some fake image
- The discriminator has to tell the difference between fake and real images



At first, the **discriminator's job is easy...**

The Model

- The generator will learn to adapt to the distribution matching the real images
- Eventually the GAN will converge at a realistic facsimile



GANs in PyTorch: Architecture

```
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()

        def block(in_feat, out_feat, normalize=True):
            layers = [nn.Linear(in_feat, out_feat)]
            if normalize:
                layers.append(nn.BatchNorm1d(out_feat, 0.8))
            layers.append(nn.LeakyReLU(0.2, inplace=True))
            return layers

        self.model = nn.Sequential(
            *block(opt.latent_dim, 128, normalize=False),
            *block(128, 256),
            *block(256, 512),
            *block(512, 1024),
            nn.Linear(1024, int(np.prod(img_shape))),
            nn.Tanh()
        )

    def forward(self, z):
        img = self.model(z)
        img = img.view(img.size(0), *img_shape)
        return img
```

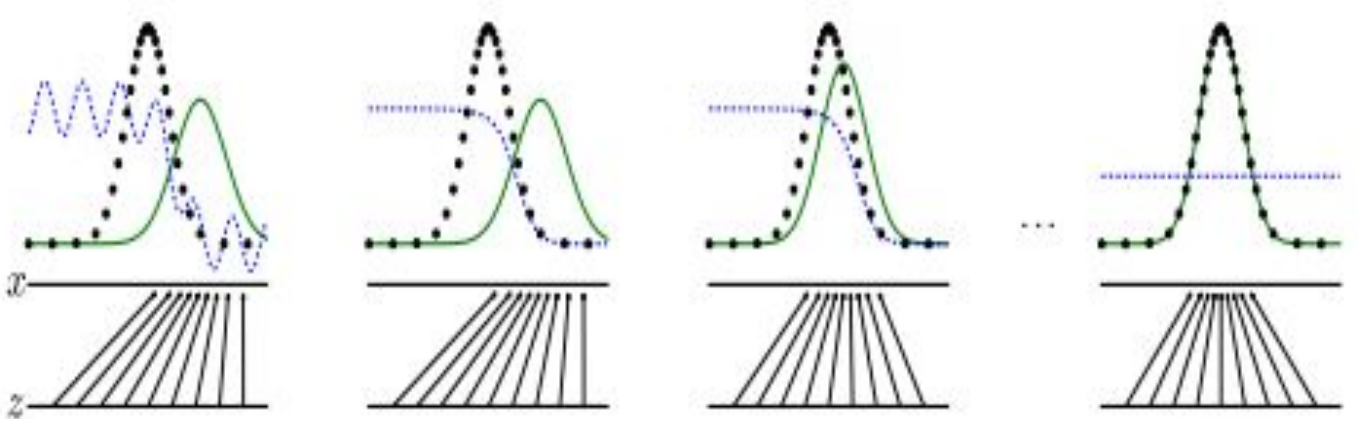
```
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()

        self.model = nn.Sequential(
            nn.Linear(int(np.prod(img_shape)), 512),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(512, 256),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(256, 1),
            nn.Sigmoid()
        )

    def forward(self, img):
        img_flat = img.view(img.size(0), -1)
        validity = self.model(img_flat)

        return validity
```

For more on this, come to recitation! The TAs are lonely :(

GAN Learning idea

Image

Random number



What properties does P_θ need to have?

Describe the manifold on which images lie

(or at least the perceptually salient aspects of it)

Onto the Specifics…

The generator

$z = (0.3, 0.2, -0.6, ...)$ $\xrightarrow{g(z)}$ 

$z \sim \mathcal{N}(0, 1)$

or

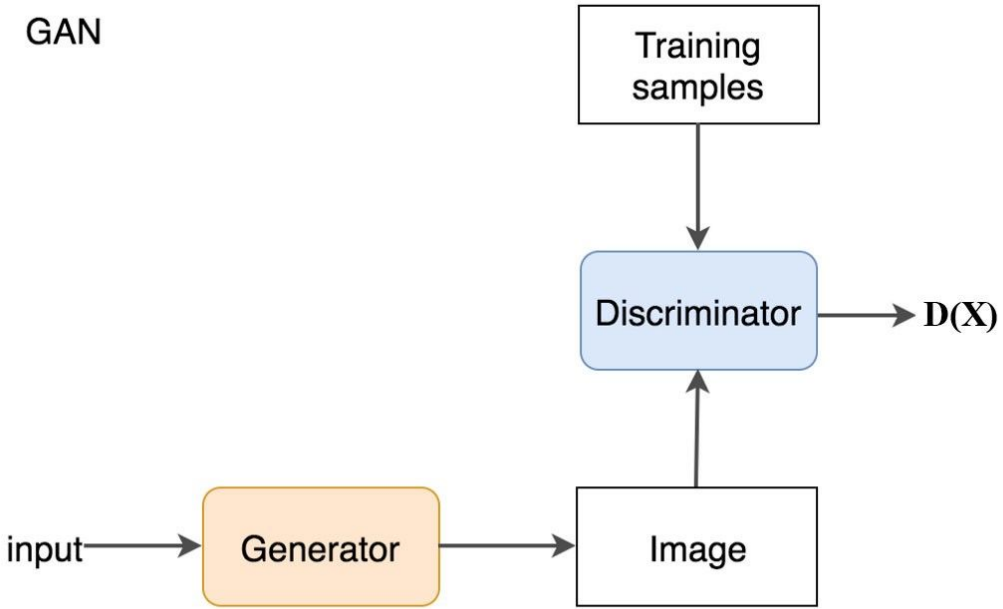
$z \sim U(-1, 1)$

$z = (-0.1, 0.1, 0.2, ...)$ $\xrightarrow{g(z)}$ 

Next 15 slides following Jonathan Hui. Awesome

@ https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b

Putting it together with the discriminator



A competitive game

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

The Discriminator Loss Function

Can *D* tell the difference between what's real and what's fake? Real = 1, Fake = 0

$$J^{(D)} = \underbrace{-\frac{1}{m_{real}}\sum_{i=1}^{m_{real}} y_{real}^{(i)}.\log(D(x^{(i)}))}_{\text{cross-entropy 1: "D should correctly label real data as 1"}} \underbrace{-\frac{1}{m_{gen}}\sum_{i=1}^{m_{gen}}(1-y_{gen}^{(i)}).\log(1-D(G(z^{(i)})))}_{\text{cross-entropy 2: "D should correctly label generated data as 0"}}$$

The optimal discriminator for a given generator

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

The Generator Loss Function

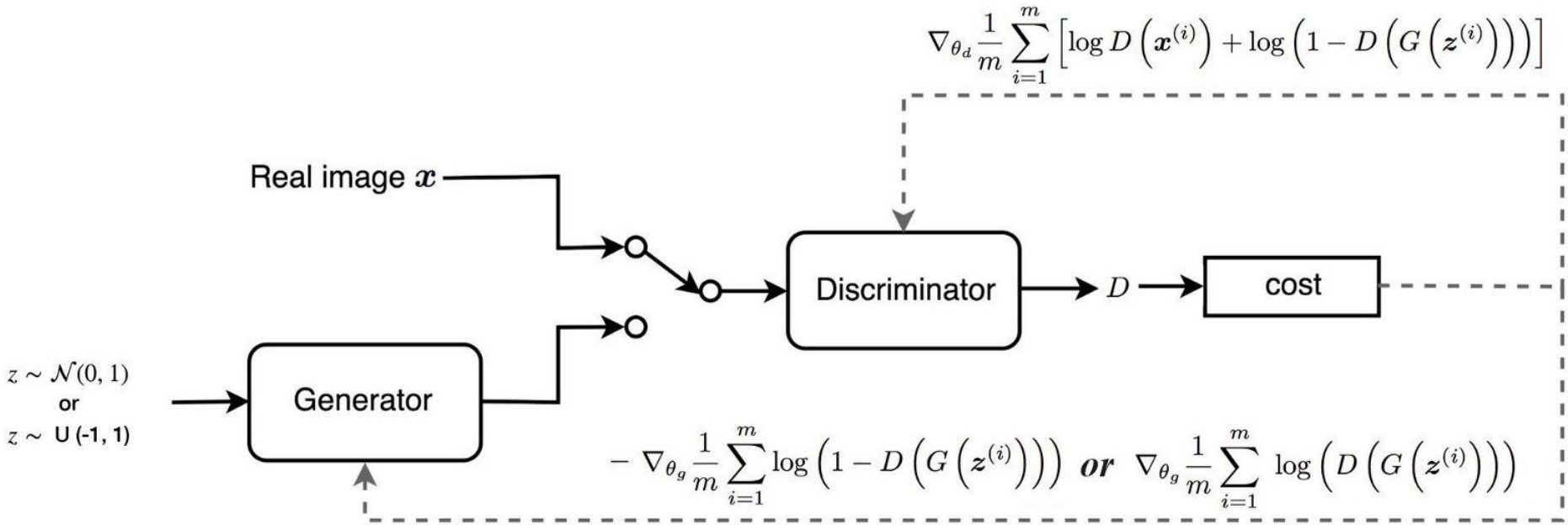
Can *G* avoid getting caught? How well did it do at fooling *D*?

$$J^{(G)} = -J^{(D)} = \frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log(1 - D(G(z^{(i)})))$$

"G should try to fool D: by minimizing the opposite of what D is trying to minimize"

This loss function has problems, though…

Overall setting



Now why is this going to be hard?

Convergence issues

- Vanishing gradients
- Instabilities

Mode collapse

High-d stats is *hard*

Super dependent on hyperparameter settings (and feels very random)

Tips and Tricks: weak evidence!

Normalize data (-1 1)

Tanh function output of generative

Sample from Gaussian not uniform

Develop sorted minibatches (all real or all fake)

