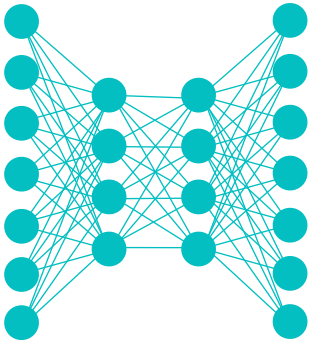
Lecture Notes for

Neural Networks and Machine Learning

Holy GAN-Zooks



Logistics and Agenda

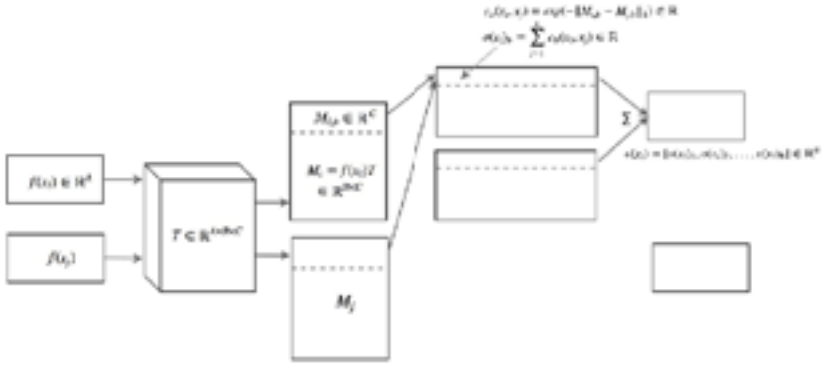
- Logistics
  - None!
- Agenda
  - More-GANs:
    - LSGAN
    - InfoGAN
    - CycleGAN
    - Paper Presentation
    - GAIA (next lecture)
    - Text-to-image Synthesis (next lecture)

Last Time

- Loss function for generator is really difficult and unstable
  - hard to optimize based on feedback only from discriminator output!
- Since generator is trying to statistically match features inside the discriminator (to fool it)
  - try to make generator match statistics of discriminator activations

$$\|\mathbf{E}_{x \leftarrow data}[\mathbf{f}(x)] - \mathbf{E}_{x \leftarrow q(z)}[\mathbf{f}(G(z))]\|_2^2$$

mean discriminator activations from **real** data — mean discriminator activations from **fake** data



- How much do I need to change one distribution to get it to match another?
- Could use KL divergence, but we already know that has many problems associated with vanishing gradients
- We will only have samples from the distributions (not the actual equations)
- **Wasserstein Distance** for continuous probabilities:

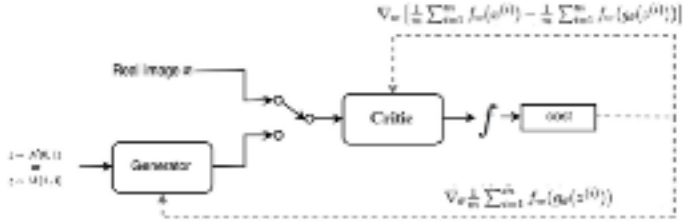
$$W(p_{data}, p_g) = \inf_{\gamma \in \prod(p_{data}, p_g)} \mathbf{E}_{x,y \sim \gamma}\left[\|x - y\|\right]$$

- inf is greatest lower bound
- gamma is completely and utterly intractable to compute

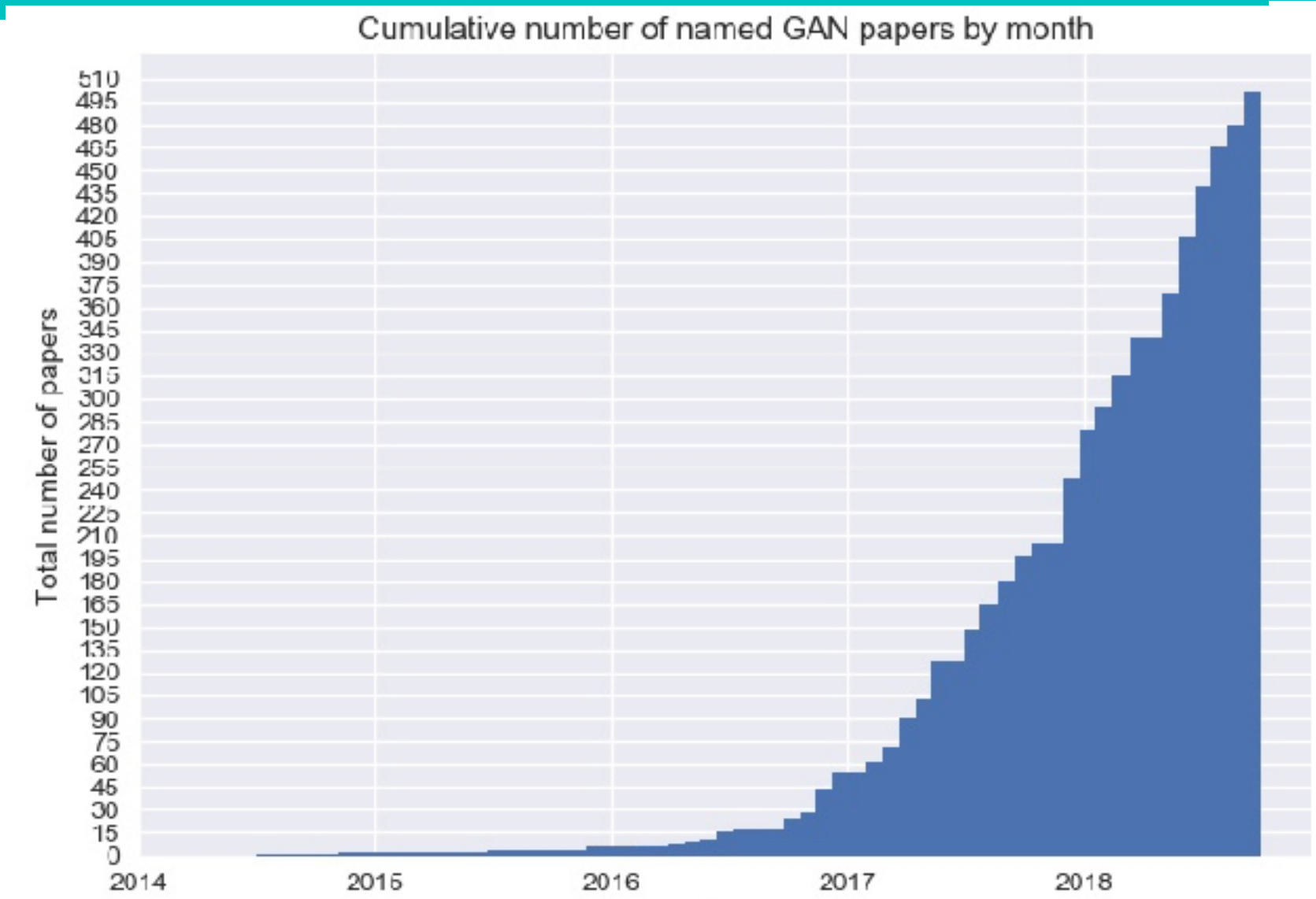
- 1-Lipschitz constraint: $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$.
- Wasserstein Duality Formula:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbf{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbf{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

- where sup is the least upper bound (which we cannot find directly, so we assume its a maximization)



Example GANs, Abridged



https://github.com/hindupuravinash/the-gan-zoo

ABC-GAN —
ABC-GAN —
AC-GAN — C
acGAN — Fa
ACGAN — Co
acGAN — On
ACtuAL — AC
AdaGAN — A
Adaptive GAN
AdvEntuRe —
AdvGAN — G
AE-GAN — A
AE-OT — Lat
AEGAN — Le
AF-DCGAN —
AffGAN — Am
AIM — Gener
AL-CGAN —
ALI — Advers
AlignGAN —
AlphaGAN —
AM-GAN — A
AmbientGAN
AMC-GAN —
AnoGAN — U
APD — Adver
APE-GAN —
ARAE — Adv
ARDA — Adv
ARIGAN — A
ArtGAN — Art
ASDL-GAN —
ATA-GAN —
Attention-GAN
AttGAN — Arb
AttnGAN — A
AVID — AVID

LSGAN

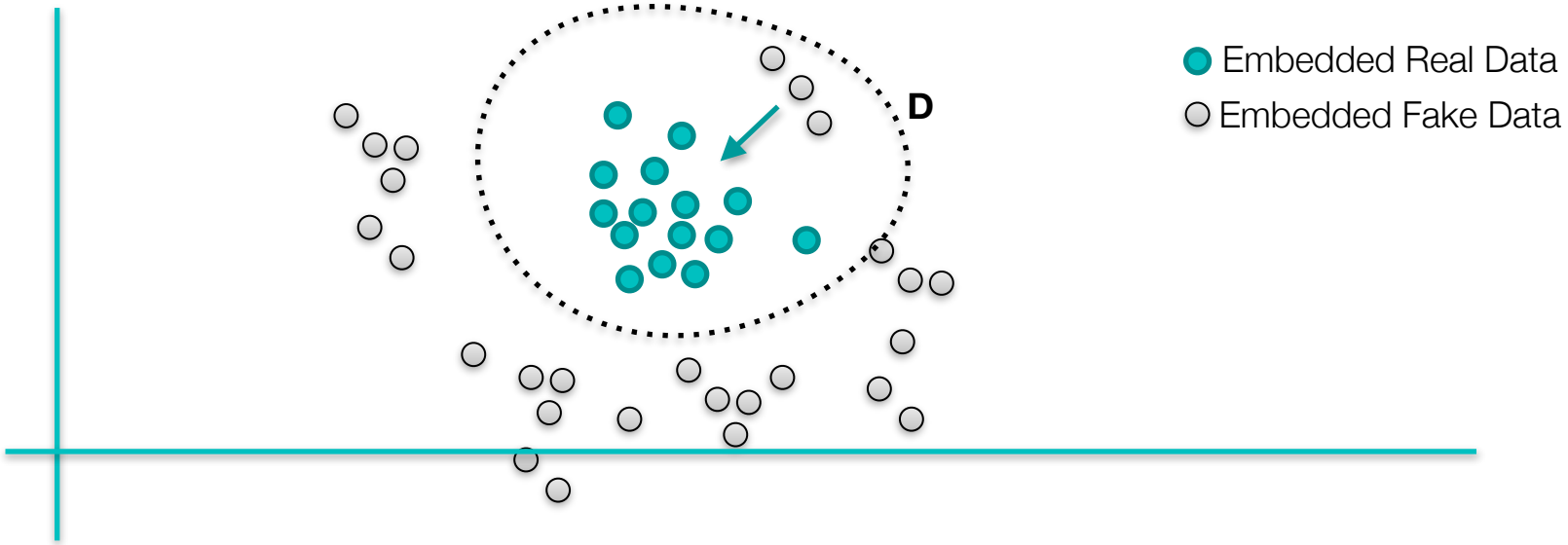
Least Squares Generative Adversarial Networks

Xudong Mao*[1], Qing Li†[1], Haoran Xie‡[2], Raymond Y.K. Lau§[3], Zhen Wang¶[4], and Stephen Paul Smolley‖[5]

[1]Department of Computer Science, City University of Hong Kong
[2]Department of Mathematics and Information Technology, The Education University of Hong Kong
[3]Department of Information Systems, City University of Hong Kong
[4]Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University
[5]CodeHatch Corp.

The Least Squares GAN

- **Observation**: Generated points may (by chance) be classified as real by Discriminator—but they are still not representative of the real data
- **Solution**: Incentivize even correctly classified labels to move toward real data distribution



Mao, Xudong, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. "Least squares generative adversarial networks." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794-2802. 2017.

Incentivizing with Least Squares

- Assume a=fake label, b=real label, c=misleading label
- The new loss function is:

$$\min_D V_{\mathrm{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]$$

$$\min_G V_{\mathrm{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right],$$

- Here we can take advantage of the labels, even when they are classified correct/incorrect
  - …becasue we have a distance to margin
  - …that's it!

But that results is not publishable!

- We need to find a way to complicate the math to make it less approachable and therefore publishable
  - Yay for ethics!
- **Discussion**: is this wrong for the authors to do?

In the original GAN paper [7], the authors has shown that minimizing Equation 1 yields minimizing the Jensen-Shannon divergence:

$$C(G) = KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right) - \log(4). \quad (3)$$

Here we also explore the relation between LSGANs and f-divergence. Consider the following extension of Equation 2:

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x}) - b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - a)^2\right]$$
$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x}) - c)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - c)^2\right]. \quad (4)$$

Note that adding the term $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[(D(\boldsymbol{x})-c)^2]$ to $V_{\text{LSGAN}}(G)$ does not change the optimal values since this term does not contain parameters of G.

We first derive the optimal discriminator D for a fixed G as below :

$$D^*(\boldsymbol{x}) = \frac{bp_{\text{data}}(\boldsymbol{x}) + ap_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}. \quad (5)$$

In the following equations we use p_d to denote p_{data} for simplicity. Then we can reformulate Equation 4 as follows:

$$\begin{aligned}
2C(G) &= \mathbb{E}_{\boldsymbol{x}\sim p_d}\left[(D^*(\boldsymbol{x}) - c)^2\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[(D^*(\boldsymbol{x}) - c)^2\right] \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_d}\left[\left(\frac{bp_d(\boldsymbol{x}) + ap_g(\boldsymbol{x})}{p_d(\boldsymbol{x}) + p_g(\boldsymbol{x})} - c\right)^2\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\left(\frac{bp_d(\boldsymbol{x}) + ap_g(\boldsymbol{x})}{p_d(\boldsymbol{x}) + p_g(\boldsymbol{x})} - c\right)^2\right] \\
&= \int_{\mathcal{X}} p_d(\boldsymbol{x})\left(\frac{(b-c)p_d(\boldsymbol{x}) + (a-c)p_g(\boldsymbol{x})}{p_d(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right)^2 d\boldsymbol{x} + \int_{\mathcal{X}} p_g(\boldsymbol{x})\left(\frac{(b-c)p_d(\boldsymbol{x}) + (a-c)p_g(\boldsymbol{x})}{p_d(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right)^2 d\boldsymbol{x} \\
&= \int_{\mathcal{X}} \frac{\left((b-c)p_d(\boldsymbol{x}) + (a-c)p_g(\boldsymbol{x})\right)^2}{p_d(\boldsymbol{x}) + p_g(\boldsymbol{x})} d\boldsymbol{x} \\
&= \int_{\mathcal{X}} \frac{\left((b-c)(p_d(\boldsymbol{x}) + p_g(\boldsymbol{x})) - (b-a)p_g(\boldsymbol{x})\right)^2}{p_d(\boldsymbol{x}) + p_g(\boldsymbol{x})} d\boldsymbol{x}.
\end{aligned} \quad (6)$$

If we set $b - c = 1$ and $b - a = 2$, then

$$\begin{aligned}
2C(G) &= \int_{\mathcal{X}} \frac{\left(2p_g(\boldsymbol{x}) - (p_d(\boldsymbol{x}) + p_g(\boldsymbol{x}))\right)^2}{p_d(\boldsymbol{x}) + p_g(\boldsymbol{x})} d\boldsymbol{x} \\
&= \chi^2_{\text{Pearson}}(p_d + p_g \| 2p_g),
\end{aligned} \quad (7)$$

where χ^2_{Pearson} is the Pearson χ^2 divergence. Thus minimizing Equation 4 yields minimizing the Pearson χ^2 divergence between $p_d + p_g$ and $2p_g$ if a, b, and c satisfy the conditions of $b - c = 1$ and $b - a = 2$.

LS-GAN Parameter Selection

3.2.3 Parameters Selection

One method to determine the values of a, b, and c in Equation 2 is to satisfy the conditions of $b - c = 1$ and $b - a = 2$, such that minimizing Equation 2 yields minimizing the Pearson χ^2 divergence between $p_d + p_g$ and $2p_g$. For example, by setting $a = -1$, $b = 1$, and $c = 0$, we get the following objective functions:

$$\begin{aligned}\min_D V_{\text{LSGAN}}(D) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-1)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))+1)^2\right] \\ \min_G V_{\text{LSGAN}}(G) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})))^2\right].\end{aligned} \tag{8}$$

Another method is to make G generate samples as real as possible by setting $c = b$. For example, by using the 0-1 binary coding scheme, we get the following objective functions:

$$\begin{aligned}\min_D V_{\text{LSGAN}}(D) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-1)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})))^2\right] \\ \min_G V_{\text{LSGAN}}(G) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-1)^2\right].\end{aligned} \tag{9}$$

In practice, we observe that Equation 8 and Equation 9 show similar performance. Thus either one can be selected. In the following sections, we use Equation 9 to train the models.

LS-GAN Results

- Some takeaways:
  - RMSProp works better than Adam
  - Reasonable values for a,b,c have similar performance
  - Mode collapse is still a problem, but not nearly as bad as regular GANs

Experiment:
Generate 2D samples from known Mix of Gaussian Distributions, then train 3 layer GANs to generate the same 2D data.

Does one learn the distribution?

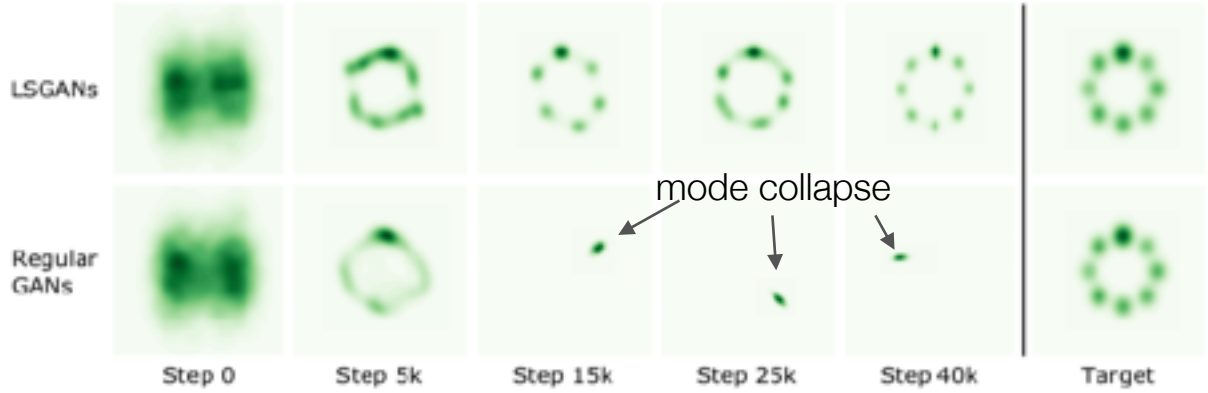
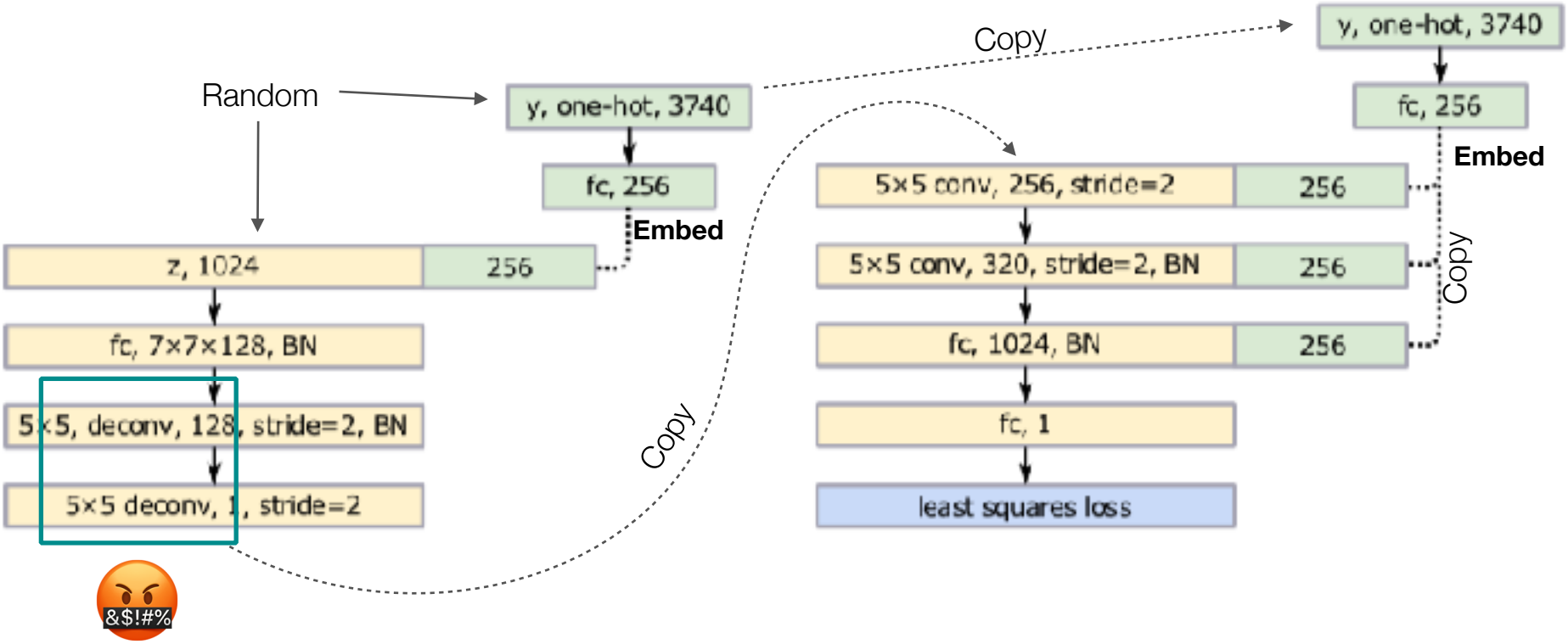


Figure 8: Dynamic results of Gaussian kernel estimation for LSGANs and regular GANs. The final column shows the real data distribution.

