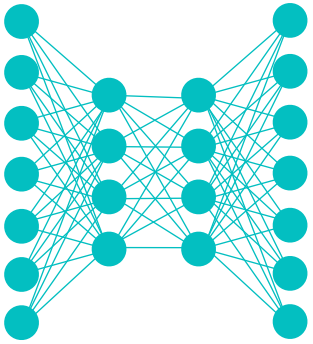
Lecture Notes for

Neural Networks and Machine Learning

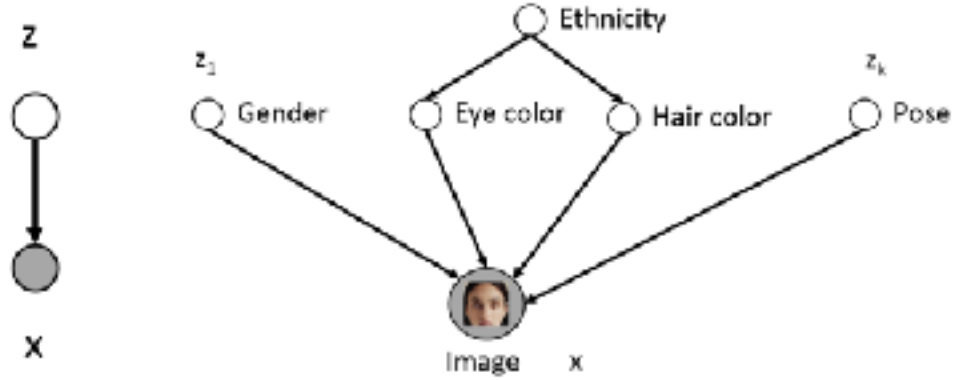
Generative Networks



Logistics and Agenda

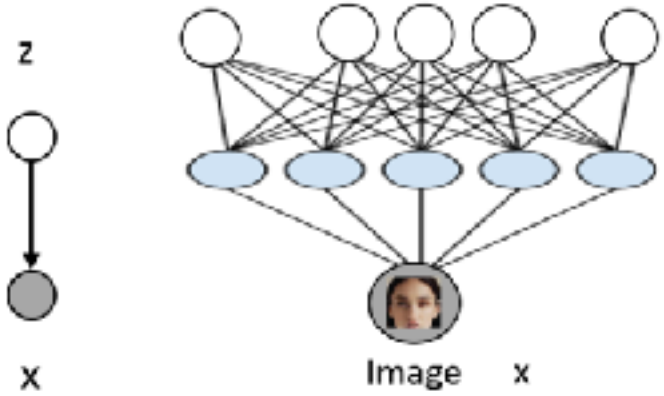
- Logistics
  - Lab Three is Posted
- Agenda
  - A historical perspective of generative Neural Networks
  - Variational Auto-Encoding
  - VAE in Keras Demo

Motivations: Latent Variables



$$p(\mathbf{x} \mid \mathbf{z})$$

Hard: **z** is expertly chosen



$$p(\mathbf{x} \mid \mathbf{z})$$

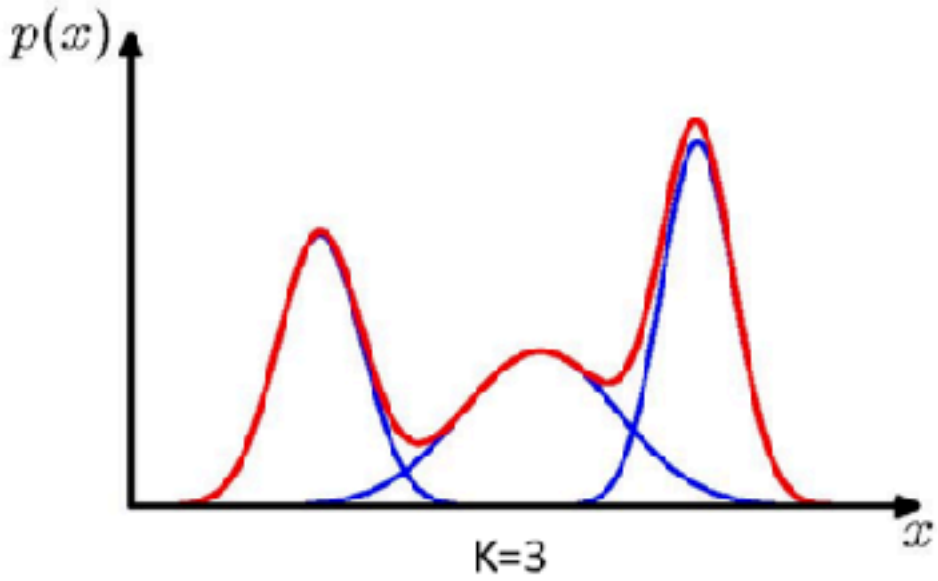
Not as Hard: **z** is trained, latent variables are uncontrolled

Want: $$p(\mathbf{x}) \approx \sum_{\mathbf{z}} p(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z})$$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Motivation: Mixtures for Simplicity

Want: $$p(\mathbf{x}) \approx \sum_{\mathbf{z}} p(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z})$$



- Each latent variable mostly independent of other latent variables
- The sum of various mixtures can approximate most any ditribution
- Good choice for conditional is Normal Distribution
- Can parameterize $p(x|z)$ to be a Neural Network

$$p_\theta(\mathbf{x} \mid \mathbf{z} = k) = \mathcal{N}(\mathbf{x}, \mu_k, \Sigma_k)$$

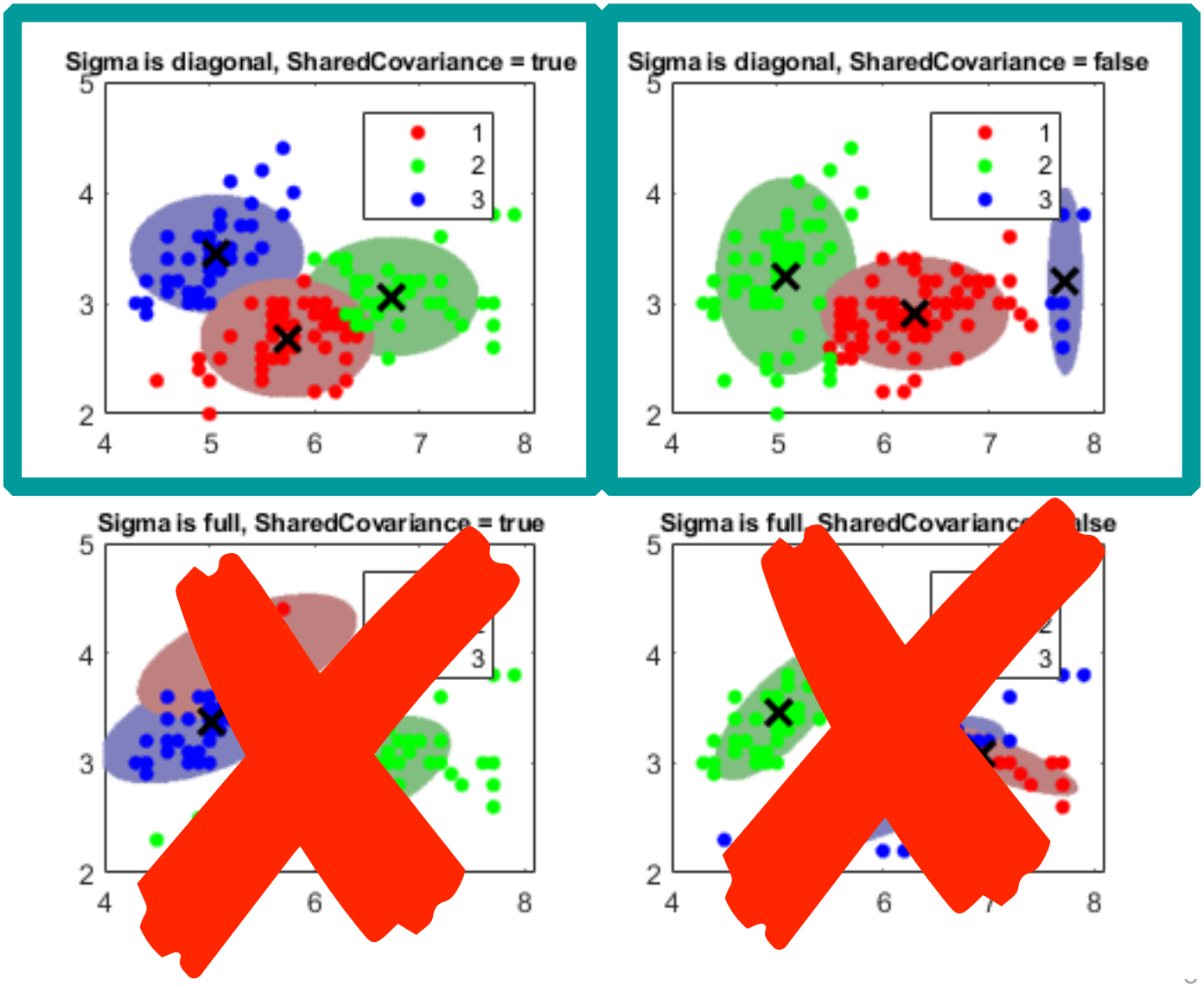
mean and covariance learned

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

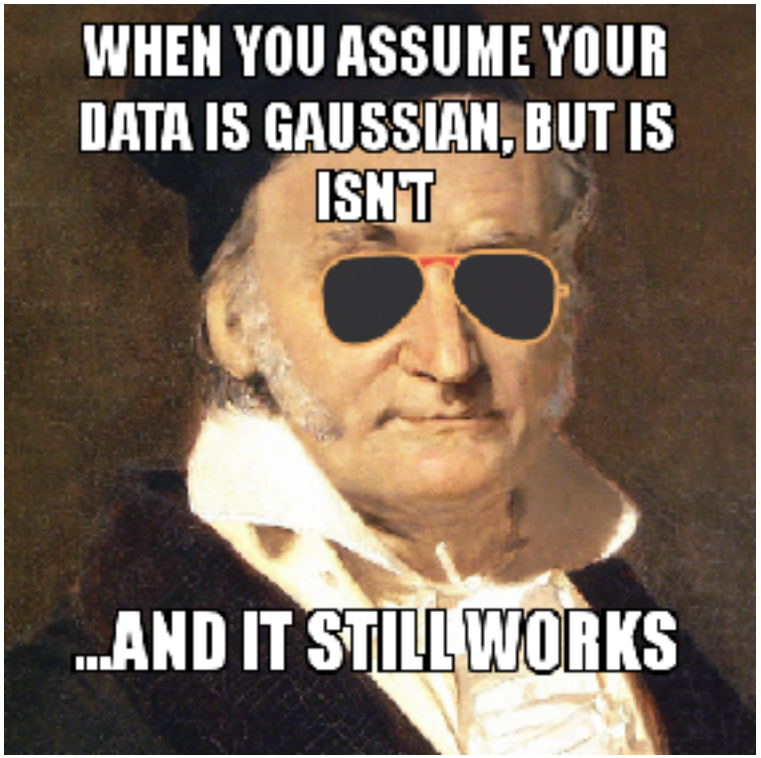
Motivation: Mixtures for Simplicity

$$= \mathcal{N}(\mathbf{x}, \mu_k, \Sigma_k)$$

mean and covariance learned



A History of Generative Networks



Taxonomy of Generative Models

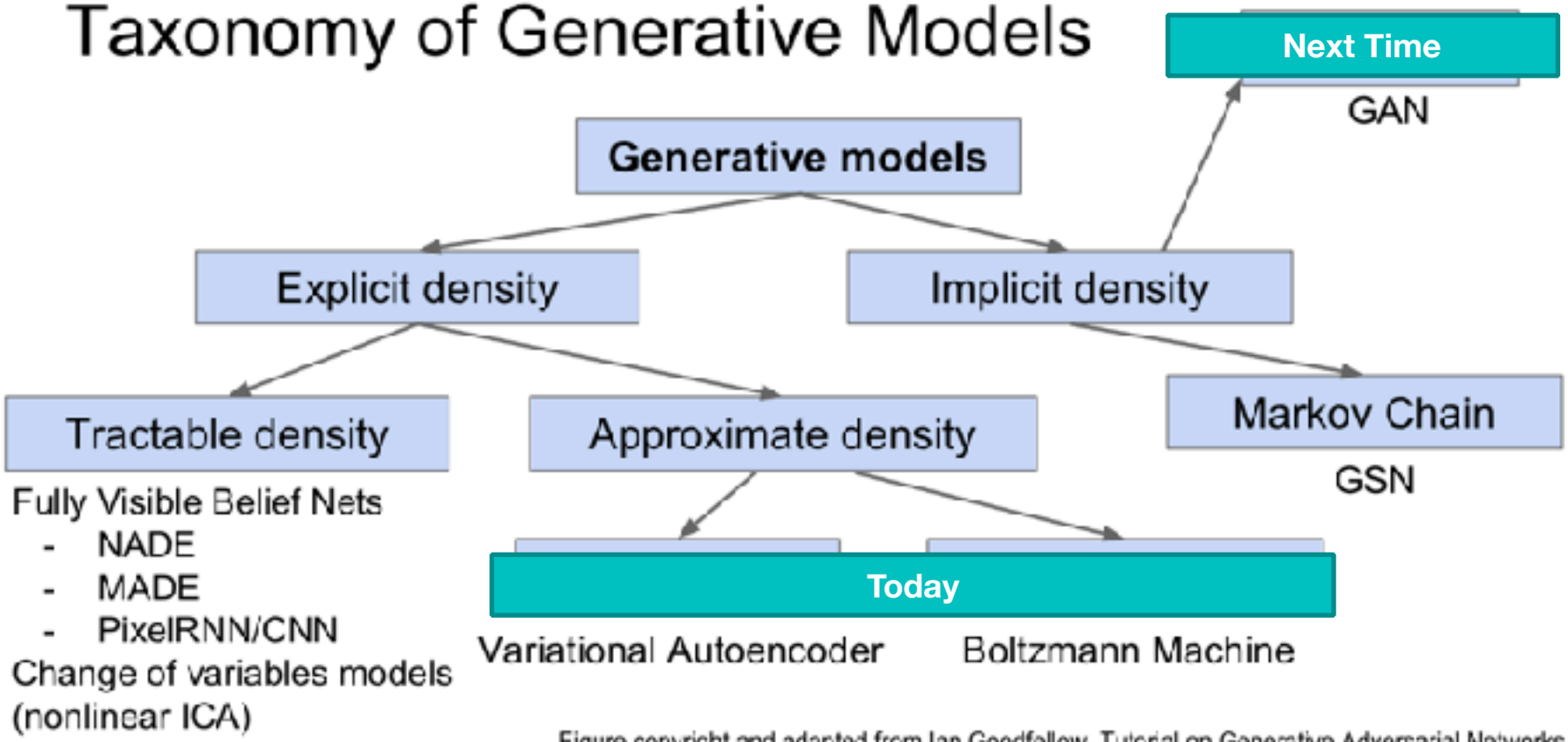


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Deep Generative Models, 2006

- Deep Belief Networks (DBNs)
- Iterative Layer Training (not feed forward)

energy function of the RBM

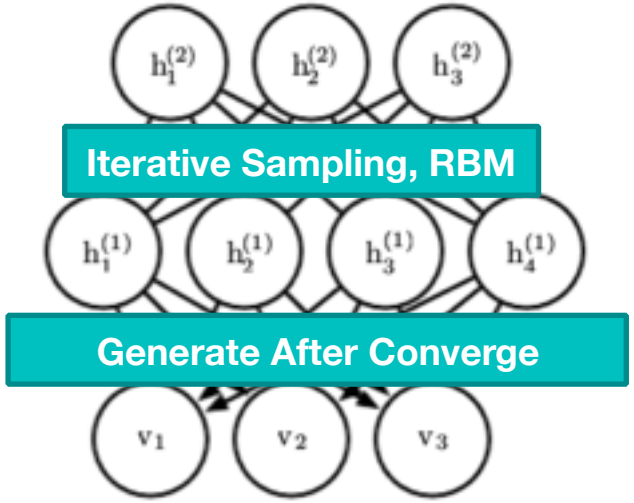
$$P(\boldsymbol{h}^{(l)}, \boldsymbol{h}^{(l-1)}) \propto \exp\left(\boldsymbol{b}^{(l)\top}\boldsymbol{h}^{(l)} + \boldsymbol{b}^{(l-1)\top}\boldsymbol{h}^{(l-1)} + \boldsymbol{h}^{(l-1)\top}\boldsymbol{W}^{(l)}\boldsymbol{h}^{(l)}\right), \quad (20.17)$$

$$P(h_i^{(k)} = 1 \mid \boldsymbol{h}^{(k+1)}) = \sigma\left(b_i^{(k)} + \boldsymbol{W}_{:,i}^{(k-1)\top}\boldsymbol{h}^{(k+1)}\right) \forall i, \forall k \in 1, \ldots, l-2, \quad (20.18)$$

$$P(v_i = 1 \mid \boldsymbol{h}^{(1)}) = \sigma\left(b_i^{(0)} + \boldsymbol{W}_{:,i}^{(1)\top}\boldsymbol{h}^{(1)}\right) \forall i. \quad (20.19)$$

In the case of real-valued visible units, substitute

$$\mathbf{v} \sim \mathcal{N}\left(v; \boldsymbol{b}^{(0)} + \boldsymbol{W}^{(1)\top}h^{(1)}, \boldsymbol{\beta}^{-1}\right) \quad (20.20)$$



$$\mathbf{E}_{v \leftarrow p}[\log p(v)]$$

$p(v)$ is intractable

$$\mathbf{E}_{v \leftarrow p}\left[\mathbf{E}_{h^{(1)} \leftarrow p^{(1)}(h^{(1)}|v)}[\log p^{(2)}(h^{(1)})]\right]$$

optimize with constrastive divergence

i.e., approximation of EM, not Back Prop

To train a deep belief network, one begins by training an RBM to maximize $\mathbb{E}_{\mathbf{v} \sim p_{\text{data}}} \log p(\boldsymbol{v})$ using contrastive divergence or stochastic maximum likelihood. The parameters of the RBM then define the parameters of the first layer of the DBN. Next, a second RBM is trained to approximately maximize

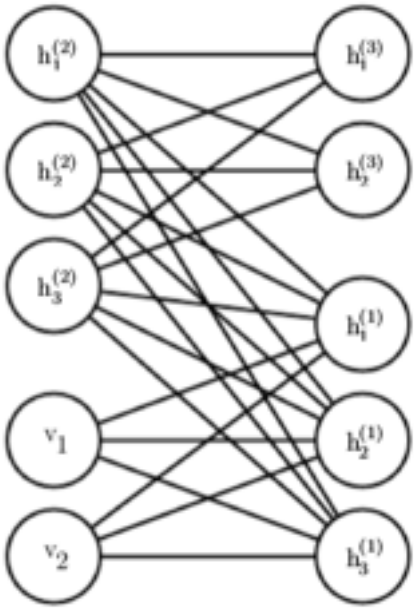
$$\mathbb{E}_{\mathbf{v} \sim p_{\text{data}}} \mathbb{E}_{\mathbf{h}^{(1)} \sim p^{(1)}(\boldsymbol{h}^{(1)}|\boldsymbol{v})} \log p^{(2)}(\boldsymbol{h}^{(1)}) \quad (20.21)$$

where $p^{(1)}$ is the probability distribution represented by the first RBM and $p^{(2)}$ is the probability distribution represented by the second RBM. In other words, the second RBM is trained to model the distribution defined by sampling the hidden units of the first RBM, when the first RBM is driven by the data. This

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

Generative Models, 2009

- Deep Boltzmann Machine



$$P\left(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}\right) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}; \boldsymbol{\theta})\right). \tag{20.24}$$

To simplify our presentation, we omit the bias parameters below. The DBM energy function is then defined as follows:

$$E(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}; \boldsymbol{\theta}) = -\boldsymbol{v}^\top \boldsymbol{W}^{(1)} \boldsymbol{h}^{(1)} - \boldsymbol{h}^{(1)\top} \boldsymbol{W}^{(2)} \boldsymbol{h}^{(2)} - \boldsymbol{h}^{(2)\top} \boldsymbol{W}^{(3)} \boldsymbol{h}^{(3)}. \tag{20.25}$$

We now develop the mean field approach for the example with two hidden layers. Let $Q(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)} \mid \boldsymbol{v})$ be the approximation of $P(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)} \mid \boldsymbol{v})$. The mean field assumption implies that

$$Q(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)} \mid \boldsymbol{v}) = \prod_j Q(h_j^{(1)} \mid \boldsymbol{v}) \prod_k Q(h_k^{(2)} \mid \boldsymbol{v}). \tag{20.29}$$

Not tractable: Can only optimize the Evidence lower bound, ELBO

Approximate via MCMC like **Gibbs Sampling**

$$\mathrm{KL}(Q \| P) = \sum_{\boldsymbol{h}} Q(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)} \mid \boldsymbol{v}) \log\left(\frac{Q(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)} \mid \boldsymbol{v})}{P(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)} \mid \boldsymbol{v})}\right). \tag{20.30}$$

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

Image Generation from Samples

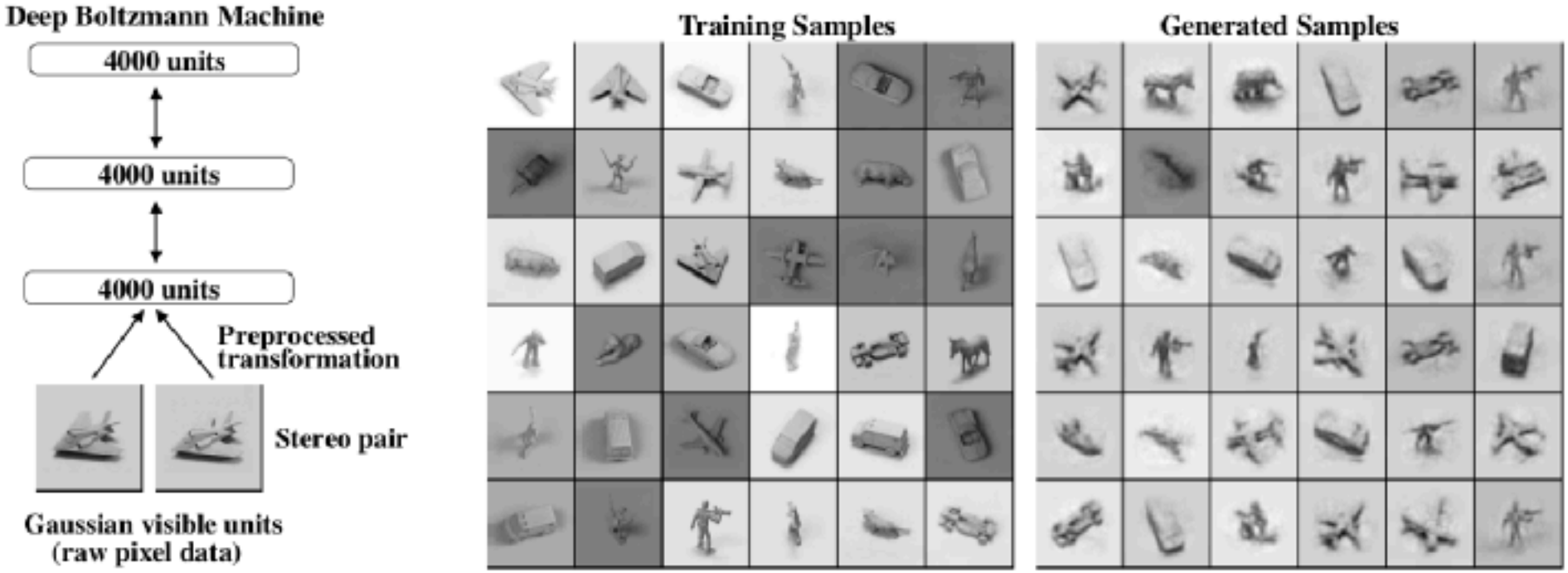


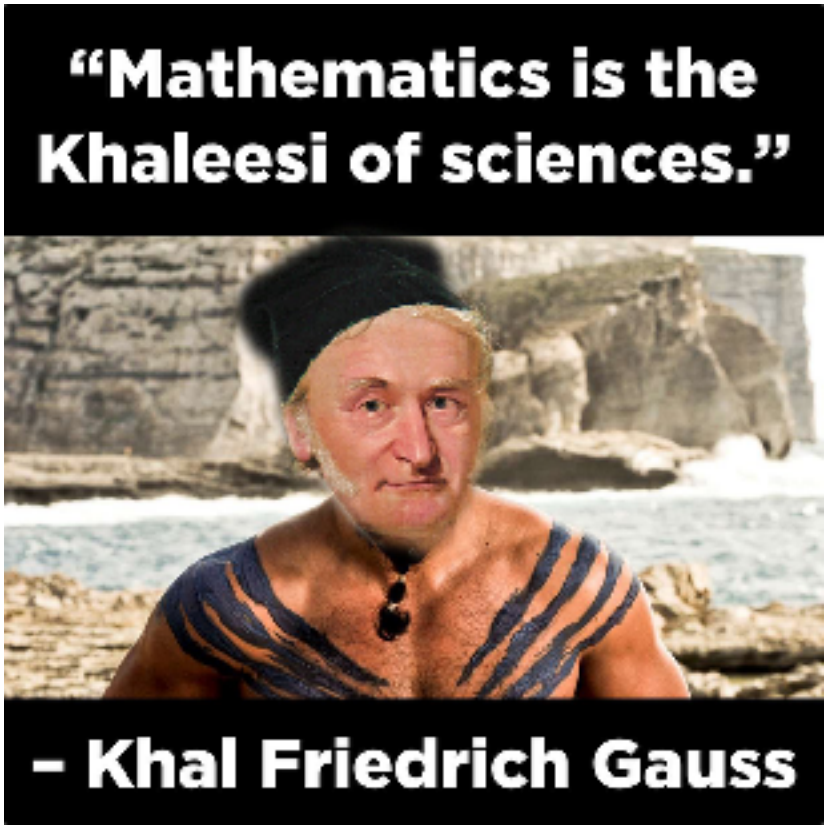
Figure 5: **Left:** The architecture of deep Boltzmann machine used for NORB. **Right:** Random samples from the training set, and samples generated from the deep Boltzmann machines by running the Gibbs sampler for 10,000 steps.