Adam in Generator, SGD in Discriminator

DCGAN

Dropout in Generator

Bad Images



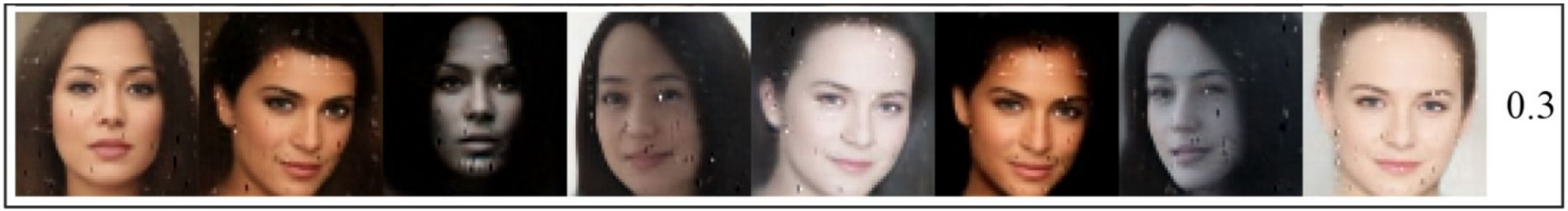




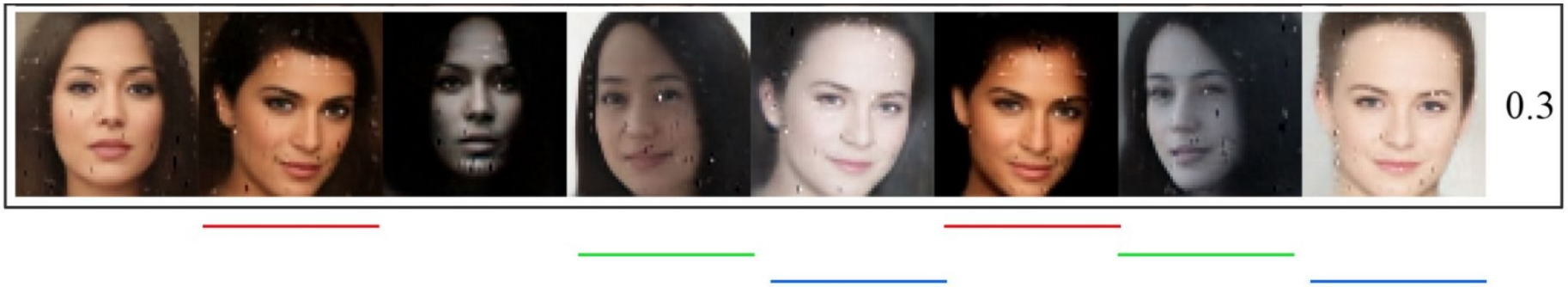




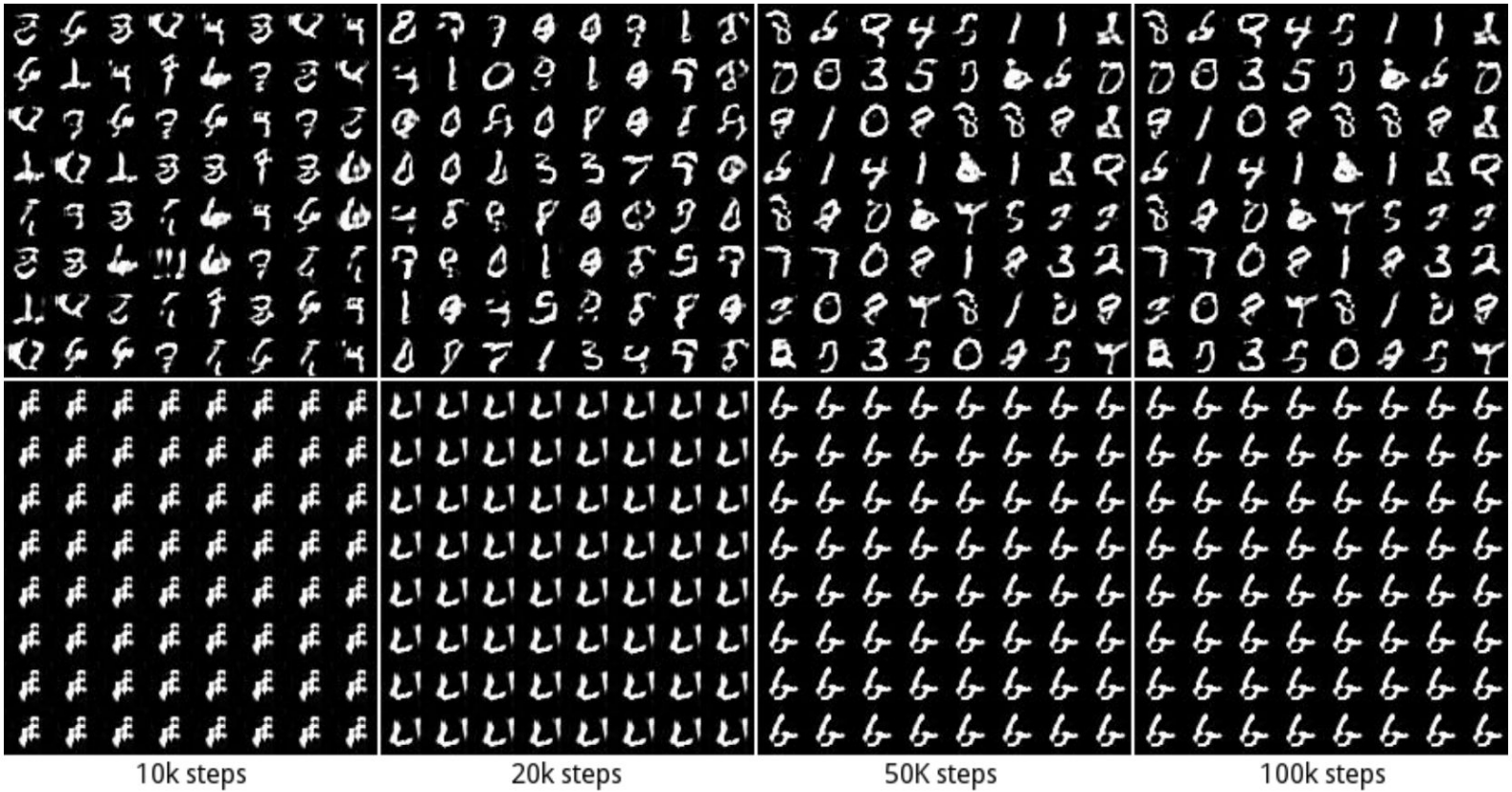

What is going on here?



Mode collapse



Mode collapse



Nash equilibrium

Neither side can improve their outcome by changing its actions

any change in generator makes it more detectable

Any change in discriminator makes things less detectable

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

Lets analyze the KL divergence problem

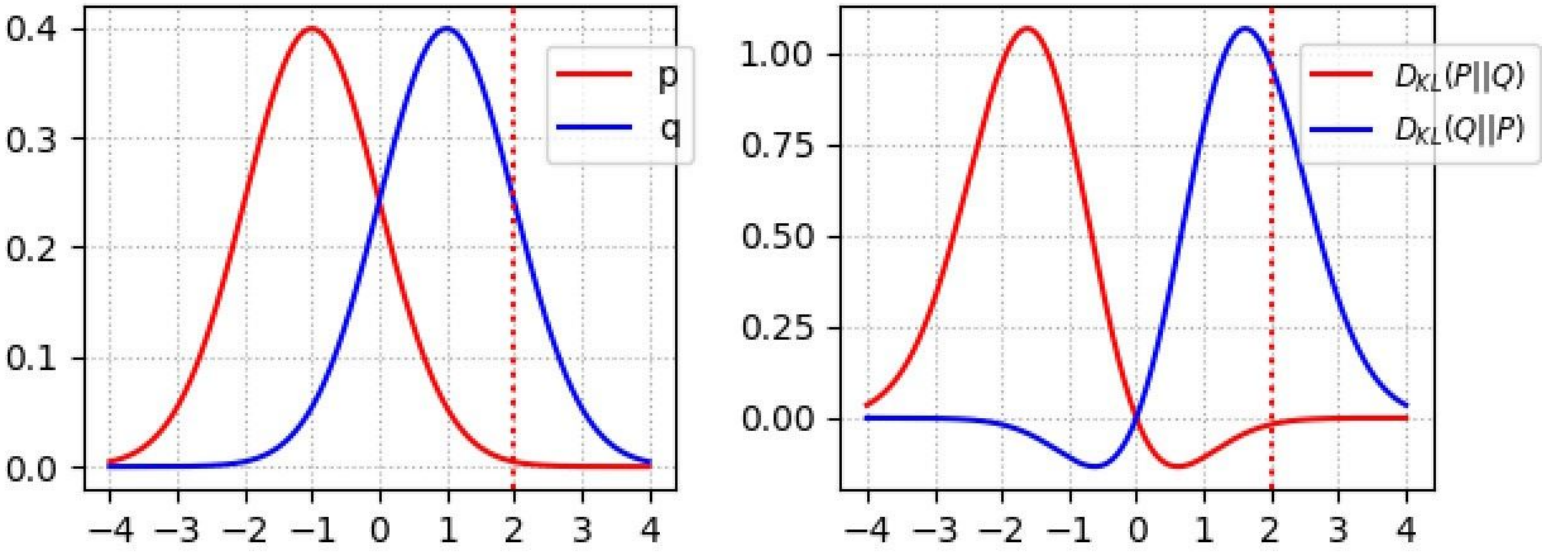
$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{N} p(x_i|\theta)$$

Equivalent to maximizing

$$D_{KL}(p||q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$$

Not symmetrical

$$D_{KL}(p||q) \neq D_{KL}(q||p)$$



DL(P,Q)

Penalizes generator if it misses mode

Penalty high if p(x)>0, q(x)->0

Not so bad if some images don’t look good

DL(Q,P)

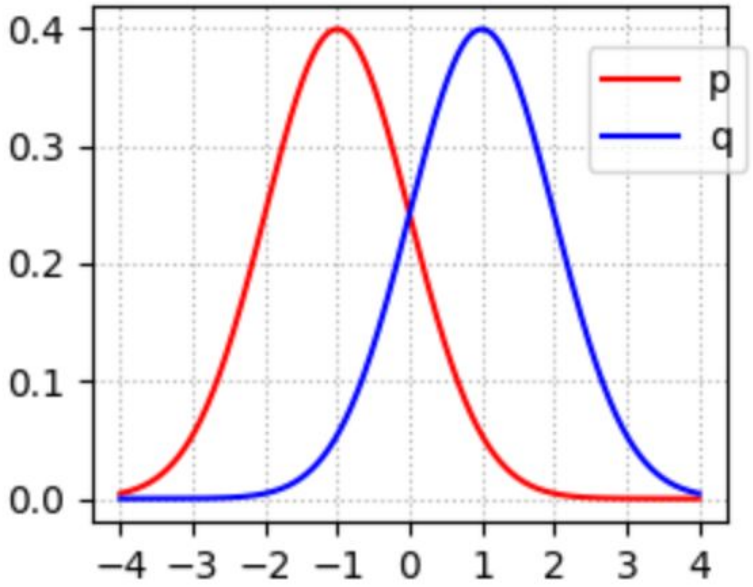
Penalizes generator if the images don't look good

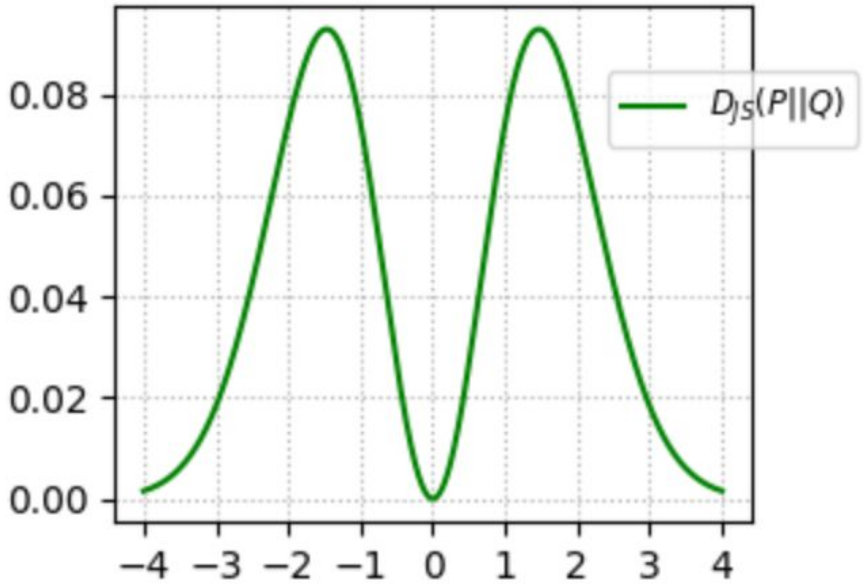
Penalty high if $q(x)>0$, $p(x)->0$

Not so bad if a mode is missed

Jensen Shannon (JS) Divergence

$$D_{JS}(p||q) = \frac{1}{2}D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q||\frac{p+q}{2})$$