Architecture Employed



Qualitative Results

(a) Church outdoor.

(b) Dining room.

(c) Kitchen.

(d) Conference room.

More Qualitative Results

(a) Generated by LSGANs.

(b) Generated by DCGANs (Reported in [11]).

(c) Generated by EBGANs (Reported in [26]).

How can we trust the results?

- Do we trust that the authors are not cherry picking the results?
- If I perform random seed optimization and run my algorithms for longer, can I always claim its better
  - …but maybe not for the reasons I publish…
- Without strong quantitative evaluation criteria for image generation, can there be a solution to this?
  - Require human subjects evaluation?
  - But can't they still tune the results for their algorithm before the human subjects testing?
  - What about open sourcing the weights of a tuned algorithm?

InfoGAN

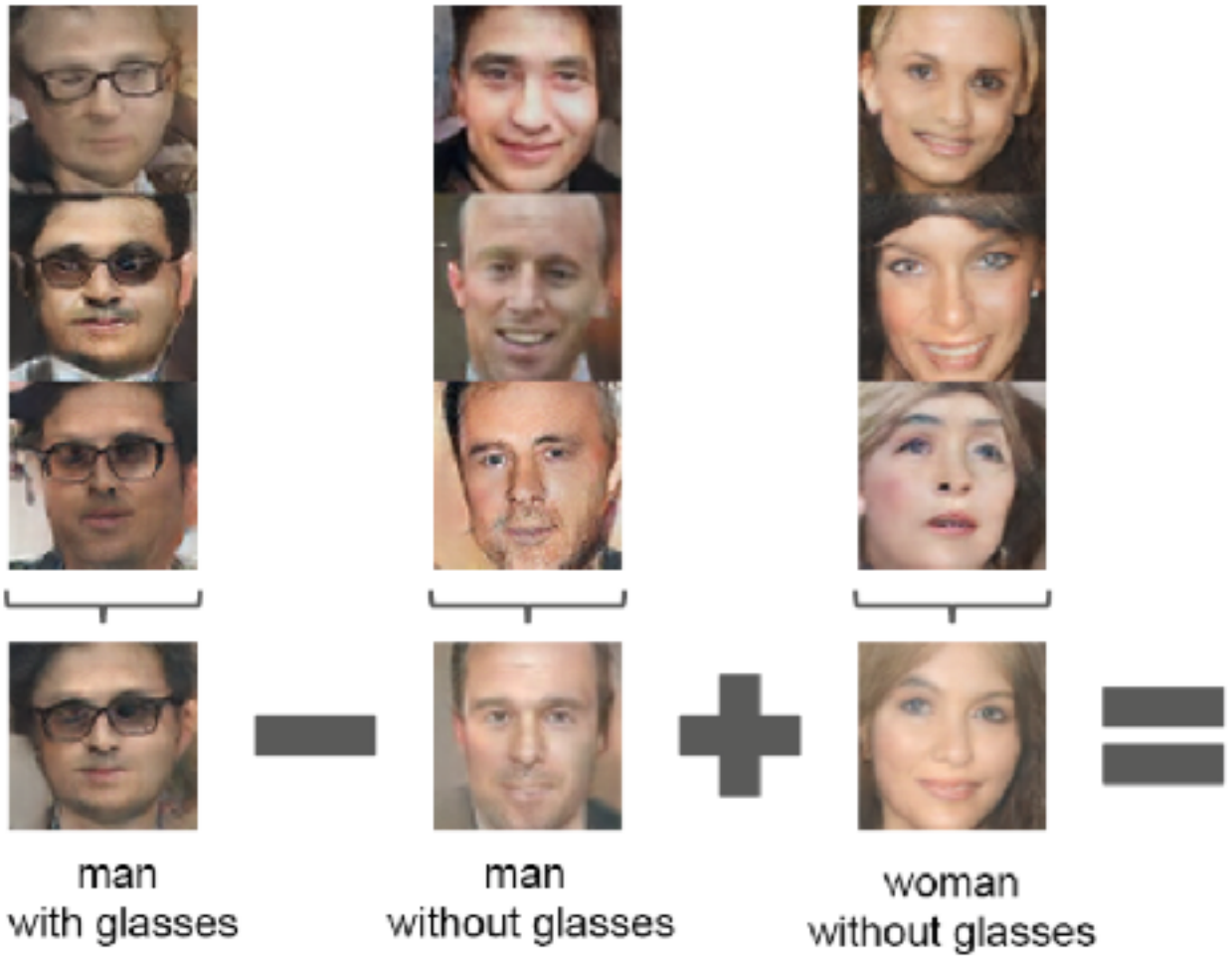
InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

Xi Chen[†‡], Yan Duan[†‡], Rein Houthooft[†‡], John Schulman[†‡], Ilya Sutskever[‡], Pieter Abbeel[†‡]
† UC Berkeley, Department of Electrical Engineering and Computer Sciences
‡ OpenAI

InfoGAN Problem Statement

- We need to disentangle information in the latent space for it to be truly useful
- Disentangling should be ascertained by the ability of the generator to have interpretable latent space axes
- Each axis should have little impact on the other axes, which is not imposed by any GAN structure yet
  - *For example, for a dataset of faces, a useful disentangled representation may allocate a separate set of dimensions for each of the following attributes: facial expression, eye color, hairstyle, presence or absence of eyeglasses, and the identity of the corresponding person.*

How to perform latent space arithmetic?



NOT Systematic!

woman with glasses

Need to find a bunch of these

Need to find a bunch of these

Need to find a bunch of these

To generate al these

The InfoGAN Money Back Guarantee

However, many domains naturally decompose into a set of semantically meaningful factors of variation. For instance, when generating images from the MNIST dataset, it would be ideal if the model automatically chose to allocate a discrete random variable to represent the numerical identity of the d… the digit's angle a… ndent and salient, … by simply specify… dependent continuous variables.

Authors: by Maximizing Mutual Information the network will learn to disentangle latent space

- Decompose latent variable representation into
  - z: continuous incompressible noise
  - c: latent code, hopefully learns semantic features
- We want mutual information of c and $G(z,c)$ to be large
- Using standard entropy definition: $I(X;Y)=H(X)-H(X|Y)$
- New Loss Function: $\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c;G(z,c))$

Regular GAN Loss — MI Loss

Maximizing Mutual Information

- You guessed it, Mutual Informaiton with P is intractable to compute!
- So let's dust off our approximation pants and put some ELBO grease into this
- Variational Inference Maximization, Assume there is an approximation of P from Q:

$$
\begin{aligned}
I(c; G(z,c)) &= H(c) - H(c|G(z,c)) \\
&= \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log P(c'|x)]] + H(c) \\
&= \mathbb{E}_{x\sim G(z,c)}[\underbrace{D_{\mathrm{KL}}(P(\cdot|x) \parallel Q(\cdot|x))}_{\text{make Q close to P } \geq 0} + \mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \\
&\geq \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c)
\end{aligned}
$$

Expand Def. of Conditional Entropy

Insert Q Approximation (Variational)

sample from Q

Define a Lower Bound to Maximize because we don't like other terms…

Lemma 5.1 *For random variables X, Y and function $f(x,y)$ under suitable regularity conditions:* $\mathbb{E}_{x\sim X, y\sim Y|x}[f(x,y)] = \mathbb{E}_{x\sim X, y\sim Y|x, x'\sim X|y}[f(x',y)]$.

Maximizing Mutual Information (kinda)

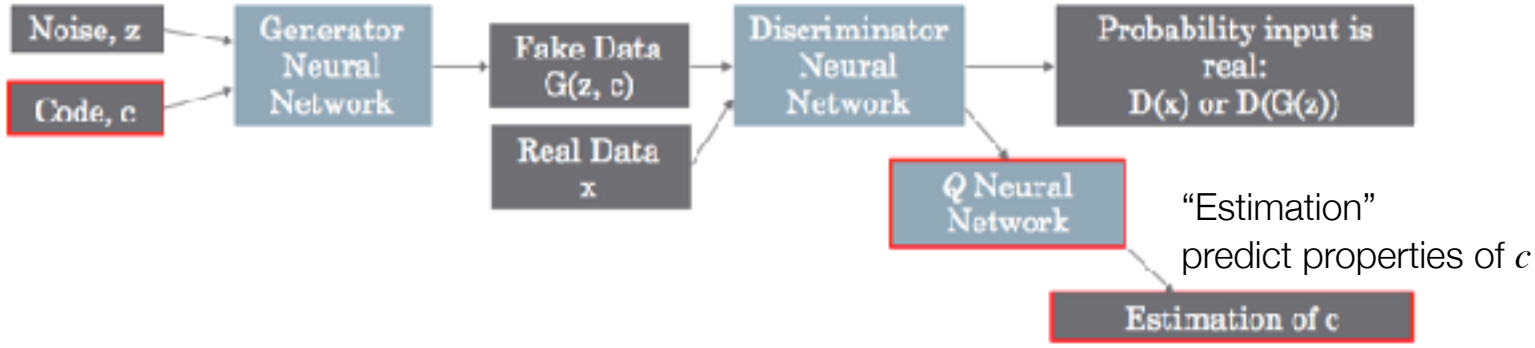
$$I(c; G(z,c)) \geq \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c)$$

Lemma 5.1 *For random variables X, Y and function $f(x, y)$ under suitable regularity conditions:* $\mathbb{E}_{x\sim X, y\sim Y|x}[f(x,y)] = \mathbb{E}_{x\sim X, y\sim Y|x, x'\sim X|y}[f(x', y)]$.

$$\begin{aligned} L_I(G,Q) &= E_{c\sim P(c), x\sim G(z,c)}[\log Q(c|x)] + H(c) \\ &= E_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \\ &\leq I(c; G(z,c)) \end{aligned}$$

Re-parameterize:
Q will be good if we can measure c from the Generator network

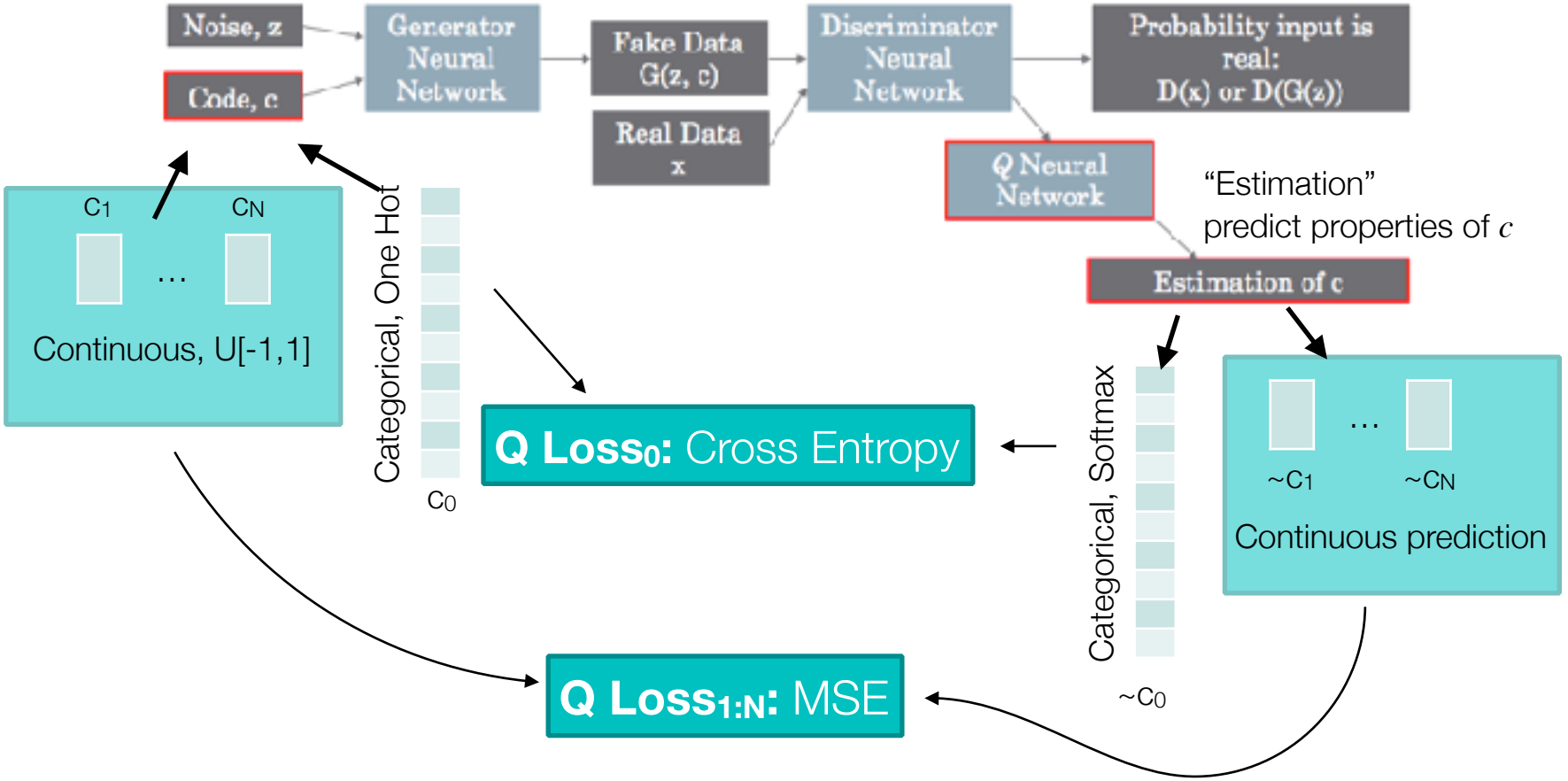
New Objective Function Becomes: $\min_{G,Q} \max_{D} V_{\text{InfoGAN}}(D,G,Q) = V(D,G) - \lambda L_I(G,Q)$



"Estimation"
predict properties of c

https://towardsdatascience.com/infogan-generative-adversarial-networks-part-iii-380c0c6712cd

Measuring Latent Codes in Loss



Intuition: If the generator starts to use these codes and variables semantically, then the Q network (and therefore the discriminator) can start learning to predict these semantics. By learning these semantics, the generator is incentivized to produce disentangled, meaningful features from the codes.

Implementation: D and Q

```
# define the standalone discriminator model
def define_discriminator(n_cat, in_shape=(28,28,1)):
	# weight initialization
	init = RandomNormal(stddev=0.02)
	# image input
	in_image = Input(shape=in_shape)
	# downsample to 14x14
	d = Conv2D(64, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(in_image)
	d = LeakyReLU(alpha=0.1)(d)
	# downsample to 7x7
	d = Conv2D(128, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(d)
	d = LeakyReLU(alpha=0.1)(d)
	d = BatchNormalization()(d)
	# normal
	d = Conv2D(256, (4,4), padding='same', kernel_initializer=init)(d)
	d = LeakyReLU(alpha=0.1)(d)
	d = BatchNormalization()(d)
	# flatten feature maps
	d = Flatten()(d)
	# real/fake output
	out_classifier = Dense(1, activation='sigmoid')(d)
	# define d model
	d_model = Model(in_image, out_classifier)
	# compile d model
	d_model.compile(loss='binary_crossentropy', optimizer=Adam(lr=0.0002, beta_1=0.5))
	# create q model layers
	q = Dense(128)(d)
	q = BatchNormalization()(q)
	q = LeakyReLU(alpha=0.1)(q)
	# q model output
	out_codes = Dense(n_cat, activation='softmax')(q)
	# define q model
	q_model = Model(in_image, out_codes)
	return d_model, q_model
```

https://machinelearningmastery.com/how-to-develop-an-information-maximizing-generative-adversarial-network-infogan-in-keras/

Implementation: G and Q Trained

```
# define the combined discriminator, generator and q network model
def define_gan(g_model, d_model, q_model):
	# make weights in the discriminator (some shared with the q model) as not trainable
	d_model.trainable = False
	# connect g outputs to d inputs
	d_output = d_model(g_model.output)
	# connect g outputs to q inputs
	q_output = q_model(g_model.output)
	# define composite model
	model = Model(g_model.input, [d_output, q_output])
	# compile model
	opt = Adam(lr=0.0002, beta_1=0.5)
	model.compile(loss=['binary_crossentropy', 'categorical_crossentropy'], optimizer=opt)
	return model
```

Q network will update with generator
(for variables not shared with discriminator)

```
# generate points in latent space as input for the generator
def generate_latent_points(latent_dim, n_cat, n_samples):
	# generate points in the latent space
	z_latent = randn(latent_dim * n_samples)
	# reshape into a batch of inputs for the network
	z_latent = z_latent.reshape(n_samples, latent_dim)
	# generate categorical codes
	cat_codes = randint(0, n_cat, n_samples)
	# one hot encode
	cat_codes = to_categorical(cat_codes, num_classes=n_cat)
	# concatenate latent points and control codes
	z_input = hstack((z_latent, cat_codes))
	return [z_input, cat_codes]
```

Only uses codes, not continuous “c” vars

Input to generator and labels for Q network returned (repeats “c”)

https://machinelearningmastery.com/how-to-develop-an-information-maximizing-generative-adversarial-network-infogan-in-keras/