Salakhutdinov, Ruslan, and Geoffrey Hinton. "Deep boltzmann machines." In Artificial intelligence and statistics, pp. 448-455. 2009.

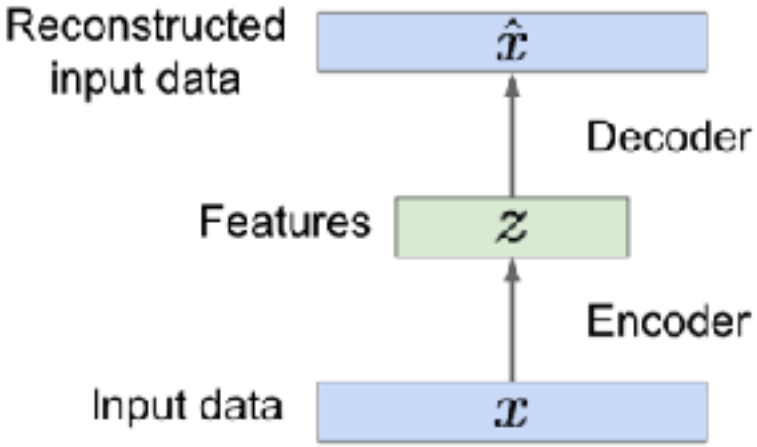
Contemporary Modeling

- DBNs and DBMs have mostly been abandoned
  - Mathematics detracts from popular understanding
  - Often methods using sampling are not scalable
  - Cannot directly use Gradients (no Back Prop) 😢
- Evidence Lower Bound (ELBO) considered "good enough" because … we can't do better computationally
- Popular method for calculating generative networks with ELBO approximation:
  - Variational Auto Encoding
    - Guaranteed NOT to find global minimum
    - But scalable and will converge in finite time

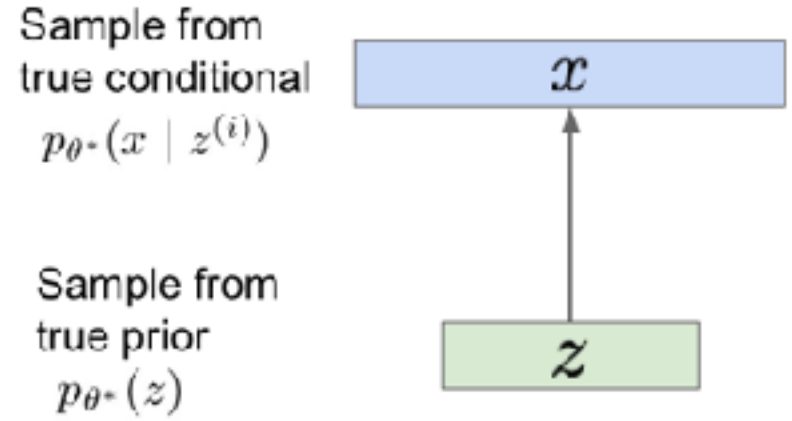
Variational Auto Encoding



Aside: Auto Encoding



Once trained, is it possible to generate data?



- Does this work for simple auto encoding?
  - Nope.
- Learned space is not continuous
- How to sample from the latent space?
- Features could be highly correlated
- Need to define some constraints on the latent space…

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

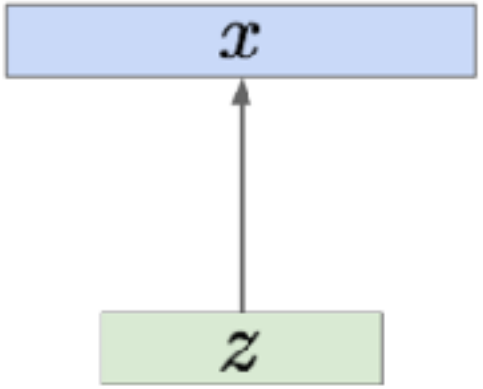
Reasonable constraints for p(z)?

Sample from true conditional

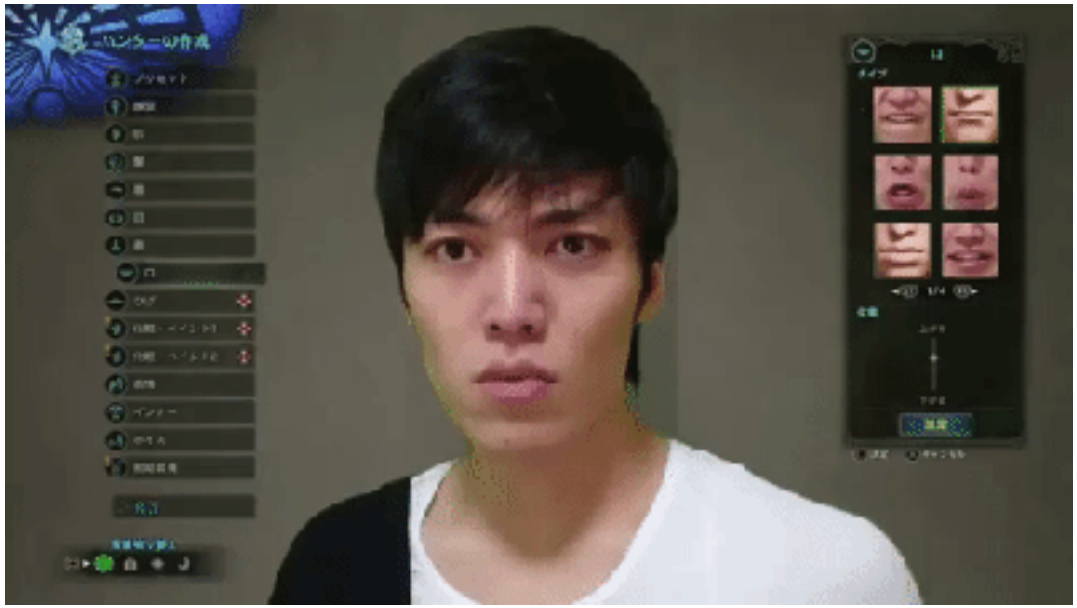
$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$p_{\theta^*}(z)$



- Should be simple, easily to sample from: Normal
- Each component should be independent: Covariance
  - Encourages features that are semantic



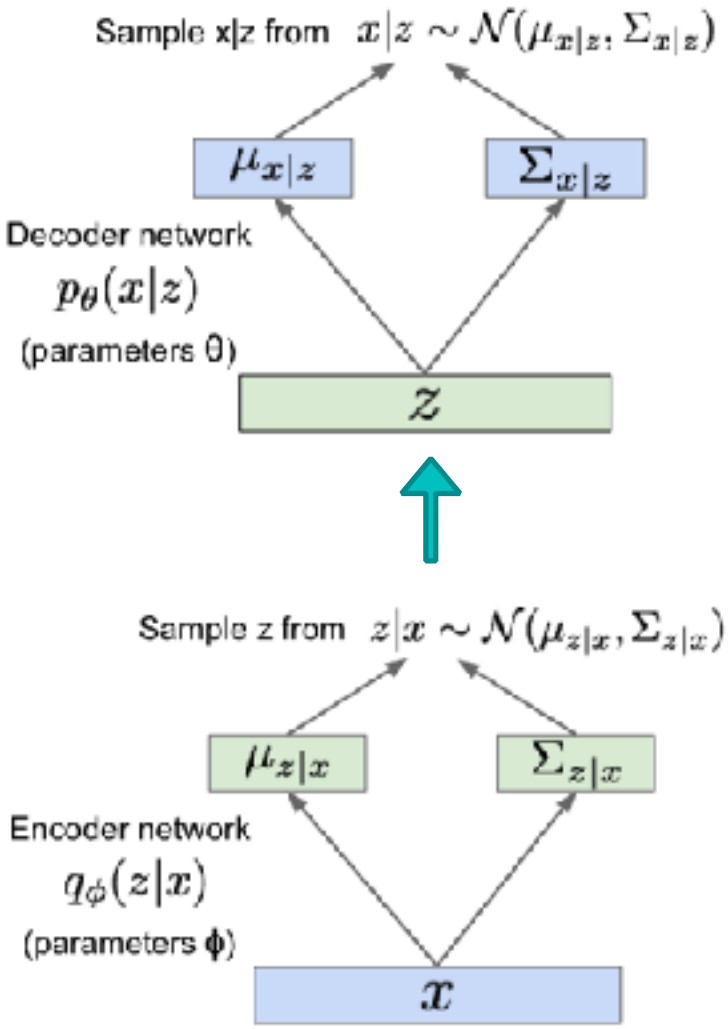
Optimizing

- We want generated samples from latent space to be as close as possible to the true p(x)…
- How can we optimize this?

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

- We can't! Intractable computation for every "z"
- So let's define this with variational inference:
  - Only needs to work for z with observed $x^{(i)}$
  - Encode observed $x^{(i)}$ using network q,
    - so $q = p(z \mid x^{(i)})$
  - Optimize decoder to minimize reconstruction

Need a new formulation



if we optimize **z**, this is MLE

$$\log p(x^{(i)}) = \mathbf{E}_{\mathbf{z} \leftarrow q(z|x)} \left[\log p(x^{(i)}) \right]$$

$$= \sum_{z \in \mathcal{Z}|x^{(i)}} q(z \,|\, x^{(i)}) \cdot \log p(x^{(i)})$$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Need a new formulation

$$\log p(x^{(i)}) = \mathbf{E}_{\mathbf{z} \leftarrow q(z|x)} \left[\log p(x^{(i)}) \right]$$

$$= \mathbf{E}_{\mathbf{z}} \left[\log \frac{p(x^{(i)} | z) p(z)}{p(z | x^{(i)})} \frac{q(z | x^{(i)})}{q(z | x^{(i)})} \right]$$

$$= \mathbf{E}_{\mathbf{z}} \left[\log p(x^{(i)} | z) \right] + \mathbf{E}_{\mathbf{z}} \left[\log \frac{p(z)}{q(z | x^{(i)})} \right] + \mathbf{E}_{\mathbf{z}} \left[\log \frac{q(z | x^{(i)})}{p(z | x^{(i)})} \right]$$

$$= \mathbf{E}_{\mathbf{z}} \left[\log p(x^{(i)} | z) \right] - \mathbf{E}_{\mathbf{z}} \left[\log \frac{q(z | x^{(i)})}{p(z)} \right] + \mathbf{E}_{\mathbf{z}} \left[\log \frac{q(z | x^{(i)})}{p(z | x^{(i)})} \right]$$

$$= \mathbf{E}_{\mathbf{z}} \left[\log p(x^{(i)} | z) \right] - D_{KL} \left[q(z | x^{(i)}) || p(z) \right] + D_{KL} \left[q(z | x^{(i)}) || p(z | x^{(i)}) \right]$$

$$\log p(x^{(i)}) \geq \mathbf{E}_{\mathbf{z}} \left[\log p(x^{(i)} | z) \right] - D_{KL} \left[q(z | x^{(i)}) || p(z) \right]$$ Maximize Lower Bound

What have we really done here? Is this still MLE?

The Loss Function

Maximize through
Error of Reconstruction
This is just negative cross entropy

Here we
assume p(z) is Normal
because we like Normal

$$D_{KL}[N(\mu(X),\Sigma(X))\|N(0,1)] = \frac{1}{2}\left(\mathrm{tr}(\Sigma(X)) + \mu(X)^T\mu(X) - k - \log\det(\Sigma(X))\right)$$

$$D_{KL}[N(\mu(X),\Sigma(X))\|N(0,1)] = \frac{1}{2}\left(\sum_k \Sigma(X) + \sum_k \mu^2(X) - \sum_k 1 - \log\prod_k \Sigma(X)\right)$$

$$= \frac{1}{2}\left(\sum_k \Sigma(X) + \sum_k \mu^2(X) - \sum_k 1 - \sum_k \log\Sigma(X)\right)$$

$$\geq \mathbf{E}_{\mathbf{z}}\left[\log p(x^{(i)}\,|\,z)\right] - D_{KL}\left[q(z\,|\,x^{(i)})\,||\,p(z)\right]$$

$$= \frac{1}{2}\sum_k \left(\Sigma(X) + \mu^2(X) - 1 - \log\Sigma(X)\right)$$

https://wiseodd.github.io/techblog/2016/12/10/variational-autoencoder/

The Loss Function

$$\geq \mathbf{E}_{\mathbf{z}}\left[\log p(x^{(i)} \mid z)\right] - D_{KL}\left[q(z \mid x^{(i)}) \mid\mid p(z)\right]$$

Maximize through
Error of Reconstruction
This is just negative cross entropy

Here we
assume p(z) is Normal
because we like Normal

$$= \frac{1}{2}\sum_{k}\left(\Sigma(X) + \mu^2(X) - 1 - \log\Sigma(X)\right)$$

covariance is not numerically stable because of underflow

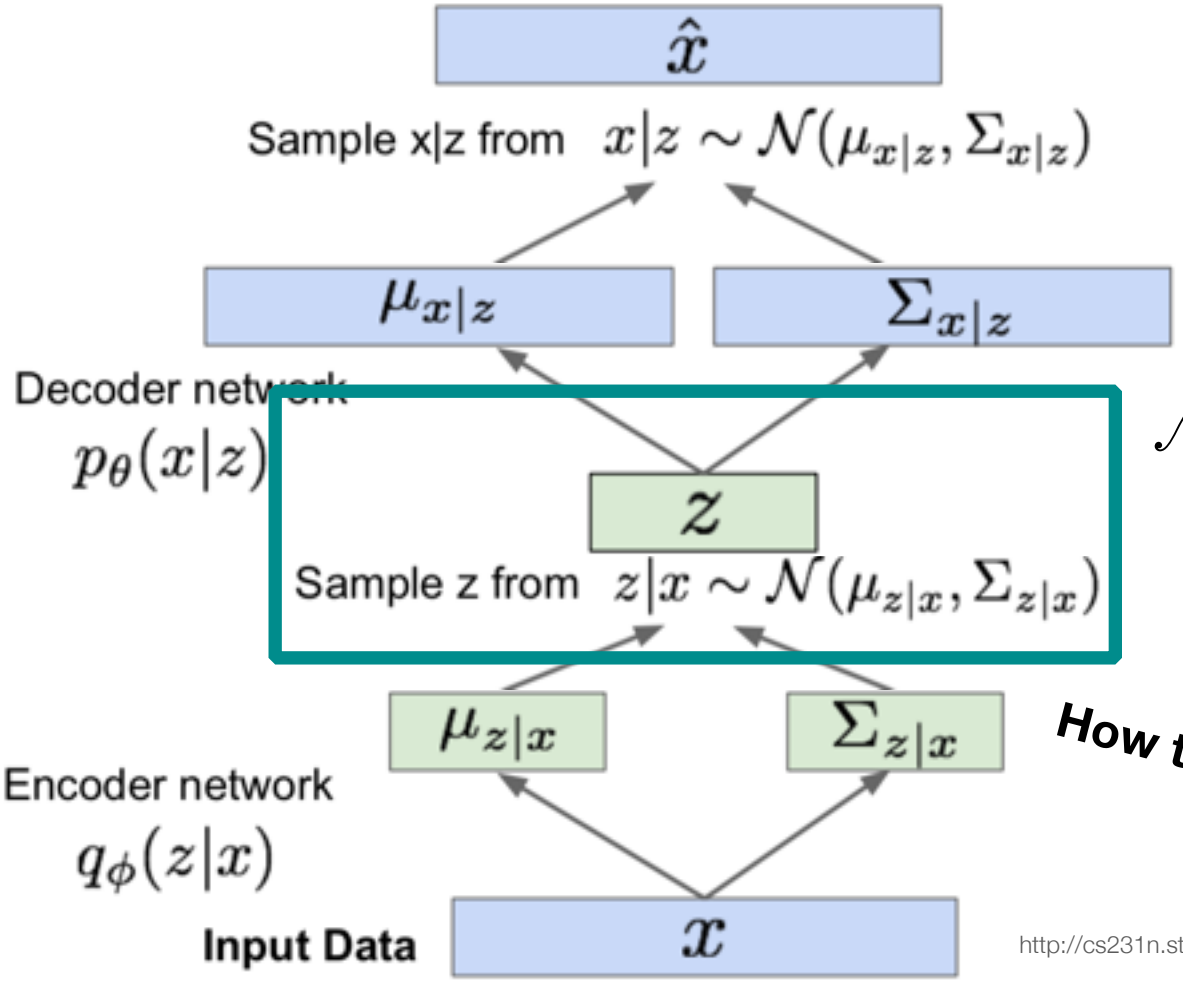
$$D_{KL}[N(\mu(X), \Sigma(X)) \| N(0,1)] = \frac{1}{2}\sum_{k}\left(\exp(\Sigma(X)) + \mu^2(X) - 1 - \Sigma(X)\right)$$

so we actually optimize log variance
(remember we assume diagonal covariance)

https://wiseodd.github.io/techblog/2016/12/10/variational-autoencoder/

Back Propagating

$$\geq \mathbf{E}_{\mathbf{z}}\left[\log p(x^{(i)}\,|\,z)\right] - D_{KL}\left[q(z\,|\,x^{(i)})\,||\,p(z)\right]$$



This is partially differentiable by chain rule…

$$\mathcal{N}(\mu, \Sigma) = z$$
$$= \mu(x^{(i)}) + \Sigma(x^{(i)}) \cdot \mathcal{N}(0,1)$$

How to back propagate?

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

The Loss Function Implementation

```
# Encode the input into a mean and variance parameter
z_mean, z_log_variance = encoder(input_img)

# Draw a latent point using a small random epsilon
z = z_mean + exp(z_log_variance) * epsilon

# Then decode z back to an image
reconstructed_img = decoder(z)

# Instantiate a model
model = Model(input_img, reconstructed_img)
```

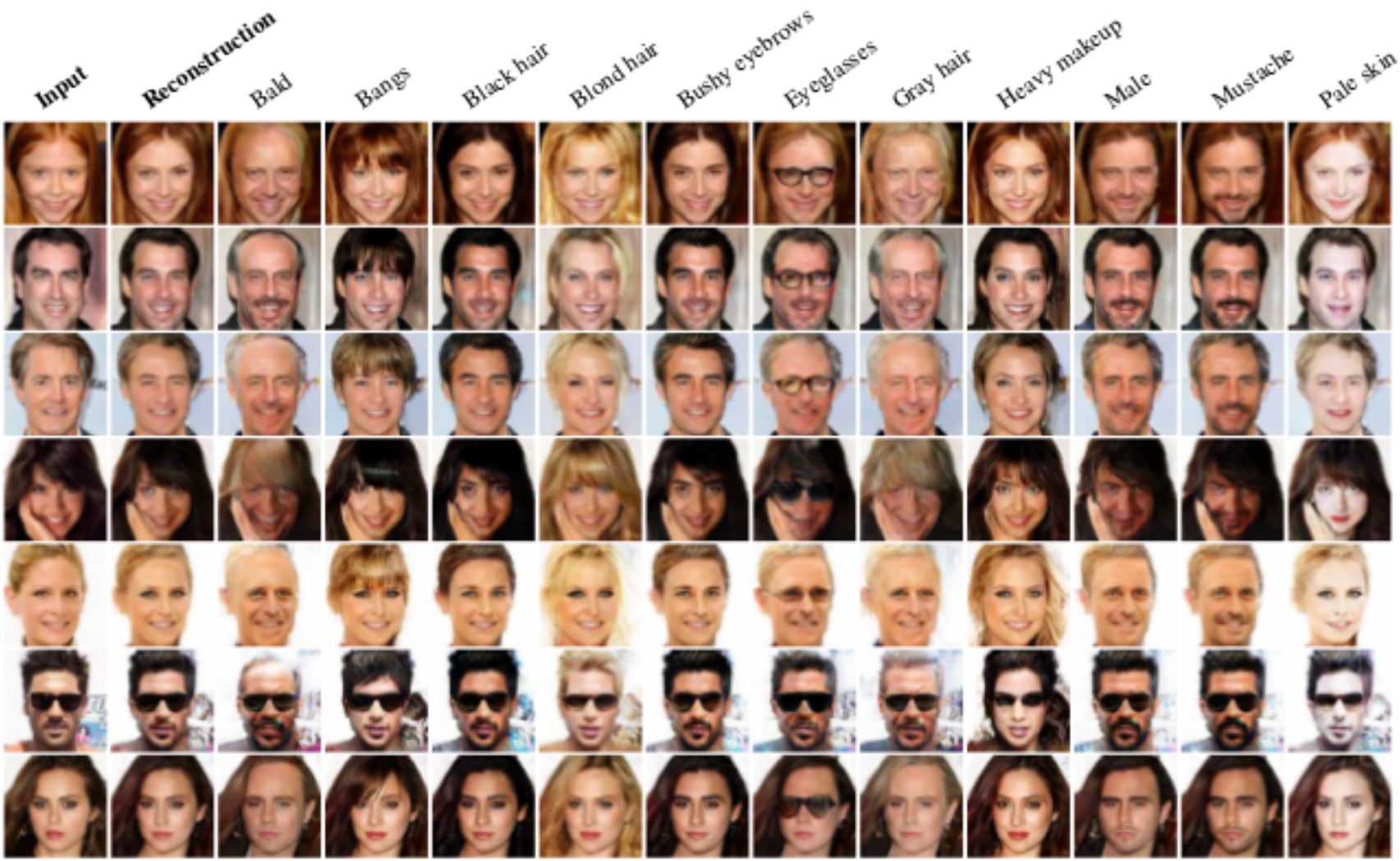
$$\mathcal{N}(\mu, \Sigma) = z$$
$$= \mu(x^{(i)}) + \Sigma(x^{(i)}) \cdot \mathcal{N}(0,1)$$

```
def vae_loss(self, x, z_decoded):
    x = K.flatten(x)
    z_decoded = K.flatten(z_decoded)
    xent_loss = keras.metrics.binary_crossentropy(x, z_decoded)
    kl_loss = -5e-4 * K.mean(
        1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis=-1)
    return K.mean(xent_loss + kl_loss)
```

$$\mathbf{E}_{\mathbf{z}}\left[\log p(x^{(i)} \mid z)\right] - D_{KL}\left[q(z \mid x^{(i)}) \mid\mid p(z)\right]$$
$$= \mathbf{E}_{\mathbf{z}}\left[\log p(x^{(i)} \mid z)\right] - \sum_{k} \exp(\Sigma(x^{(i)}) + \mu(x^{(i)})^2 - 1 - \Sigma(x^{(i)})$$