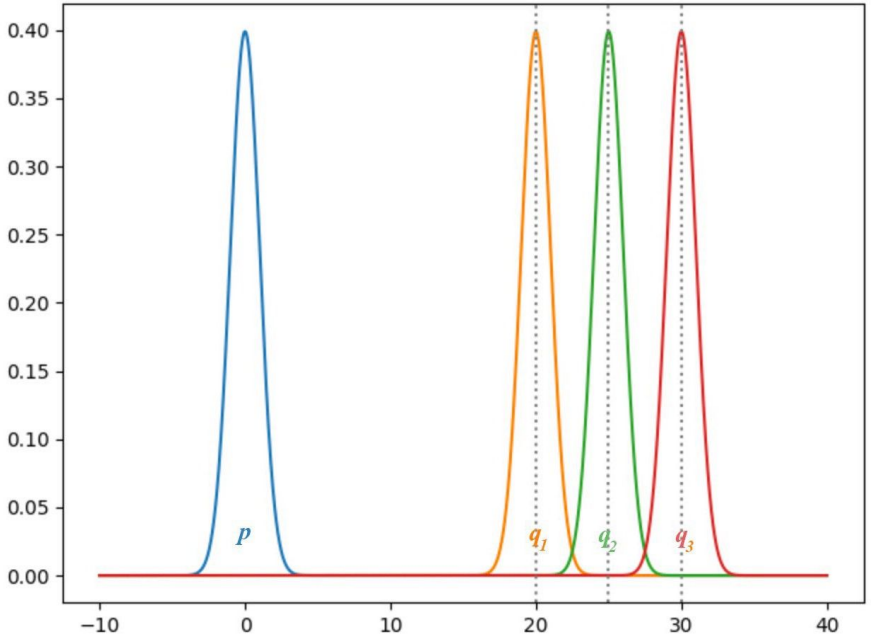




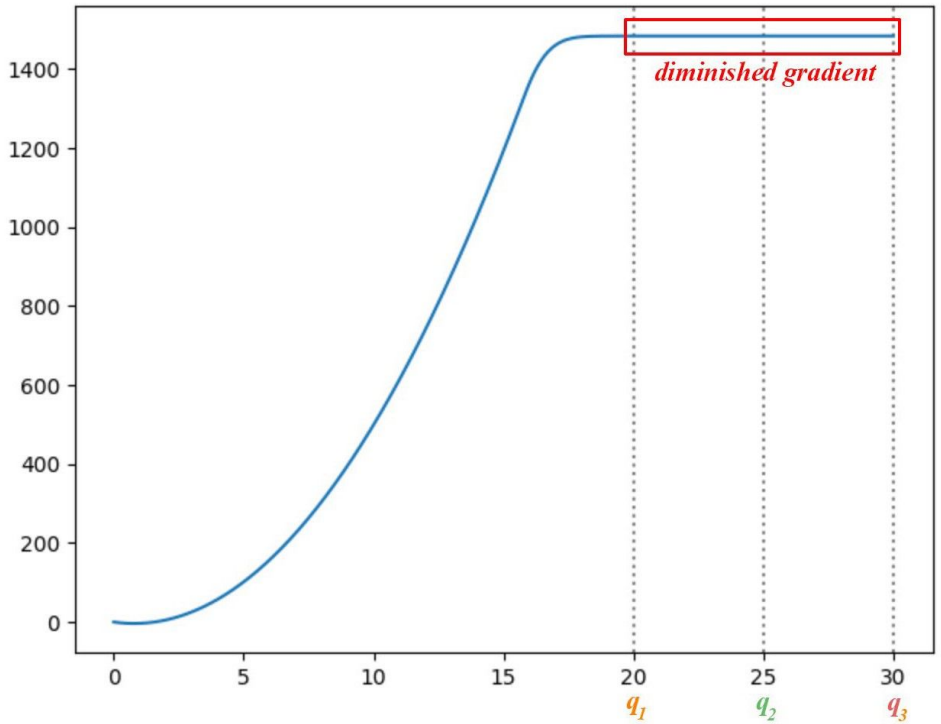
The best generator for the best discriminator

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} [\log(1 - D_G^*(G(\boldsymbol{z})))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \right] + \mathbb{E}_{\boldsymbol{x} \sim p_g} \left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \right]
\end{aligned}
$$

Vanishing gradients



Gradient vanishes



Replace cost function to make it less bad

$$\nabla_{\theta_g} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \rightarrow 0 \quad \textit{change to} \quad \nabla_{\theta_g} - \log\left(D\left(G\left(z^{(i)}\right)\right)\right)$$

$$\mathbb{E}_{z\sim p(z)}\left[-\nabla_\theta \log D^*(g_\theta(z))|_{\theta=\theta_0}\right] = \nabla_\theta\left[KL(\mathbb{P}_{g_\theta}\|\mathbb{P}_r) - 2JSD(\mathbb{P}_{g_\theta}\|\mathbb{P}_r)\right]|_{\theta=\theta_0}$$

Has anti KL term! - more quality, less diversity!

The math is just a thin veneer to hide what we do not understand

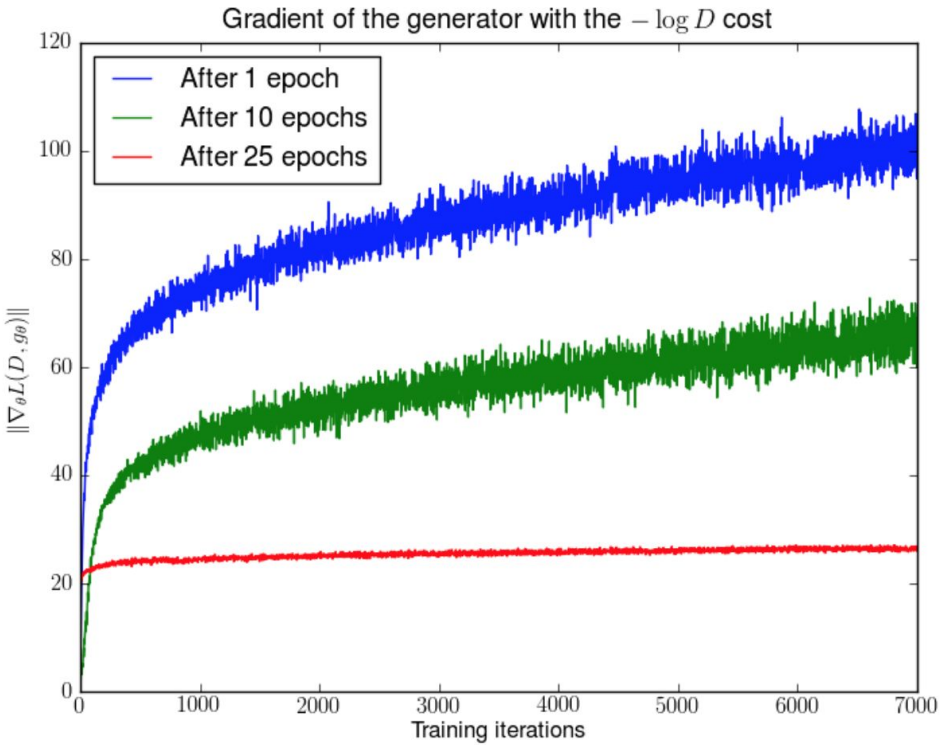
But maybe read

On How Well Generative Adversarial Networks Learn Densities:
Nonparametric and Parametric Results

Tengyuan Liang*[1]

[1]University of Chicago, Booth School of Business

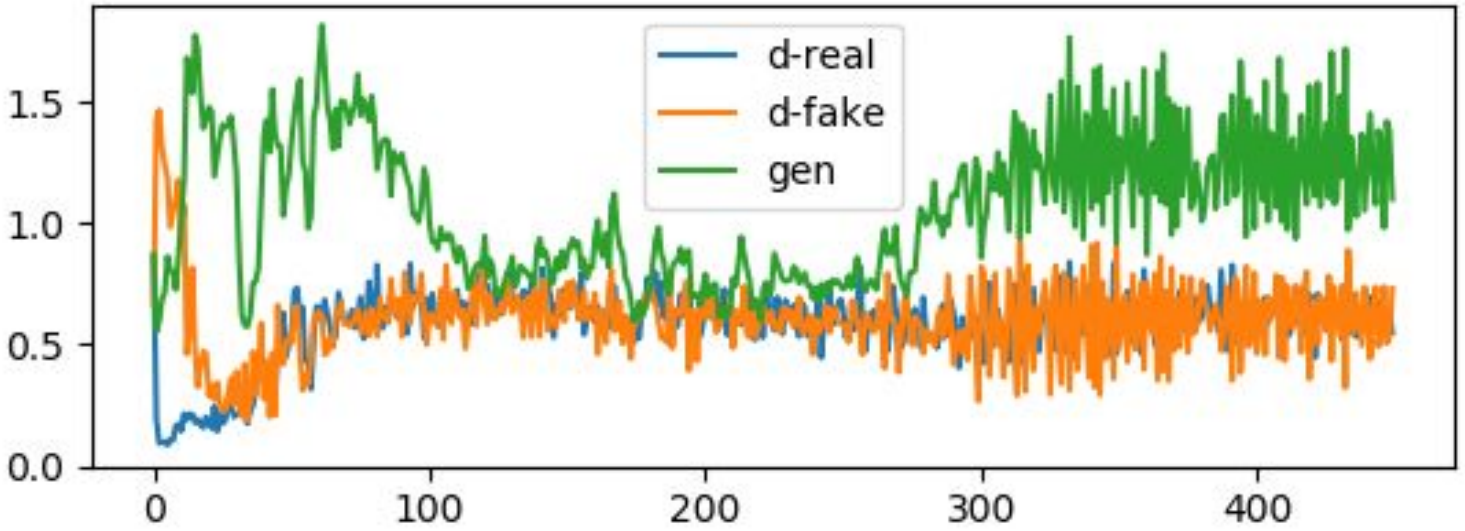
What happens if we freeze the generator?



The longer we train,
The quicker the D gradient explodes!

Unstable!