Implementation: Training

```
# train the generator and discriminator
def train(g_model, d_model, gan_model, dataset, latent_dim, n_cat, n_epochs=100, n_batch=64):
	# calculate the number of batches per training epoch
	bat_per_epo = int(dataset.shape[0] / n_batch)
	# calculate the number of training iterations
	n_steps = bat_per_epo * n_epochs
	# calculate the size of half a batch of samples
	half_batch = int(n_batch / 2)
	# manually enumerate epochs
	for i in range(n_steps):
		# get randomly selected ‘real' and ‘fake’ samples
		X_real, y_real = generate_real_samples(dataset, half_batch)
		X_fake, y_fake = generate_fake_samples(g_model, latent_dim, n_cat, half_batch)
		# update discriminator and q model weights
		d_loss1 = d_model.train_on_batch(X_real, y_real)
		d_loss2 = d_model.train_on_batch(X_fake, y_fake)
		# prepare points in latent space as input for the generator
		z_input, cat_codes = generate_latent_points(latent_dim, n_cat, n_batch)
		# create inverted labels for the fake samples
		y_mislead = ones((n_batch, 1))
		# update the g via the d and q error
		_,g_1,g_2 = gan_model.train_on_batch(z_input, [y_mislead, cat_codes])
		# summarize loss on this batch
		print('>%d, d[%.3f,%.3f], g[%.3f] q[%.3f]' % (i+1, d_loss1, d_loss2, g_1, g_2))
		# evaluate the model performance every 'epoch'
		if (i+1) % (bat_per_epo * 10) == 0:
			summarize_performance(i, g_model, gan_model, latent_dim, n_cat)
```

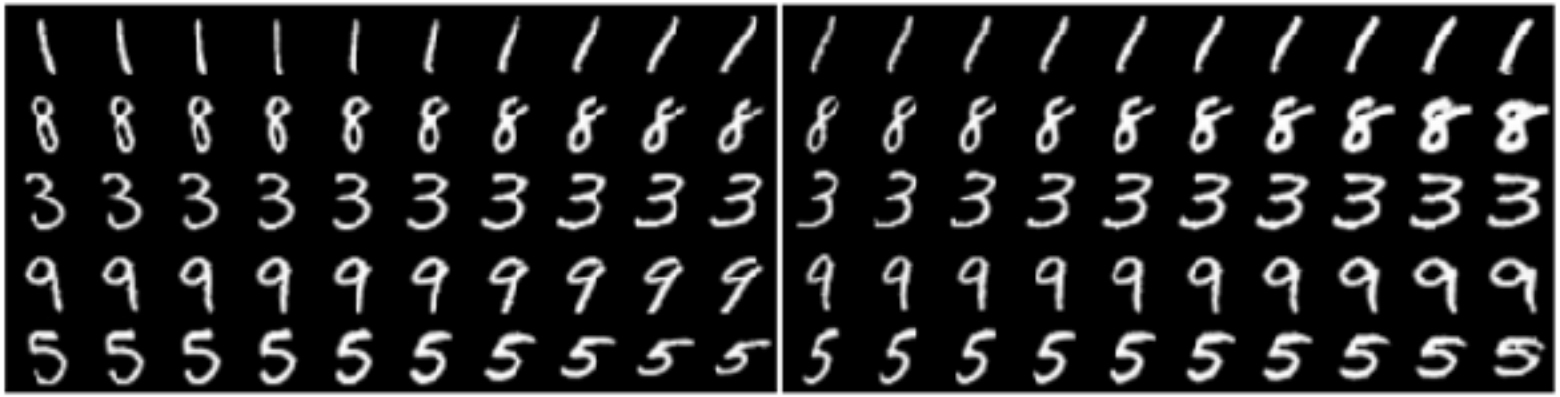
https://machinelearningmastery.com/how-to-develop-an-information-maximizing-generative-adversarial-network-infogan-in-keras/

Did they really disentangle?

c1 = 10 discrete codes with equal probability 0.1
c2 and c3 = uniformly sampled data from -1 to 1

(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)



(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Columns: Hold other variables constant, Rows: Vary named variable

Did they really disentangle?

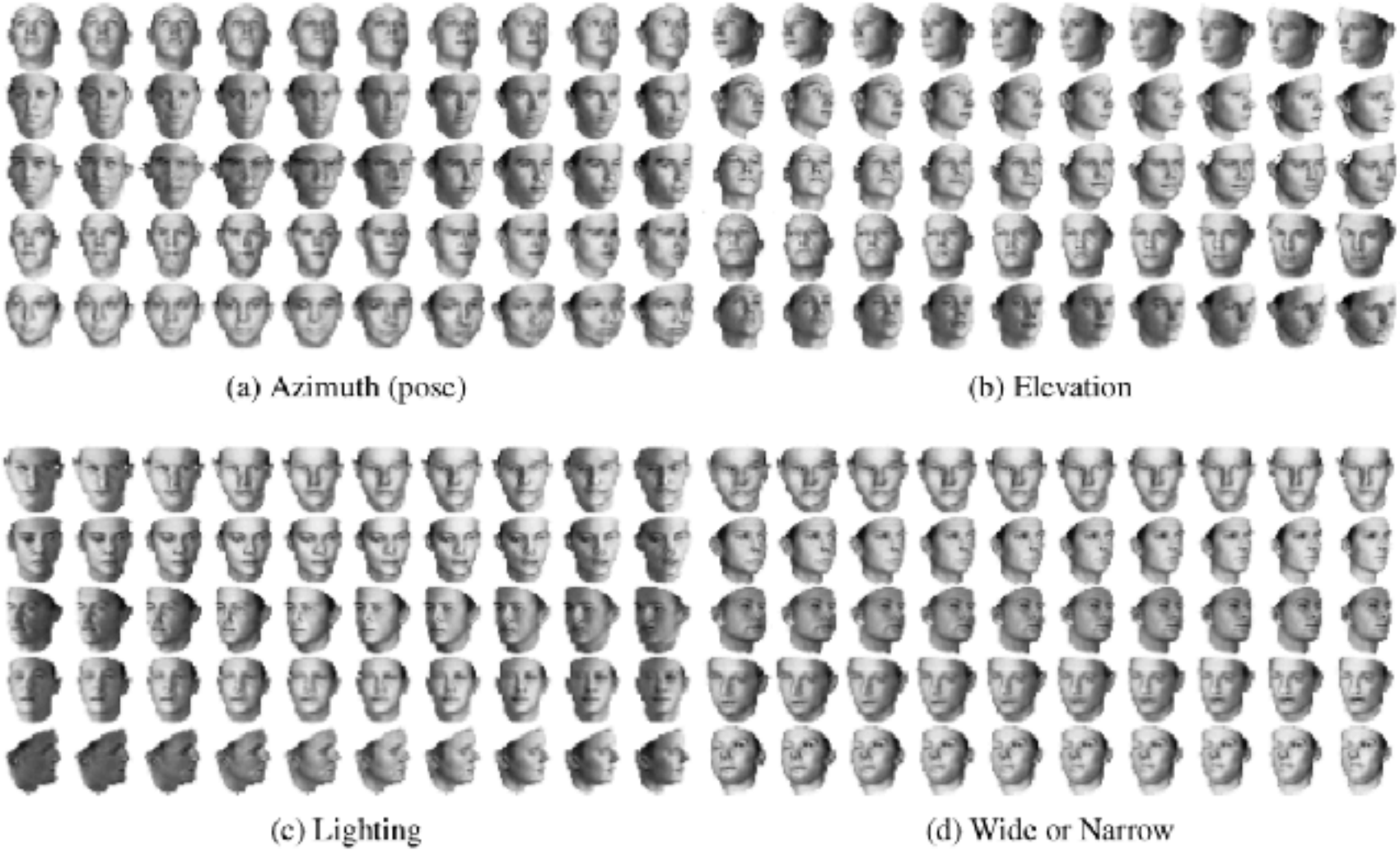


Figure 3: **Manipulating latent codes on 3D Faces:** We show the effect of the learned continuous

Did they really disentangle?

(a) Azimuth (pose)

(b) Presence or absence of glasses

(c) Hair style

(d) Emotion

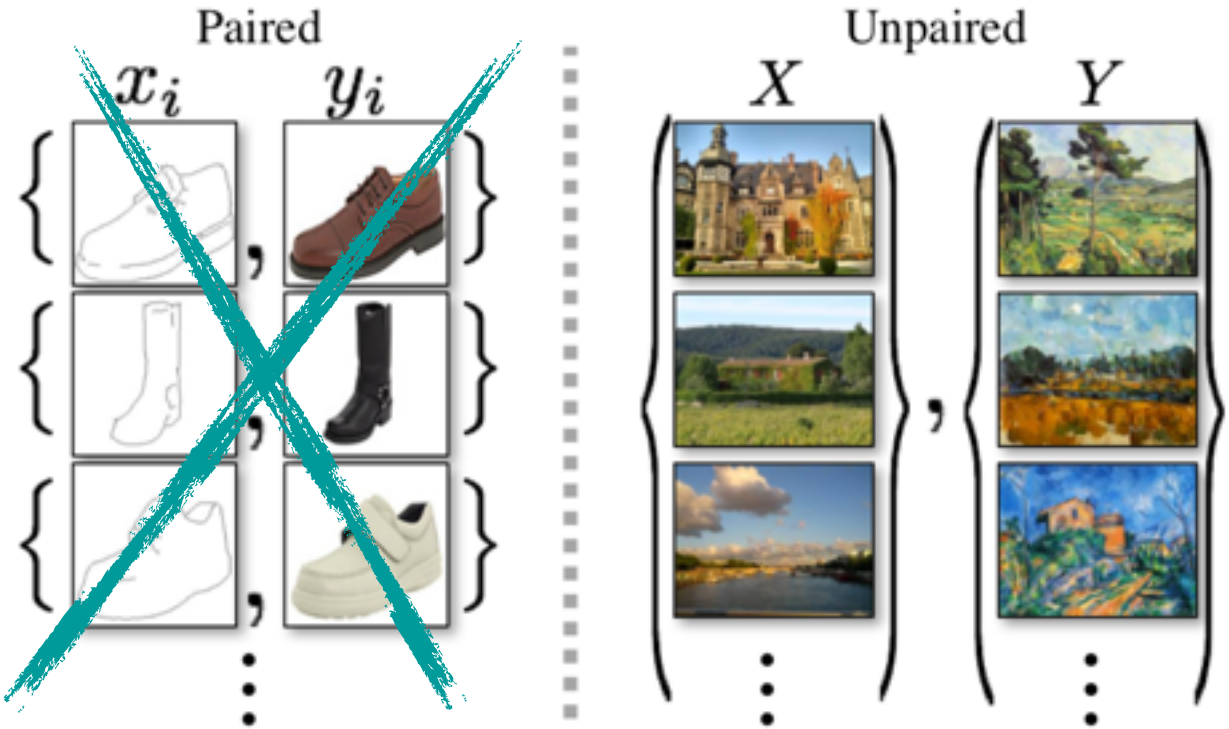
CycleGAN

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

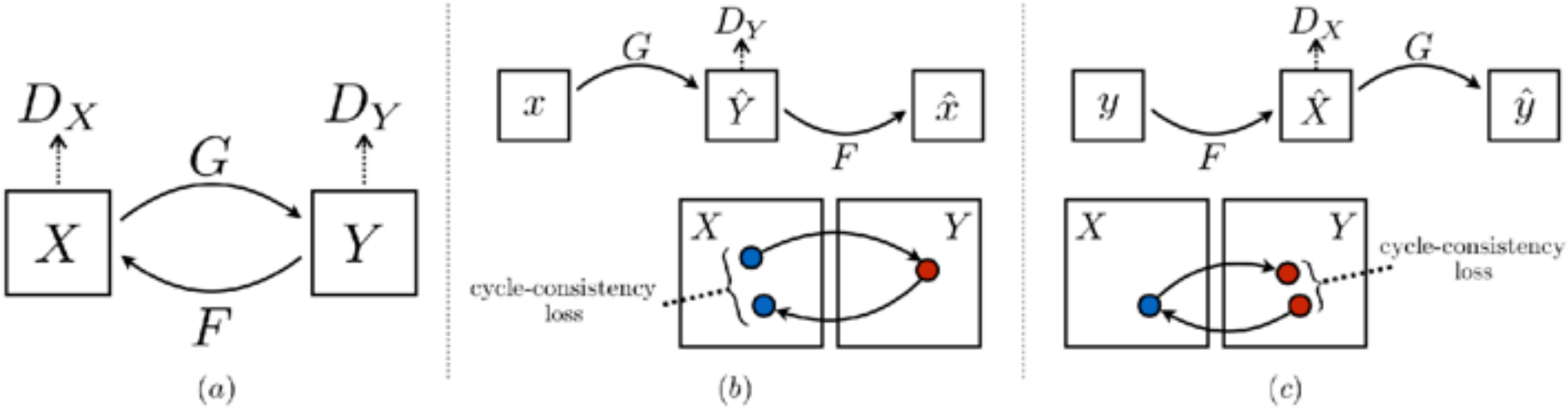
Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros
Berkeley AI Research (BAIR) laboratory, UC Berkeley

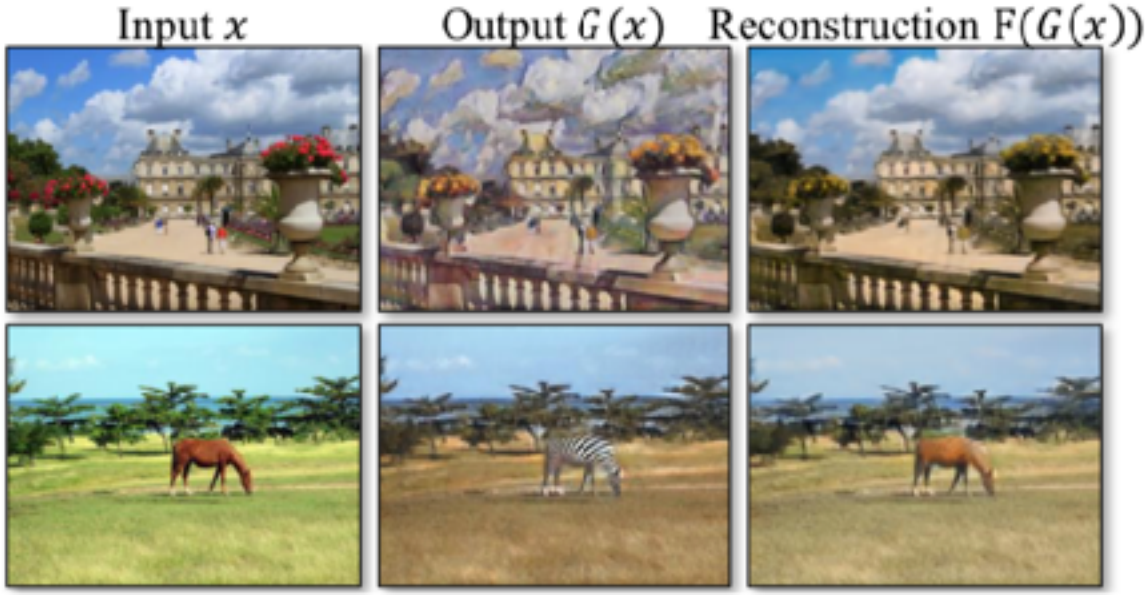
Unpaired Cycles

- Idea: Try to find specialized unpaired translations that are consistent between domains



Defining Cycle Consistency





Adding to the loss function

- Have two discriminator losses for two auto encoders:

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$

- Enforce cycle consistency and self consistency:

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

$$\mathcal{L}_{\text{identity}}(G, F) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(x) - x\|_1]$$

- Put it all together:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$

Training Notes

Training details We apply two techniques from recent works to stabilize our model training procedure. First, for $\mathcal{L}_{\text{GAN}}$ (Equation 1), we replace the negative log likelihood objective by a least-squares loss [35]. This loss is more stable during training and generates higher quality results. In particular, for a GAN loss $\mathcal{L}_{\text{GAN}}(G, D, X, Y)$, we train the G to minimize $\mathbb{E}_{x \sim p_{\text{data}}(x)}[(D(G(x)) - 1)^2]$ and train the D to minimize $\mathbb{E}_{y \sim p_{\text{data}}(y)}[(D(y) - 1)^2] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[D(G(x))^2]$.

Second, to reduce model oscillation [15], we follow nators using a history of generated images rather than the ones produced by the latest generators. We keep an image buffer that stores the 50 previously created images.

For all the experiments, we set $\lambda = 10$ in Equation 3. We use the Adam solver [26] with a batch size of 1. All networks were trained from scratch with a learning rate of 0.0002. We keep the same learning rate for the first 100 epochs and linearly decay the rate to zero over the next 100 epochs. Please see the appendix (Section 7) for more details about the datasets, architectures, and training procedures.

So, not using Entropy?

Yes! Experience Replay!

Linear learning rate decay.

Show me the Code!

Disciminators

```
# Build and compile the discriminators
self.d_A = self.build_discriminator()
self.d_B = self.build_discriminator()
self.d_A.compile(loss='mse',
    optimizer=optimizer,
    metrics=['accuracy'])
self.d_B.compile(loss='mse',
    optimizer=optimizer,
    metrics=['accuracy'])
```

Cycle Producers

```
# Translate images to the other domain
fake_B = self.g_AB(img_A)
fake_A = self.g_BA(img_B)
# Translate images back to original domain
reconstr_A = self.g_BA(fake_B)
reconstr_B = self.g_AB(fake_A)
# Identity mapping of images
img_A_id = self.g_BA(img_A)
img_B_id = self.g_AB(img_B)
```

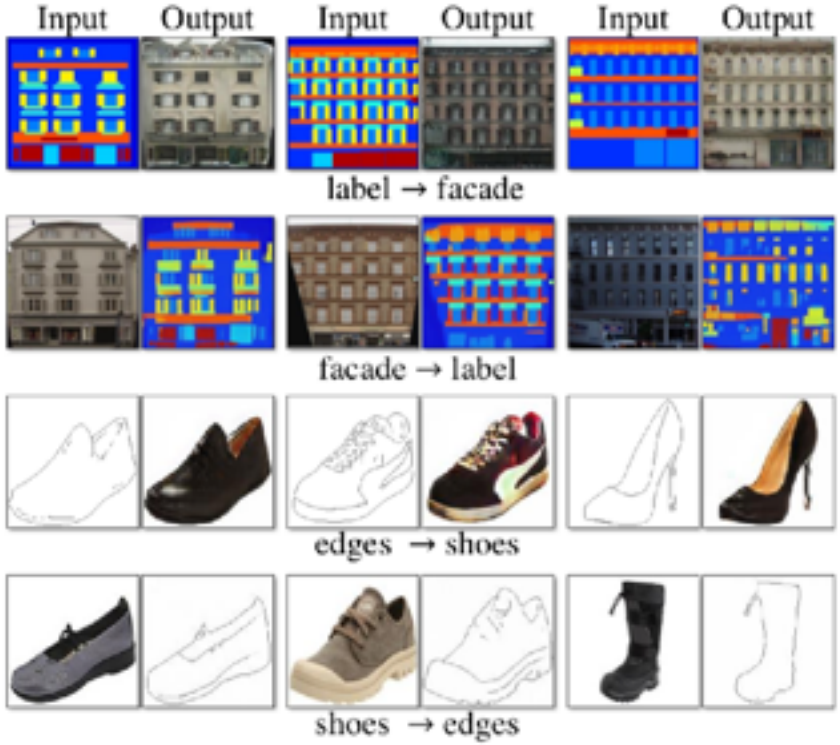
```
# Combined model trains generators to fool discriminators
self.combined = Model(inputs=[img_A, img_B],
                      outputs=[ valid_A, valid_B,
                                reconstr_A, reconstr_B,
                                img_A_id, img_B_id ])
self.combined.compile(loss=['mse', 'mse',
                            'mae', 'mae',
                            'mae', 'mae'],
                      loss_weights=[  1, 1,
                                      self.lambda_cycle, self.lambda_cycle,
                                      self.lambda_id, self.lambda_id ],
                      optimizer=optimizer)
```