VAE Examples

VAE Examples



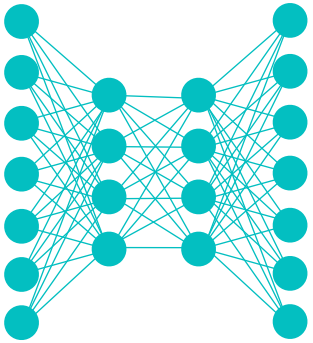
Lecture Notes for

Neural Networks and Machine Learning

Generative Networks

Next Time:
GANs

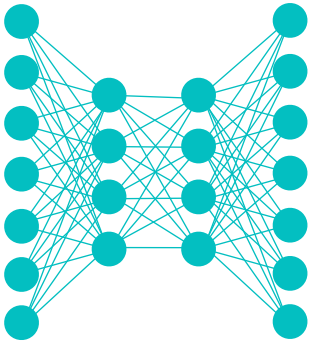
Reading: Chollet CH8



Lecture Notes for

Neural Networks and Machine Learning

Generative Adversarial Networks



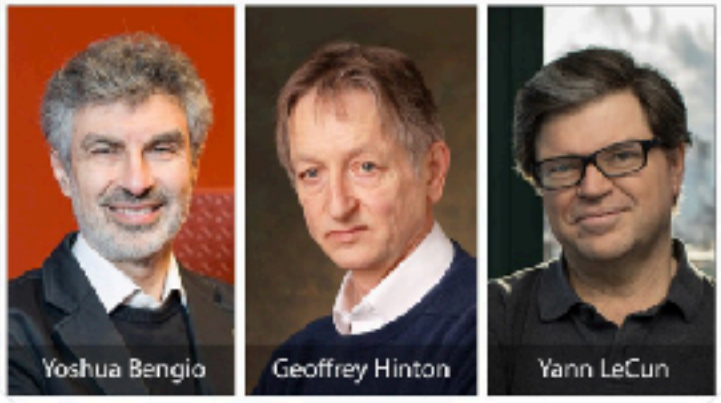
Logistics and Agenda

- Logistics
  - None!
  - How is the ChEMBL filtering going?
- Agenda
  - Working with ChEMBL
  - VAE Demo
  - Simple Generative Adversarial Networks

Working with ChEMBL

Official ACM @TheOfficialACM

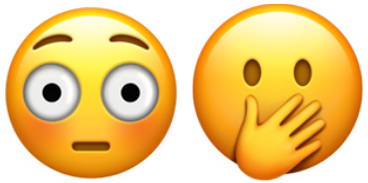
Yoshua Bengio, Geoffrey Hinton and Yann LeCun, the fathers of #DeepLearning, receive the 2018 #ACMTuringAward for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing today. bit.ly/2HVJtdV



Olivier Grisel liked

Volodymyr Kuleshov @volkuleshov · 11h

Congratulations! I hope Jurgen Schmidhuber is not too upset that he wasn't included. Could lead to some awkward questions at the award ceremony ;)



Working with ChEMBL Tables

- **Assays**: need assay type to be binding affinity
- **Activities**:
  - Has **IC50** column, **molregno**, and target (**record_id**)
  - Should calculate “top” records here
  - Do filtering here to find needed compounds
- **Compound_Structures**
  - Has **molregno** and **canonical_smiles** representation
- Building final dataset is simply a join operation on calculated fingerprints and binary IC50 results

Working with ChEMBL

activities

activity_id	assay_id	doc_id	record_id	molregno	star	standard_value	standard_units	stan	standard_type
31863	54505	6424	206172	180094	>	100000	nM	1	IC50
31864	83907	6432	208970	182268	=	2500	nM	1	IC50
31865	88152	6432	208970	182268	>	50000	nM	1	IC50
31866	83907	6432	208987	182855	=	9000	nM	1	IC50

assays

assay_id	doc_id	description	assay_type
1	11087	The compou	B
2	684	Compound w	F
3	15453		B
4	17841	Binding affin	B

compound_structures

molregno	mo	sta	sta	canonical_smiles
1		InC	OV	Cc1cc(ccc1C(=O)c2ccccc2Cl)N3N=CC(=O)NC3=O
2		InC	ZJ	Cc1cc(ccc1C(=O)c2ccc(cc2)C#N)N3N=CC(=O)NC3=O
3		InC	YO	Cc1cc(cc(C)c1C(O)c2ccc(Cl)cc2)N3N=CC(=O)NC3=O
4		InC	PS	Cc1ccc(cc1)C(=O)c2ccc(cc2)N3N=CC(=O)NC3=O
5		InC	KE	Cc1cc(ccc1C(=O)c2ccc(Cl)cc2)N3N=CC(=O)NC3=O

Back to VAEs

Geoffrey Hinton @geoffreyhinton · 23h

Replying to @JeffDean @ylecun and @TheOfficialACM

Thanks to my graduate students and postdocs whose work won a Turing award. Thanks to my visionary mentors Inman Harvey, David Rumelhart and Terry Sejnowski. And thanks to Jeff Dean for creating the brain team that turns basic research in neural nets into game-changing products.

40 313 2,208

Yann LeCun @ylecun · 20h

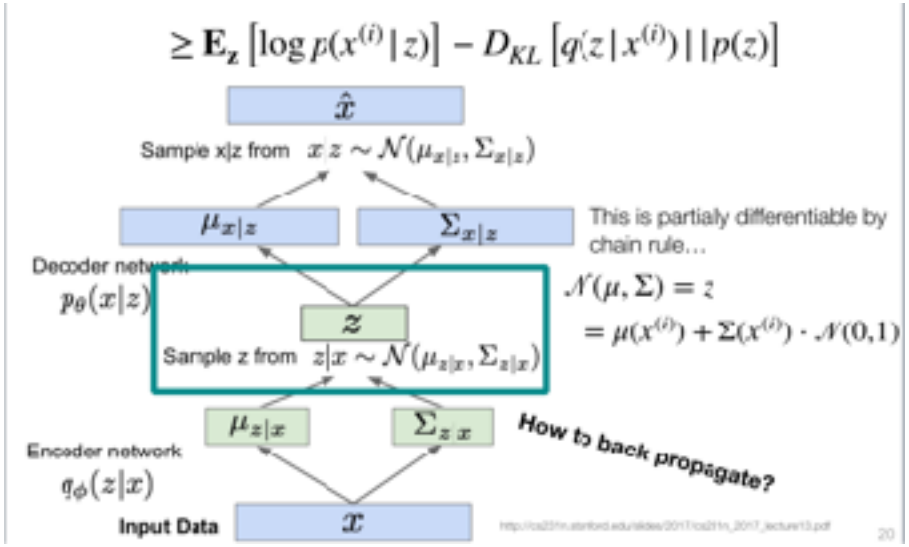
Congratulations Geoff, from one of your former postdocs ;-)

3 11 312

This is why we like Hinton.

Last Time

- Variational auto encoding

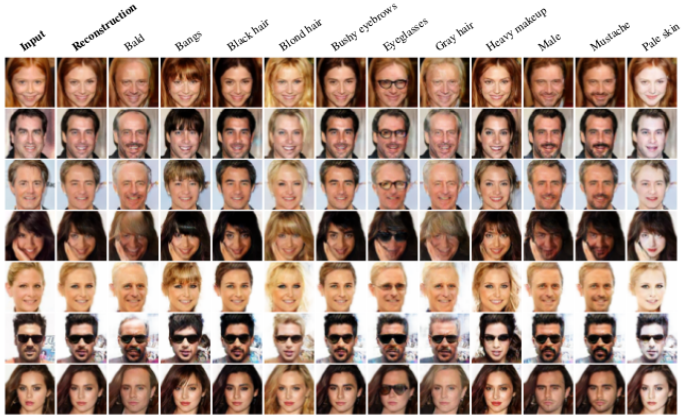



```
# Encode the input into a mean and variance parameter
z_mean, z_log_variance = encoder(input_img)

# Draw a latent point using a small random epsilon
z = z_mean + exp(z_log_variance) * epsilon

# Then decode z back to an image
reconstructed_img = decoder(z)

# Instantiate a model
model = Model(input_img, reconstructed_img)
```

$$\mathcal{N}(\mu, \Sigma) = z$$
$$= \mu(x^{(i)}) + \Sigma(x^{(i)}) \cdot \mathcal{N}(0,1)$$

```
def vae_loss(self, x, z_decoded):
    x = K.flatten(x)
    z_decoded = K.flatten(z_decoded)
    xent_loss = keras.metrics.binary_crossentropy(x, z_decoded)
    kl_loss = -5e-4 * K.mean(
        1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis=-1)
    return K.mean(xent_loss + kl_loss)
```

$$\mathbf{E}_{\mathbf{z}}\left[\log p(x^{(i)}|z)\right] - D_{KL}\left[q(z|x^{(i)})||p(z)\right]$$
$$= \mathbf{E}_{\mathbf{z}}\left[\log p(x^{(i)}|z)\right] - \sum \exp(\Sigma(x^{(i)}) + \mu(x^{(i)})^2 - 1 - \Sigma(x^{(i)}))$$



VAEs in Keras

Implementation from Book on MNIST

Demo by Francois Chollet

Follow Along: https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/8.4-generating-images-with-vaes.ipynb

The Simple GAN

Geoff Hinton after writing the paper on backprop in 1986

Latent Variable Approximation with GANS

- **Idea**: transform random input data into target of interest
  - Like an image!
- Rather than training an encoder with variational inference, we need a transformation protocol
  - Transformer will be a…
  - Neural Network (of course!)
- Transformer is a “complex” sampling method
- How to optimize and provide training examples?
  - Use two Neural Networks!
  - … Deep Learning is like the chiropractory of the machine learning…

Generative Adversarial Network

- Generator: $x = g_w(z)$
- Discriminator: $\{0,1\} = d_w(x)$
- Mini-max, turn-based game:

$$\hat{w} = \arg \min_{g} \max_{d} v(g,d)$$

- Nice differentiable choice for v (zero sum game):

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

only portion dependent on g
generator minimizes

everything dependent on d
discriminator maximizes

Taking turns

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

$$\hat{w} = \arg\min_{g}\max_{d} v(g,d)$$

- If only the problem was convex! This would be easy to solve…
- But v(g,d) is not convex, not even close
  - Saddle points, saddle points everywhere
- Taking turns on the gradient descent will probably never reach equilibrium
  - There is no convergence guarantee
  - But practically we like when discriminator == chance

Practical Objectives

- Discriminator objective, gradient ascent:

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

- Generator objective, gradient descent:

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

- But **practically** generator is really hard to train, amplified by a decreasing gradient
- Goodfellow tried to solve this mathematically through a number of different formulations
  - Nothing seemed to work, and then…

Practical Objectives

- Original Generator objective, gradient descent:

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

- Goodfellow et al. gave up on using rigorous mathematics (intractable) and relied on heuristic:
  - Instead of minimizing when discriminator is right
  - Why not maximize when its wrong?
    - not trying to win, just make the other person lose

$$\max_g \mathbf{E}_{x \leftarrow g(z)} \left[\log(d(x)) \right]$$ **No longer a zero sum game?!**

Final Loss Functions

- Discriminator:

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

Same as minimizing the binary cross entropy of the model with: (1) labeled data and (2) generated data!

- Generator:

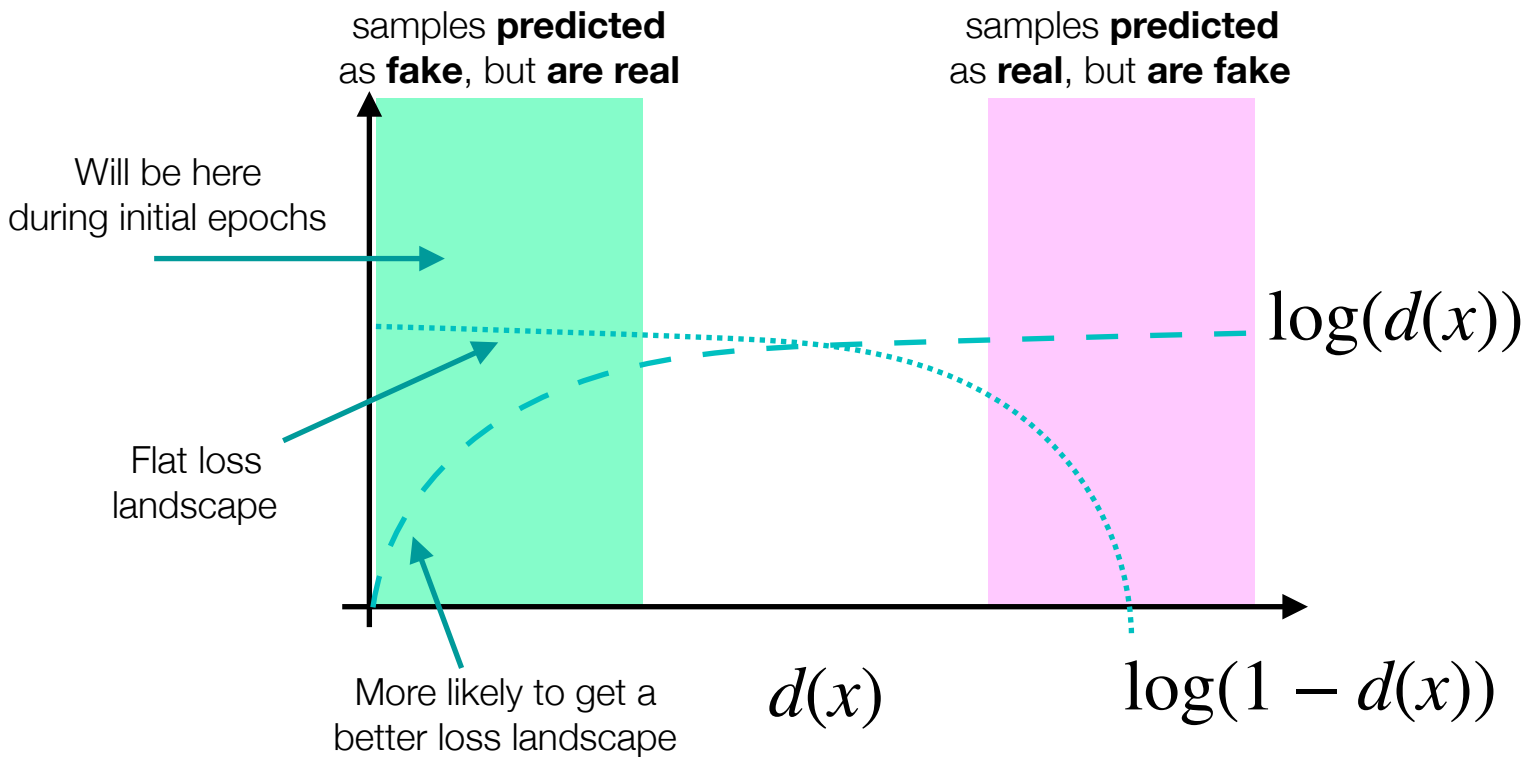
$$\max_{g} \mathbf{E}_{x \leftarrow g(z)} \left[\log(d(x))\right]$$

Freeze Discriminator Weights

Same as minimizing the binary cross entropy of "mislabeled" generated data!

Why maximize incorrect?

- Training two networks is inherently unstable, need heuristic
- Let's assume generator is not any good for initial epochs!



How to Train your (dra) GAN

Every time Ian Goodfellow has a tutorial, It should be called:

"GAN-splaining with Ian"

—Geoffrey Hinton, Probably

Simple Training Approach

- Some caveats on why training will be difficult:
  - The latent space discovered by the Generator has almost no guarantees regarding continuity and structure (different than VAEs)
  - Convergence of GAN is not possible, only equilibrium
    - even saying discriminator is chance is not enough!
  - Each iteration through, the entire loss function changes because generator changes
    - Also we are sampling different points from generator…

Simple Training Approach

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

  end for

  - Sample minibatch of m noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  - Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

Does this work?! **Not really! Why?**

Goodfellow et al. “Generative Adversarial Networks” (NeurIPS 2014)

A bag of tricks from F. Chollet

The process of training GANs and tuning GAN implementations is notoriously difficult. There are a number of known tricks you should keep in mind. Like most things in deep learning, it's **more alchemy than science**: **these tricks are heuristics, not theory-backed guidelines**. They're supported by a level of intuitive understanding of the phenomenon at hand, and they're known to work well empirically, although not necessarily in every context.

—Francois Chollet, DL with Python

- Use tanh for generator output, not a sigmoid
  - We typically normalize inputs to a NN to be from [-1, 1], generator needs to mirror this squashing
- Sample from Normal Distribution
  - Everyone else is doing it (even though no assumption of Gaussian in GAN formulation)
- Random is more robust
  - GANs get stuck a lot
- Sparse gradients are not your friend here
  - No max pooling, no ReLU 😳
- Make encoder and decoder symmetric
  - Unequal pixel coverage (checkerboards)

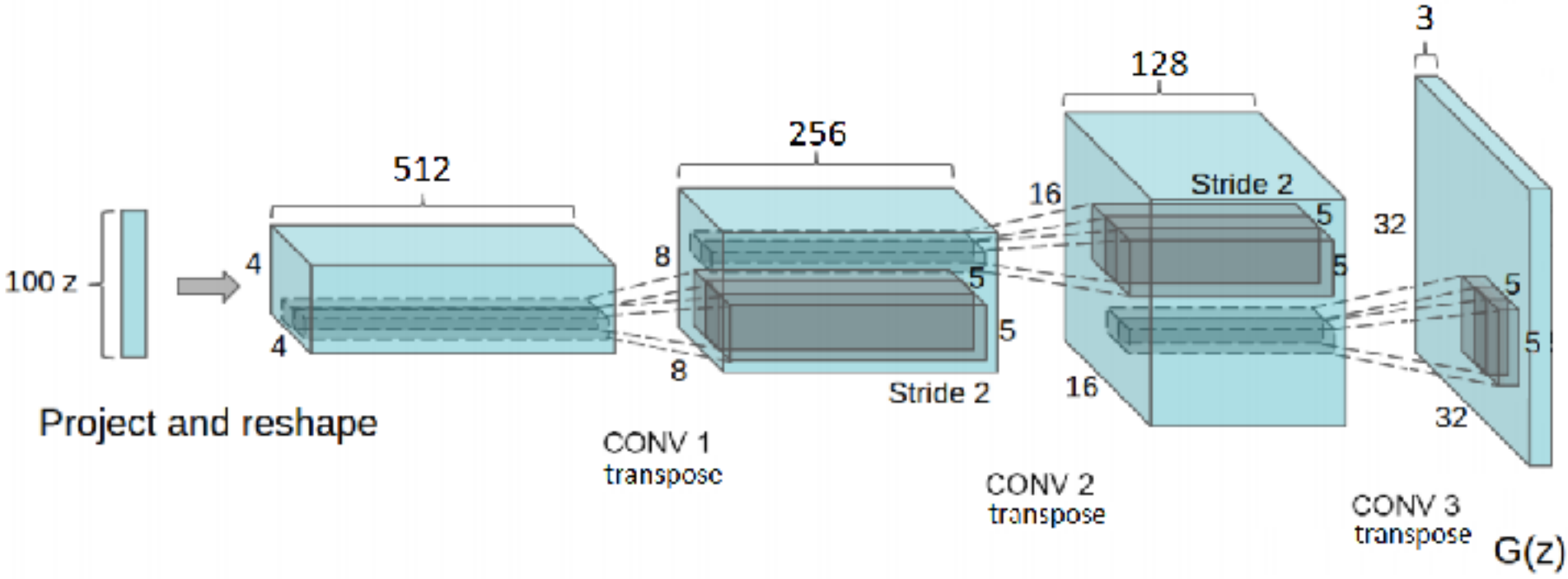
Some Results of Generation

Goodfellow et al. "Generative Adversarial Networks" (NeurIPS 2014)

Highlight: Nearest Training Sample

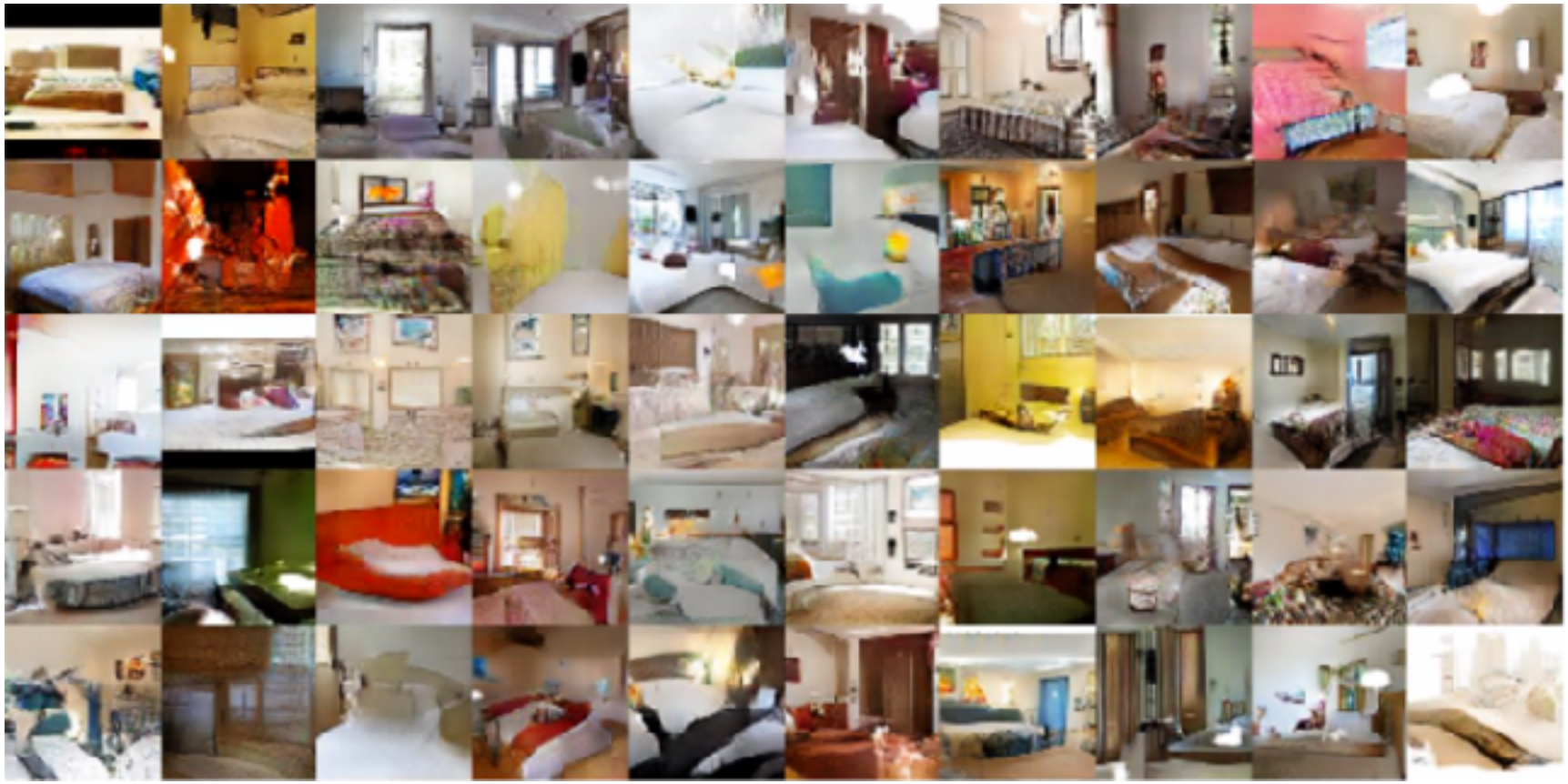
Deep Convolutional GANs

- Just need to reshape and use upsampling



https://sthalles.github.io/intro-to-gans/

Some Results of Generation



Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434* (2015).