GAN Training



Making GANs better

Many ideas

GAN Type	Key Take-Away
GAN	The original (JSD divergence)
WGAN	EM distance objective
Improved WGAN	No weight clipping on WGAN
LSGAN	L2 loss objective
RWGAN	Relaxed WGAN framework
McGAN	Mean/covariance minimization objective
GMMN	Maximum mean discrepancy objective
MMD GAN	Adversarial kernel to GMMN
Cramer GAN	Cramer distance
Fisher GAN	Chi-square objective
EBGAN	Autoencoder instead of discriminator
BEGAN	WGAN and EBGAN merged objectives
MAGAN	Dynamic margin on hinge loss from EBGAN

https://towardsdatascience.com/gan-objective-functions-gans-and-their-variations-ad77340bce3c

Intuition earth mover's distance

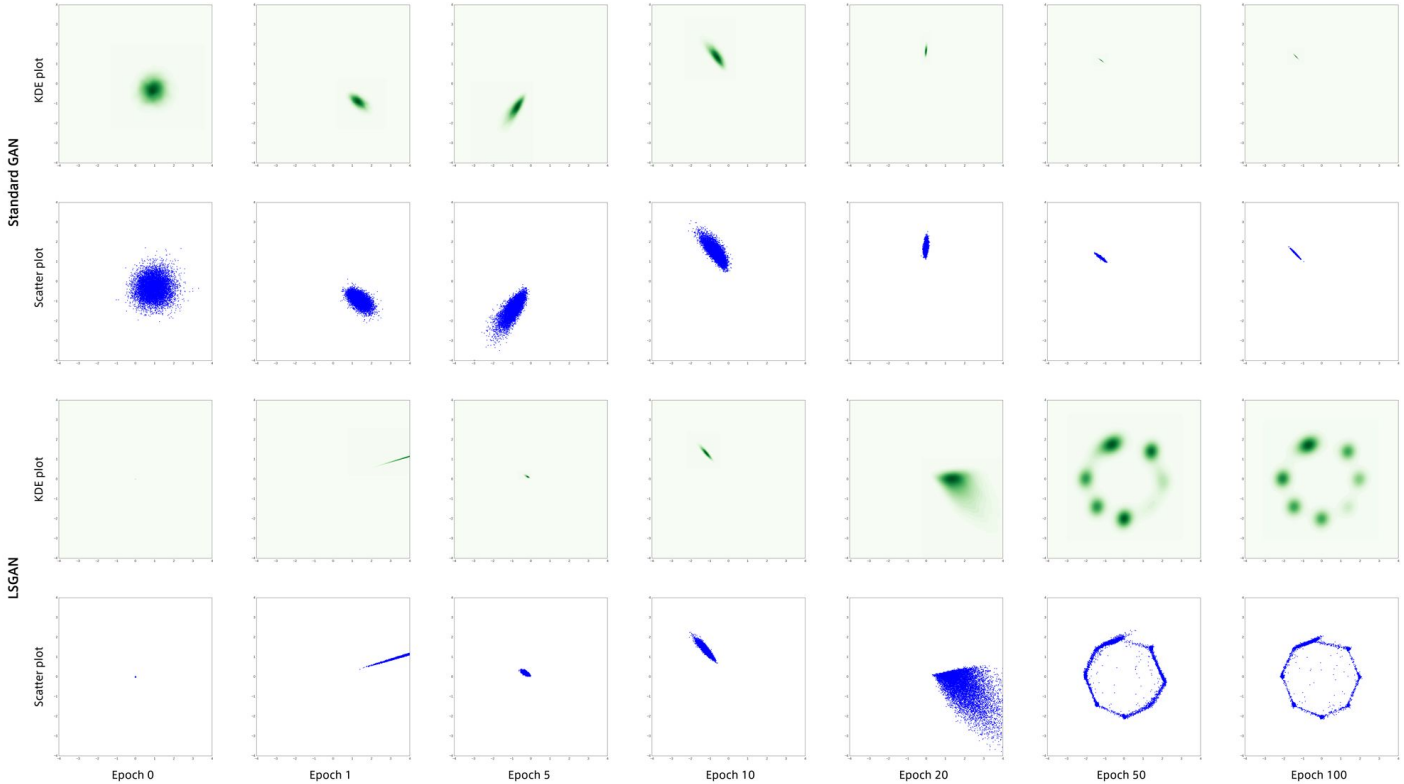
Wasserstein GANs

Turn off weight clipping

DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Baseline (*G*: DCGAN, *D*: DCGAN)			




L2 loss (instead of JSD)



Did they succeed?

Did anyone find the magical well-working formulation?

Measuring how good a GAN does

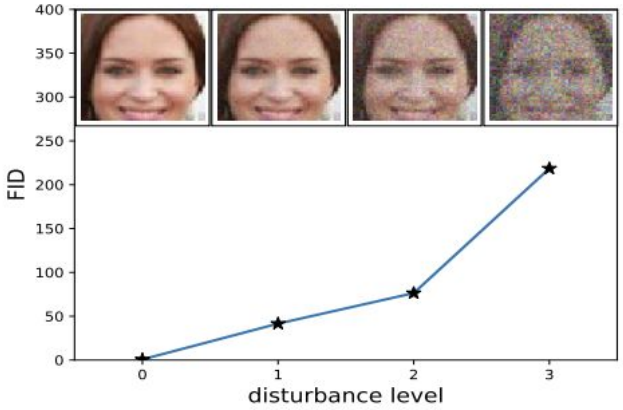
Take inception network

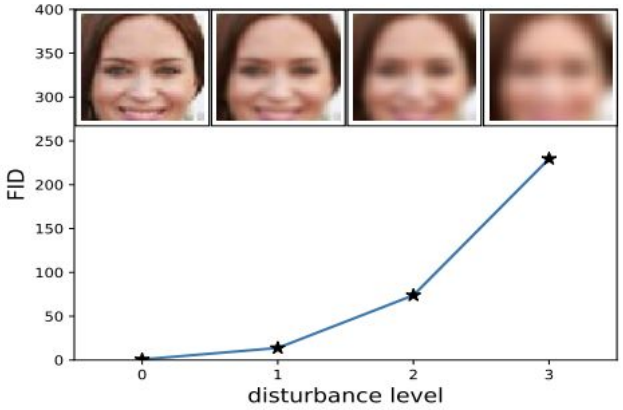
Use FC 2048 width layer

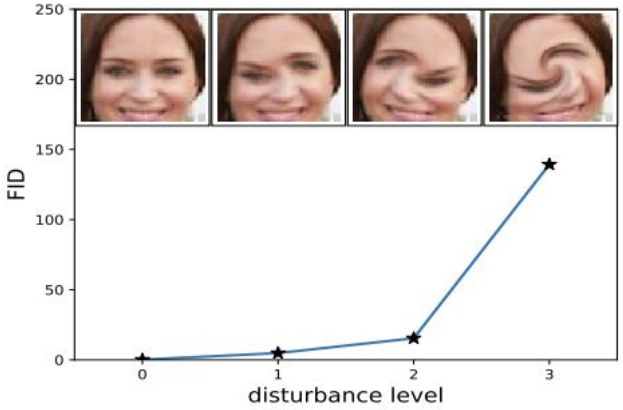
Probability distributions should be matched

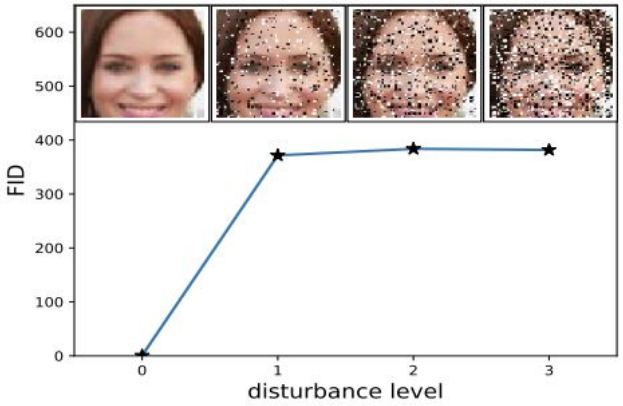
$$\text{FID} = ||\mu_r - \mu_g||^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$$