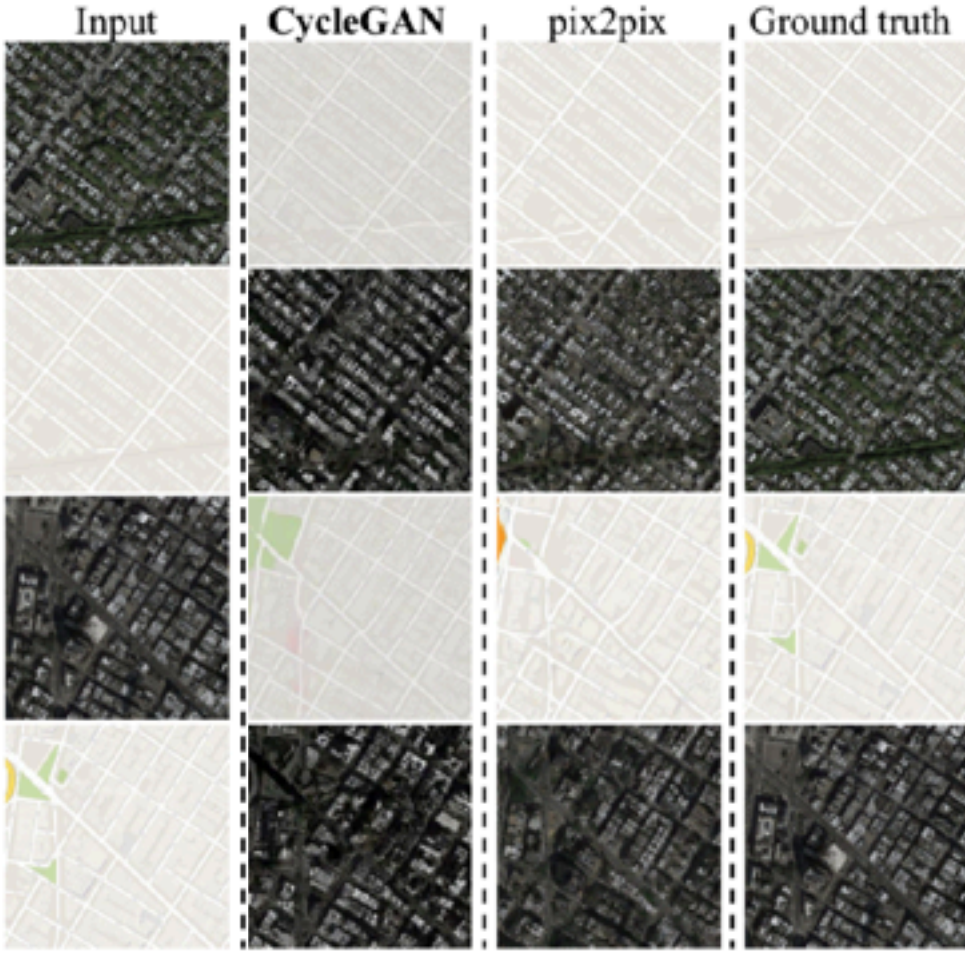
discriminator MSE, 0,1
cycle consistency
self consistency

https://github.com/eriklindernoren/Keras-GAN/blob/master/cyclegan/cyclegan.py

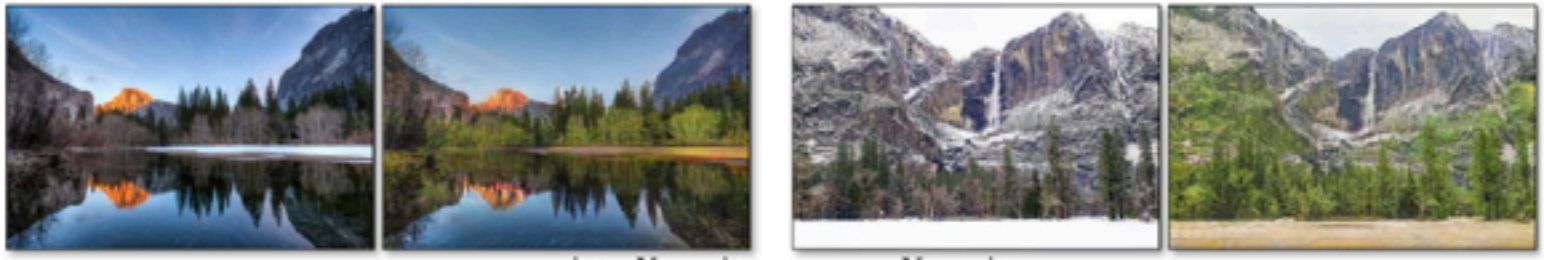
Results


Input
CycleGAN
pix2pix
Ground truth


Input
Output
Input
Output
Input
Output
label → facade
facade → label
edges → shoes
shoes → edges


Input
CycleGAN
pix2pix
Ground truth

Fun Results



winter Yosemite → summer Yosemite

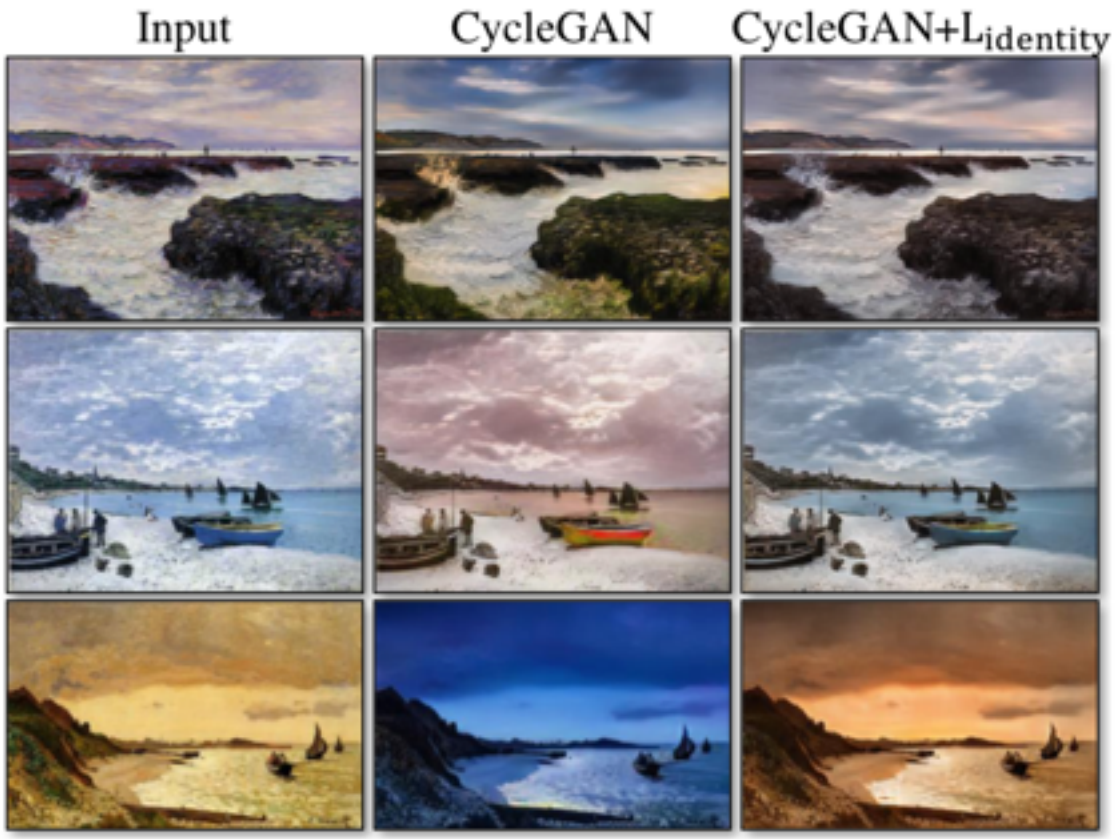
summer Yosemite → winter Yosemite

apple → orange

orange → apple

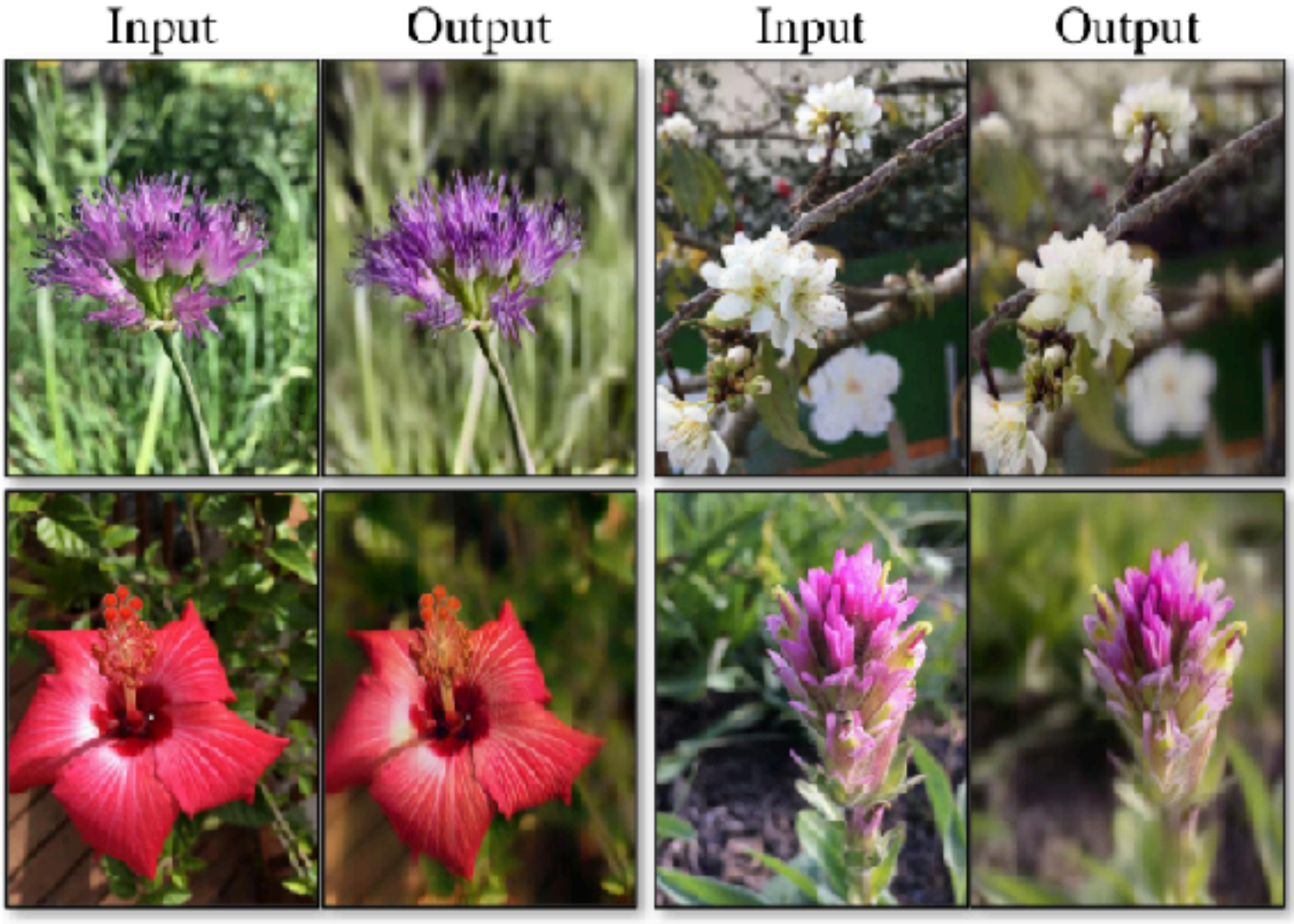
Results: Fixing Hue Changes

- Self consistency is mostly about preserving color



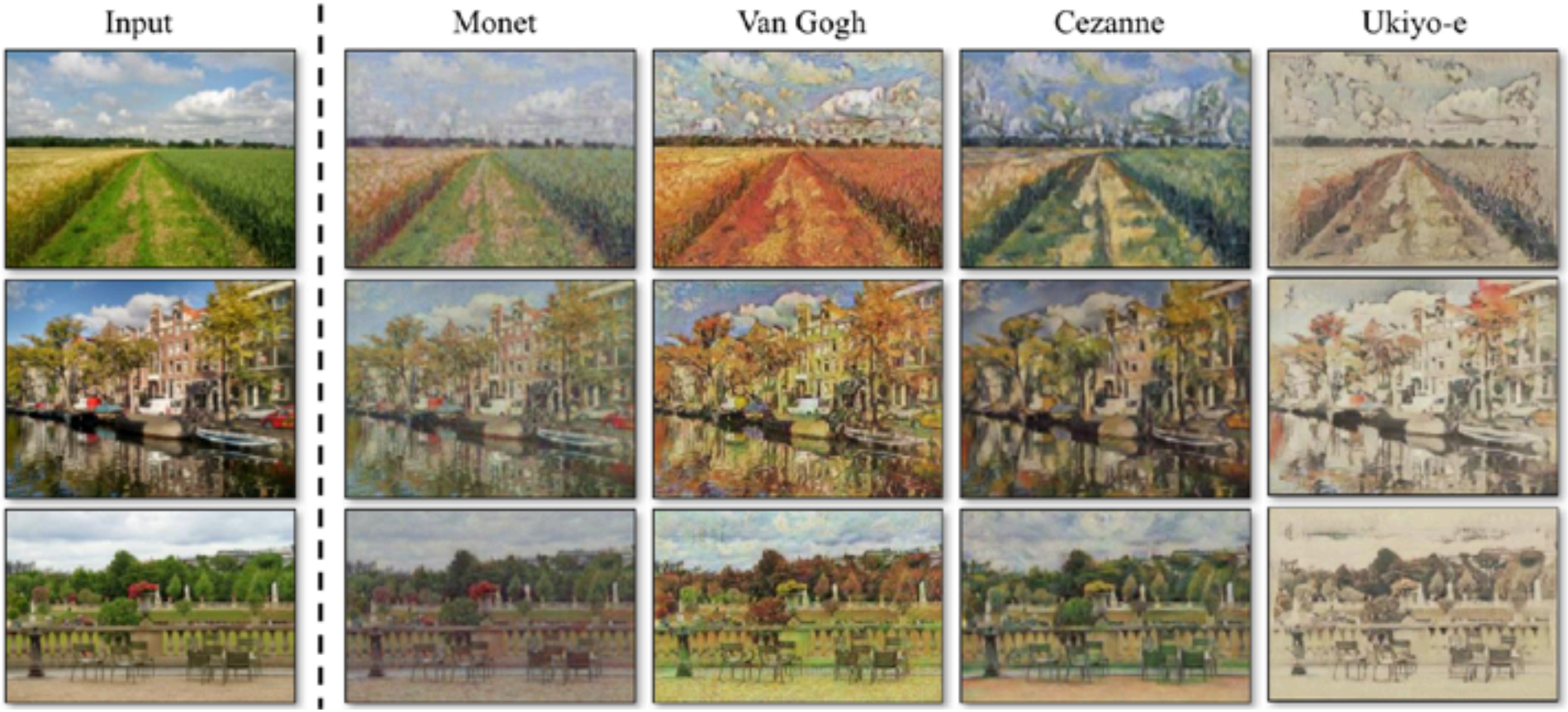
$$\mathcal{L}_{\text{identity}}(G, F) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(x) - x\|_1]$$

Image Enhancement

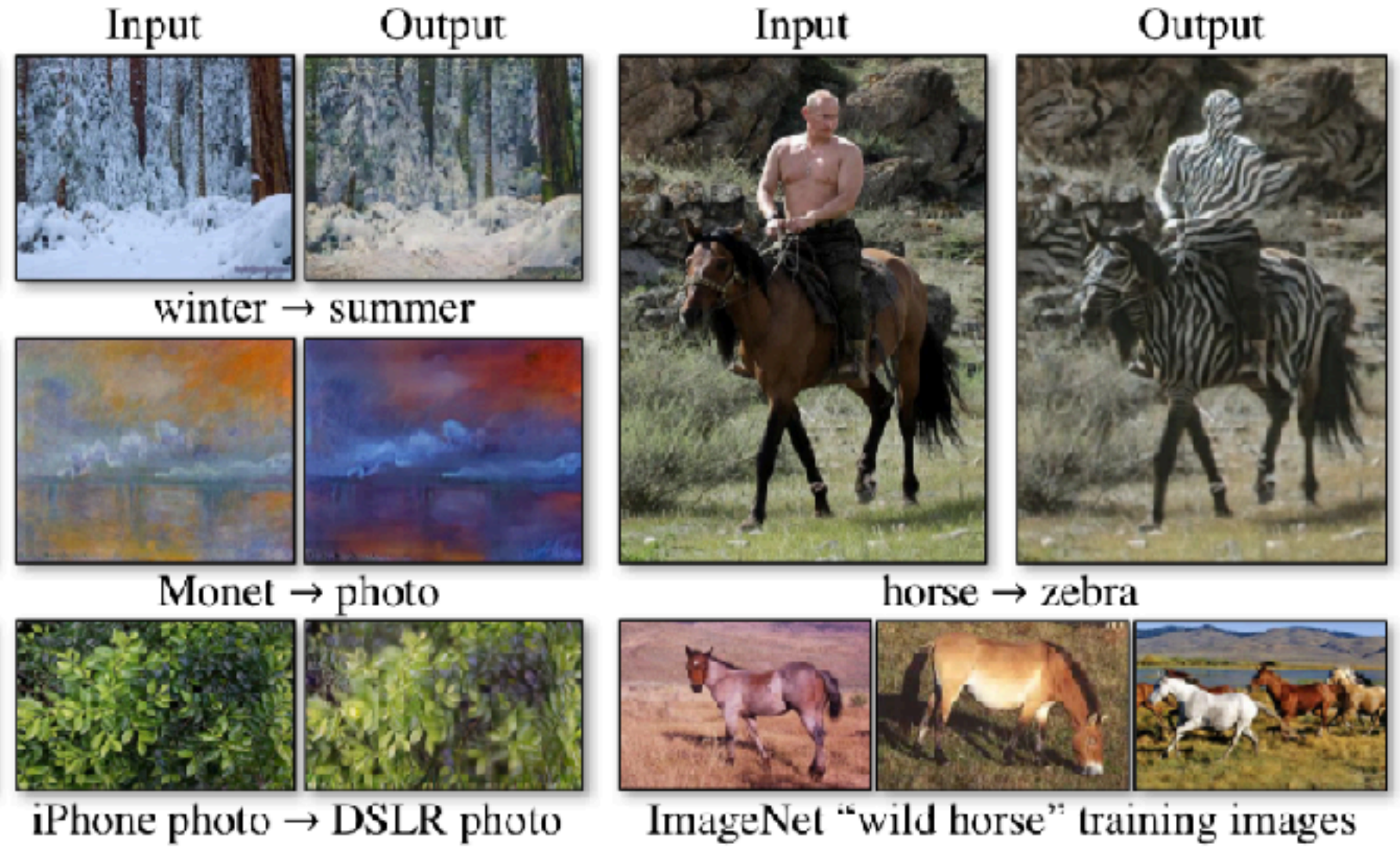


Map from iPhone Images to DSLR Photos

Full Circle: Back to Style Transfer



They can't all be gold



GAN Town Hall

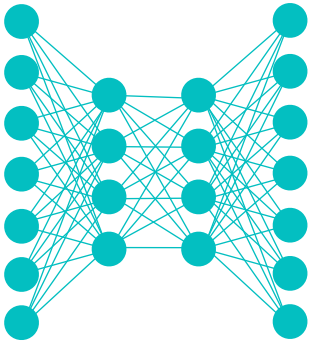
Lecture Notes for

Neural Networks and Machine Learning

Holy GAN Zooks

Next Time:
Continued

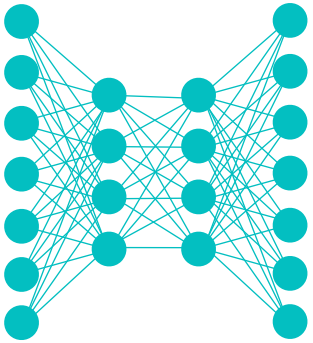
Reading: None



Lecture Notes for

Neural Networks and Machine Learning

Holy GAN-Zooks
Continued



Logistics and Agenda

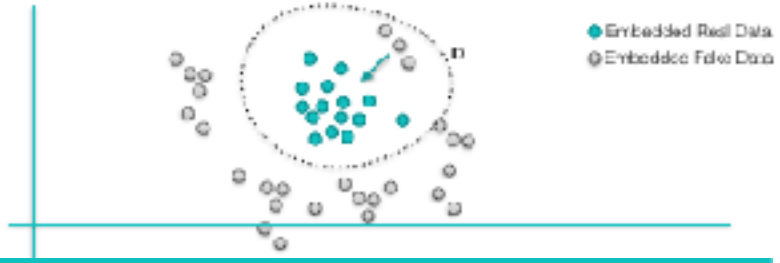
- Logistics

- Agenda
  - More GANs:
    - GAIA
    - Conditional GAN
    - Text-to-image Synthesis
    - BigGAN