GANs in Keras

Implementation from Book on "Frogs from CIFAR"

Demo by Francois Chollet

Follow Along: https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/8.5-introduction-to-gans.ipynb

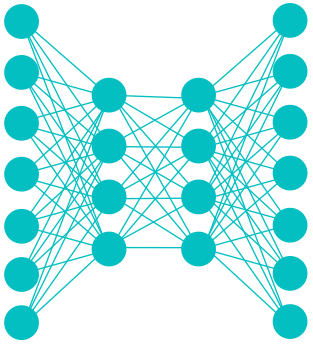
Lecture Notes for

Neural Networks and Machine Learning

GANs

Next Time:
Practical GANs

Reading: Chollet CH8

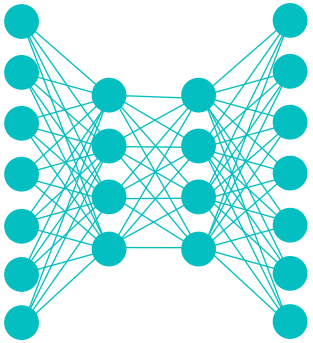


Lecture Notes for

Neural Networks and Machine Learning

Practical

GAN Training



Logistics and Agenda

- Logistics
  - Paper Presentation, Next Lecture: GANs
  - Which paper?
- Agenda
  - Review from Last Time, GAN Demo
  - Training GANs in different loss landscapes
  - Wasserstein GAN

Last Time

- Simple GANS

- Discriminator:

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

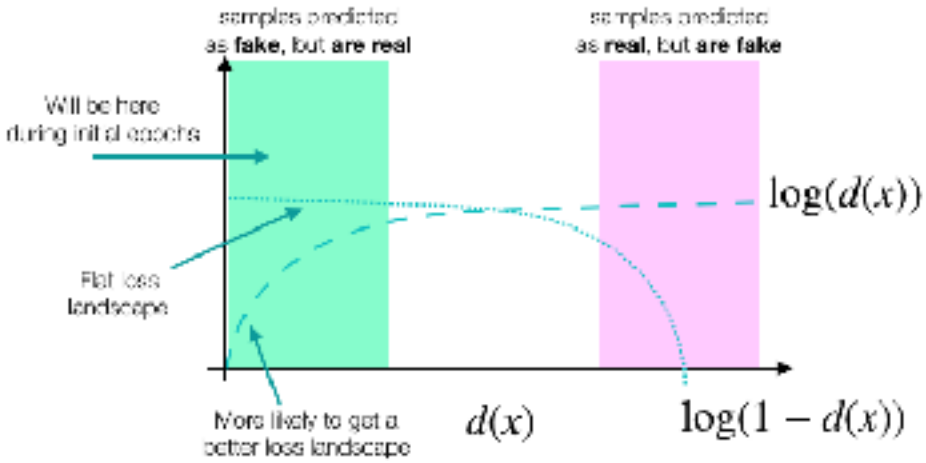
Same as minimizing the binary cross entropy of the model with: (1) labeled data and (2) generated data!

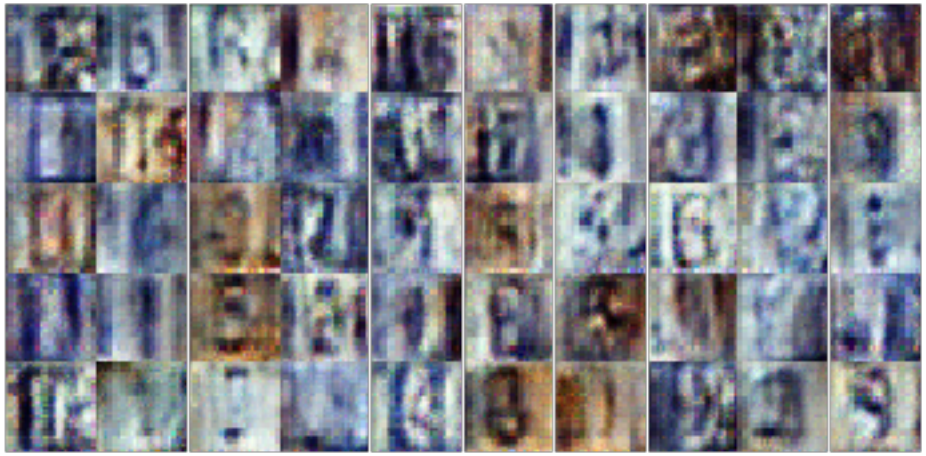
- Generator:

$$\max_g \mathbf{E}_{x \leftarrow g(z)} \left[\log(d(x))\right]$$

Freeze Discriminator Weights

Same as minimizing the binary cross entropy of "mislabeled" generated data!





A bag of tricks from F. Chollet

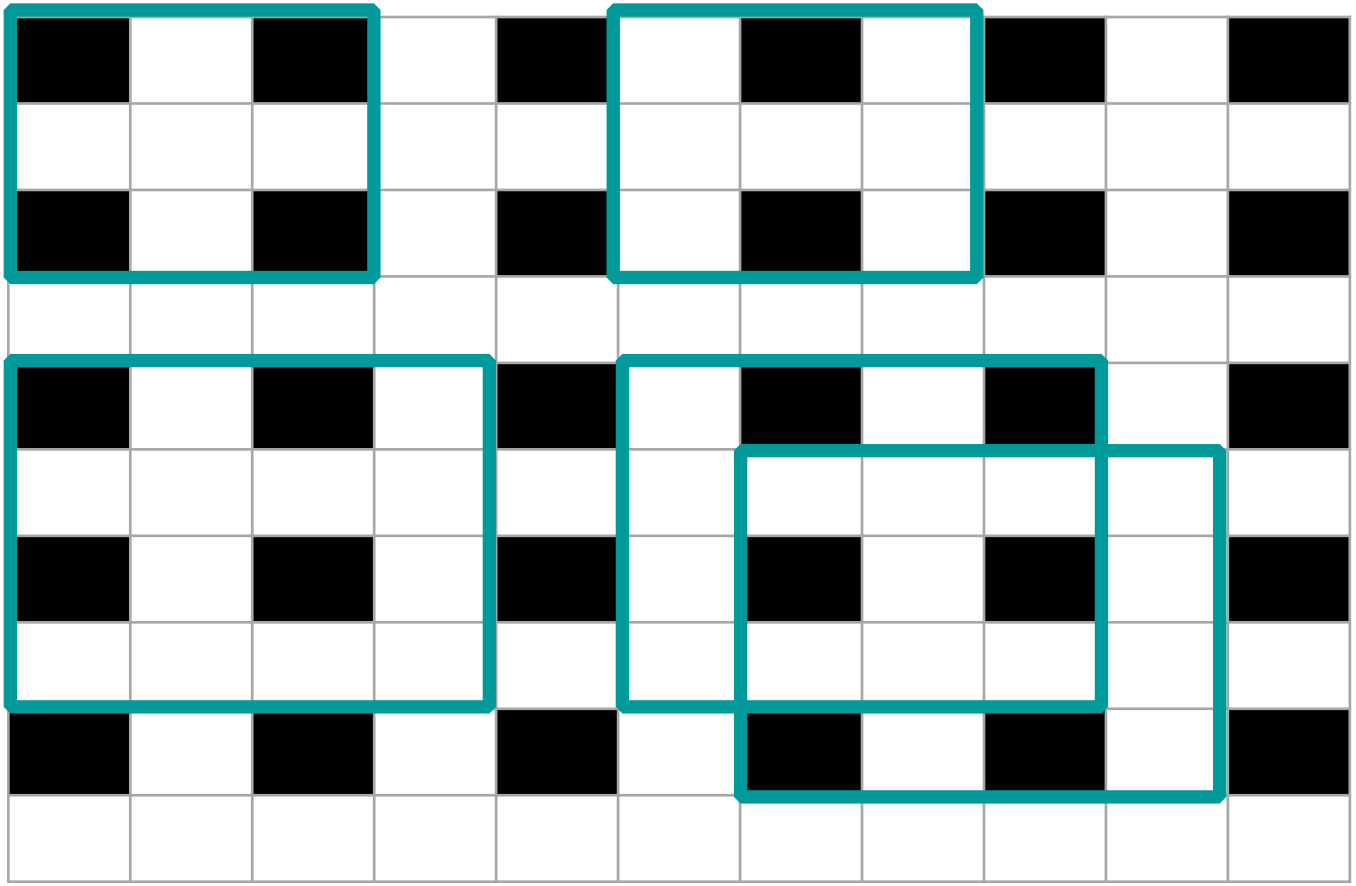
The process of training GANs and tuning GAN implementations is notoriously difficult. There are a number of known tricks you should keep in mind. Like most things in deep learning, it's **more alchemy than science**: **these tricks are heuristics, not theory-backed guidelines**. They're supported by a level of intuitive understanding of the phenomenon at hand, and they're known to work well empirically, although not necessarily in every context.

—Francois Chollet, DL with Python

- Use tanh for generator output, not a sigmoid
  - We typically normalize inputs to a NN to be from [-1, 1], generator needs to mirror this squashing
- Sample from Normal Distribution
  - Everyone else is doing it (even though no assumption of Gaussian in GAN formulation)
- Random is more robust
  - GANs get stuck a lot
- Sparse gradients are not your friend here
  - No max pooling, no ReLU 😳
- Make encoder and decoder symmetric
  - Unequal pixel coverage (checkerboards)

Another note on checkerboard patterns

- Kernel size should be a multiple of the stride



Bias needs to account for both when 2 pixels and four pixels are in the kernel space

Multiple of stiride ensures that same number of active pixels are there.

GANs in Keras

Implementation from Book on
"Frogs from CIFAR"

Game:
Can you catch the errors!?

Demo by Francois Chollet

Follow Along: https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/8.5-introduction-to-gans.ipynb

GANs Loss Landscape

Meta-meta-learning for Neural Architecture Search through arXiv Descent

Antreas Antoniou
MetaMind
aa@mm.ai

Nick Pawlowski
Google x^2
nick@x.x

Jack Turner
slow.ai
jack@slow.ai

James Owers
Facebook AI Research Team
jim@fart.org

Joseph Mellor
Institute of Yellow Jumpers
joe@anditwasall.yellow

Elliot J. Crowley
ClosedAI
elliot@closed.ai

Abstract

Recent work in meta-learning has set the deep learning community alight. From minute gains on few-shot learning tasks, to discovering architectures that are slightly better than chance, to solving intelligence itself[1], meta-learning is proving a popular solution to every conceivable problem ever conceivably conceived ever.

In this paper we venture deeper into the computational insanity that is meta-learning, and potentially risk exiting the simulation of reality itself, by attempting to meta-learn at a third learning level. We showcase the resulting approach—which we call *meta-meta-learning*—for neural architecture search. Crucially, instead of *meta-learning* a neural architecture differentiably as in DARTS (Liu et al., 2018) we *meta-meta-learn* an architecture by searching through arXiv. This *arXiv descent* is GPU-free and only requires a handful of graduate students. Further, we introduce a regulariser, called *college-dropout*, which works by randomly removing a single graduate student from our system. As a consequence, procrastination levels decrease significantly, due to the increased workload and sense of responsibility each student attains.

The code for our experiments is publicly available at [redacted].
Edit: we have decided not to release our code as we are concerned that it may be used for malicious purposes.

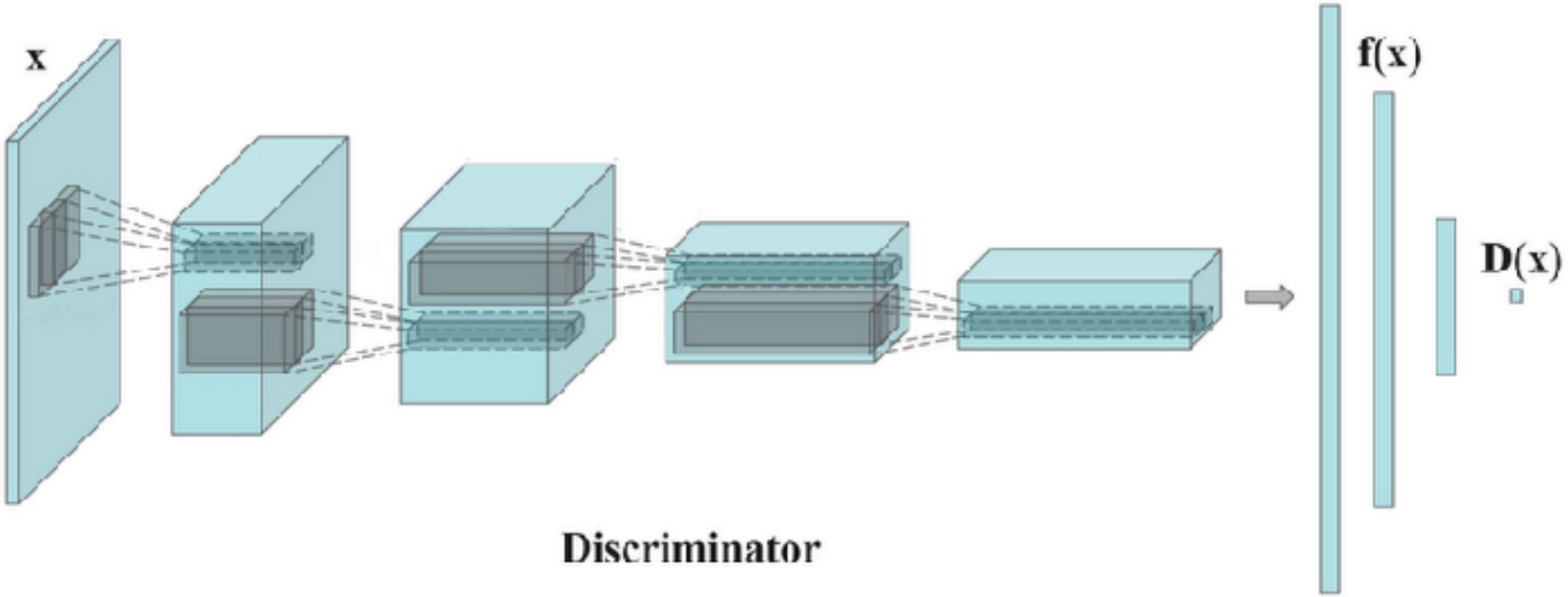
The GAN Loss Landscape

- A collection of Heuristics and Observations that is more Alchemy than Science
  - Feature Matching loss
  - Mini-batch discrimination
  - Historical averaging
  - Virtual Batch normalization
  - Experience Replay
  - Wasserstein Loss (W-GANs…)

There are no GuAraNtees

- From Salimans, Goodfellow, et al., :
  - "*In this work, we introduce several techniques intended to encourage convergence of the GANs game. These techniques are motivated by a heuristic understanding of the non-convergence problem. They lead to improved semi-supervised learning peformance and improved sample generation. We hope that some of them may form the basis for future work, providing formal guarantees of convergence.*"

Before we go too far



Feature Matching

- 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

Salimans, Tim, **Ian Goodfellow**, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved techniques for training gans." In *Advances in neural information processing systems*, pp. 2234-2242. 2016.

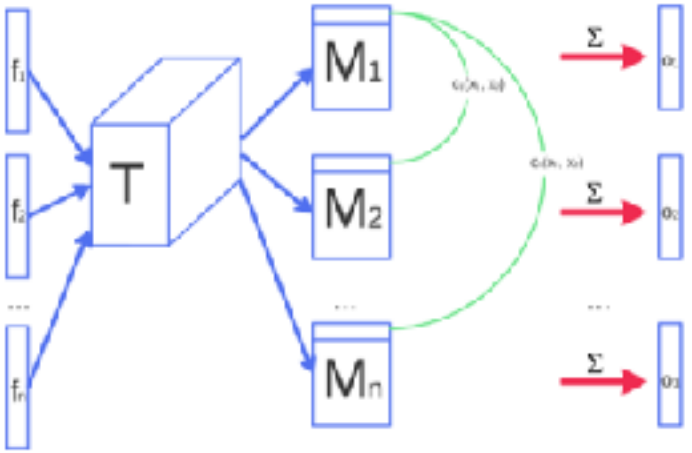
Mini-Batch Discrimination

- Mode collapse of generator happens when parameters emit the same point $G(z)$
- Solution: Incentivize dissimilarity of generated samples by letting discriminator see batches (detect mode collapse early, incentivize generator to prevent it)
- Use discriminator activation $\mathbf{f}(\ G(z)\ )$

$$o(x_i) = \sum_{j \in Batch} \exp\left(-\|\mathbf{f}(x_i) \cdot \mathbf{T} - \mathbf{f}(x_j) \cdot \mathbf{T}\|\right)$$

i^{th} sample in batch

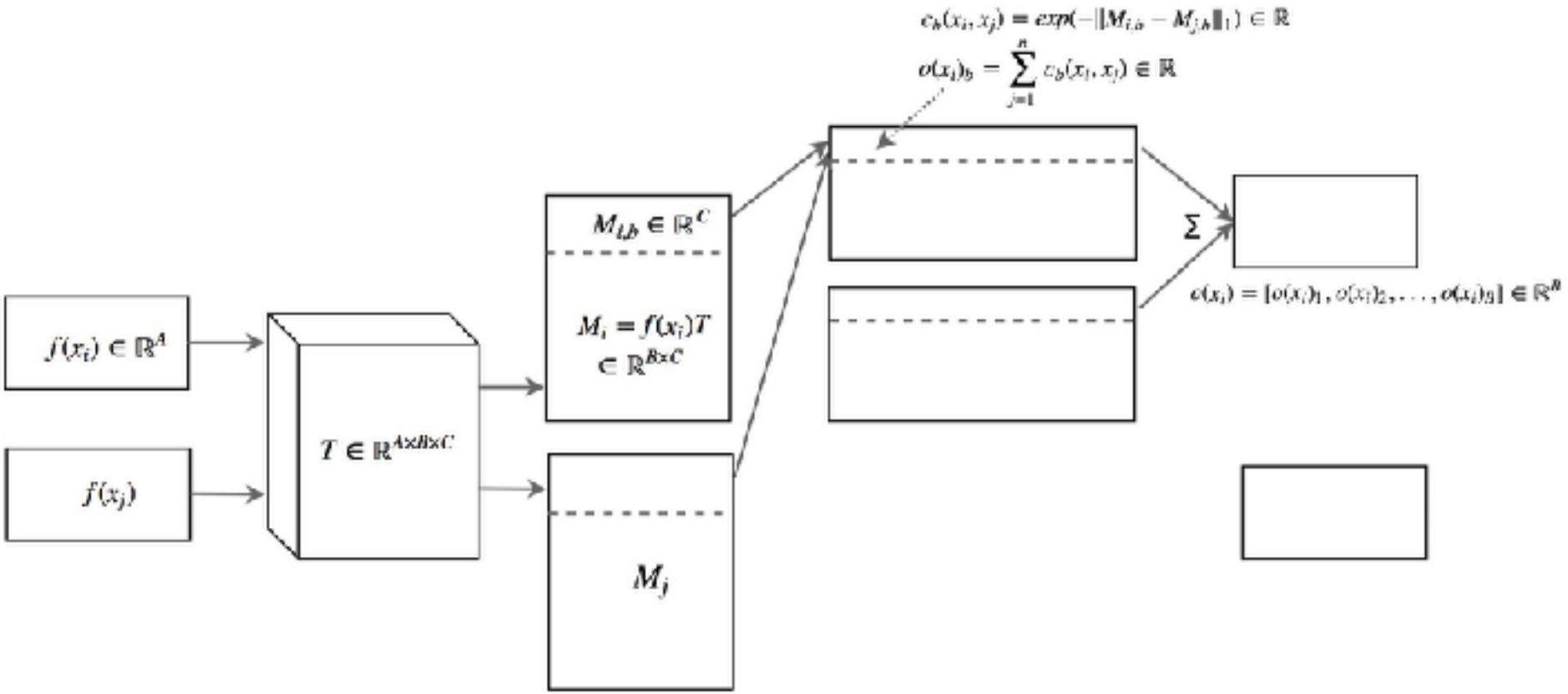
j^{th} sample in batch



- Concatenate and Feed $o(x_{1:n})$ as a feature into next discriminator layer

Salimans, Tim, **Ian Goodfellow**, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved techniques for training gans." In *Advances in neural information processing systems*, pp. 2234-2242. 2016.

Mini-Batch Discrimination



Salimans, Tim, **Ian Goodfellow**, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved techniques for training gans." In *Advances in neural information processing systems*, pp. 2234-2242. 2016.

Historical Averaging

- Because we cannot converge, parameters should constantly be exploring different solutions
- But mode collapse and other issues might cause parameters to converge
- Solution: Incentivize changes directly in loss functions

$$\left\| w[T] - \frac{1}{t} \sum_{i=T-n}^{T-1} w[t] \right\|^2$$

Salimans, Tim, **Ian Goodfellow**, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved techniques for training gans." In *Advances in neural information processing systems*, pp. 2234-2242. 2016.

Virtual Batch Normalization

- Batch normalization is really swell
- But for GANs, it makes current x's dependent on other generated x's (and it was already hard to train…)
- Why not use a reference batch for statistics? And just use statistics of the reference batch activations for normalizing?
  - Every time we update W's in the network, we need to recompute the statistics for the reference batch
  - That's it!

$$\mu^{ref} = \frac{1}{M}\sum_i a_i^{ref} \quad \sigma^{ref\,2} = \frac{1}{M}\sum_i (a_i^{ref} - \mu^{ref})^2 \quad z_i = \gamma \cdot \frac{(a_i - \mu^{ref})}{\sqrt{\sigma^{ref\,2} + \epsilon}} + \beta$$

Salimans, Tim, **Ian Goodfellow**, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved techniques for training gans." In *Advances in neural information processing systems*, pp. 2234-2242. 2016.

Experience Replay

- The discriminator can overpower the generator quickly and it over-learns to a particular epoch
- Solution: feed it more than just the current epoch
- Save previous epochs and randomly grab from them when updating the generator!
- Grab a coffee and chill…

Salimans, Tim, **Ian Goodfellow**, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved techniques for training gans." In *Advances in neural information processing systems*, pp. 2234-2242. 2016.

Figure 4: Samples generated during semi-supervised training on CIFAR-10 with feature matching (Section 3.1, *left*) and minibatch discrimination (Section 3.2, *right*).