Kinda makes sense

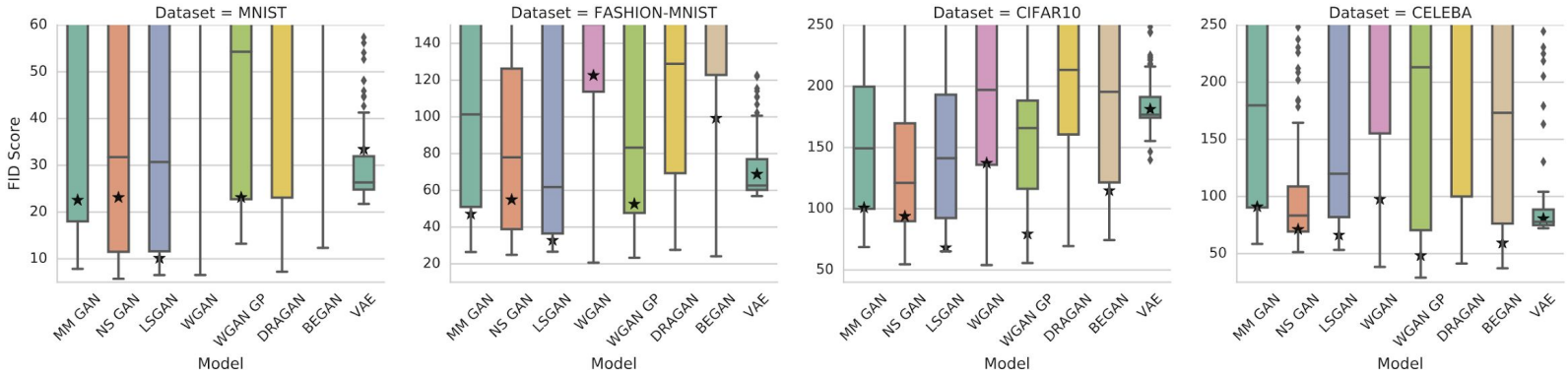




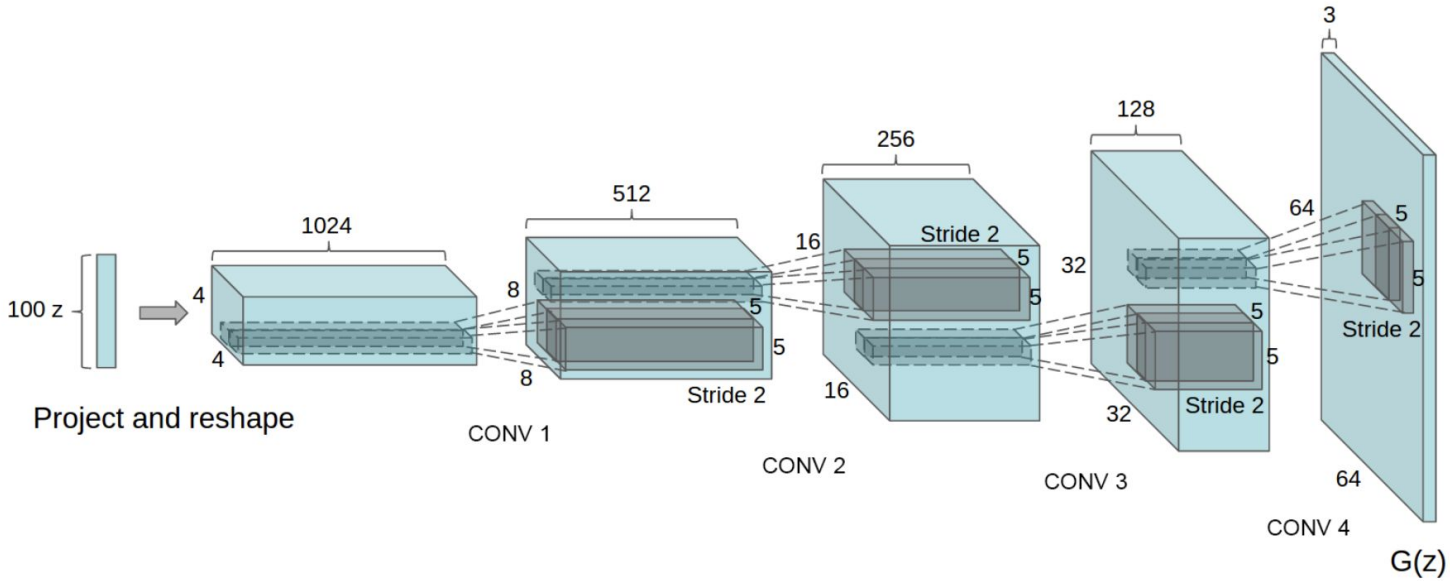




Surprisingly all our innovations seem pretty useless



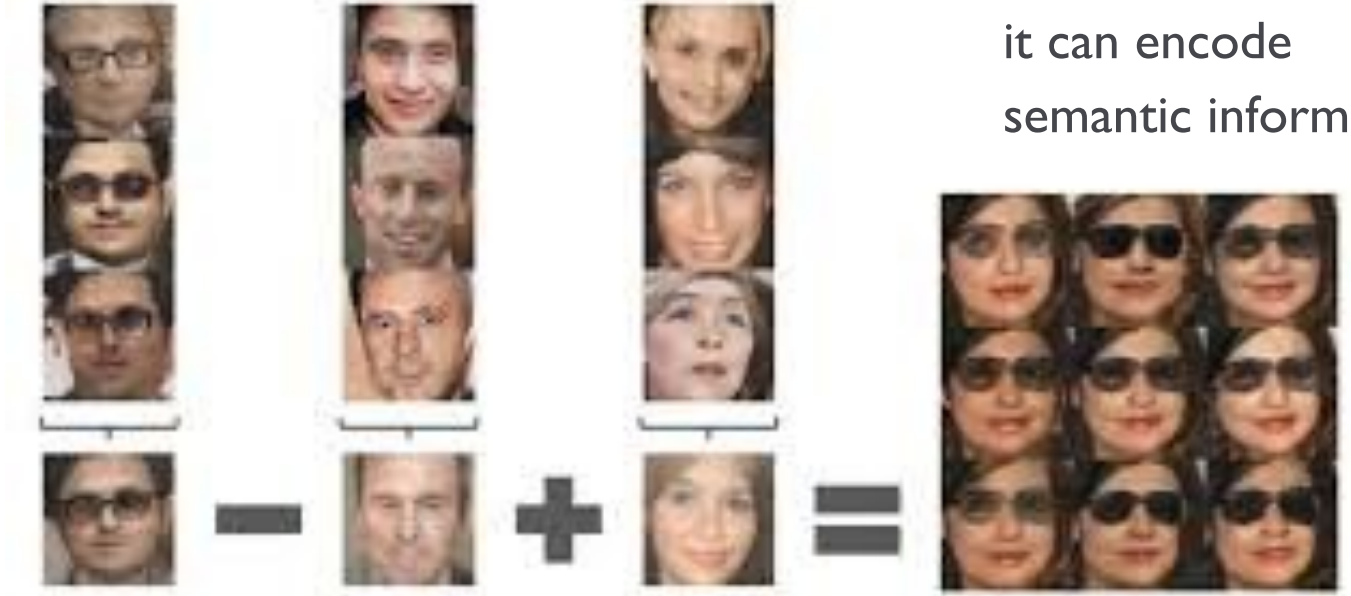
Networks behind successful GANs



DCGAN

GAN representations

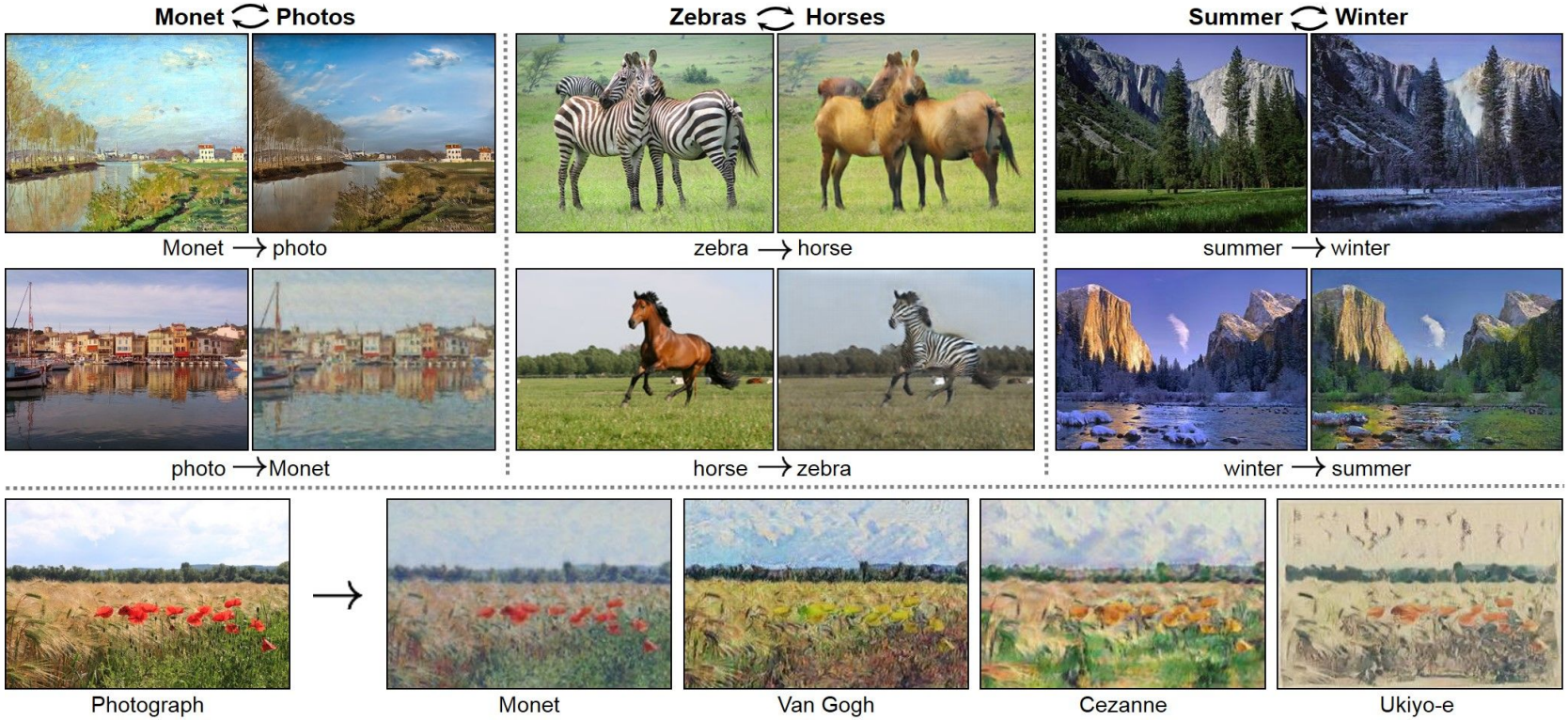
Remember that input vector Z? Turns out it can encode semantic information



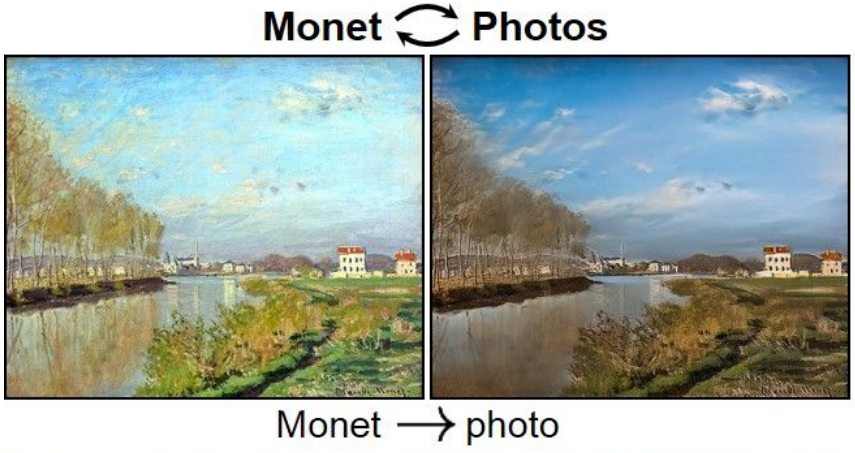
Man with glasses − Man + Woman = Woman with glasses