Last Time

The Least Squares GAN

- **Observation**: Generated points may (by chance) be classified as real by Discriminator—but they are still not representative of the real data
- **Solution**: Incentivize even correctly classified labels to move toward real data distribution



Maximizing Mutual Information (kinda)

$$I(c;G(z,c)) \geq \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c)$$

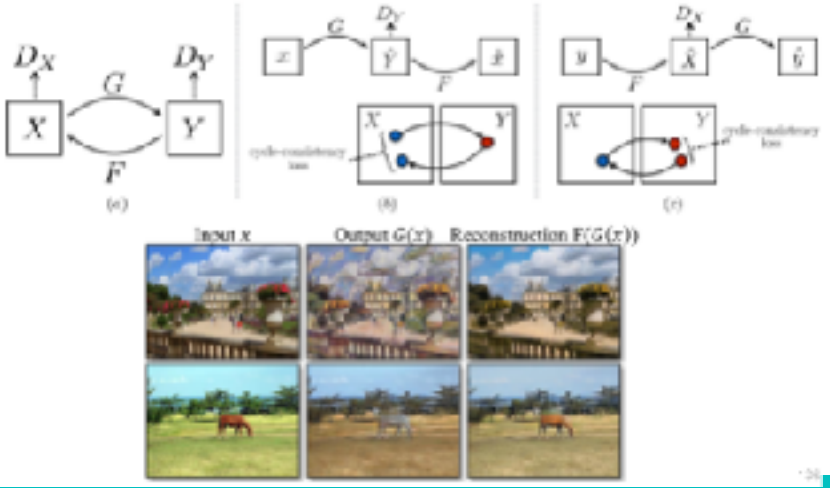
Lemma 5.1 *For random variables X, Y and function $f(x, y)$ under suitable regularity conditions:* $\mathbb{E}_{x\sim X, y\sim Y|x}[f(x,y)] = \mathbb{E}_{x\sim X, y\sim Y|x, x'\sim X|y}[f(x', y)]$

$$\begin{aligned} L_I(G,Q) &= \mathbb{E}_{c\sim P(c),x\sim G(z,c)}[\log Q(c|x)] + H(c) \\ &= \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \\ &\leq I(c;G(z,c)) \end{aligned}$$

Re-parameterize: Make Q arbitrarily rich so you want to sample from!

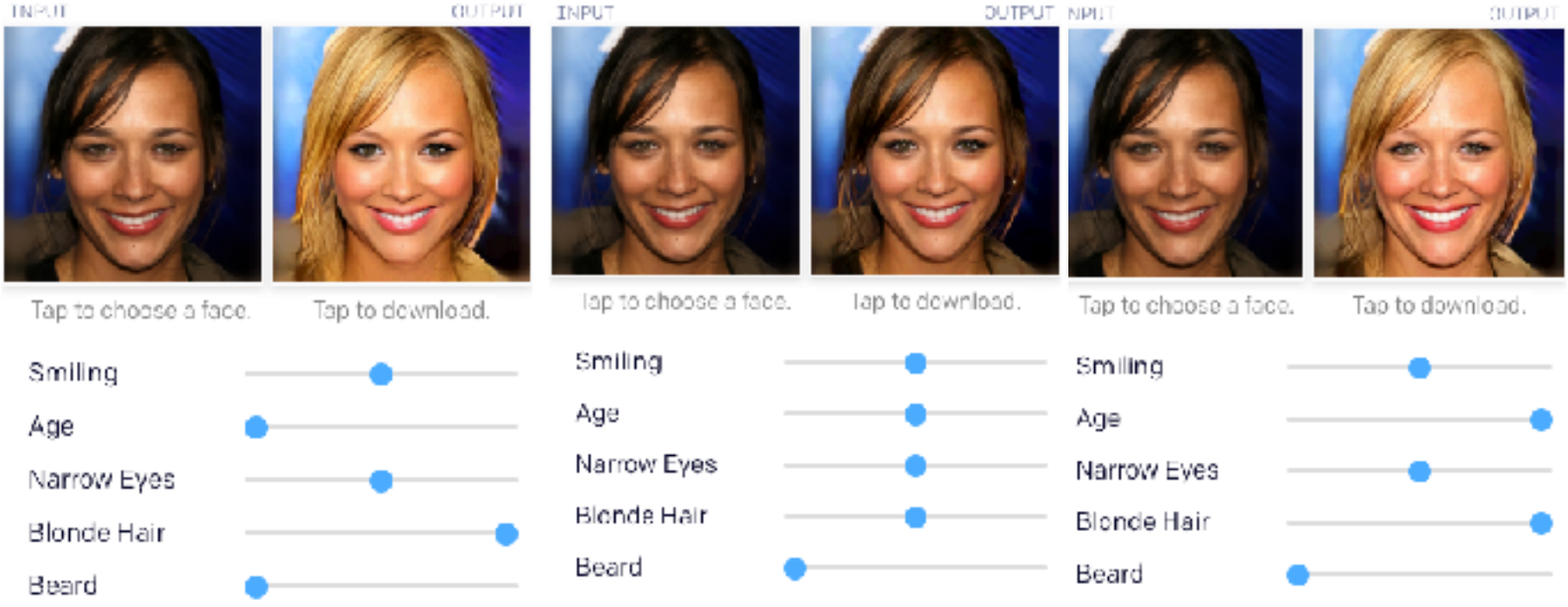
How Objective Function Becomes $\min_{G,Q}\max_{D} V_{InfoGAN}(D,G,Q) = V(D,G) - \lambda L_I(G,Q)$

Defining Cycle Consistency



Full Circle: Back to Style Transfer

GLOW


INPUT
OUTPUT
Tap to choose a face.
Tap to download.
Smiling
Age
Narrow Eyes
Blonde Hair
Beard
INPUT
OUTPUT
Tap to choose a face.
Tap to download.
Smiling
Age
Narrow Eyes
Blonde Hair
Beard
INPUT
OUTPUT
Tap to choose a face.
Tap to download.
Smiling
Age
Narrow Eyes
Blonde Hair
Beard

GAIA

Tim Sainburg

PhD Student @ UCSD studying Psychology, Neuroscience, Anthropogeny, Animal Communication, and Machine Learning

Generative adversarial interpolative autoencoding: adversarial training on latent space interpolations encourage convex latent distributions

Tim Sainburg
UC San Diego
tsainbur@ucsd.edu

Marvin Thielk
UC San Diego
mthielk@ucsd.edu

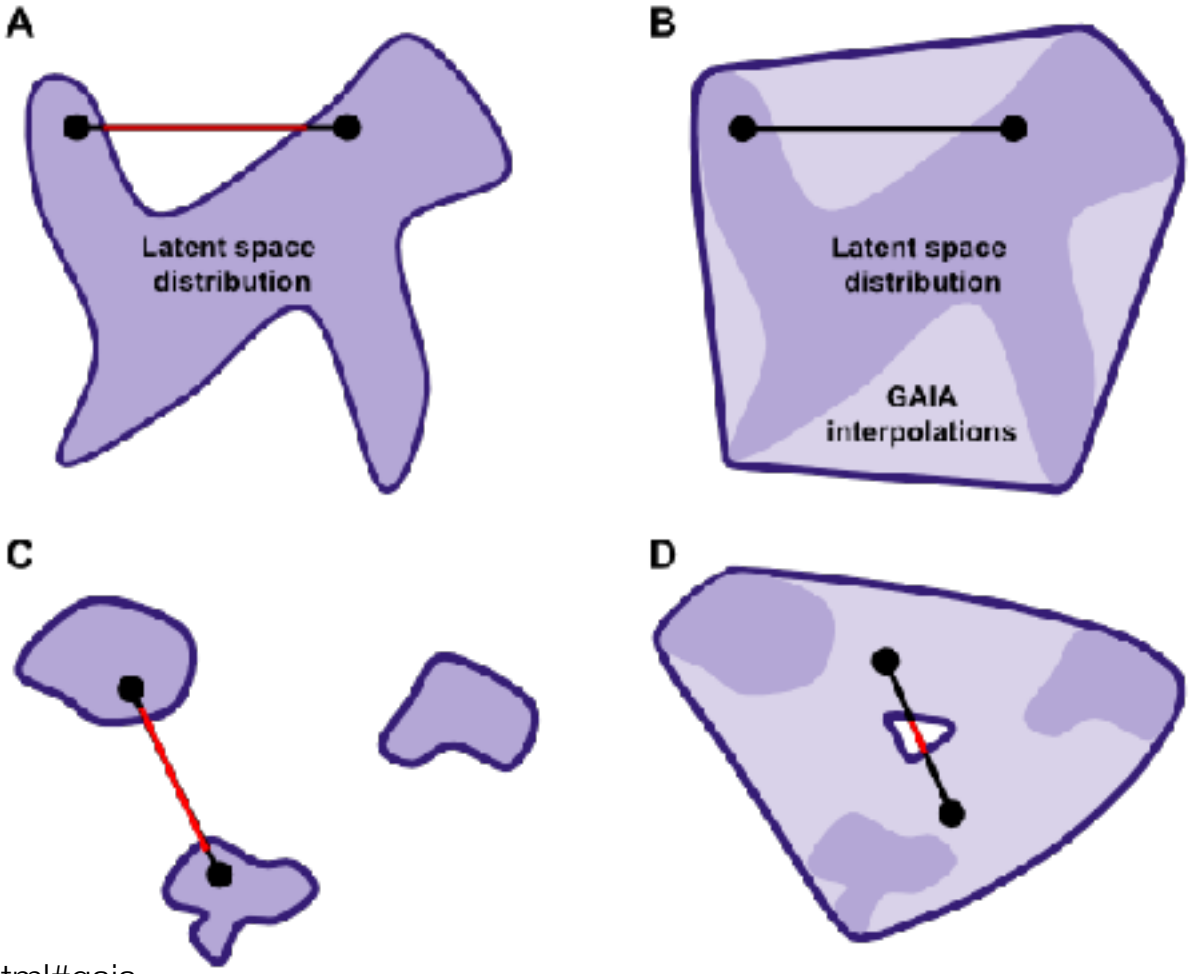
Brad Theilman
UC San Diego
btheilma@ucsd.edu

Benjamin Migliori
SPAWAR
migliori@spawar.navy.mil

Timothy Gentner
UC San Diego
tgentner@ucsd.edu

What does GAIA address?

- Explicitly Interpolate to make latent space as **convex** as possible

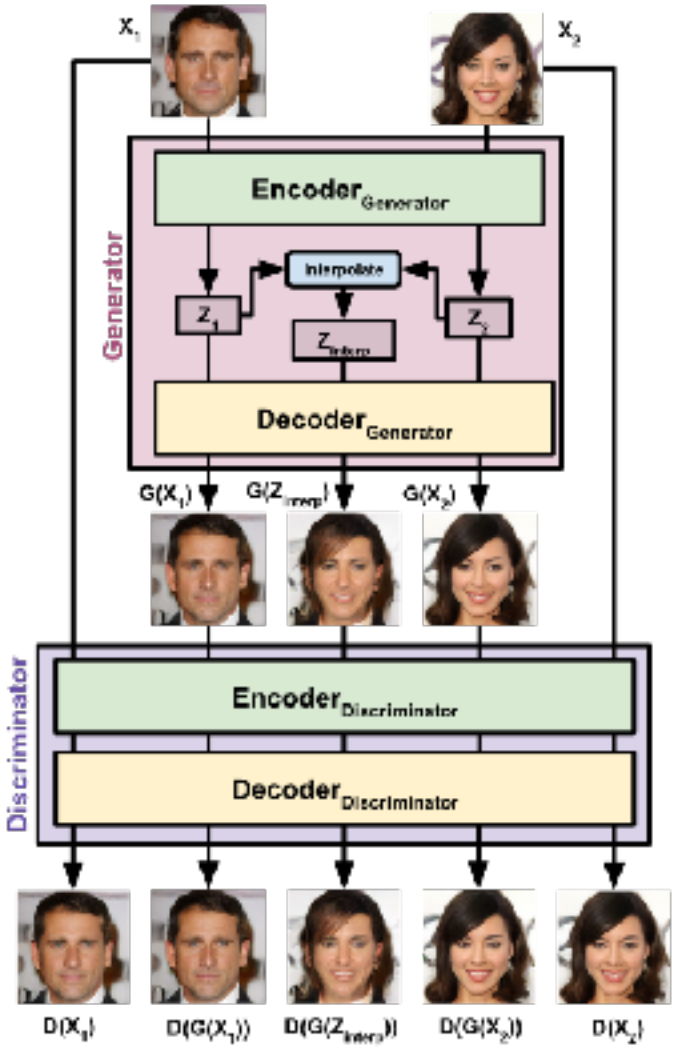


https://timsainburg.com/gaia.html#gaia

How to enforce convexity?

- Change discriminator function entirely
  - Discriminator also auto-encodes an image
  - Maximize **reconstruction error** in generated image for **fake**
  - Minimize **reconstruction error** in generated image for **real**
- Generator takes two images and outputs three
  - Two are simply auto encoded
  - Third Image is interpolated

How to enforce convexity?



Generator: Minimize

$$\mathcal{L}_{Gen} = \|X - D(G(X))\|_1 + \|G(Z_{interp}) - D(G(Z_{interp}))\|_1$$

Discriminator: Minimize

$$\mathcal{L}_{Disc} = \|X - D(X)\|_1 + -\|G(X) - D(G(X))\|_1 + -\|G(Z_{interp}) - D(G(Z_{interp}))\|_1$$

Regularize, for each mini-batch:

$$\mathcal{L}_{dist}(X,Z) = \frac{1}{B}\sum_{i,j}^{B}\left[log_2\left(1 + \frac{(X_i - X_j)^2}{\frac{1}{N}\sum_{i,j}(X_i - X_j)^2}\right) - log_2\left(1 + \frac{(Z_i - Z_j)^2}{\frac{1}{N}\sum_{i,j}(Z_i - Z_j)^2}\right)\right]$$

Results

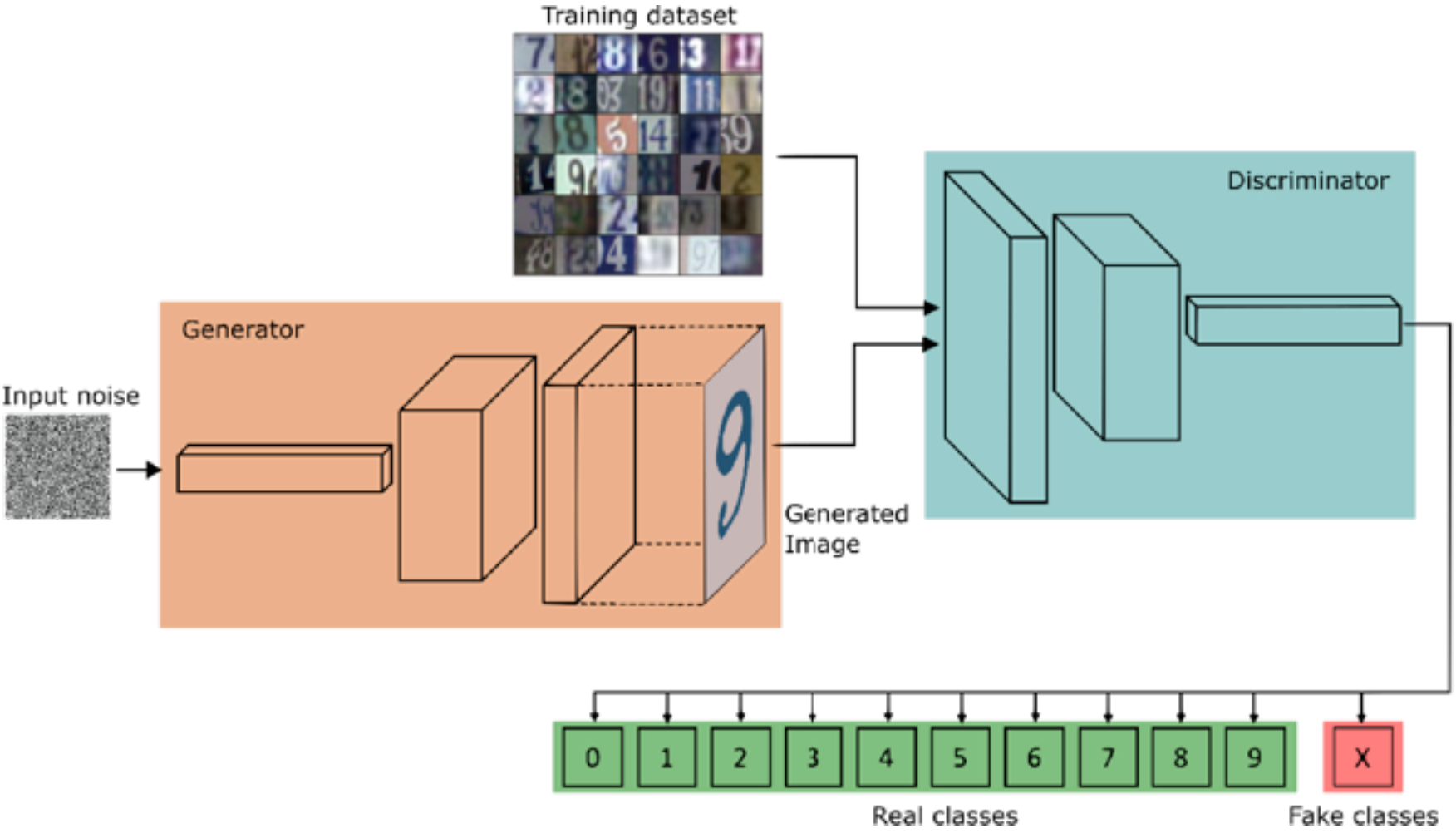
Results


Original
Reconstruction
Young
Old
Male
Female
Bangs
Eyeglasses
Rosy Cheeks
Goatee
Smiling
Pale Skin