Change the loss function entirely

Name	Value Function
GAN	$L_D^{GAN} = E[\log(D(x))] + E[\log(1 - D(G(z)))]$ $L_G^{GAN} = E[\log(D(G(z)))]$
LSGAN	$L_D^{LSGAN} = E[(D(x) - 1)^2] + E[D(G(z))^2]$ $L_G^{LSGAN} = E[(D(G(z)) - 1)^2]$
WGAN	$L_D^{WGAN} = E[D(x)] - E[D(G(z))]$ $L_G^{WGAN} = E[D(G(z))]$ $W_n \leftarrow clip\_by\_value(W_n, -0.01, 0.01)$
WGAN_GP	$L_D^{WGAN\_GP} = L_D^{WGAN} + \lambda E[(\|\nabla D(\alpha x - (1-\alpha G(z)))\| - 1)^2]$ $L_G^{WGAN\_GP} = L_G^{WGAN}$
DRAGAN	$L_D^{DRAGAN} = L_D^{GAN} + \lambda E[(\|\nabla D(\alpha x - (1-\alpha x_p))\| - 1)^2]$ $L_G^{DRAGAN} = L_G^{GAN}$
CGAN	$L_D^{CGAN} = E[\log(D(x,c))] + E[\log(1 - D(G(z),c))]$ $L_G^{CGAN} = E[\log(D(G(z),c))]$
InfoGAN	$L_{D+Q}^{InfoGAN} = L_D^{GAN} - \lambda L_1(c,c')$ $L_G^{InfoGAN} = L_G^{GAN} - \lambda L_1(c,c')$
ACGAN	$L_{D+Q}^{ACGAN} = L_D^{GAN} + E[P(class = c\|x)] + E[P(class = c\|G(z))]$ $L_G^{ACGAN} = L_G^{GAN} + E[P(class = c\|G(z))]$
EBGAN	$L_D^{EBGAN} = D_{AE}(x) + \max(0, m - D_{AE}(G(z)))$ $L_G^{EBGAN} = D_{AE}(G(z)) = \lambda \cdot PT$
BEGAN	$L_D^{BEGAN} = D_{AE}(x) - k_t D_{AE}(G(z))$ $L_G^{BEGAN} = D_{AE}(G(z))$ $k_{t+1} = k_t + \lambda(\gamma D_{AE}(x) - D_{AE}(G(z)))$

The best loss function to use:

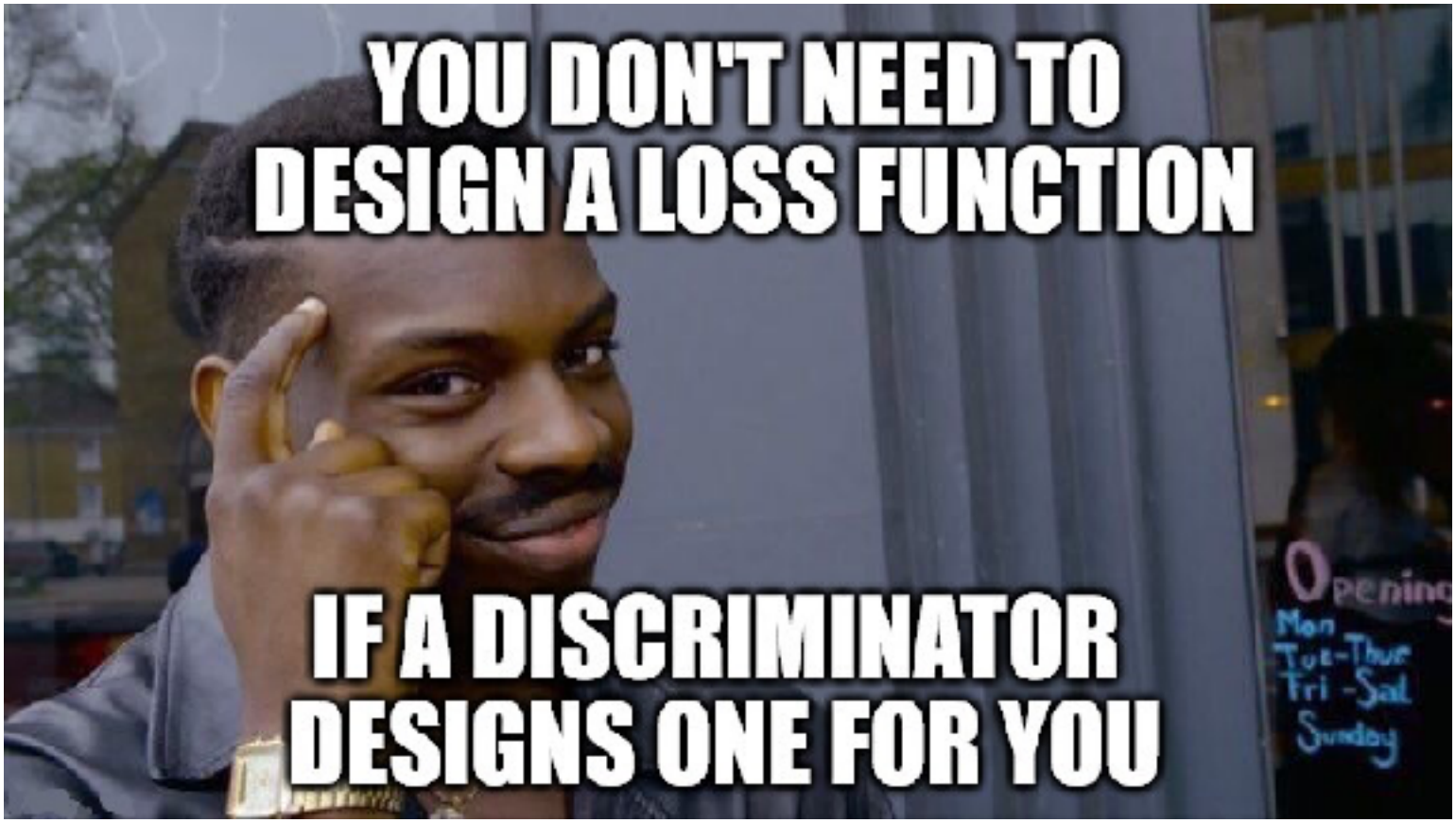
¯_(ツ)_/¯

	MNIST	FASHION	CIFAR	CELEBA
MM GAN	9.8 ± 0.9	29.6 ± 1.6	72.7 ± 3.6	65.6 ± 4.2
NS GAN	6.8 ± 0.5	26.5 ± 1.6	58.5 ± 1.9	55.0 ± 3.3
LSGAN	[illegible]	[illegible]	[illegible]	[illegible]
WGAN	6.7 ± 0.4	21.5 ± 1.6	55.2 ± 2.3	41.3 ± 2.0
WGAN GP	[illegible]	[illegible]	[illegible]	[illegible]
DRAGAN	7.6 ± 0.4	27.7 ± 1.2	69.8 ± 2.0	42.3 ± 3.0
BEGAN	13.1 ± 1.0	22.9 ± 0.9	71.4 ± 1.6	38.9 ± 0.9
VAE	23.8 ± 0.6	58.7 ± 1.2	155.7 ± 11.6	85.7 ± 3.8

Academic blasphemy:
Maybe it doesn't really matter

https://towardsdatascience.com/gan-ways-to-improve-gan-performance-acf37f9f59b

Wasserstein GANs



Caveat

- Wasserstein Loss is actually very useful
- But the mathematics pinning it down are based on approximations of approximations
- So its not really true to say its actually working for the motivated reasons
- But we cannot say that isn't why it working either
- So until we have a better understanding of the mechanisms, we need to go through the math motivated in the paper

Wasserstein

- *Wasserstein GAN adds few tricks to allow the Discriminator to approximate Wasserstein (aka Earth Mover's) distance between real and model distributions. Wasserstein distance roughly tells "how much work is needed to be done for one distribution to be adjusted to match another" and is remarkable in a way that it is defined even for non-overlapping distributions.*
- That should help training even when the generator and discriminator are not matching
- …but Wasserstein Distance is not tractable to calculate so its time to start approximating!

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

The Optimal Discriminator

- Discriminator is maximizing:

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

$$\max_d \int_{x \in [P_{data}, P_g]} \left(p_{data}(x) \cdot \log d(x) + p_g(x) \cdot \log(1 - d(x)) \right) dx$$

$$\nabla_f = 0 = \frac{1}{\partial d(x)} \left(p_{data}(x) \cdot \log d(x) + p_g(x) \cdot \log(1 - d(x)) \right)$$

... math ...

$$d(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

-**Conclusion**: Discriminator is optimal when this condition occurs.

Generator is optimal when

p_g and p_{data} are close together.

So we want to optimize the overlap of these distributions

Wasserstein Distance

- 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:

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

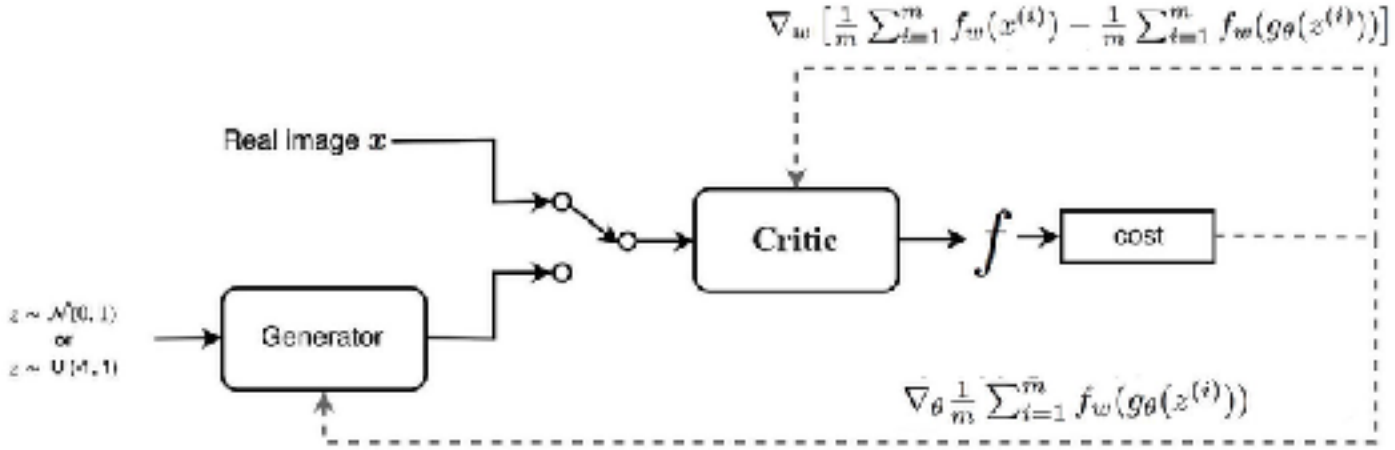
- $\inf$ is greatest lower bound
- gamma is completely and utterly intractable to compute

Wasserstein Duality

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

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

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



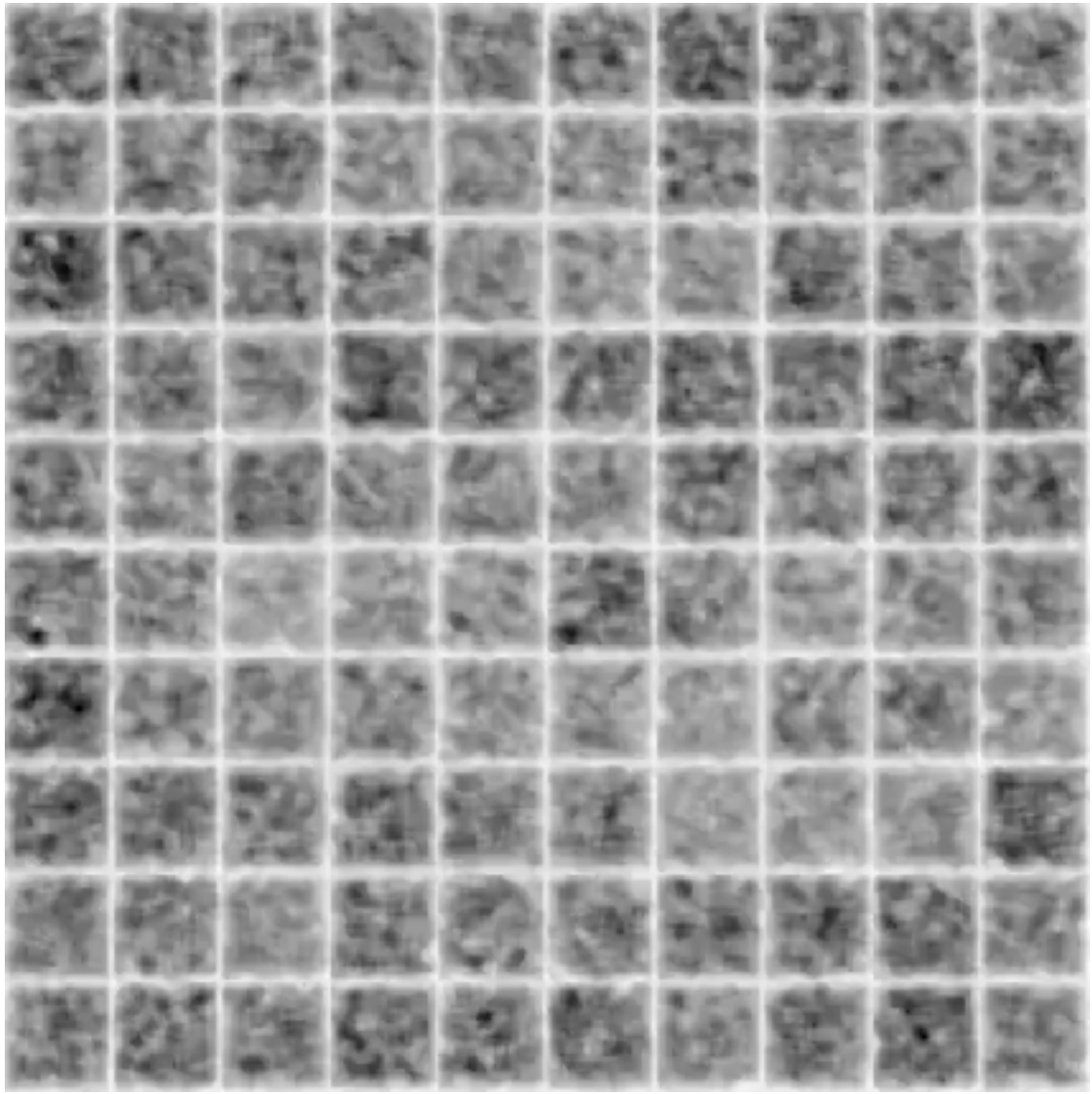
https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

Practical Wasserstein Loss

- Discriminator (critic) becomes **f**, and its output can be Lipschitz if all weights are small (squashes inputs)
  - so we clip them to -c to c (c~0.01)
  - critic output will be linear (and is now a measure of Wasserstein distance from samples input)
  - and maximize for discriminator: $\nabla_w \frac{1}{m}\sum_{i=1}^{m}\left[f(x^{(i)}) - f(G(z^{(i)}))\right]$
  - and maximize for generator: $\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f(G(z^{(i)}))$
- In their empirical findings, no mode collapse problems
- And training longer resulted in good quality images (motivates that distributions overlap)
- Still some some of the most competitive GAN results

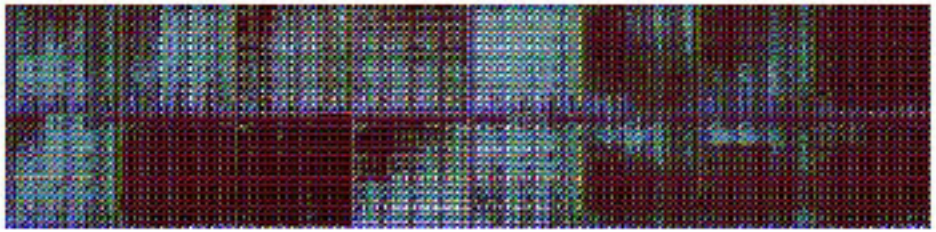
https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

WGAN Example from Keras (MNIST)

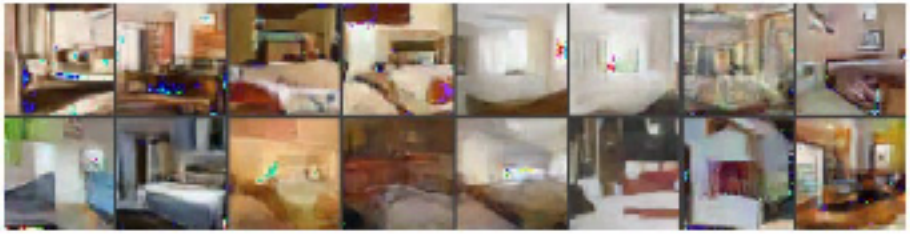


WGAN with DCGAN generator

GAN with DCGAN generator



Without batch normalization &
constant number of filters at each layer



Using a MLP as the generator

All critics and discriminators follow the same discriminator design in DCGAN

So we should always use Wasserstein!

- **Actually not really**
- Its really, really slow
- Clipping:

Weight clipping is a clearly terrible way to enforce a Lipschitz constraint. If the clipping parameter is large, then it can take a long time for any weights to reach their limit, thereby making it harder to train the critic till optimality. If the clipping is small, this can easily lead to vanishing gradients when the number of layers is big, or batch normalization is not used (such as in RNNs).

- Others have made improvements by incorporating gradient constraints, which also adds training time (lots of time)

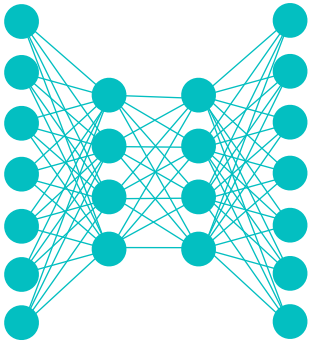
Lecture Notes for

Neural Networks and Machine Learning

Practical GANs

Next Time:
Beyond Generation

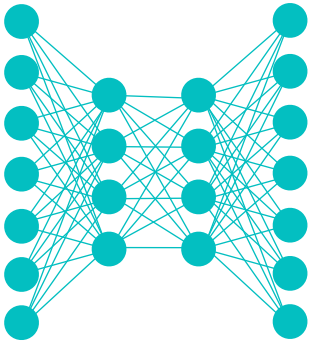
Reading: Chollet CH8



Lecture Notes for

Neural Networks and Machine Learning

Beyond Generation with GANs



Logistics and Agenda

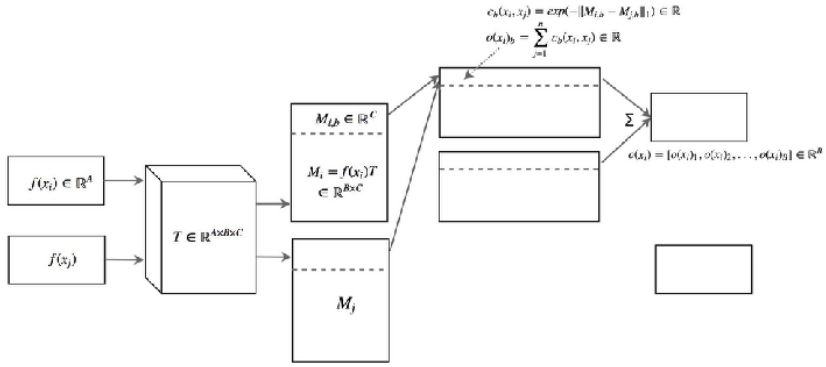
- Logistics
  - ChEMBL ok?
- Agenda
  - GANs for increased classification
  - Adversarial Auto Encoding
  - Paper Presentation: GANs

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:

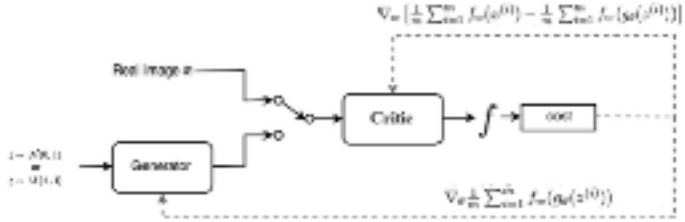
$$W(p_{data}, p_g) = \inf_{\gamma \in \prod(p_{data}, p_g)} \mathbf{E}_{x,y \sim \gamma}\left[\|x - y\|\right]$$

  - 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)



Date	Paper Name	Problems Raised	Novel approach
May 2015	On Distinguishability Criteria For Estimating Generative Models [6]	The expected gradient of generator does not match that of MLE. Non convergence of independent SGD results in under-fitting in gans	In gans G being the primary model causes it to differ from MLE. No close relation between NCE and GAN.
June 2016	Improved Techniques for GANs [24]	Overtraining of the discriminator Mode collapse of Generator Gradient descent may not converge Vulnerable to adversarial examples. GAN outputs depend on the inputs within a same batch due to BN	Feature Matching Mini-batch Discrimination Historical Averaging (Fictitious play) Label-smoothing Virtual Batch normalization
Jan 2017	Towards Principled Methods for Training GANs [4]	Perfect Discriminator resulting in zero grads when distributions are in low dimensional manifolds The $-\log D$ alternative causes unstable updates	Softer Metrics- Adding Gaussian Noise (Instantaneous Divergence) Earth-Mover Distance
Mar 2017	WGAN [3]	Learning distributions supported in lower dimension manifolds. JSD maxes and $KL \to \infty$ Mode Collapse	Define EM Distance (KR Duality) with well defined gradients and avoids balancing the critic and generator. Consistent decrease in loss with training. Weakest metric
May 2017	Improved Techniques on WGAN [8]	Lipschitz enforced by weight clipping. Capacity underuse (critic has high bias). Exploding and vanishing gradients.	Introduced a Gradient Penalty term. unit samples on following was observed. Optimal critic has unit-grad norm between P_r and P_g
Mar 2017	Loss Sensitive GAN on Lipschitz Densities[21]	Over pessimistic D as fake samples closer to data are still negative Assuming infinite capacity which leads to mode collapse and vanishing grads WGAN is unbounded from above as critic is minimized over gen samples	Loss having a data dependent margin Limited to the space of Lipschitz-continuous functions Pairwise comparisons as against WGAN which decomposes the loss into two first order moments
ICLR 2017	EBGAN [25]	With binary logistic loss only two targets are possible, so gradients in a mini batch are far from optimal. Mode Collapse.	Auto encoder Discriminator to evade the need of negative samples Repelling regularizer to prevent mode collapse
Feb 2017	Least Squares GAN [13]	Vanishing gradients with sigmoid cross entropy loss when updating the gen using fake samples that are on the correct side of decision boundary	Pearson χ^2 Divergence. The following constants , a - encoding for real data b - encoding for fakes data c- encoding for value that G wants D to believe is false
June 2016	f-GAN[20]	No convergence to saddle point when using single steps GD	General f divergence variational estimation of $f-$ divergence Variational Divergence Minimization
Nov 2017	Stabilizing Training of GANs through Regularization [23]	Empirical estimation Density misspecification Dimensional misspecification Addition of high dim noise introduces large variance in parameter estimation	Convolving with noise Noise induced regularization
NIPS 2017	The Numerics of GANs [16]	Failure of simult mirror gradient descent as the eigenvalues gradients have zero real part and large imaginary part	Convenient Optimization: Alternative method for finding the Nash Equilibrium Introduced norm of the gradients w.r.t. parameters in the loss

Numerous Recent Works that Address Loss Functions!