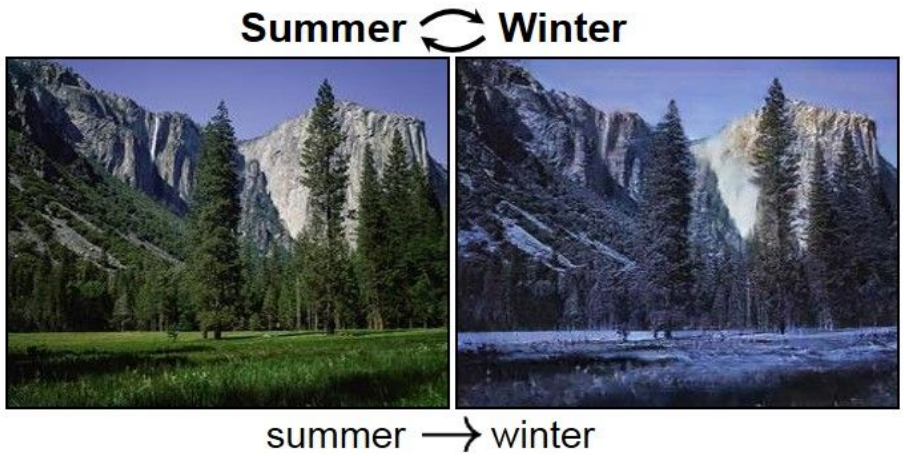
Radford et al 2015

CycleGANs

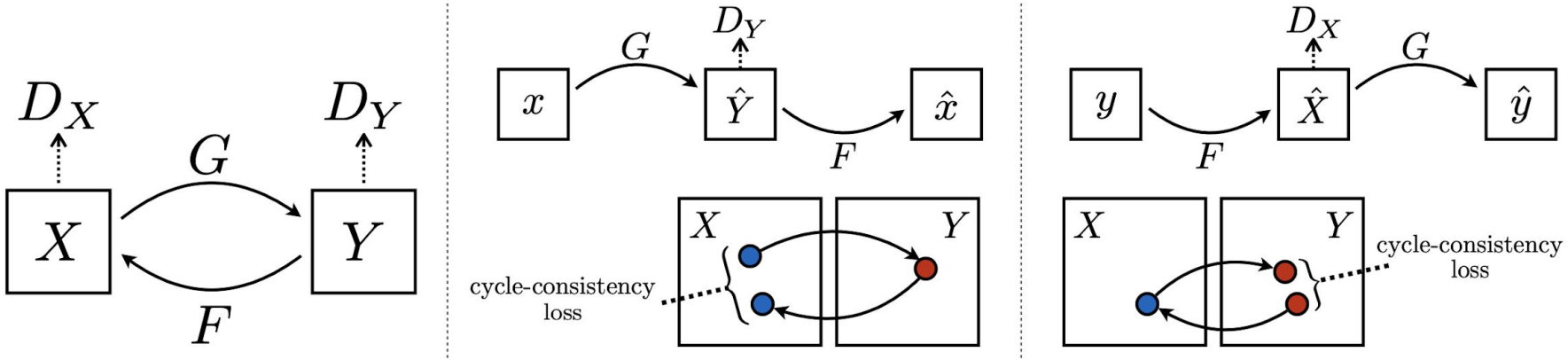

Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

CycleGANs

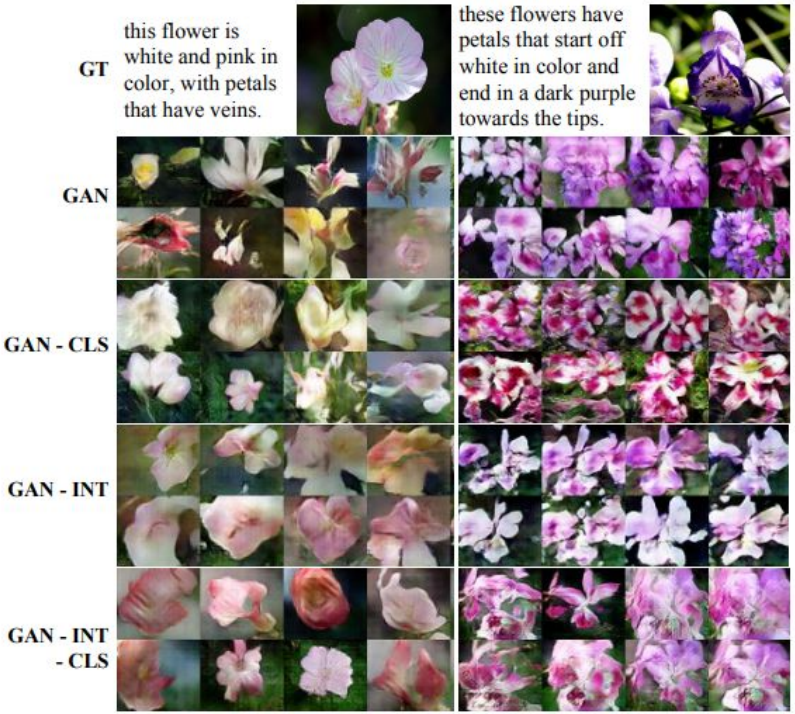




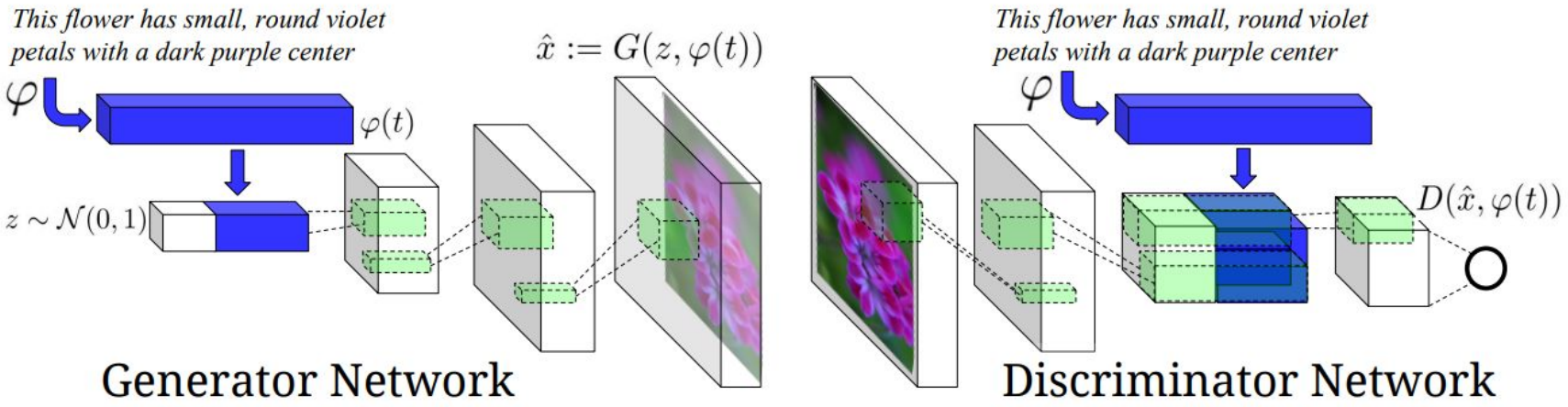
CycleGan Logic



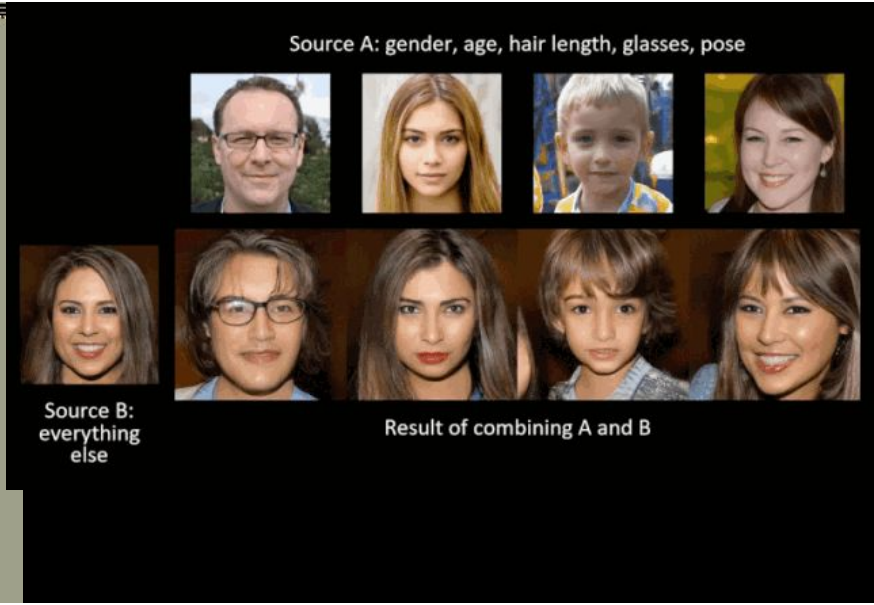
GANs with extra text input



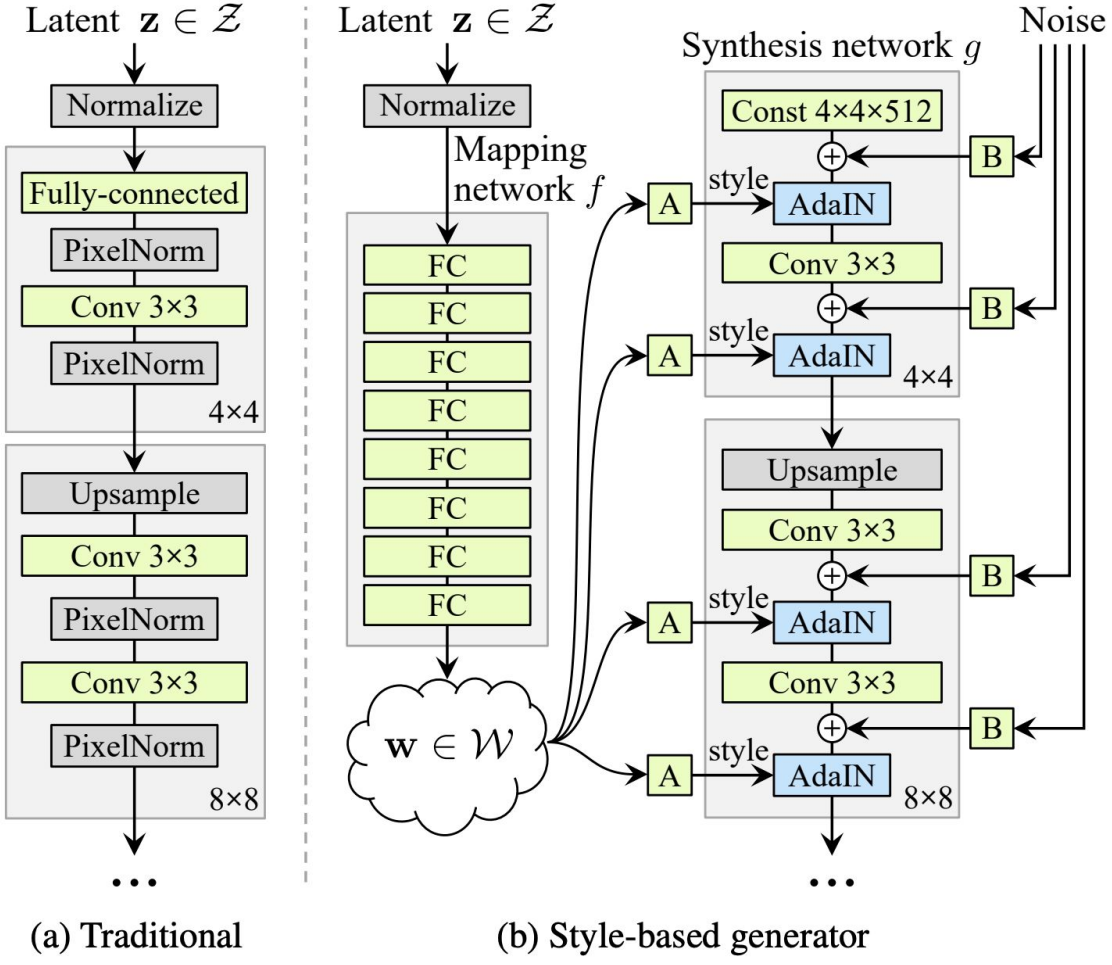
GANs with extra text input



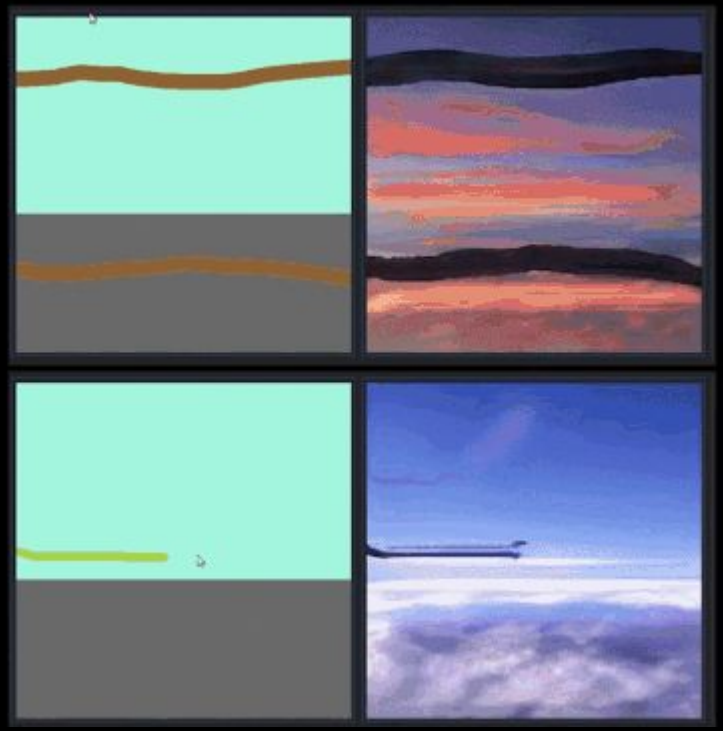
StyleGAN


Source A: gender, age, hair length, glasses, pose
Source B: everything else
Result of combining A and B