GANs for Improving Classifiers

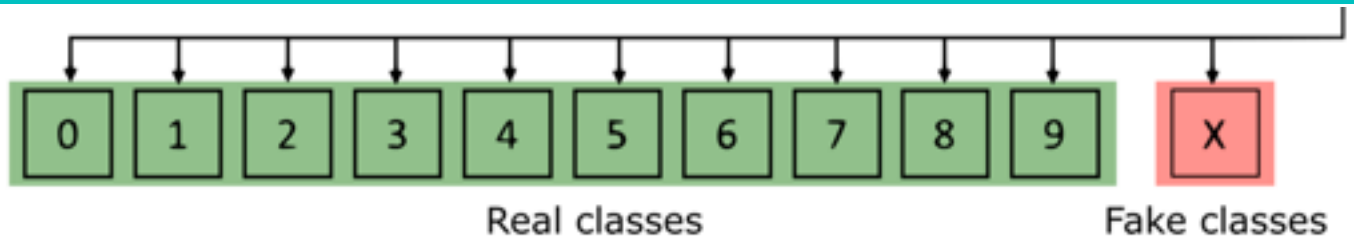
hadou-GAN

Fewer Labels, Still Lots of Data



https://sthalles.github.io/semi-supervised-learning-with-gans/

Minimizing Labels



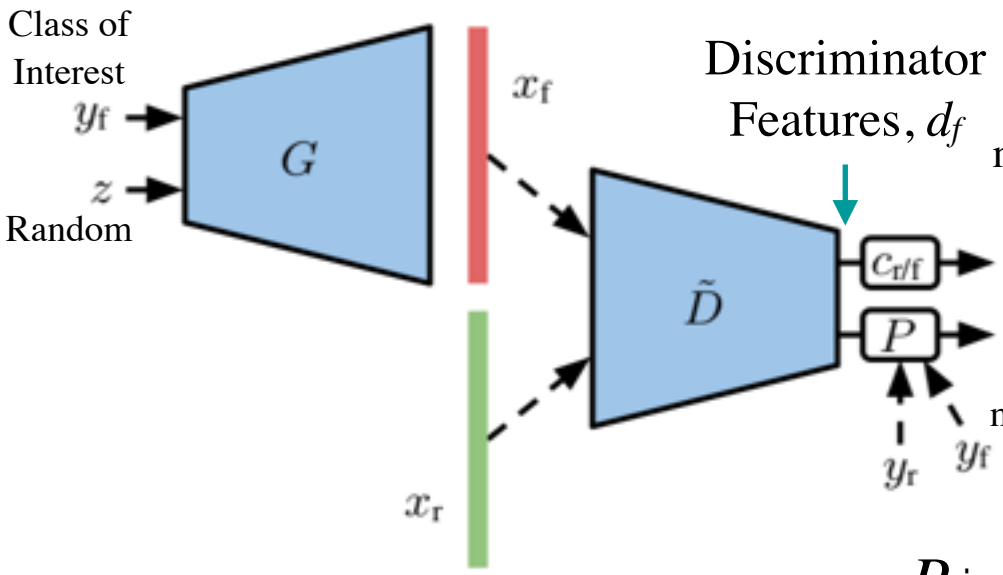
- MNIST Performance
  - 60,000 labels → 1,000 labels
  - 0.6% Error (~SOA)
- SVHN Performance
  - 73,000 labels → 1,000 labels
  - 1.3% Error (~SOA)
- ImageNet
  - 20% Labels (of 1.3M)
  - 10.3% Top-5 Error (slightly worse than SOA)

Need a slightly more complicated loss function

https://arxiv.org/pdf/1903.02271.pdf

Conditional GANs and Self-Supervision

Lucic, Mario, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. "High-Fidelity Image Generation With Fewer Labels." *arXiv preprint arXiv:1903.02271* (2019).



$$\max_d \mathbf{E}_{x \leftarrow p_{data}} \left[\log d(x)\right] + \mathbf{E}_{x \leftarrow g(z)} \left[\log(1 - d(x))\right]$$

Traditional Discriminator Loss, Goodfellow

Class Conditioned Discriminator Loss

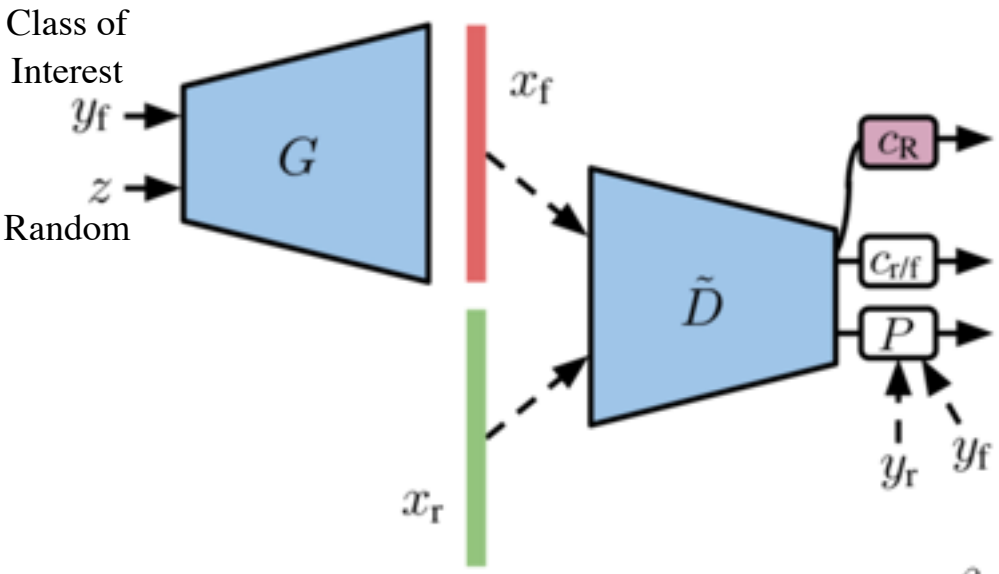
$$\max_d \mathbf{E}_{x \leftarrow p_{data}} \left[\log \tilde{d}(x_r, y_r)\right] + \mathbf{E}_{x \leftarrow g(z, y_f)} \left[\log(1 - \tilde{d}(x_f, y_f))\right]$$

P is trained using outer product

$$\tilde{d} = d_f^T W y_{enc}$$

Conditional GANs and Self-Supervision

Lucic, Mario, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. "High-Fidelity Image Generation With Fewer Labels." *arXiv preprint arXiv:1903.02271* (2019).



New task, detect rotations in image

All these equations are saying is that they classify rotations from real and fake images.

And… nothing in these equations is meaningful except α, β

$$-\frac{\beta}{|\mathcal{R}|}\sum_{r\in\mathcal{R}}\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log p(c_R(\tilde{D}(x^r)=r)]$$

and

$$-\frac{\alpha}{|\mathcal{R}|}\mathbb{E}_{(z,y)\sim p(z,y)}[\log p(c_R(\tilde{D}(G(z,y)^r)=r)],$$

Summary

- When we care about the discriminator for classification:
  - More unsupervised tasks helps feature extraction
  - Conditioning on classes with outer product helps
  - Many other attempts have been tried…
    - But not many work…
  - Open research topic!

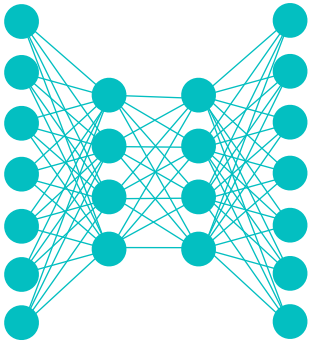
Lecture Notes for

Neural Networks and Machine Learning

Holy GAN-Zooks

Next Time:
Lapan Ch. 1

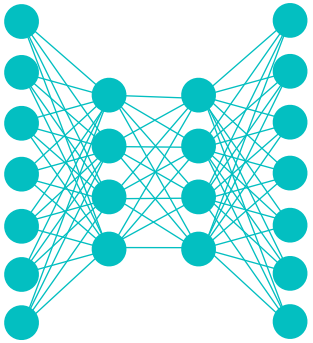
Reading: None



Lecture Notes for

Neural Networks and Machine Learning

Beyond Generation with GANs



Logistics and Agenda

- Logistics

- Agenda
  - Text to mage synthesis
  - BigGAN
  - Adversarial Auto Encoding

Text to Image Synthesis with GANs

Generative Adversarial Text to Image Synthesis

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran REEDSCOT[1], AKATA[2], XCYAN[1], LLAJAN[1]
Bernt Schiele, Honglak Lee SCHIELE[2],HONGLAK[1]

[1] University of Michigan, Ann Arbor, MI, USA (UMICH.EDU)
[2] Max Planck Institute for Informatics, Saarbrücken, Germany (MPI-INF.MPG.DE)

Correspondence Images/Text Features

- Need a correspondence function for text and image representations:

Delta is the 0,1 loss

$$\frac{1}{N}\sum_{n=1}^{N}\Delta(y_n, f_v(v_n)) + \Delta(y_n, f_t(t_n))$$

classify image — classify text

CNN — LSTM or text CNN

sampled from text for class y

$$f_v(v) = \arg\max_{y\in\mathcal{Y}} \mathbb{E}_{t\sim\mathcal{T}(y)}[\phi(v)^T\varphi(t))]$$

sampled from images for class y

$$f_t(t) = \arg\max_{y\in\mathcal{Y}} \mathbb{E}_{v\sim\mathcal{V}(y)}[\phi(v)^T\varphi(t))]$$

text description * image description

When text and image match, these products should be large, else smaller. Therefore f_v and f_t will learn to make similar text and image features.

Can also add to loss function by interpolating text descriptors of the same class

$$\mathbb{E}_{t_1,t_2\sim p_{data}}[\log(1 - D(G(z, \beta t_1 + (1-\beta)t_2)))]$$

Architecture

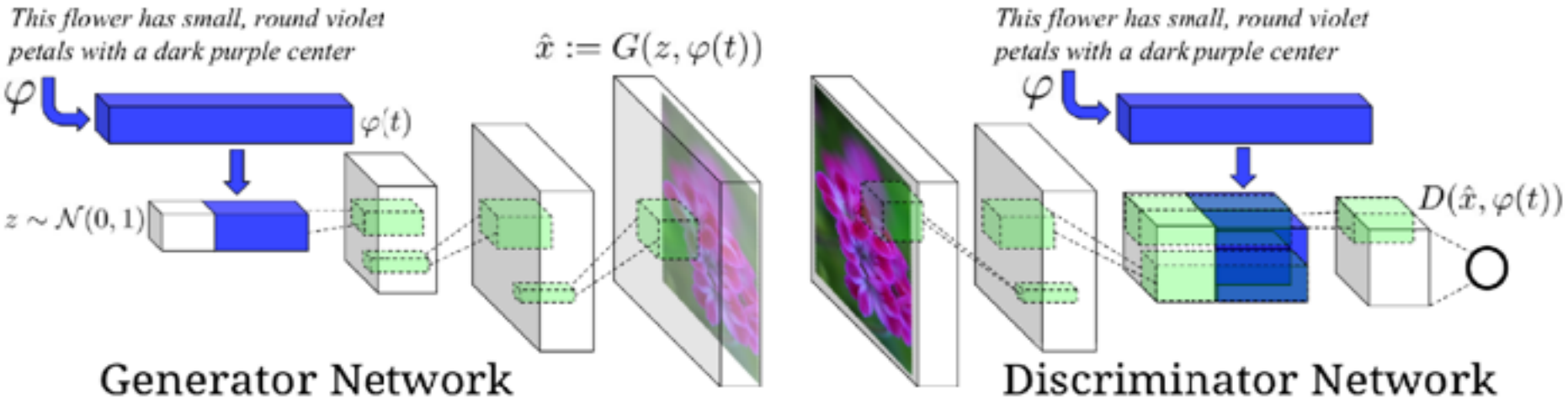


Figure 2. Our text-conditional convolutional GAN architecture. Text encoding $\varphi(t)$ is used by both generator and discriminator. It is projected to a lower-dimensions and depth concatenated with image feature maps for further stages of convolutional processing.

Algorithm 1 GAN-CLS training algorithm with step size α, using minibatch SGD for simplicity.

Input: minibatch images x, matching text t, mismatching $\hat{t}$, number of training batch steps S
for $n = 1$ **to** S **do**
  $h \leftarrow \varphi(t)$ {Encode matching text description}
  $\hat{h} \leftarrow \varphi(\hat{t})$ {Encode mis-matching text description}
  $z \sim \mathcal{N}(0, 1)^Z$ {Draw sample of random noise}
  $\hat{x} \leftarrow G(z, h)$ {Forward through generator}
  $s_r \leftarrow D(x, h)$ {real image, right text}
  $s_w \leftarrow D(x, \hat{h})$ {real image, wrong text}
  $s_f \leftarrow D(\hat{x}, h)$ {fake image, right text}
  $\mathcal{L}_D \leftarrow \log(s_r) + (\log(1 - s_w) + \log(1 - s_f))/2$
  $D \leftarrow D - \alpha \partial \mathcal{L}_D / \partial D$ {Update discriminator}
  $\mathcal{L}_G \leftarrow \log(s_f)$
  $G \leftarrow G - \alpha \partial \mathcal{L}_G / \partial G$ {Update generator}
end for

Results Not Satisfying

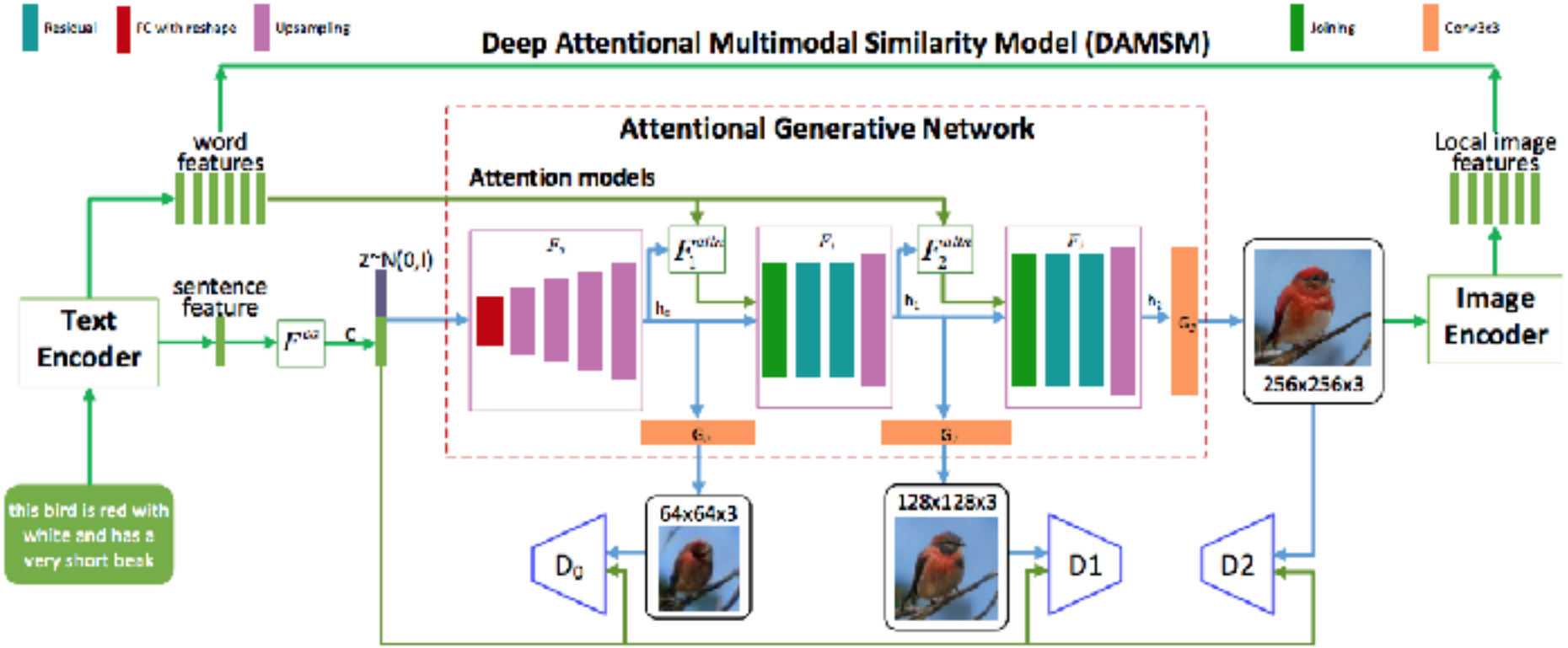
- So we need more processing!
- Add attention!

AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks

Tao Xu*[1], Pengchuan Zhang[2], Qiuyuan Huang[2],
Han Zhang[3], Zhe Gan[4], Xiaolei Huang[1], Xiaodong He[2]

[1]Lehigh University [2]Microsoft Research [3]Rutgers University [4]Duke University
{tax313, xih206}@lehigh.edu, {penzhan, qihua, xiaohe}@microsoft.com
han.zhang@cs.rutgers.edu, zhe.gan@duke.edu

New Process is Similar



Progressive Growing of Size and Multiple Discriminators

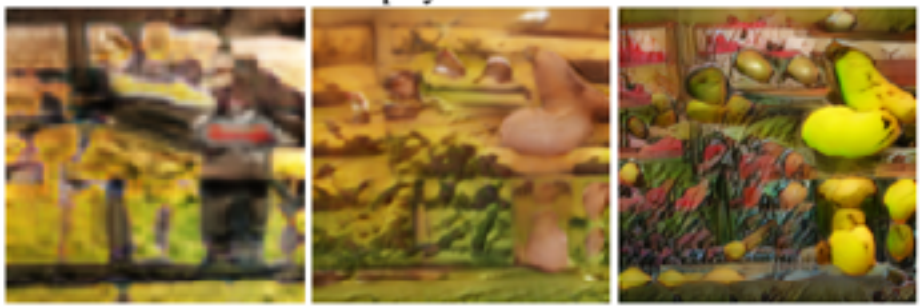
Results are very satisfying!

the bird has a yellow crown and a black eyering that is round

this bird has a green crown black primaries and a white belly

a photo of a homemade swirly pasta with broccoli carrots and onions

a fruit stand display with bananas and kiwi



this bird has wings that are black and has a white belly

this bird has wings that are red and has a yellow belly

this bird has wings that are blue and has a red belly

BigGAN

In a field with 1000's of competing papers,
BigGAN is here to use the most meaningful
Portions of each paper and put them into
One BIG paper.

BigGAN

- This is simply an agglomeration of eveything we already know about GANs from 2016-2019
- Training is hard, so use good loss and heuristics
  - large batches, feature matching
- Scale up progressively, with multiple discriminators at each scale
  - use attention, conditional classes, spectral normalization, moving average of weights, orthogonal weight initialization, skip connections, orthogonal regularizers
  - truncation trick: sample a wide range during training, then truncate for generating results

- Large Scale GAN Training for High Fidelity Natural Image Synthesis, 2018.
- Large Scale GAN Training for High Fidelity Natural Image Synthesis, ICLR 2019. 2019
- Self-Attention Generative Adversarial Networks, 2018.
- A Learned Representation For Artistic Style, 2016.
- Spectral Normalization for Generative Adversarial Networks, 2018.
- Progressive Growing of GANs for Improved Quality, Stability, and Variation, 2017.
- Exact Solutions To The Nonlinear Dynamics Of Learning In Deep Linear Neural Networks, 2013.
- Neural Photo Editing with Introspective Adversarial Networks, 2016.

https://machinelearningmastery.com/a-gentle-introduction-to-the-biggan/

BigGAN Results

- Go read that paper!

BigGAN Results

- Go read that paper!