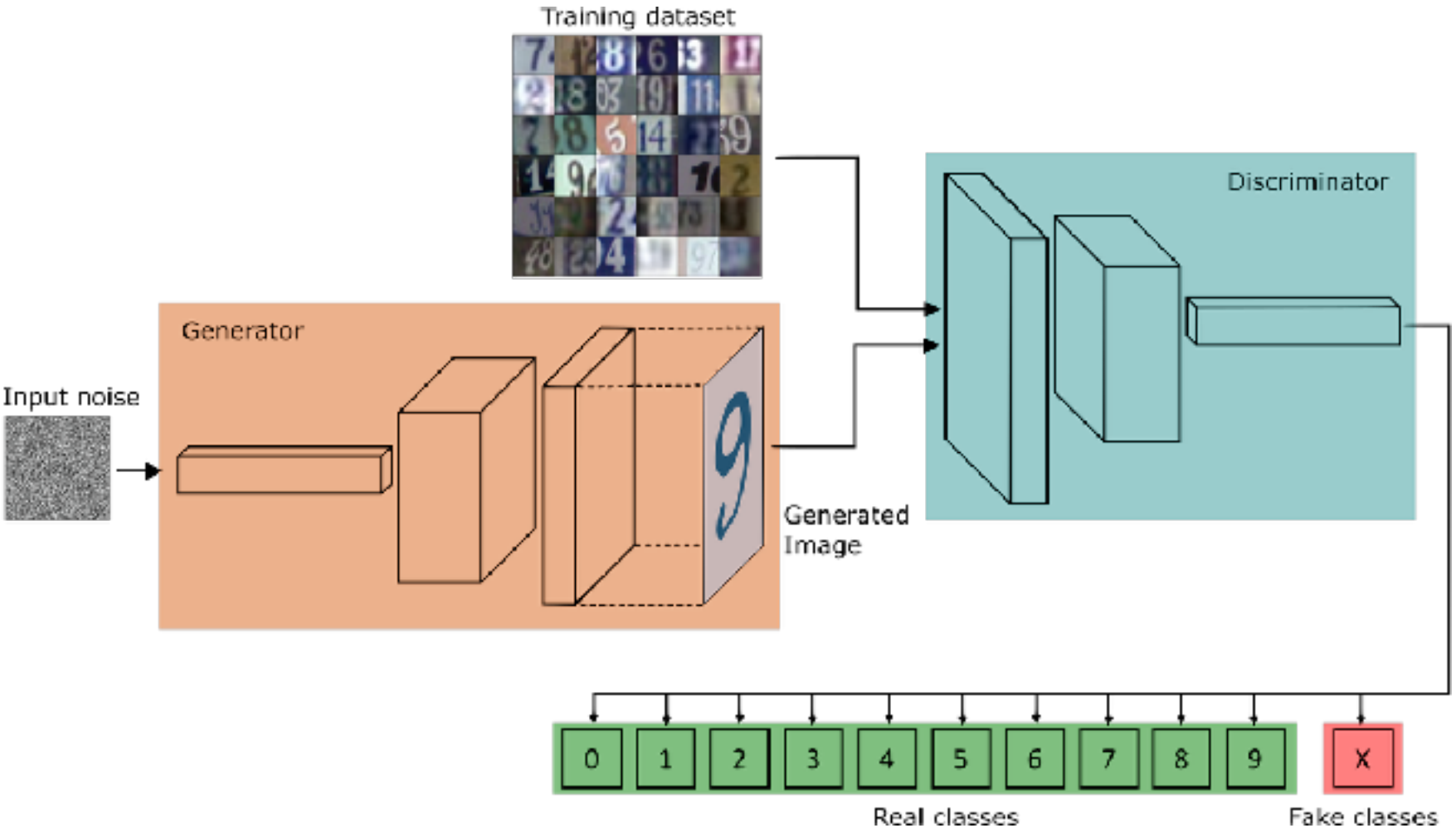
Nov 2016	Unrolled GANs [15]	Mode collapse. Oscillations of G and D. G tries to move much mass to single point, D tracks it and assigns lower probability. The cycle continues forever.	Training the D to optimality is expensive.A surrogate loss which in limit equals optimal D. The G's updates consider future steps of the discriminator.
Aug 2017	Generalization and Equilibrium in GANs [3]	Generalization is not guaranteed i.e $d(\hat{P}_{real}, P) = 0$ but $d(\hat{P}_{real}, \hat{P}) \neq 0$ where $\hat{P}$ is the empirical distribution. Non Existence of equilibrium in GAN	NN Distance which generalizes as it assumes finite capacity D. At the cost of diversity of samples. Mixed Strategy Nash using mixture of Generators which on folding results in pure NE.
May 2017	On Convergence and Stability in GANs [12]	Gradient descent is unstable and results in mode collapse. Divergence minimization hypothesis doesn't explain gan learning entire real distribution. WGAN and LS-GAN regularize the discriminator's gradients in domain space.	Regret Minimization converges to ϵ-equilibrium in non-convex case. Sharps gradients results in mode collapse hence a penalty term is introduced.
Nov 2017	Gradient Descent GAN Optimization is locally stable	It is assumed that updates occur in functional space, with models having infinite capacity. Discriminator is assumed to be fully optimized between gen updates.	local asymptotic stability of GAN (not WGAN) optimization under proper conditions. gradient-based regularizer (GAN + WGAN)

Mnisha and Gujar (2018). Generative Adversarial Networks (GANs): What it can generate and What it cannot? https://arxiv.org/pdf/1804.00140.pdf

GANs for more than Generating Images

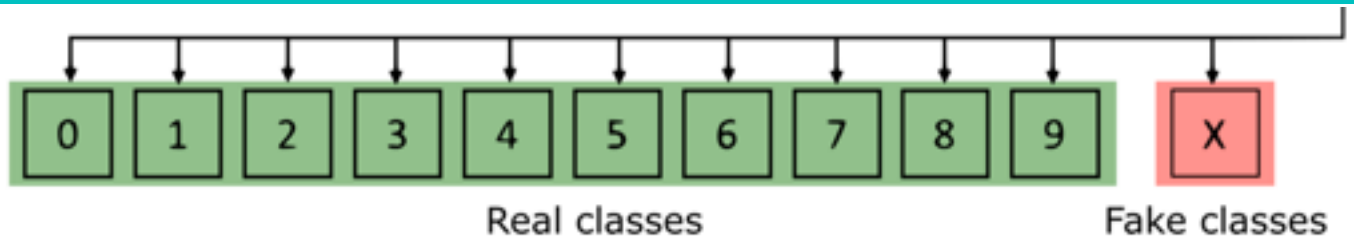
hadou-GAN

Fewer Labels, Still Lots of Data



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

Minimizing Labels



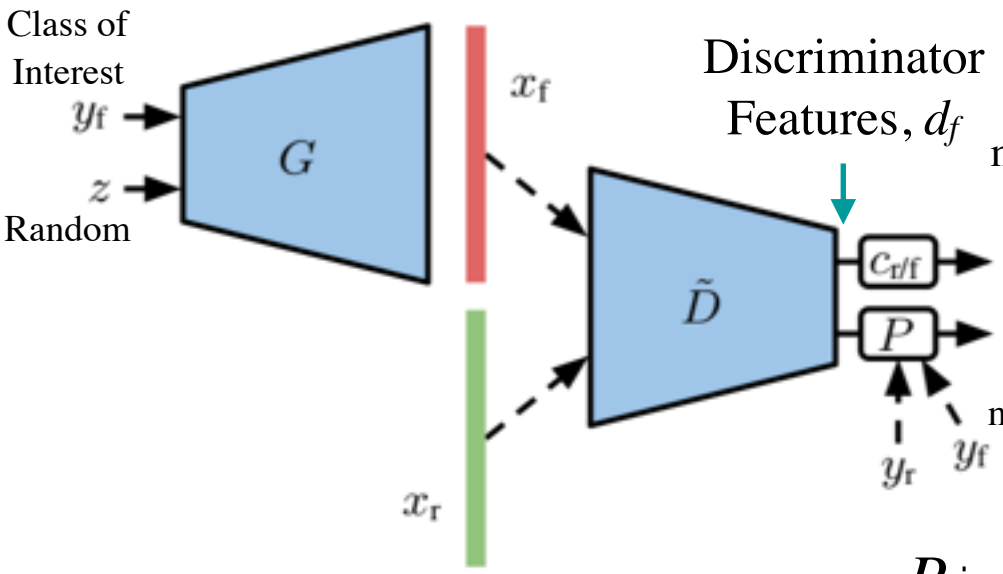
- MNIST Performance
  - 60,000 labels → 1,000 labels
  - 0.6% Error (~SOA)
- SVHN Performance
  - 73,000 labels → 1,000 labels
  - 1.3% Error (~SOA)
- ImageNet
  - 20% Labels (of 1.3M)
  - 10.3% Top-5 Error (slightly worse than SOA)

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

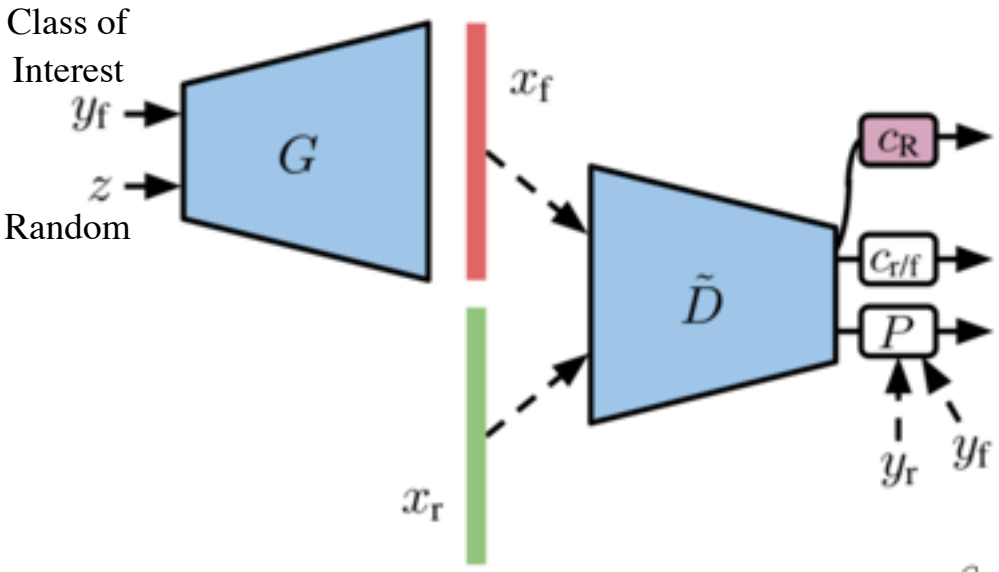
$$\max_d \mathbf{E}_{x \leftarrow p_{data}} \left[\log \tilde{d}(x_r, y_r)\right] + \mathbf{E}_{x \leftarrow g(z, y_f)} \left[\log(1 - \tilde{d}(x_f, y_f))\right]$$

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!

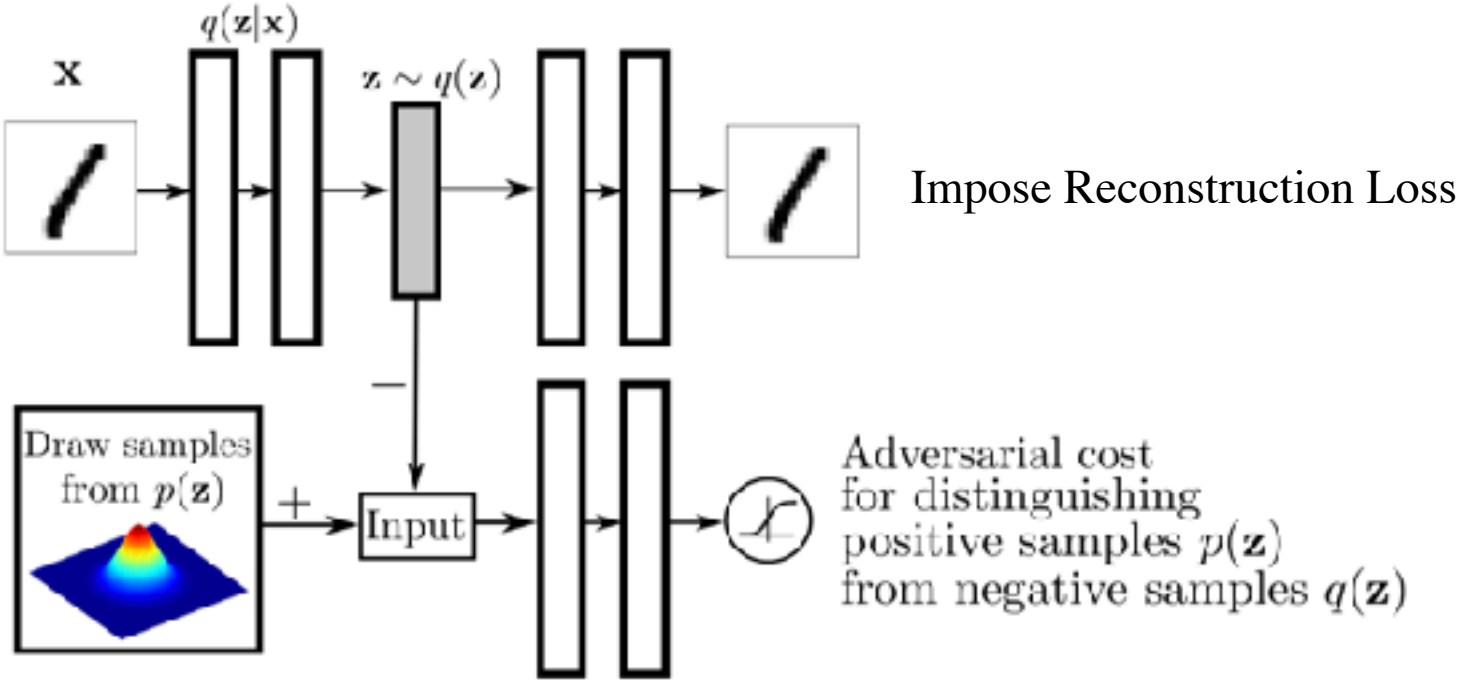
Adversarial Auto Encoding

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.

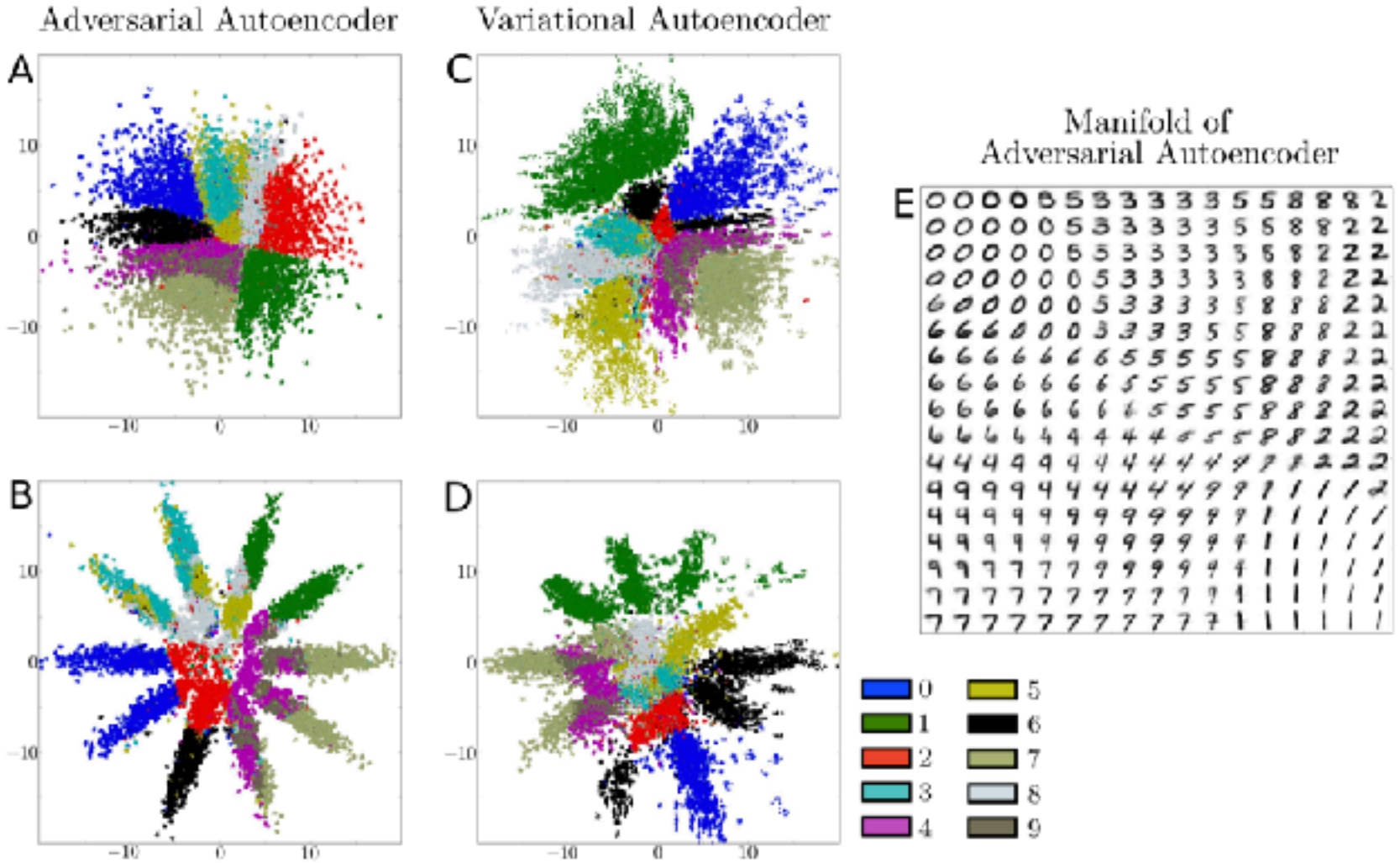
The Main Idea

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



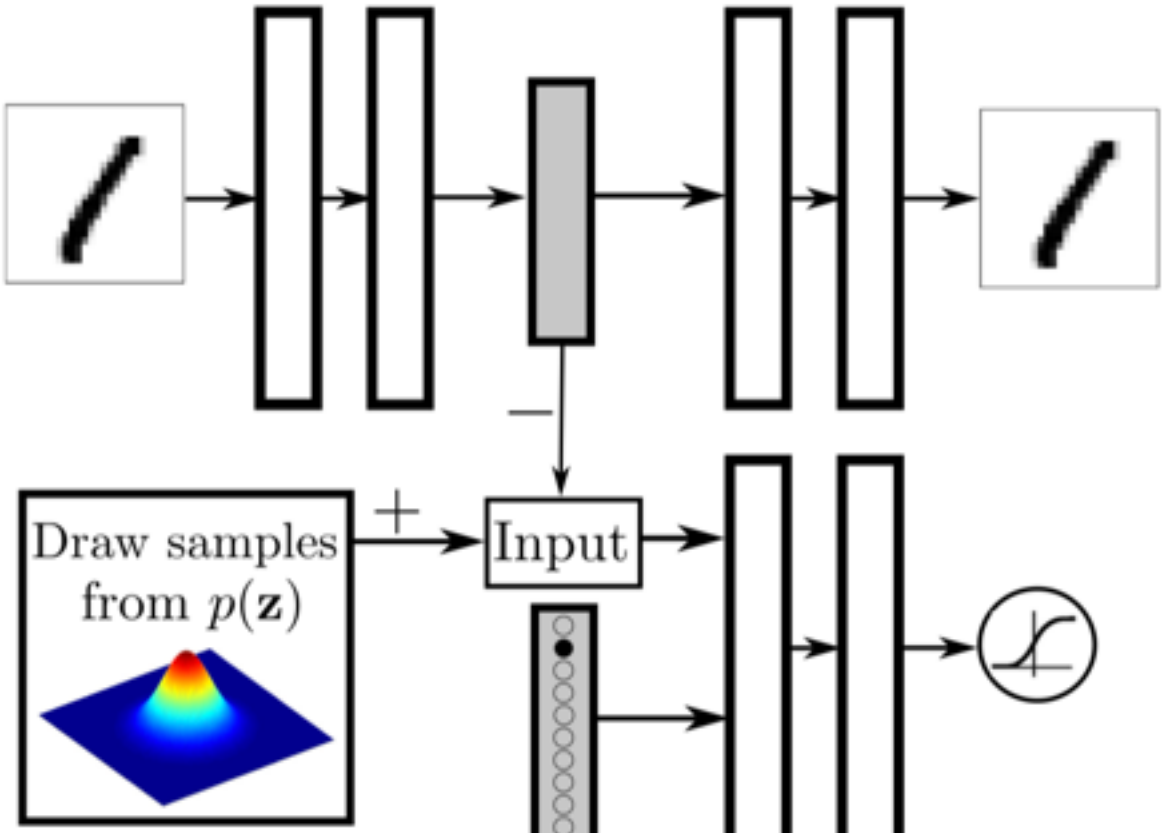
Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, **Ian Goodfellow**, and Brendan Frey. "Adversarial autoencoders." *arXiv preprint arXiv: 1511.05644* (2015).

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

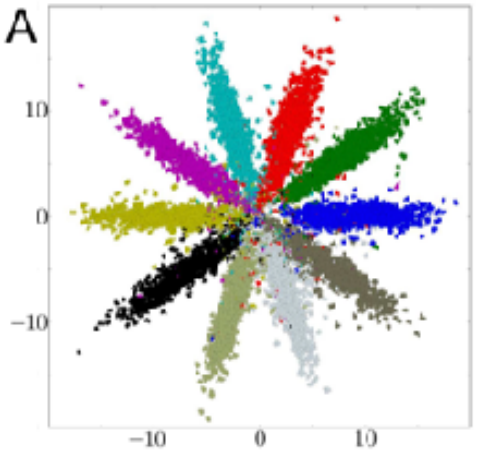


Draw Samples According to Specific Class Prior Distributions

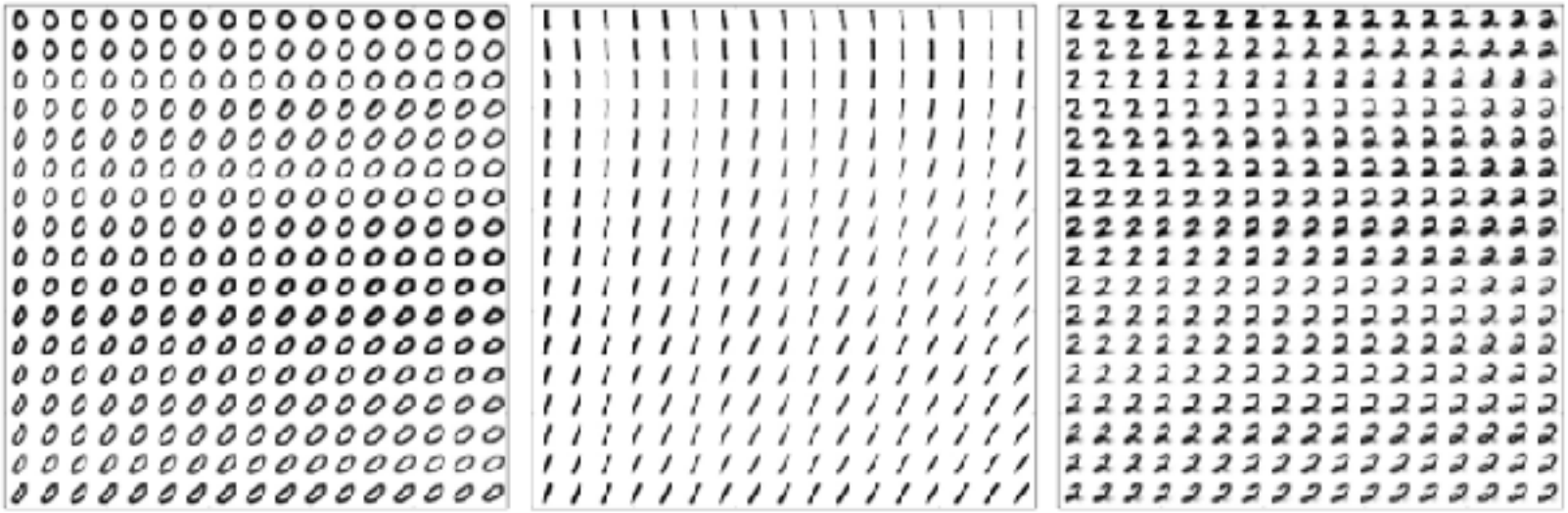
Provide Class Chosen as One Hot Vector in Discriminator