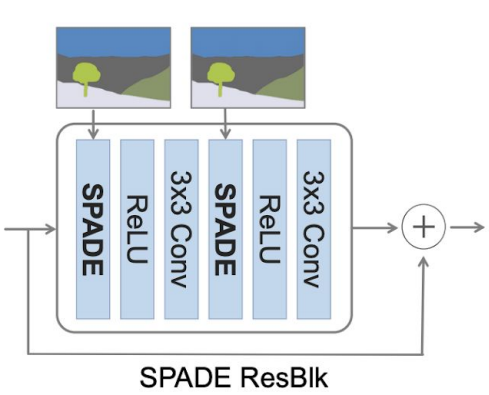
Style based architecture

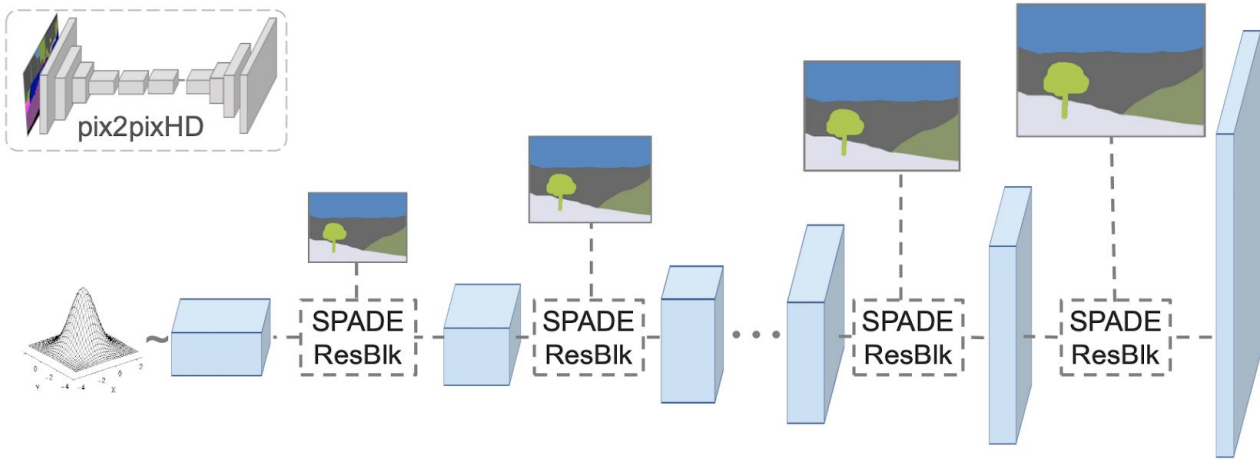


GauGAN



GauGAN structure





Trained with segmented images

This is great! What could *possibly* go wrong? (Reprise)

Ethics: What if we're fooling ourselves?

"This is a big deal," Hany Farid, computer science professor at Dartmouth College, told The Wall Street Journal. "You can literally put into a person's mouth anything you want."

Reddit bans 'deepfakes,' pornography using the faces of celebrities such as Taylor Swift and Gal Gadot

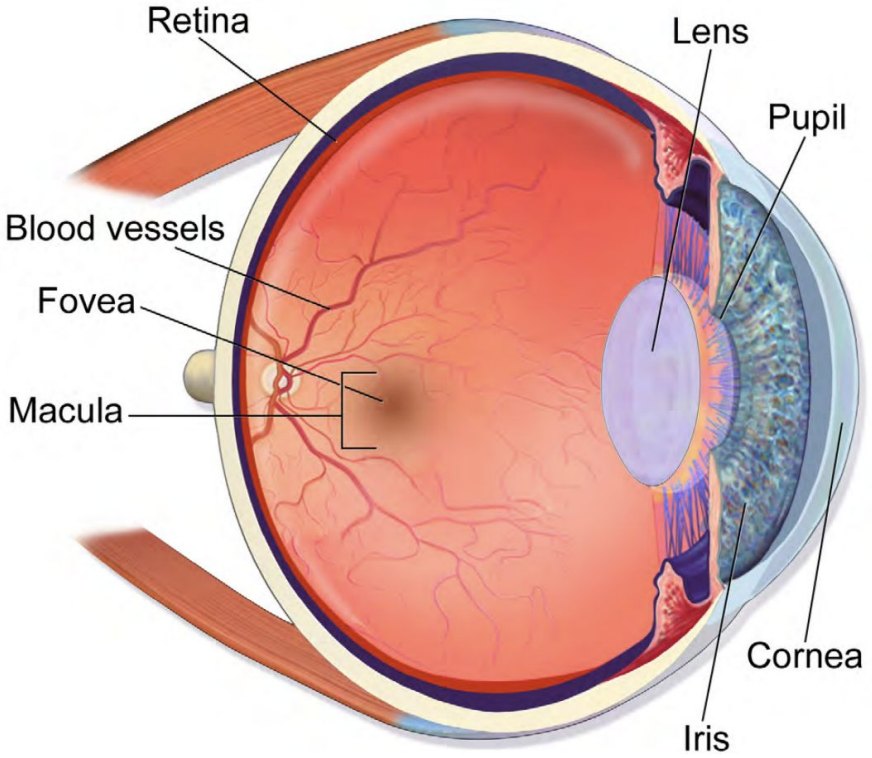
Caveat

GANs make nice images

GANs do not faithfully model high dimensional images

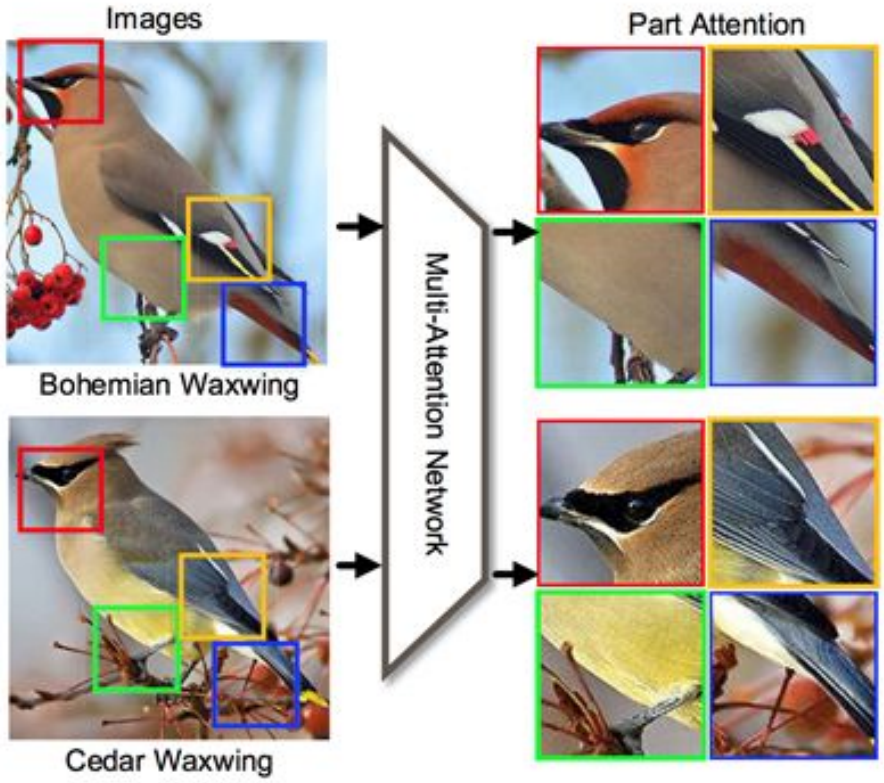
And they most certainly do not faithfully model probability distributions