256 x 256

BigGAN Results

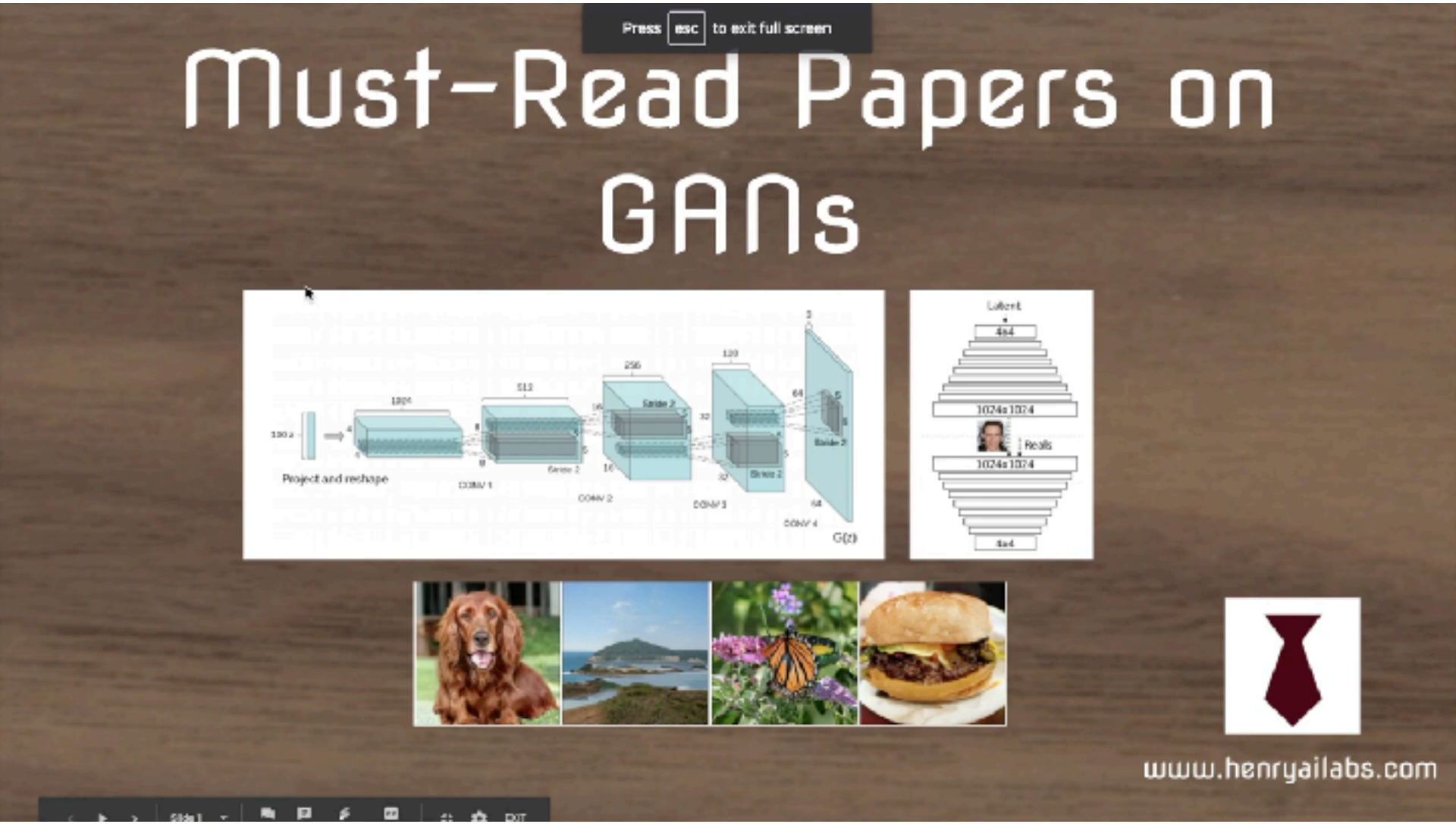
- Go read that paper!

512 x 512

The GAN Takeaways

- Approximation reigns supremum, err maximum
  - But do not be too skeptical
  - … be skeptical of your skepticism
- Intractable mathematics encourages researchers to find their method works through poor practices
- Its still unclear what loss actually works and how to incentivize different behaviors, but we are getting better

A Quick Summary



https://towardsdatascience.com/must-read-papers-on-gans-b665bbae3317

Generative Teaching Networks

- Insert Explanation here for future semesters
- Released December 2019
- https://eng.uber.com/generative-teaching-networks/

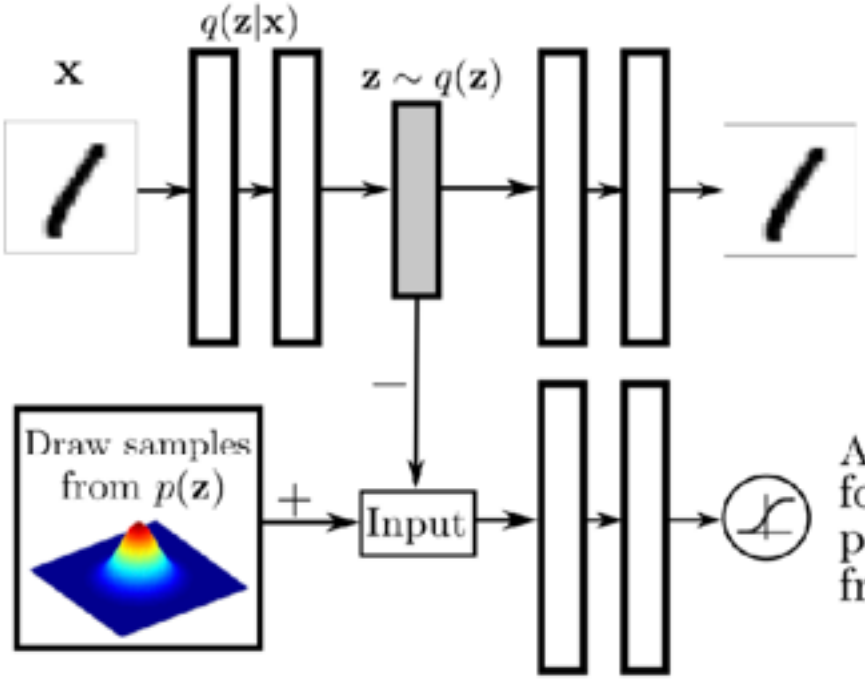
Adversarial Auto Encoding

Do we need something more than VAE?

- Arguments for Yes:
  - ELBO is flawed!
  - Assumption of Normal distributions to q(z) is limiting
  - Training tends to be slow
  - Manifold of distributions do not cover the latent space completely
  - We can't incorporate distributions separately for different classes without reformulating loss function
- Arguments for No:
  - It seems hard, how can we research methods that aren't low hanging fruit? Plus the VAE math was like really hard for me to understand so this is not going to be very fun, guaranteed. Ah, fine lets look at it.

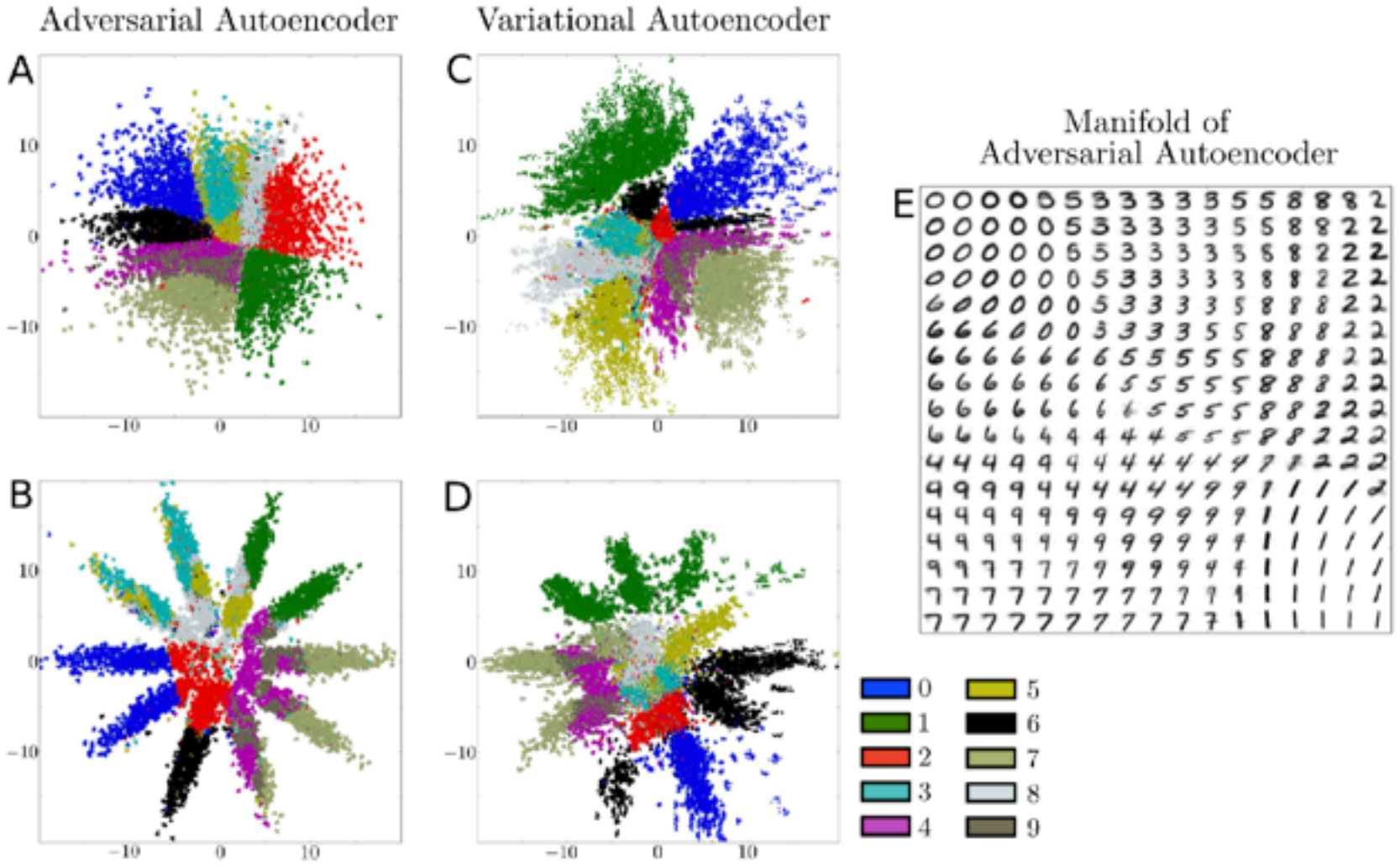
The Main Idea

- How can we enforce constraints on the latent space with GANs?



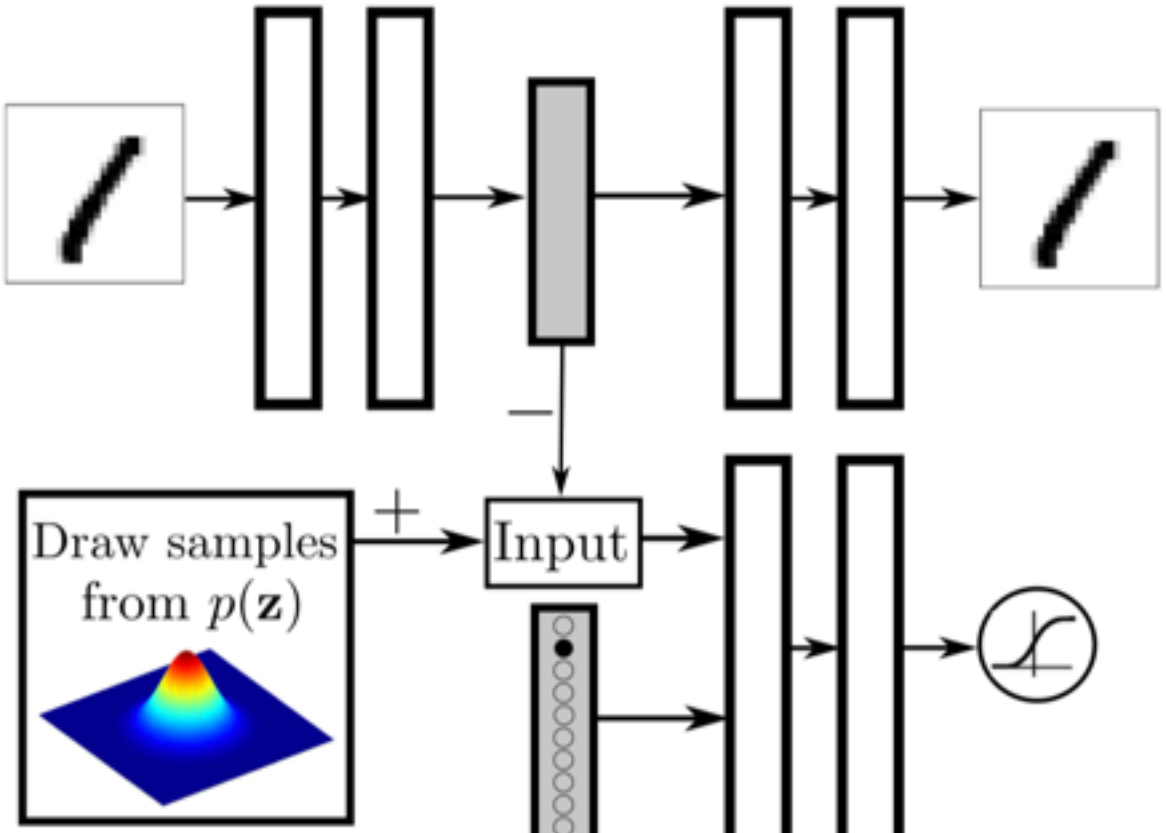
Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, **Ian Goodfellow**, and Brendan Frey. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Arbitrary Prior Distributions



Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, **Ian Goodfellow**, and Brendan Frey. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Sampling From Classes

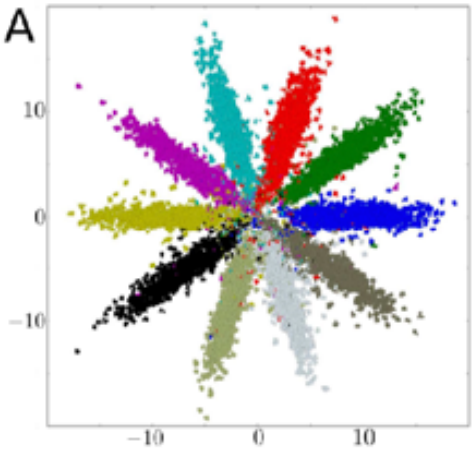


Draw Samples According to
Specific Class Prior Distributions

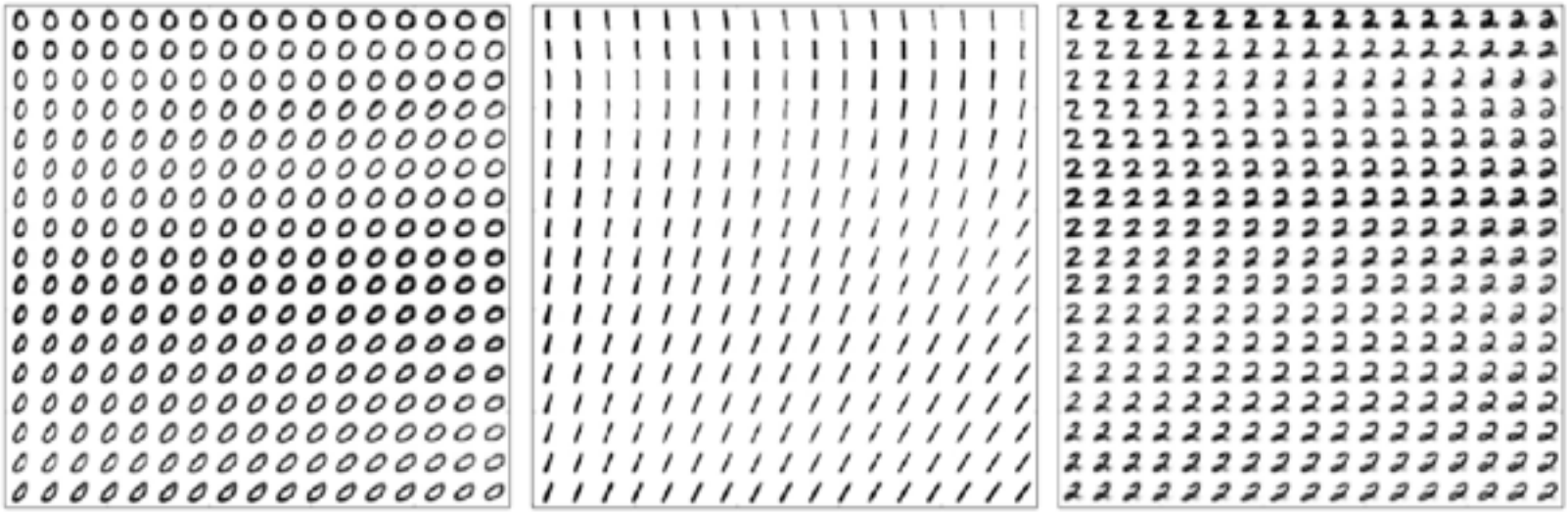
Provide Class Chosen
as One Hot Vector in Discriminator

Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, **Ian Goodfellow**, and Brendan Frey. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Conditional Class Latent Spaces

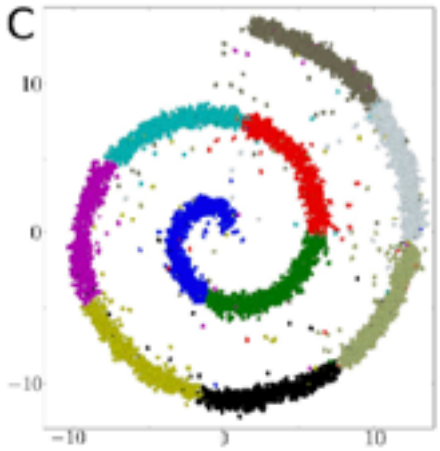


Sample Along Main Axis of the Gaussian Component for Each Digit

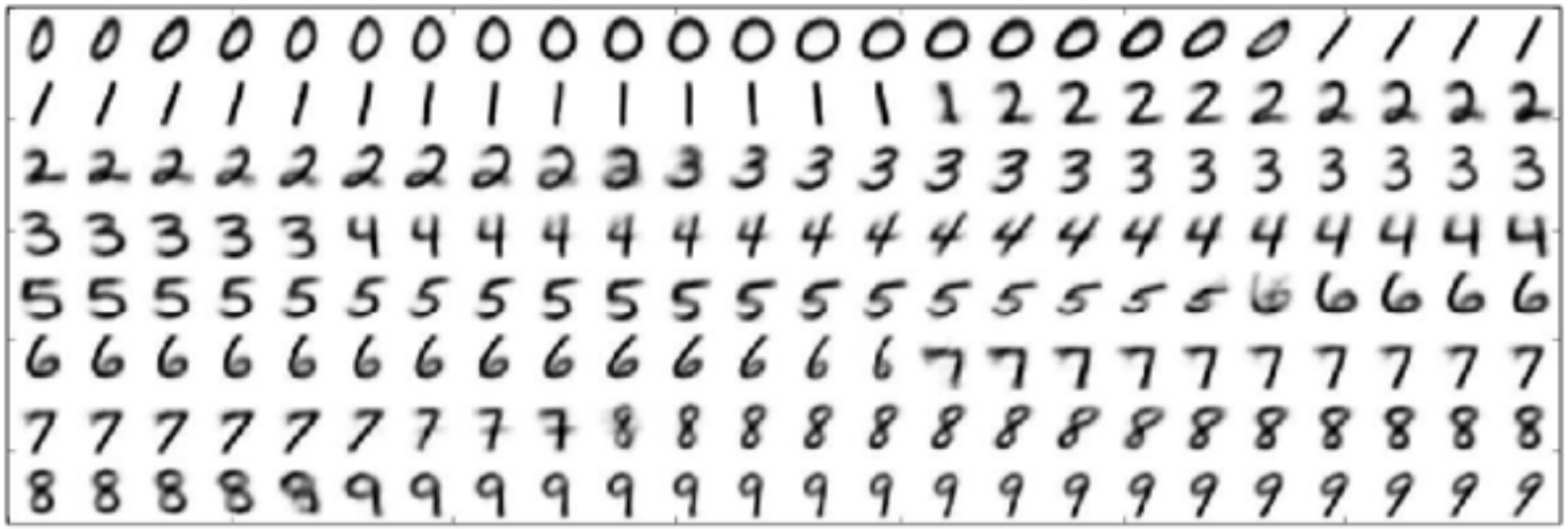


Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, **Ian Goodfellow**, and Brendan Frey. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Conditional Class Latent Spaces