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

Conditional Class Latent Spaces

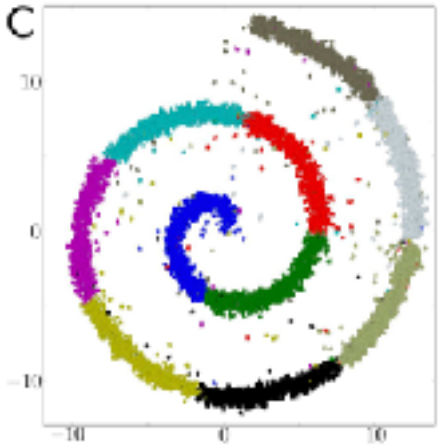


Sample Along Main Axis of the Gaussian Component for Each Digit

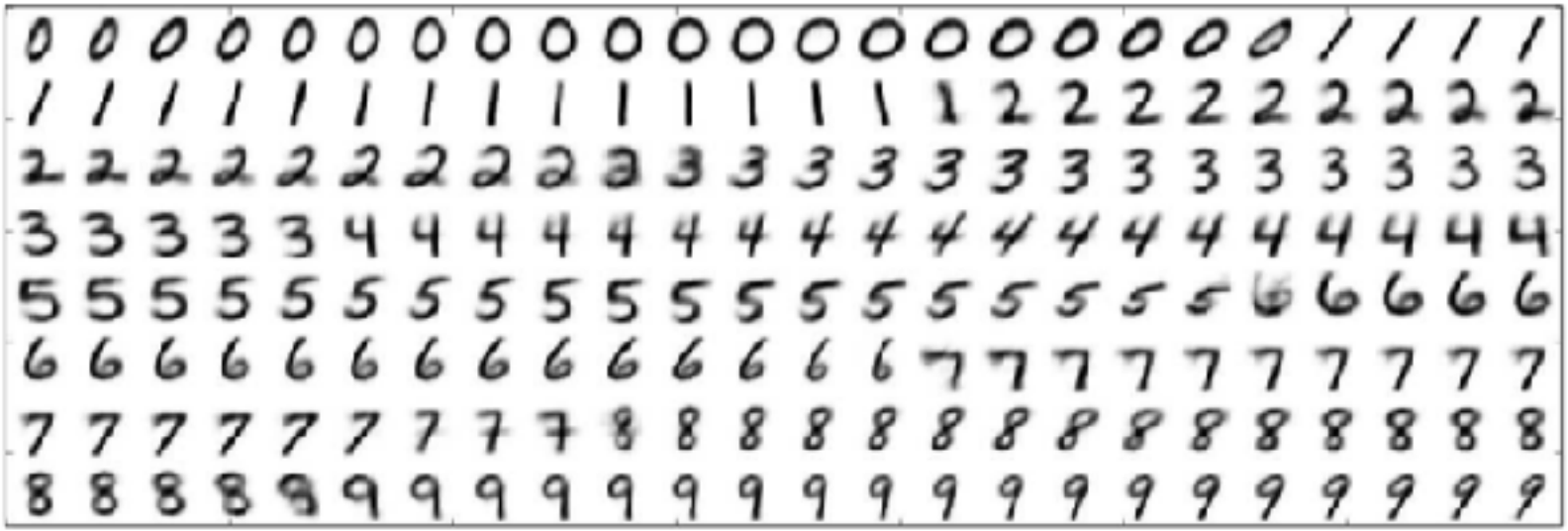


Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, **Ian Goodfellow**, and Brendan Frey. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Conditional Class Latent Spaces

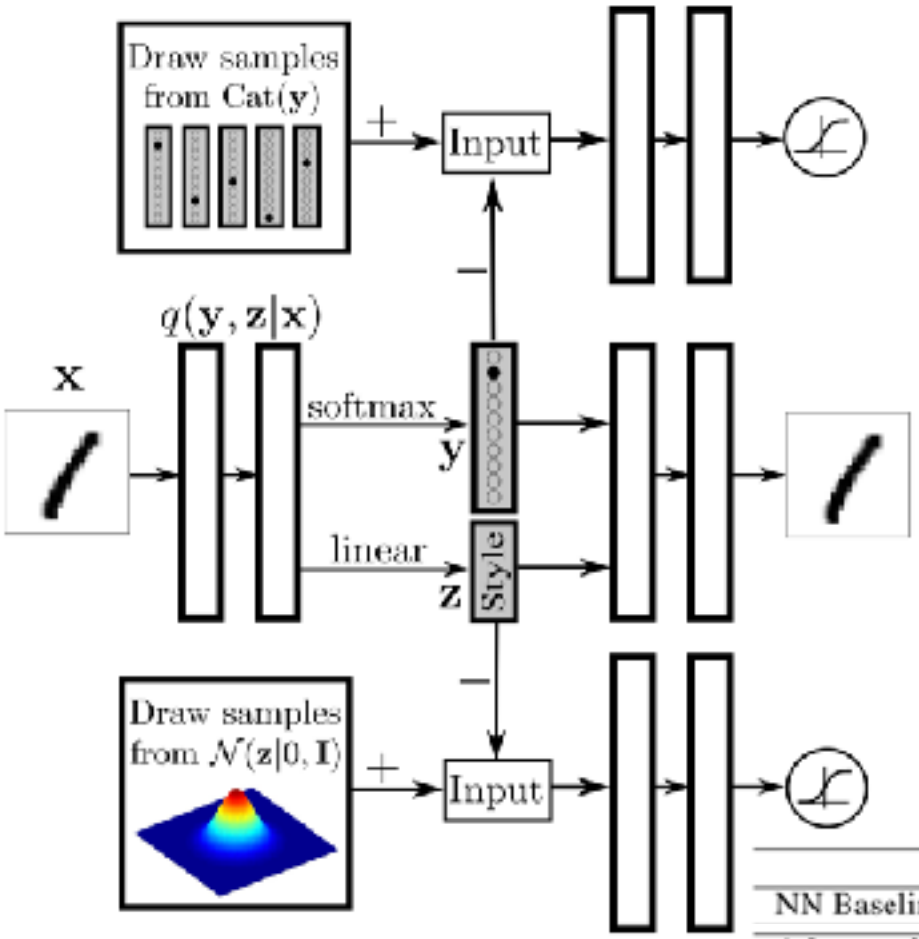


Sample Along Swiss Roll Axis



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

Semi-Supervised Classification



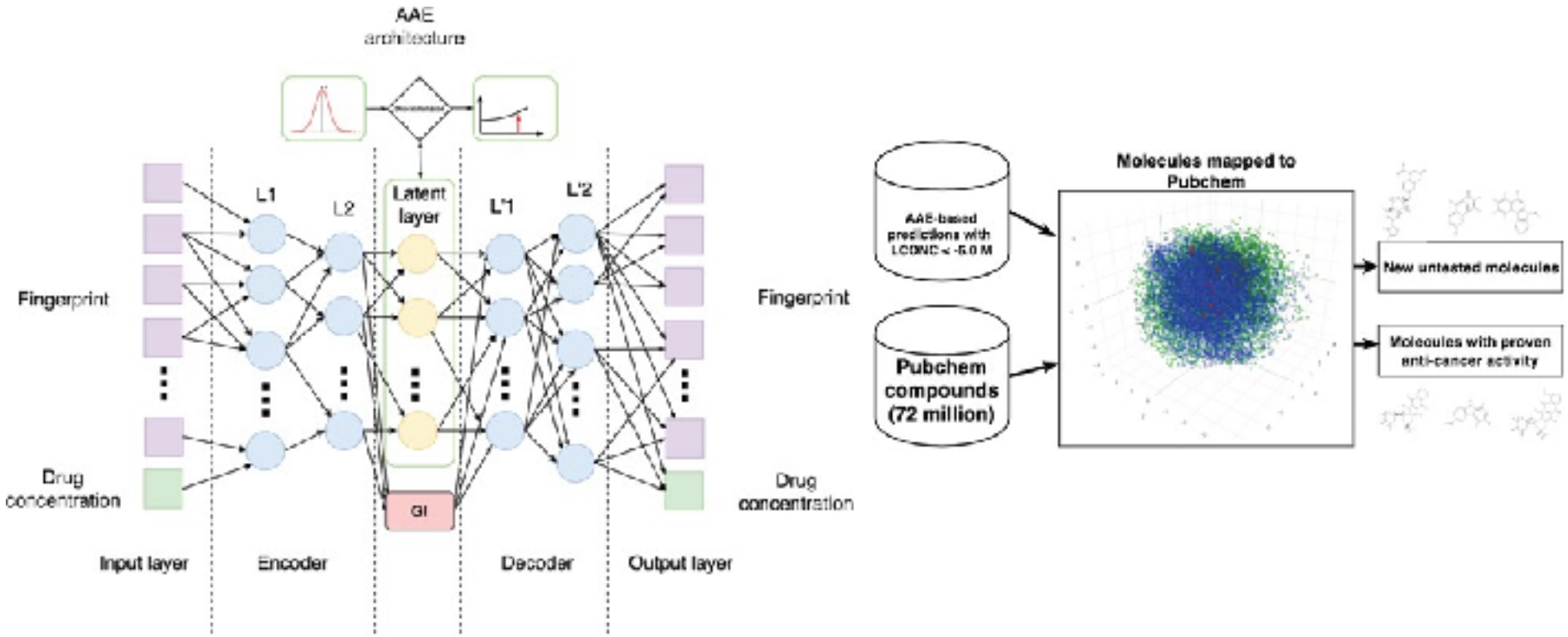
- Top Discriminator
  - Draw samples from one hot encoded labels
  - Ensures aggregated posterior matches class distributions
- Bottom Discriminator
  - Same as previous
  - Constrains latent representation
- Supervised Training
  - After GAN training:
  - Use small mini-batches on $q(y|z)$

	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

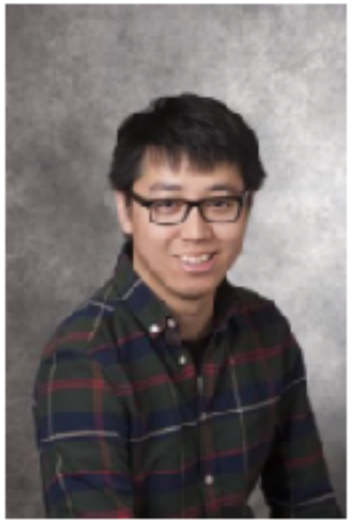
What is something cool about your field?
We use mobile devices to sensor medical quantity out of clinic and build much cheaper alternatives that could change peoples' lives.

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).

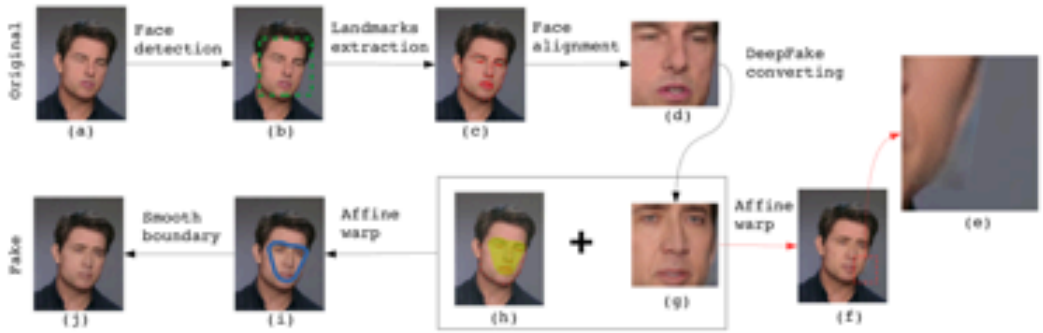
Deep Fakes: A Looming Crisis for National Security, Democracy and Privacy?

By Robert Chesney Danielle Citron Wednesday, February 21, 2018, 10:00 AM

Privacy Paradox: Rethinking Solitude

"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



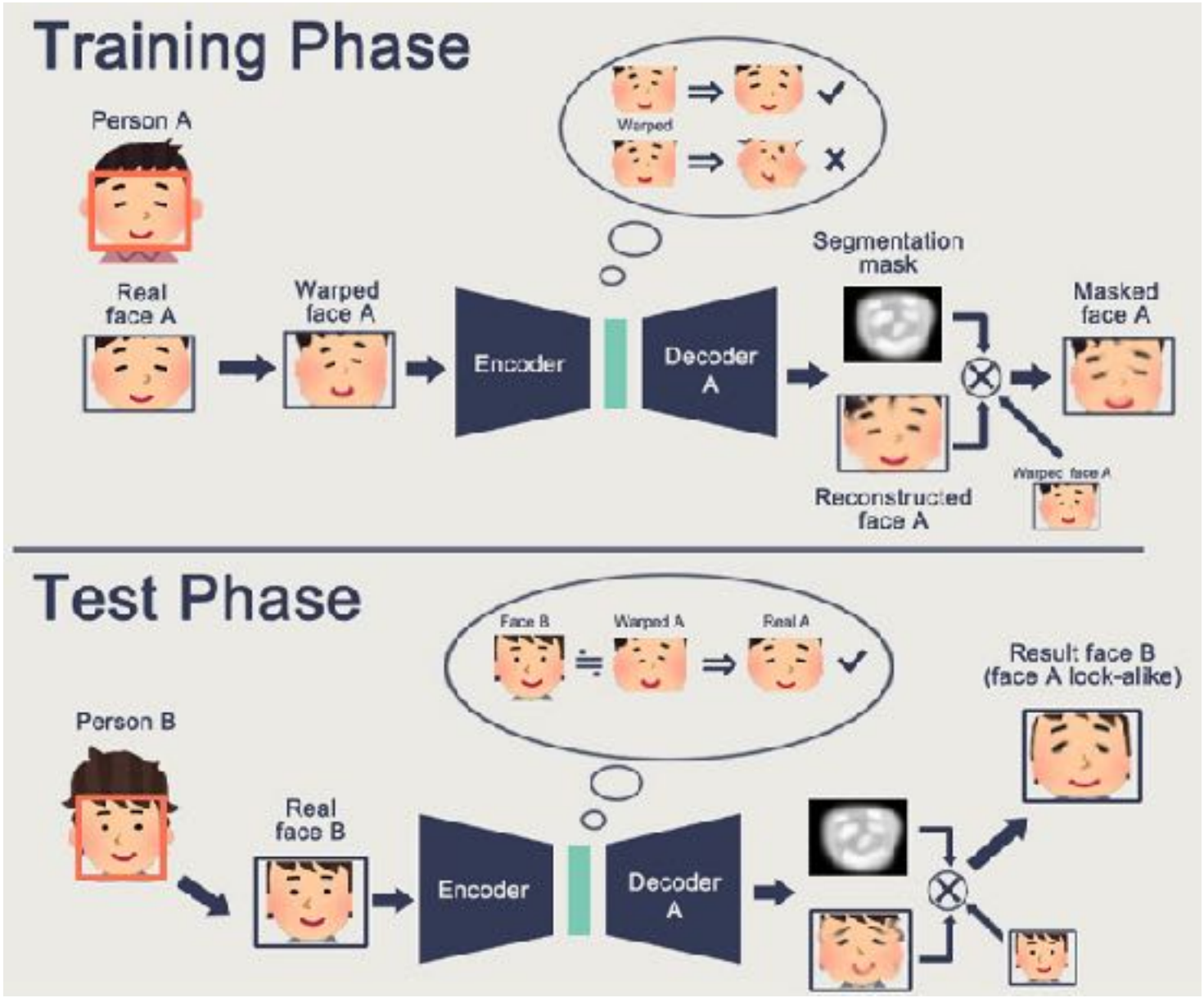
Original

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
Encoder
Decoder A
Warped
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

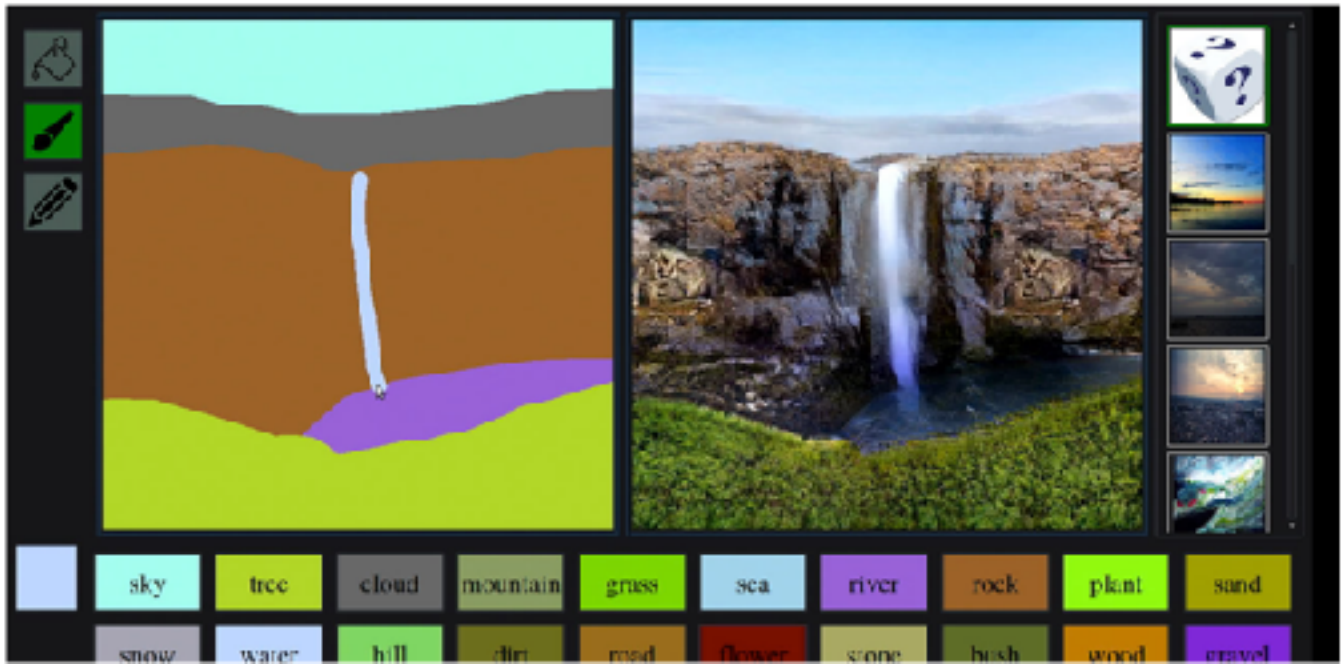
SMU | Bobby B. Lyle School of Engineering

GAN Paper Presentation

Stroke of Genius: GauGAN Turns Doodles into Stunning, Photorealistic Landscapes

NVIDIA research harnesses generative adversarial networks to create highly realistic scenes.

March 18, 2019 by EHA SALIAN



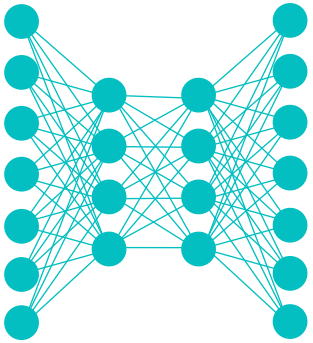
Lecture Notes for

Neural Networks and Machine Learning

Practical GANs

Next Time:
GAN Examples

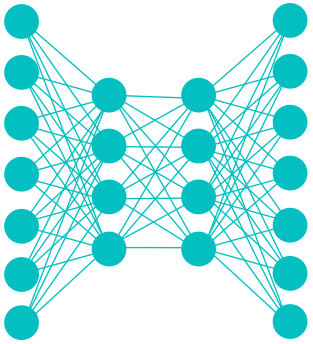
Reading: Chollet CH8



Lecture Notes for

Neural Networks and Machine Learning

The GAN-Zooks

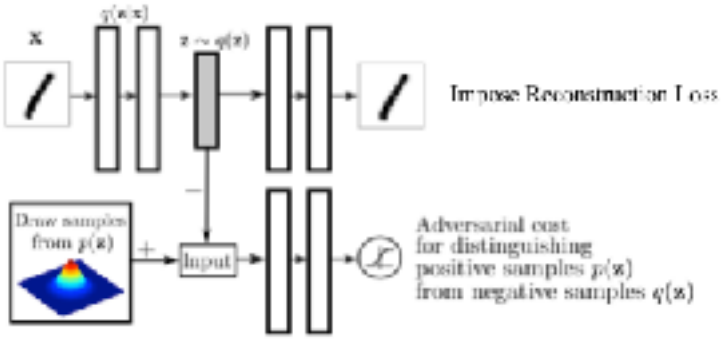


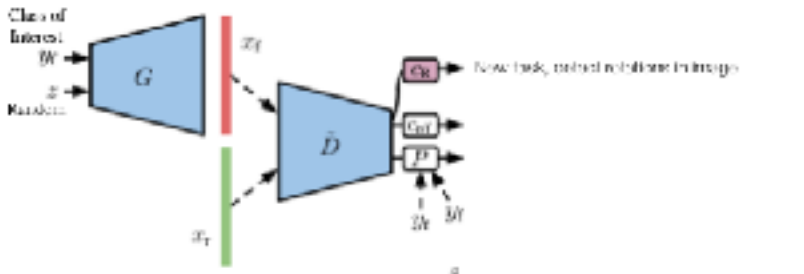
Logistics and Agenda

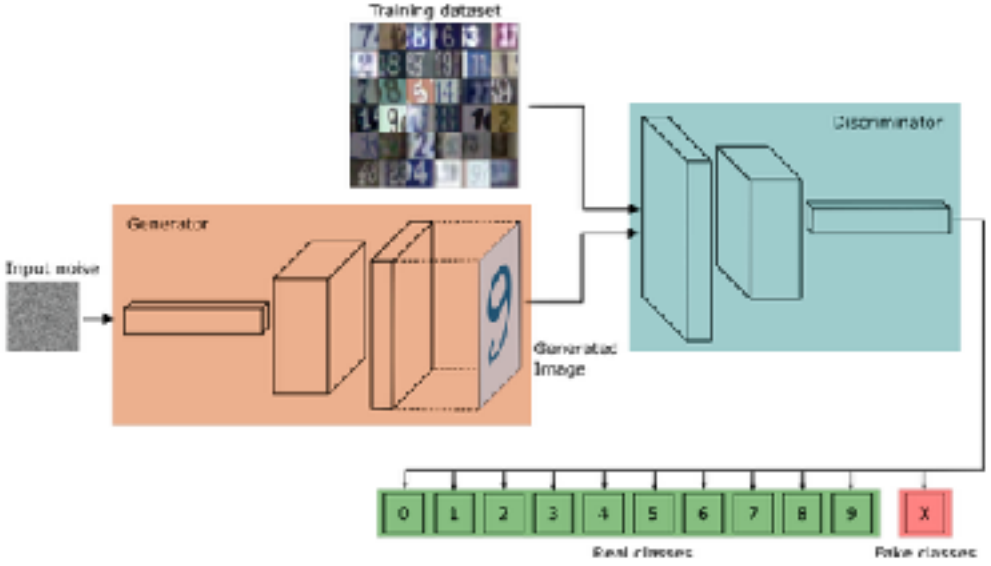
- Logistics
  - Lab 3 due Sunday!
- Agenda
  - More GANs:
    - LSGAN
    - InfoGAN
    - CycleGAN

Last Time

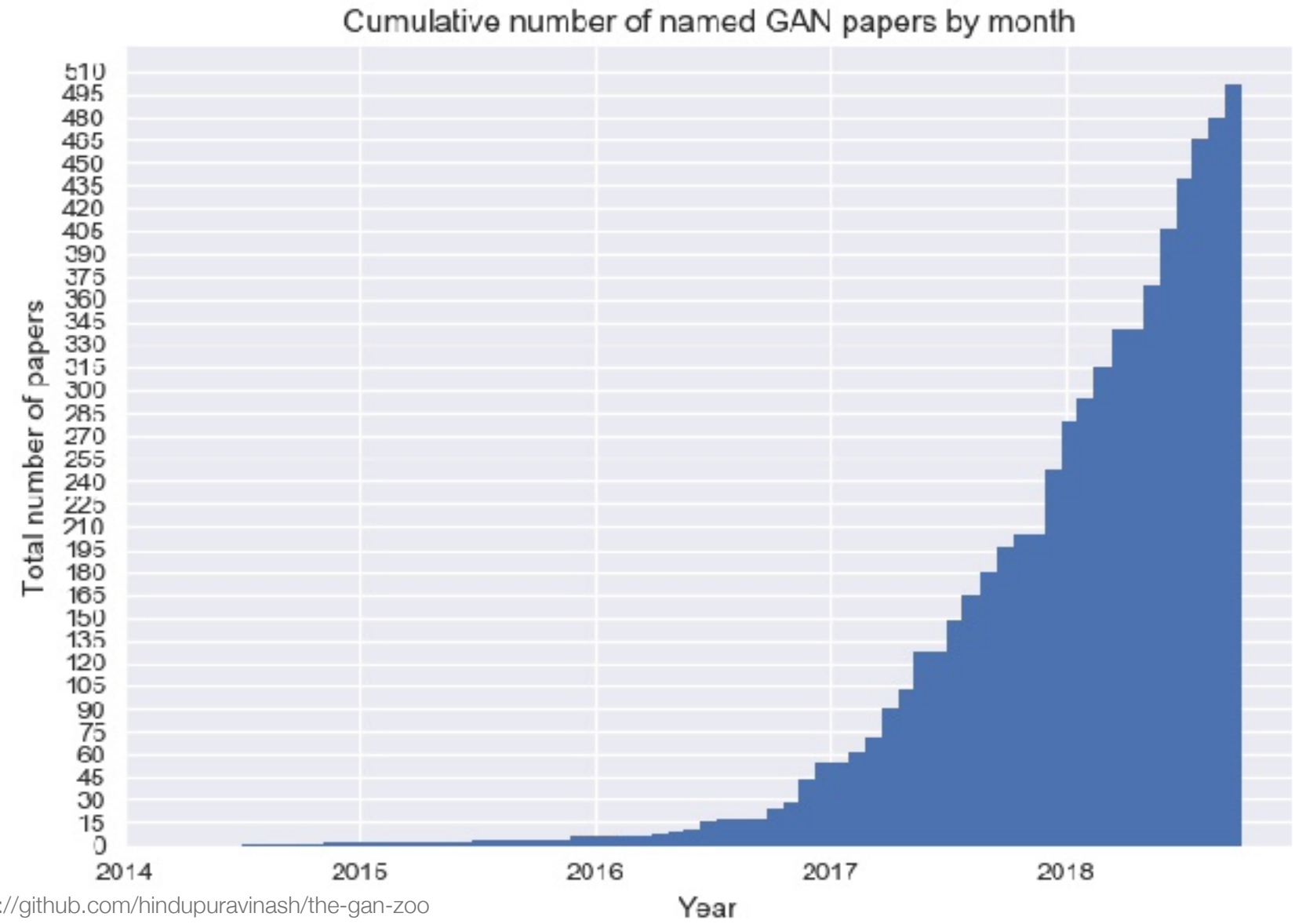
- How can we enforce constraints on the latent space with GANs?