Attention



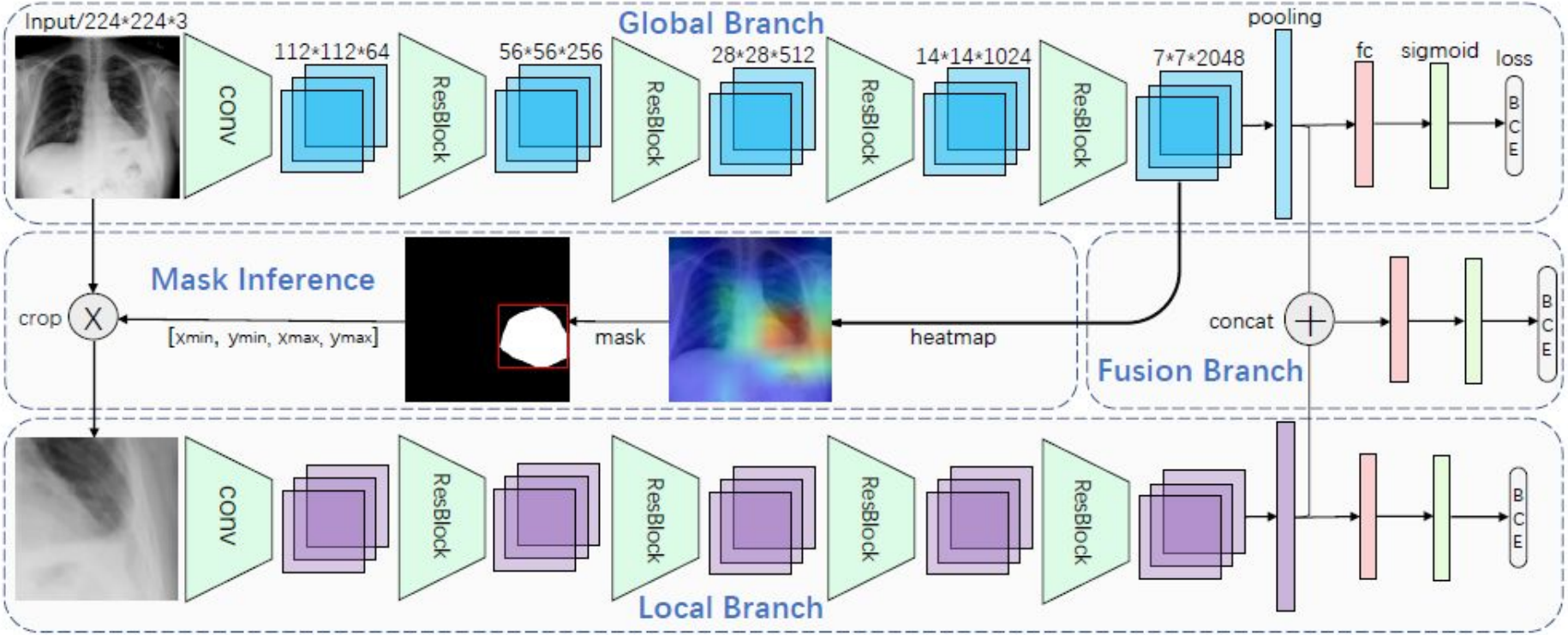
Just some architectures

Many local features



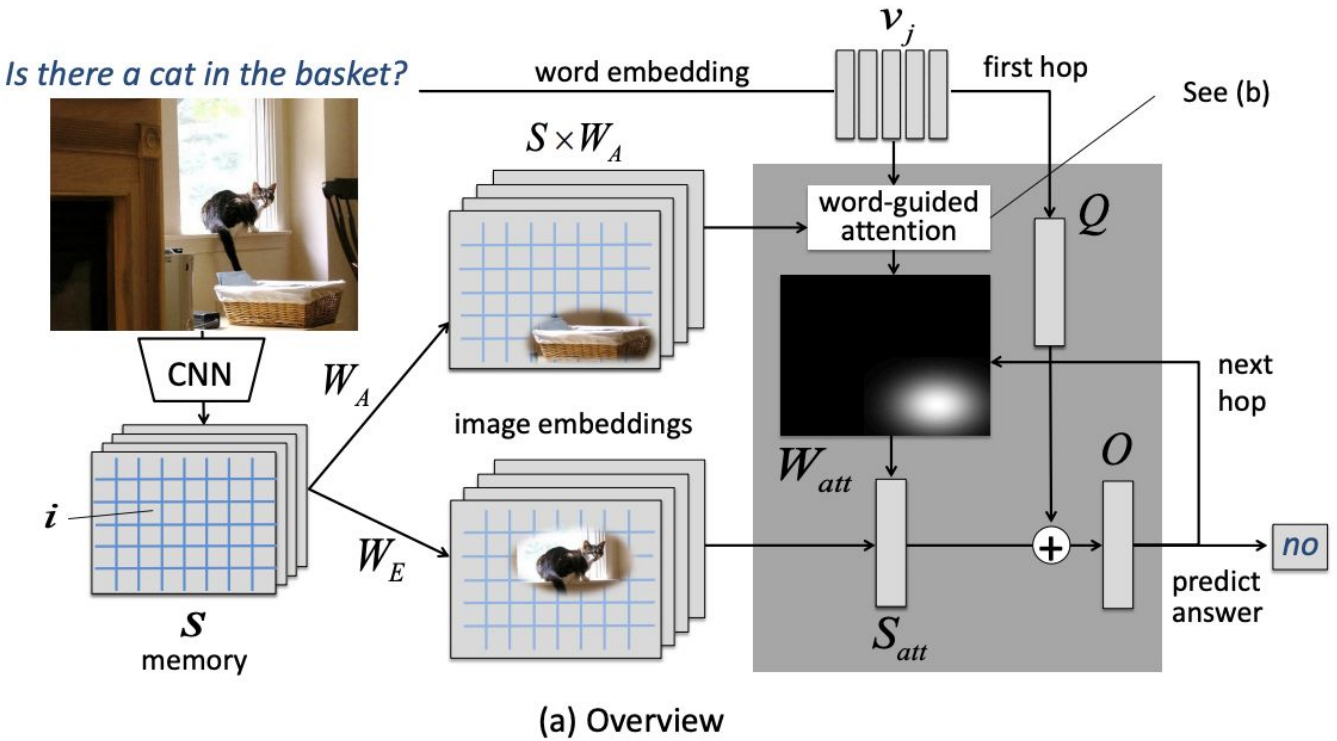
Learning Multi-Attention Convolutional Neural Network for Fine-Grained Image Recognition

Automatic radiologist



Diagnose like a radiologist: Attention guided convolutional neural network for thorax disease classification

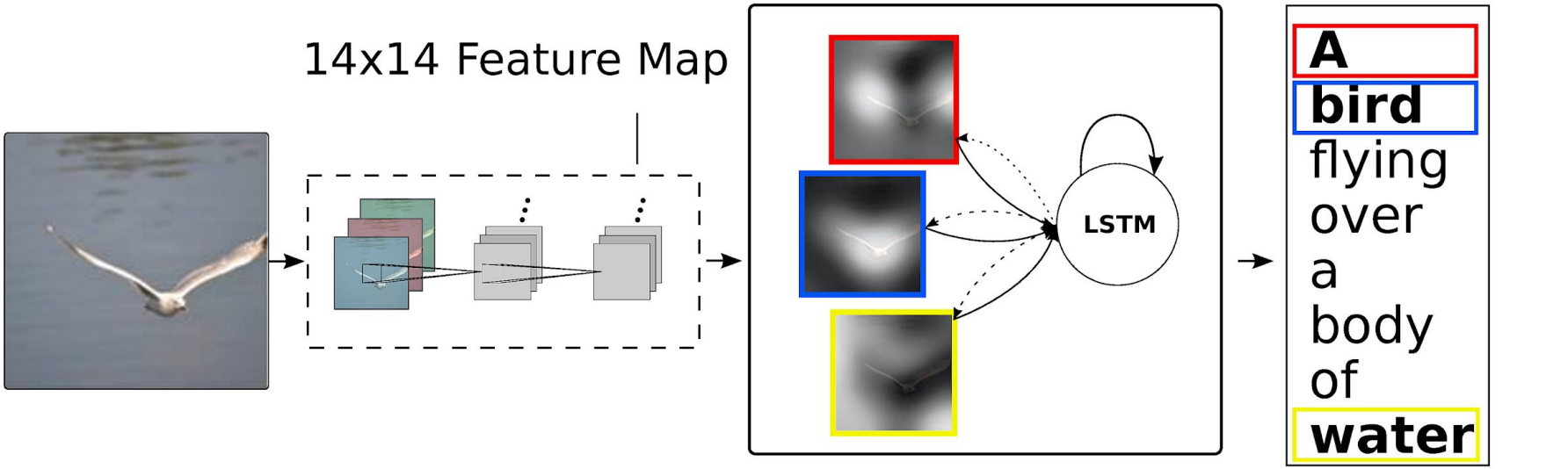
Question asking



(a) Overview

Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering

Captioning



1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

Looking forward

- Get started on the homework early
  - GANs take a *long time to train* and require a lot of attention and love
- Recitation this week will touch on how to train GANs
- [TODO]

We learned today

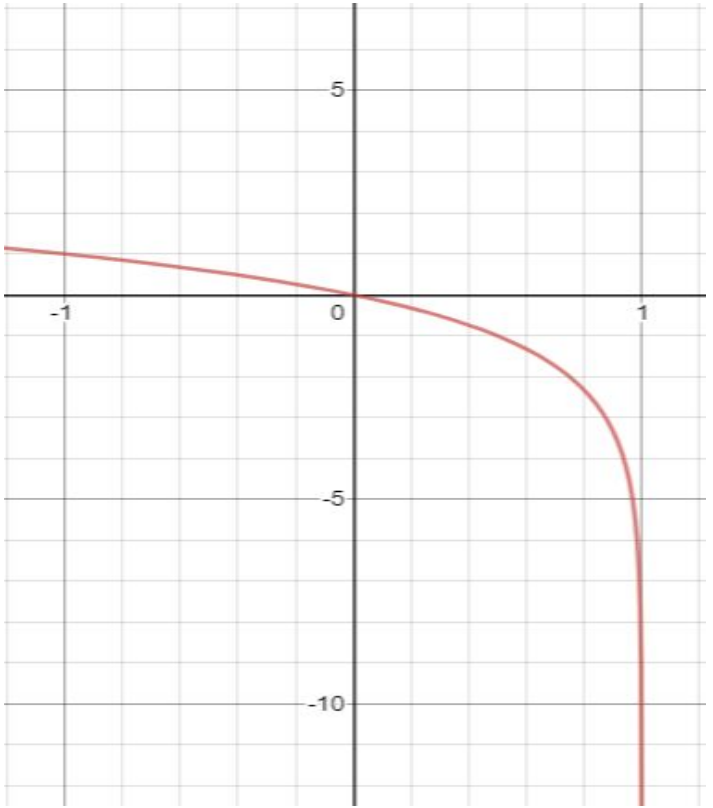
- GANs solve all kinds of image generation problems
- The basic idea of GANs and their relation to modeling probability distributions
- Why they are hard to train
- Some of the ideas behind cool GANs

- Some attention systems in CV

PollEV Feedback

Saturation Issues

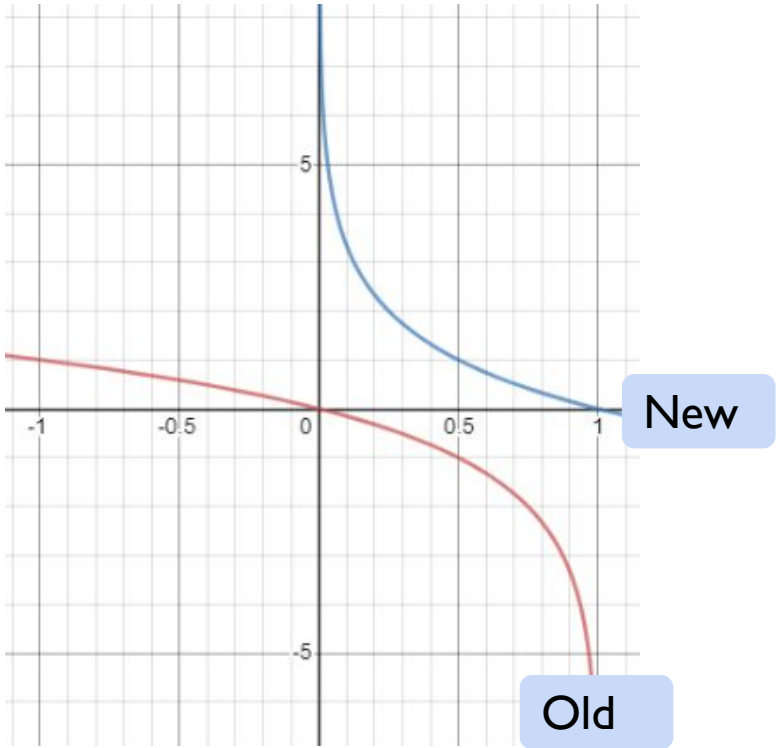
- Here's a graph of y = log(1-x)
- Remember that the proposed *G* loss function is
  y = E[log(1- D(G(x))]
- If D(G(x)) = 1, we've correctly fooled the discriminator
- What's wrong here?



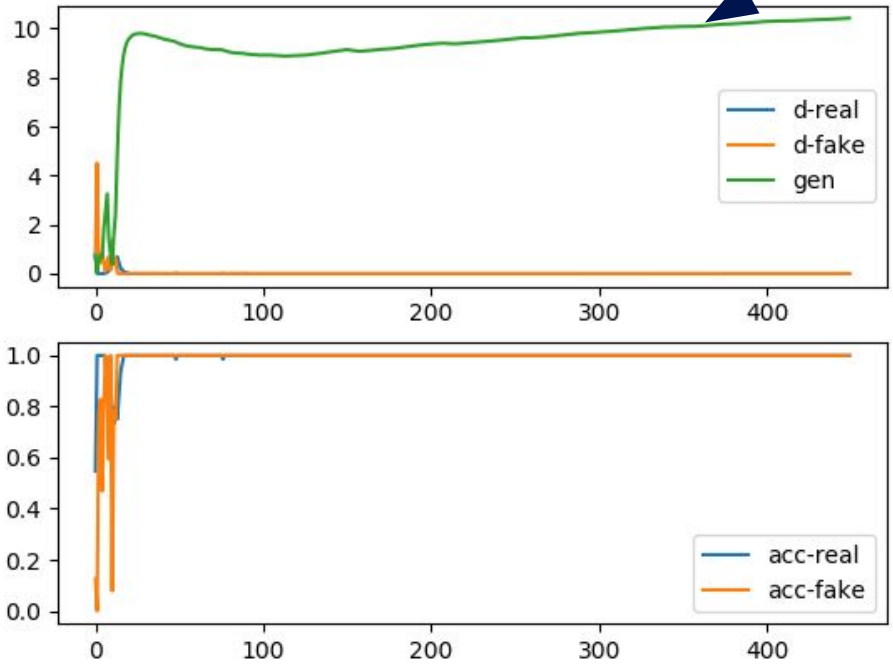
The Generator Loss Function (Modified)

$$\frac{1}{m_{gen}}\sum_{i=1}^{m_{gen}}\log(1-D(G(z^{(i)})))$$

$$-\frac{1}{m_{gen}}\sum_{i=1}^{m_{gen}}\log(D(G(z^{(i)})))$$



Non-Convergence



What's happening here? This is what giving up looks like. The generator gave up.

GANs are notorious for having convergence problems. Existing solutions: better architecture, better activations

All Together Now

The Generator Loss

$$J^{(G)} = -\frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log(D(G(z^{(i)})))$$

The Discriminator Loss

$$J^{(D)} = -\frac{1}{m_{real}} \sum_{i=1}^{m_{real}} y_{real}^{(i)} . \log(D(x^{(i)})) - \frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} (1 - y_{gen}^{(i)}) . \log(1 - D(G(z^{(i)})))$$