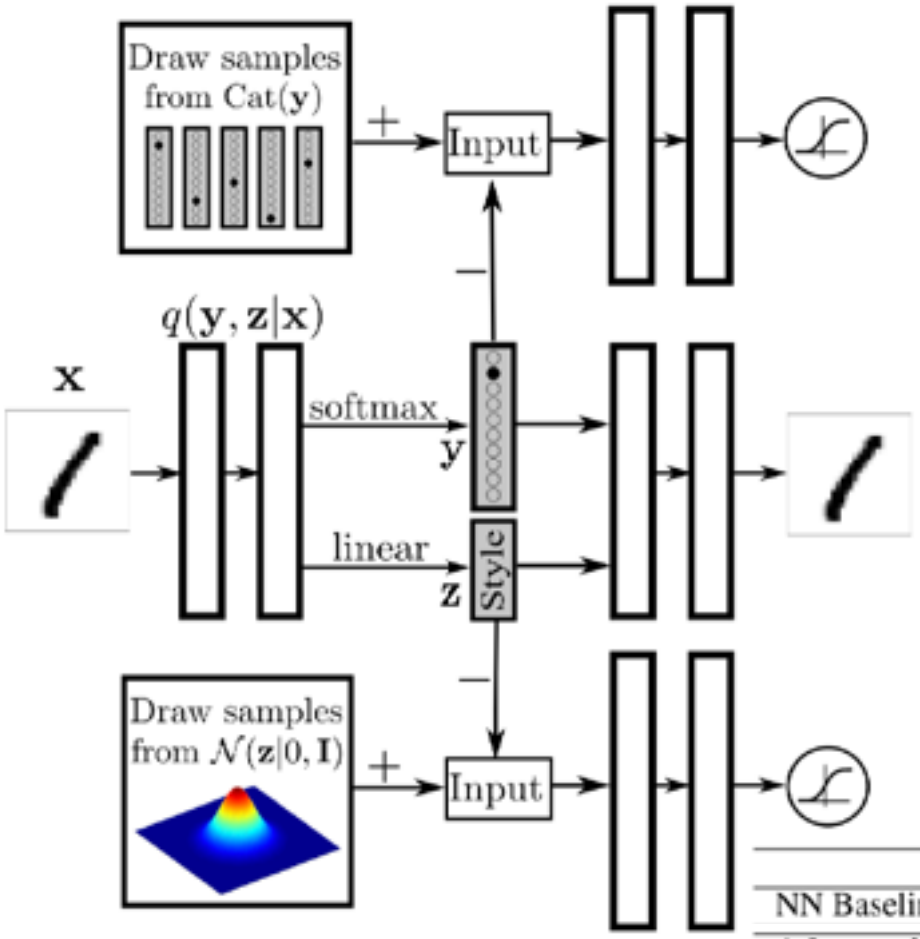


Sample Along Swiss Roll Axis



Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, **Ian Goodfellow**, and Brendan Frey. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Semi-Supervised Classification



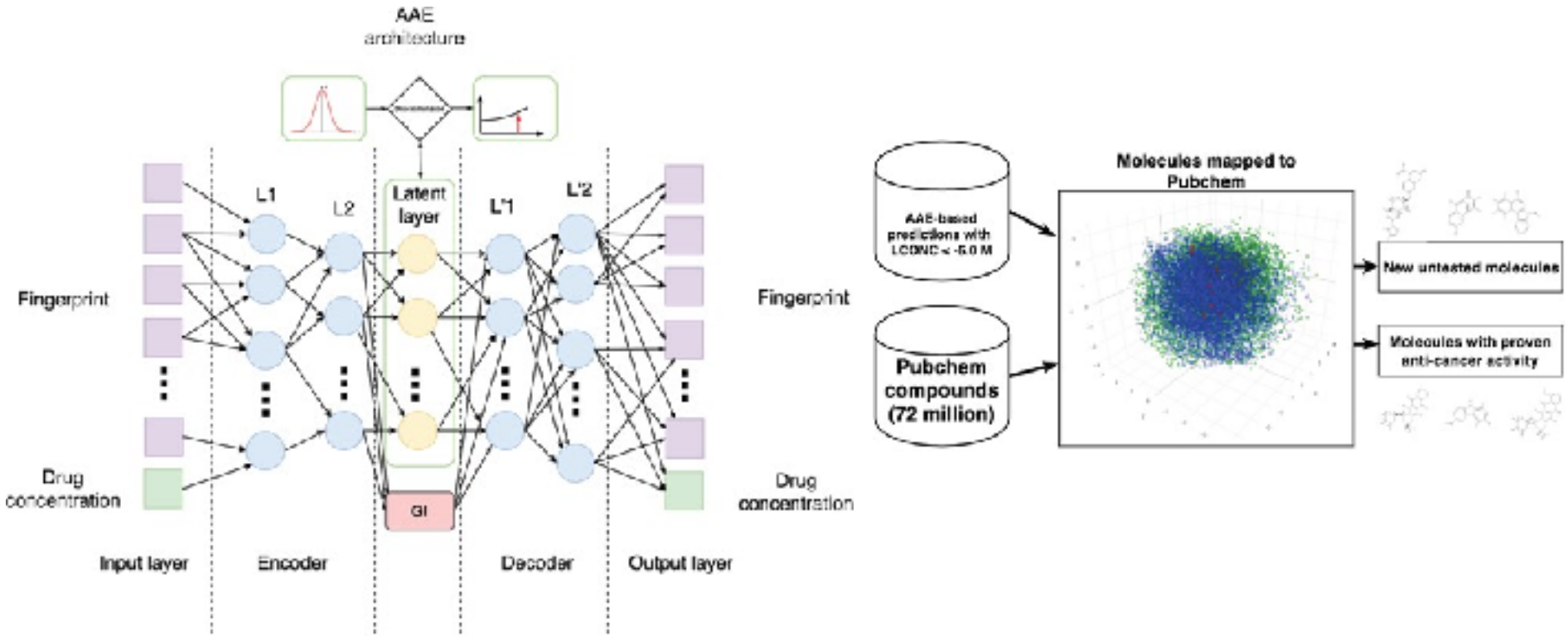
- Top Discriminator
  - Draw samples from one hot encoded labels
  - Ensures aggregated posterior matches class distributions
- Bottom Discriminator
  - Same as previous
  - Constrains latent representation
- Supervised Training
  - After GAN training:
  - Use small mini-batches on $q(y|z)$

	MNIST (100)	MNIST (1000)	MNIST (All)
NN Baseline	25.80	8.73	1.25
Adversarial Autoencoders	1.90 (±0.10)	1.60 (±0.08)	0.85 (±0.02)

Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, **Ian Goodfellow**, and Brendan Frey. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Other Uses For AAE

- Molecular Fingerprinting



Kadurin, Artur, Alexander Aliper, Andrey Kazennov, Polina Mamoshina, Quentin Vanhaelen, Kuzma Khrabrov, and Alex Zhavoronkov. "The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology." *Oncotarget* 8, no. 7 (2017): 10883.

Other Uses For AAE

- Chaos

The Newest AI-Enabled Weapon: 'Deep-Faking' Photos of the Earth

Step 1: Use AI to make undetectable changes to outdoor photos. Step 2: release them into the open-source world and enjoy the chaos.

Patrick Tucker • March 29, 2019

A Quick Aside

Meet the University Ph.D. Fellows

Funding / Ph.D. Fellows / Xinyi Ding

Xinyi Ding

Ph.D., Computer Science

Hometown: China

What is your research area?
Ubiquitous Computing, Mobile Health

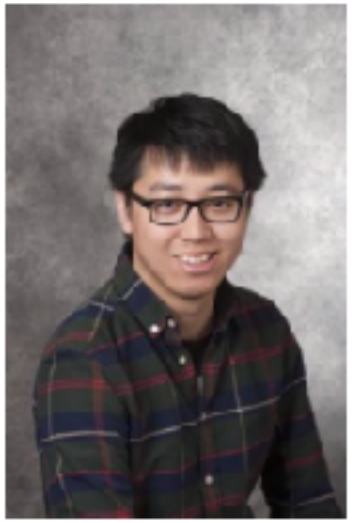
What is something cool about your field?
We use mobile devices to sensor medical quantity out of clinic and build much cheaper alternatives that could change peoples' lives.

What is the best thing you've done as a graduate student at SMU so far?
We are developing an ios app that uses computer vision technologies to detect human emotions. How exciting it is if your smart phone could read your emotions!

What is your favorite thing to do in Dallas?
I enjoy the weather here.

What do you wish you'd known before starting graduate school?
Read as many papers as you can about your intended research field.

What is your favorite leisure activity?
Watching movies and playing basketball.

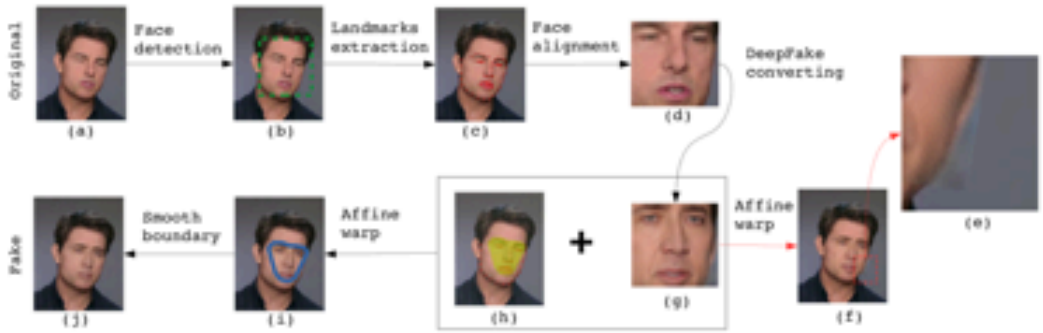


(a) (b) (c)

Figure 1: (a) The input image. (b) The result of face swapping with Nicolas Cage using our method. (c) The result of a manual face swap (source: http://niccageaseveryone.blogspot.com).

"As deepfake technology becomes more advanced and more accessible, it could pose a threat to United States public discourse and national security, with broad and concerning implications for offensive active measured campaigns targeting the United States."

"US lawmakers say AI Deepfakes have the potential to disrupt every facet of our society." US Congressional Letter



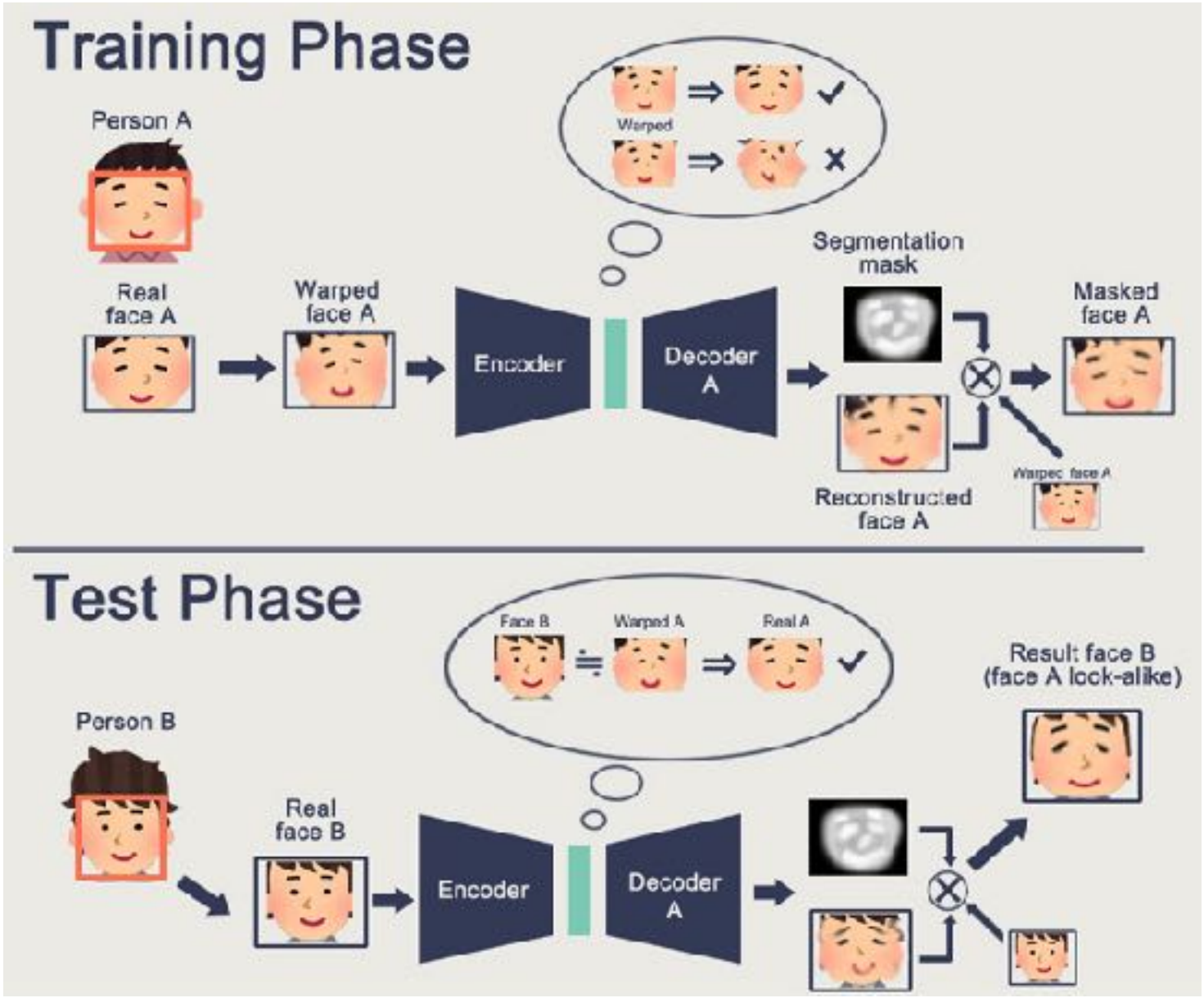
Original

Aligned

Cropped

- 50 different celebrities (from People.com listing)
- Selected celebrity pairs of faces to swap
- Swapped Faces via:
  - Auto-Encoding (Adversarial Network)
  - Pipelining with Convolutional Neural Network
- 1000 – 3000 fake images created per celebrity pair
- 400,000 images for training and evaluation

SMU | Bobby B. Lyle School of Engineering


Training Phase
Person A
Real face A
Warped face A
Warped
Encoder
Decoder A
Segmentation mask
Masked face A
Reconstructed face A
Warped face A
Test Phase
Face B
Warped A
Real A
Person B
Real face B
Encoder
Decoder A
Result: face B (face A look-alike)

Rating Swapped Images

faceface.lyle.smu.edu

- Selected subset of 400 images to rate
- Used paired selection algorithm on website to choose most "needed" image comparison
- Collected **40,000 ratings** from ~200 unique individuals
- **Result**: Sorting of images from most fake to most real

SMU | Bobby B. Lyle School of Engineering

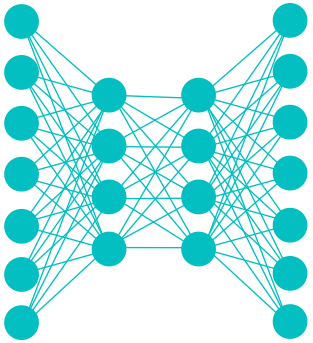
Lecture Notes for

Neural Networks and Machine Learning

Practical GANs

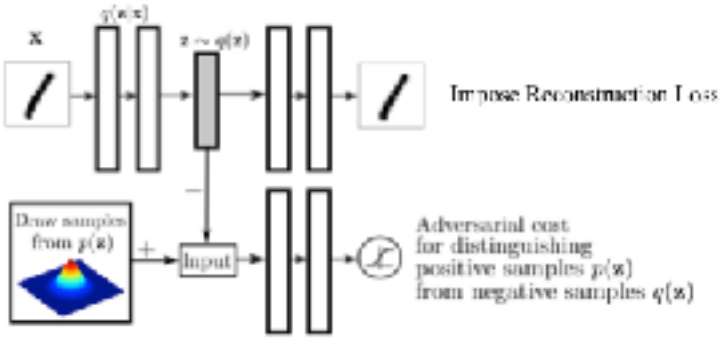
Next Time:
GAN Examples

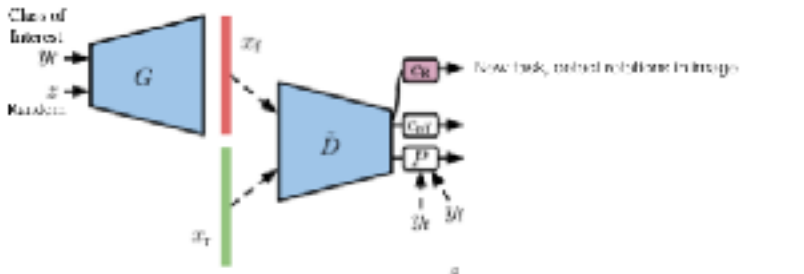
Reading: Chollet CH8



Last Time

- How can we enforce constraints on the latent space with GANs?





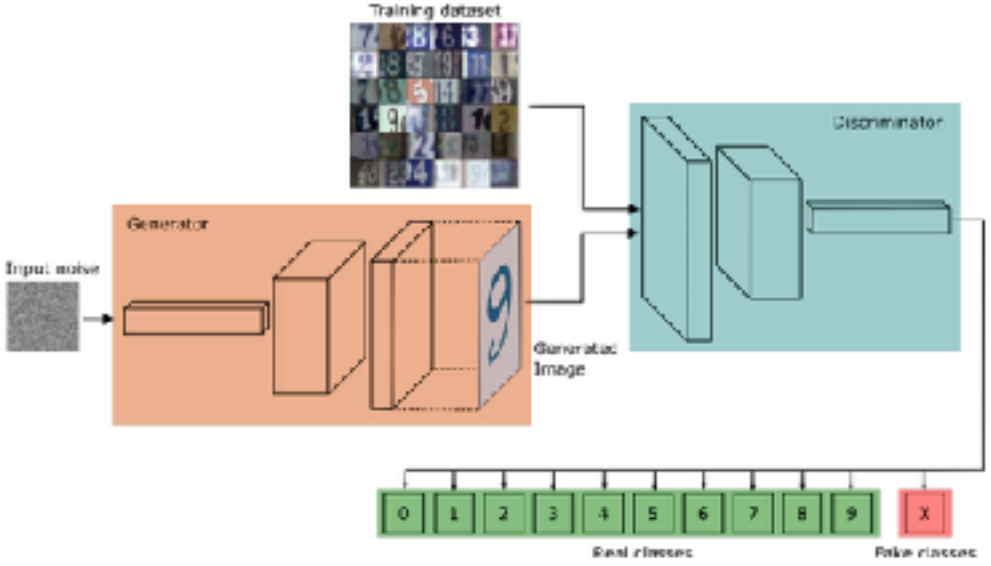
At these equations are saying that they classify real and fake images.

$$-\frac{\beta}{|\mathcal{R}|}\sum_{r\in\mathcal{R}}\mathbb{E}_{x\sim p_{\text{model}}(x)}[\log p(\text{sg}(\tilde{D}(x^r)=r)]$$

and

And... nothing in these equations is meaningful except x, β

$$-\frac{\alpha}{|\mathcal{R}|}\mathbb{E}_{(x,y)\sim q(x,y)}[\log p(\text{sg}(\tilde{D}(G(x,y)^r)=r)].$$



Backup slides

Title Between Topics

Example Slide

Title

Subtitle

`Follow Along: Notebook Name`