Example GANs, Abridged



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

LSGAN

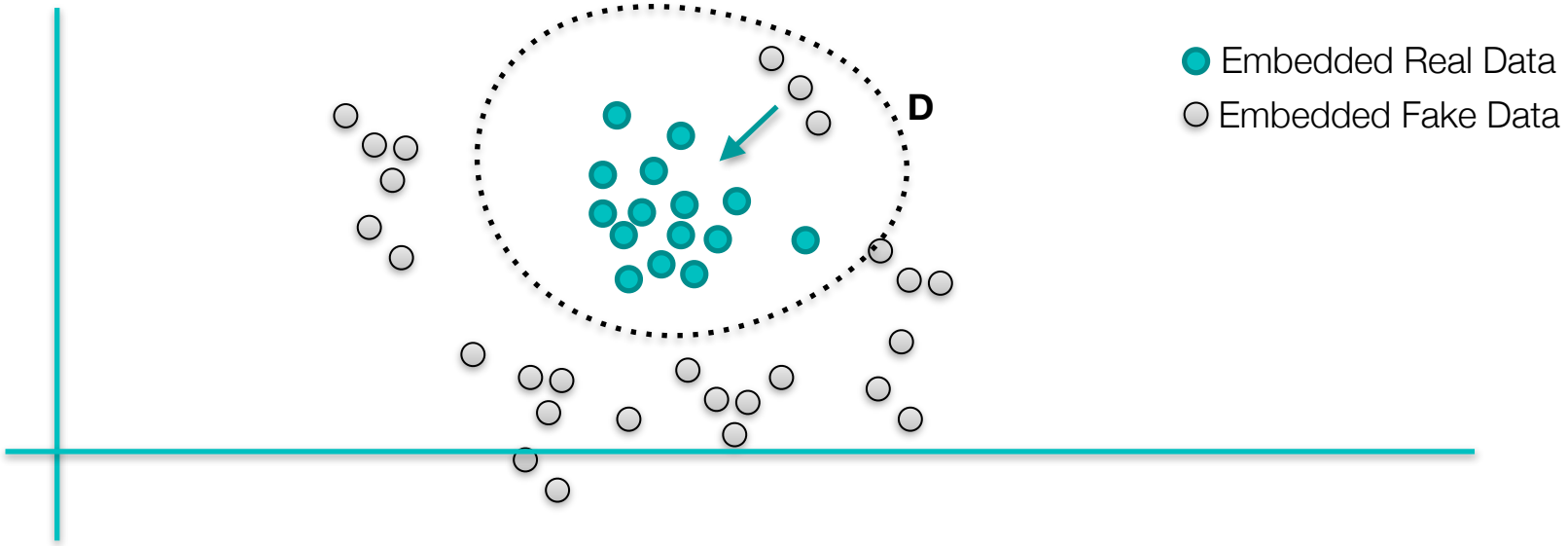
Least Squares Generative Adversarial Networks

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:

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

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

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

where χ^2_{Pearson} is the Pearson χ^2 divergence. Thus minimizing Equation 4 yields minimizing the Pearson χ^2 divergence between $p_{\text{d}} + p_g$ 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?

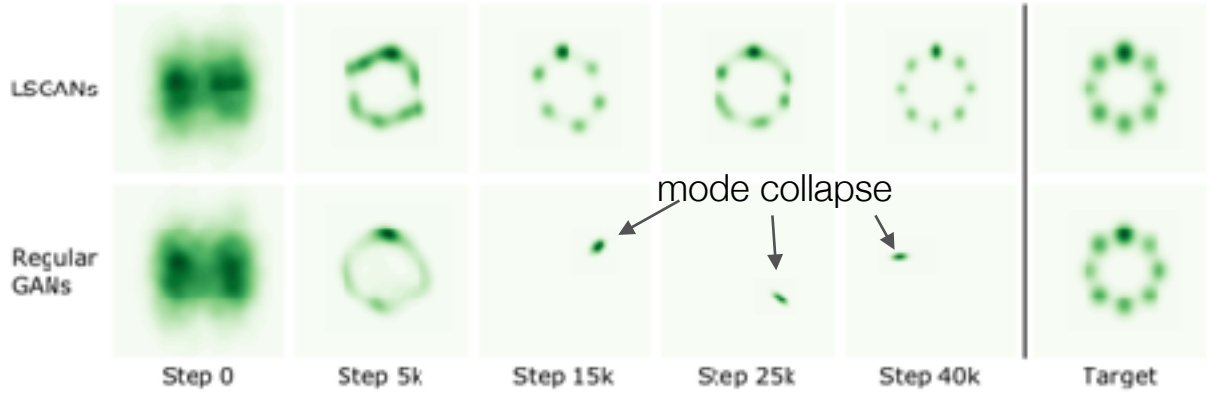
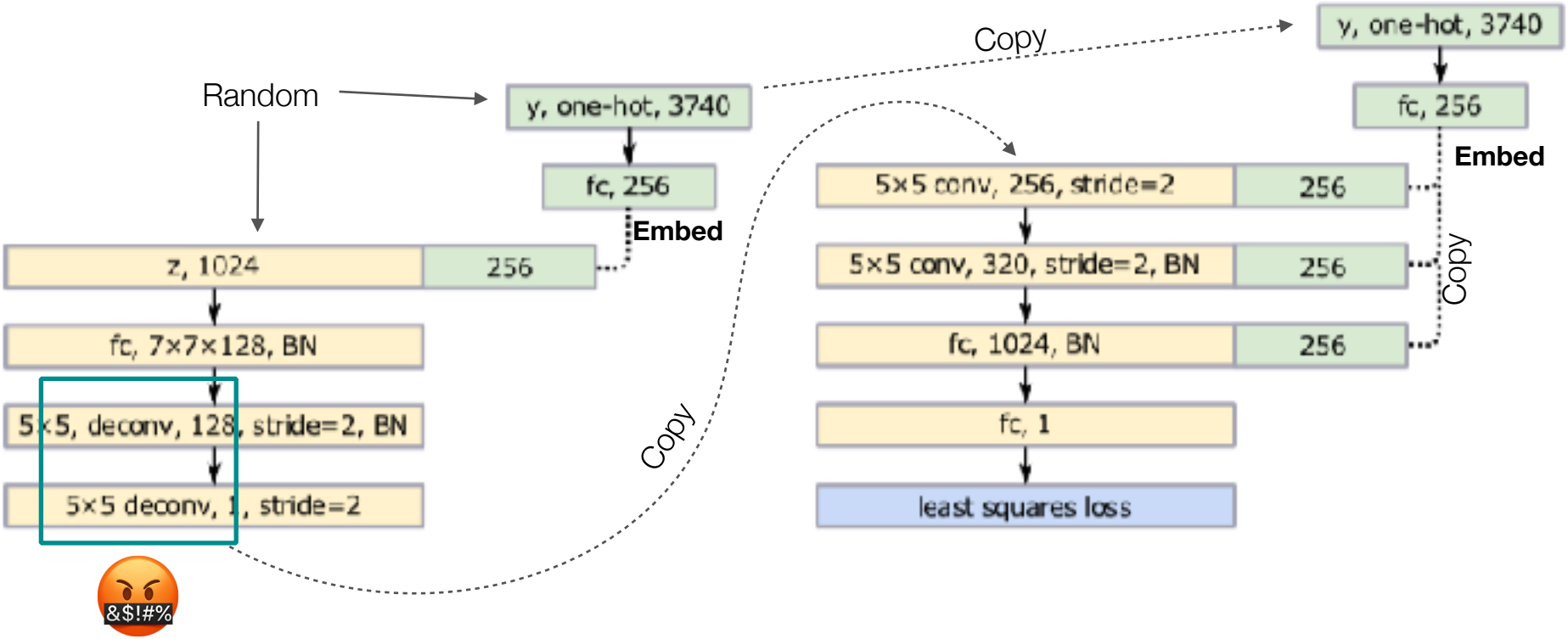


Figure 8: Dynamic results of Gaussian kernel estimation for LSGANs and regular GANs. The final column shows the real data distribution.

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

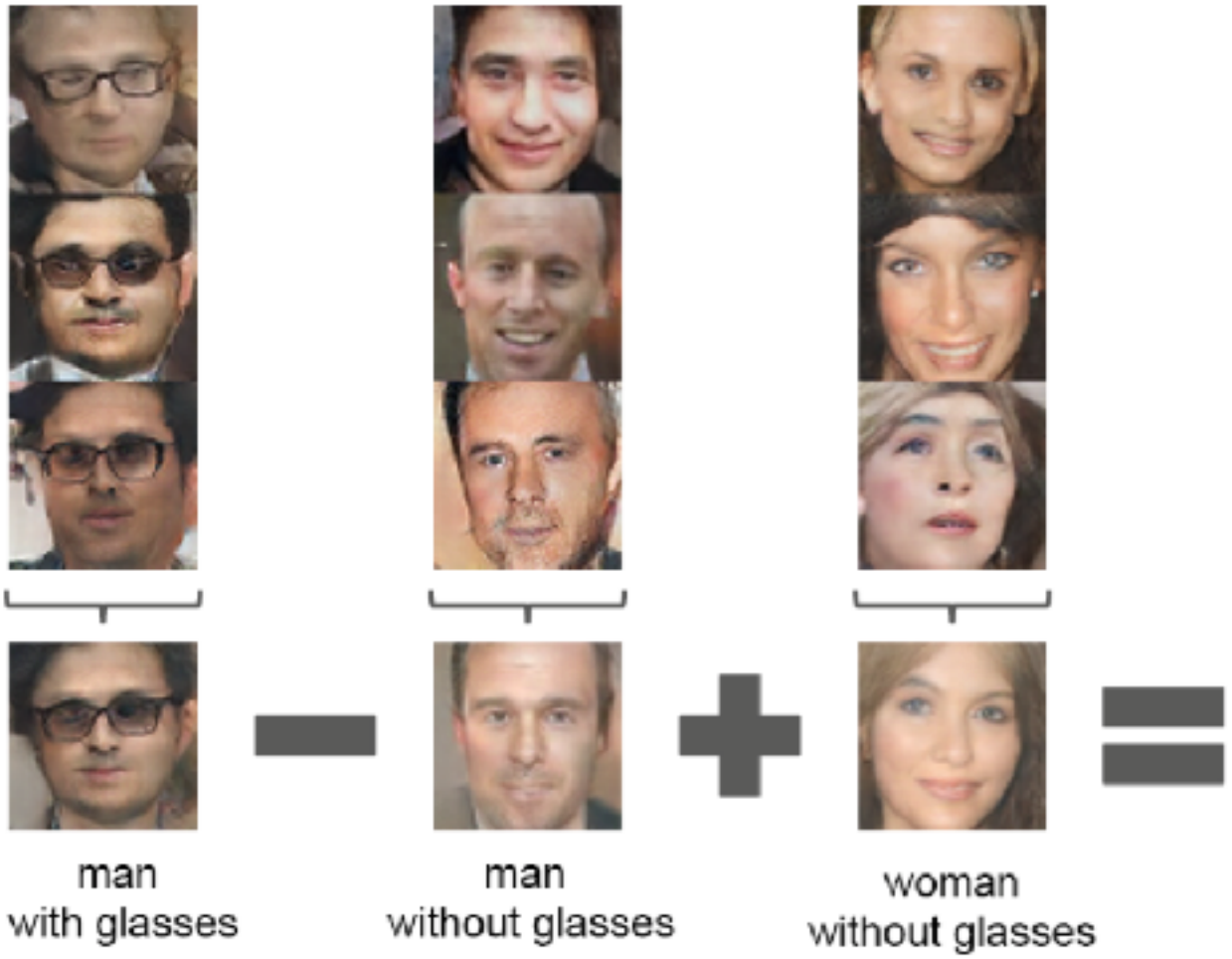
InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

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!

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, targeting 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:

$$
\begin{aligned}
I(c; G(z,c)) &= H(c) - H(c|G(z,c)) \\
&= \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log P(c'|x)]] + H(c) && \text{Expand Def. of Entropy} \\
&= \mathbb{E}_{x\sim G(z,c)}[\underbrace{D_{\mathrm{KL}}(P(\cdot|x) \parallel Q(\cdot|x))}_{\text{make Q close to P } \geq 0} + \mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) && \text{Insert Q Approximation} \\
&\geq \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) && \text{Define a Lower Bound to Maximize}
\end{aligned}
$$

sample from Q

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)

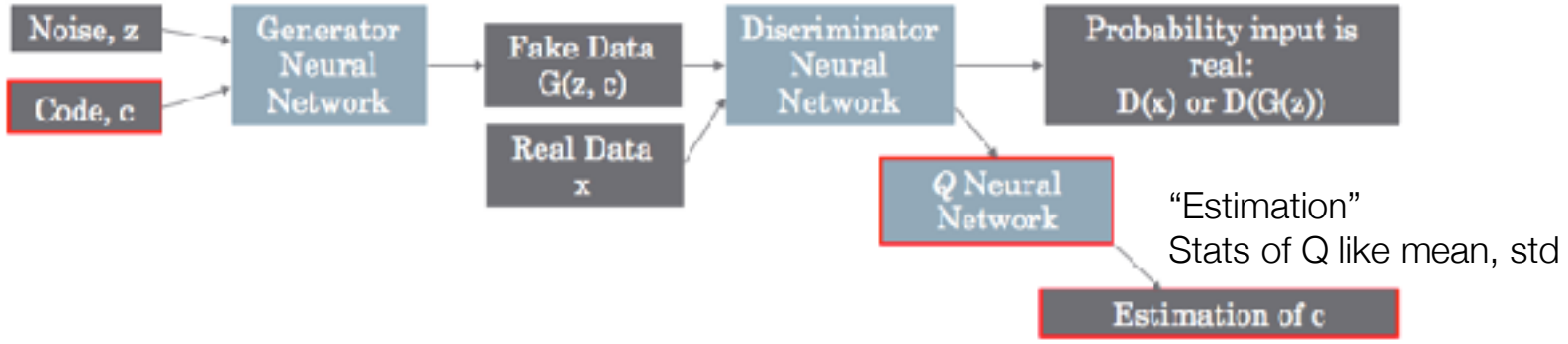
$$I(c; G(z,c)) \geq \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c)$$

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:
Make Q whatever distribution you want to sample from!

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



"Estimation"
Stats of Q like mean, std

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

Implementation

```
# Discriminator
validity = Dense(1, activation='sigmoid')(img_embedding)

# Recognition
q_net = Dense(128, activation='relu')(img_embedding)
label = Dense(self.num_classes, activation='softmax')(q_net)

# Return discriminator and recognition network
return Model(img, validity), Model(img, label)
```

```
# Build and the discriminator and recognition network
self.discriminator, self.auxilliary = self.build_disk_and_q_net()

self.discriminator.compile(loss=['binary_crossentropy'],
    optimizer=optimizer,
    metrics=['accuracy'])

# Build and compile the recognition network Q
self.auxilliary.compile(loss=[self.mutual_info_loss],
    optimizer=optimizer,
    metrics=['accuracy'])
```

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

Implementation

```
def mutual_info_loss(self, c, c_given_x):
    """The mutual information metric we aim to minimize"""
    eps = 1e-8
    conditional_entropy = K.mean(- K.sum(K.log(c_given_x + eps) * c, axis=1))
    entropy = K.mean(- K.sum(K.log(c + eps) * c, axis=1))

    return conditional_entropy + entropy
```

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

```
losses = ['binary_crossentropy', self.mutual_info_loss]
```

```
# The discriminator takes generated image as input and determines validity
valid = self.discriminator(img)
# The recognition network produces the label
target_label = self.auxilliary(img)

# The combined model  (stacked generator and discriminator)
self.combined = Model(gen_input, [valid, target_label])
self.combined.compile(loss=losses,
    optimizer=optimizer)
```

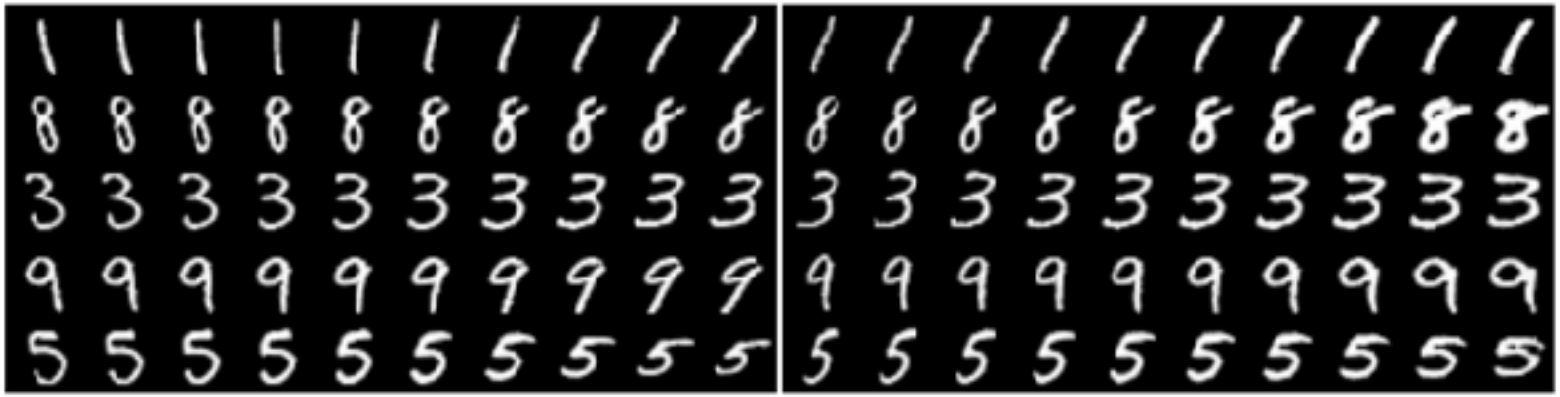
https://github.com/eriklindernoren/Keras-GAN/blob/master/infogan/infogan.py

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?

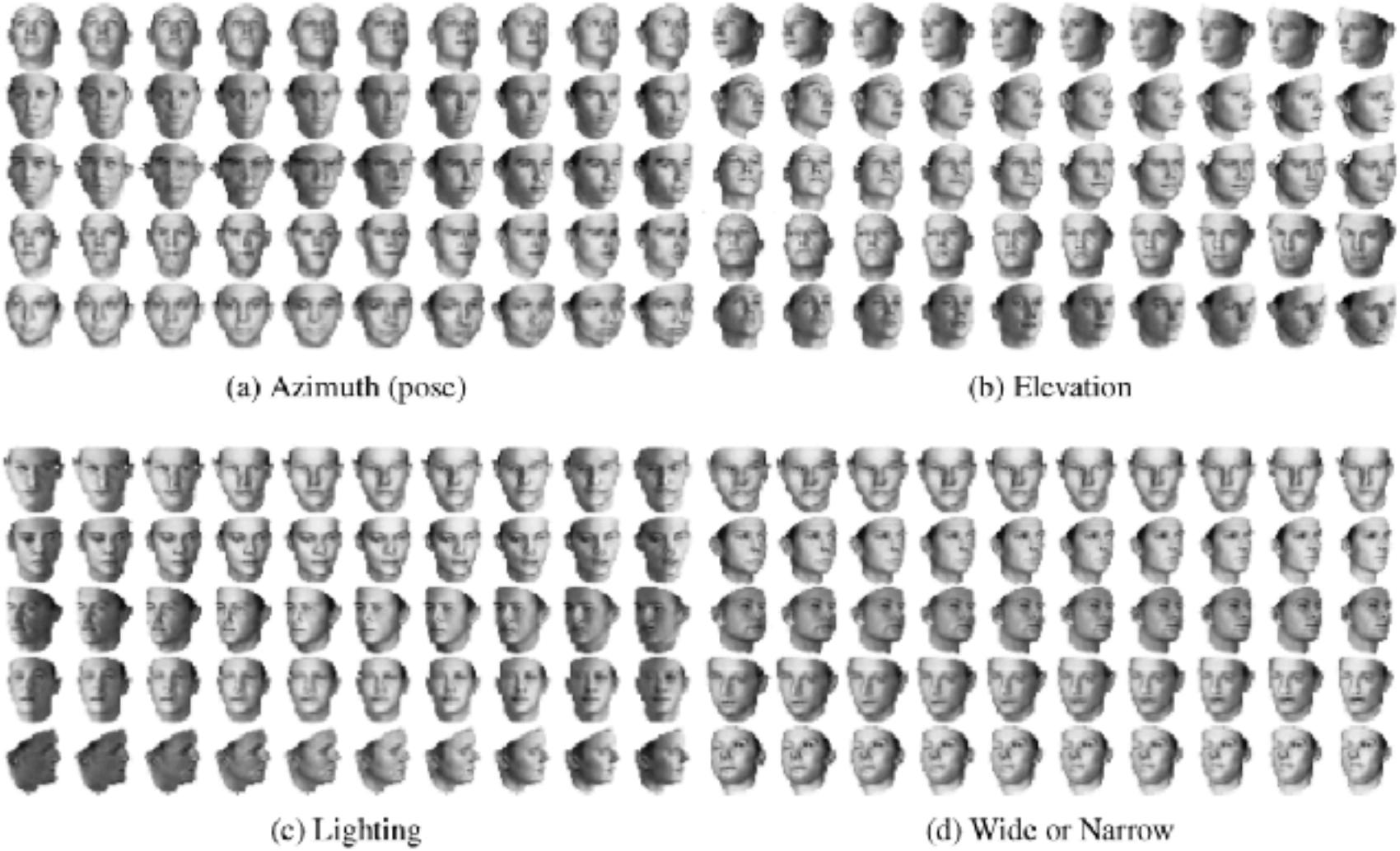


Figure 3: **Manipulating latent codes on 3D Faces:** We show the effect of the learned continuous

Did they really disentangle?

(a) Azimuth (pose)

(b) Presence or absence of glasses

(c) Hair style

(d) Emotion

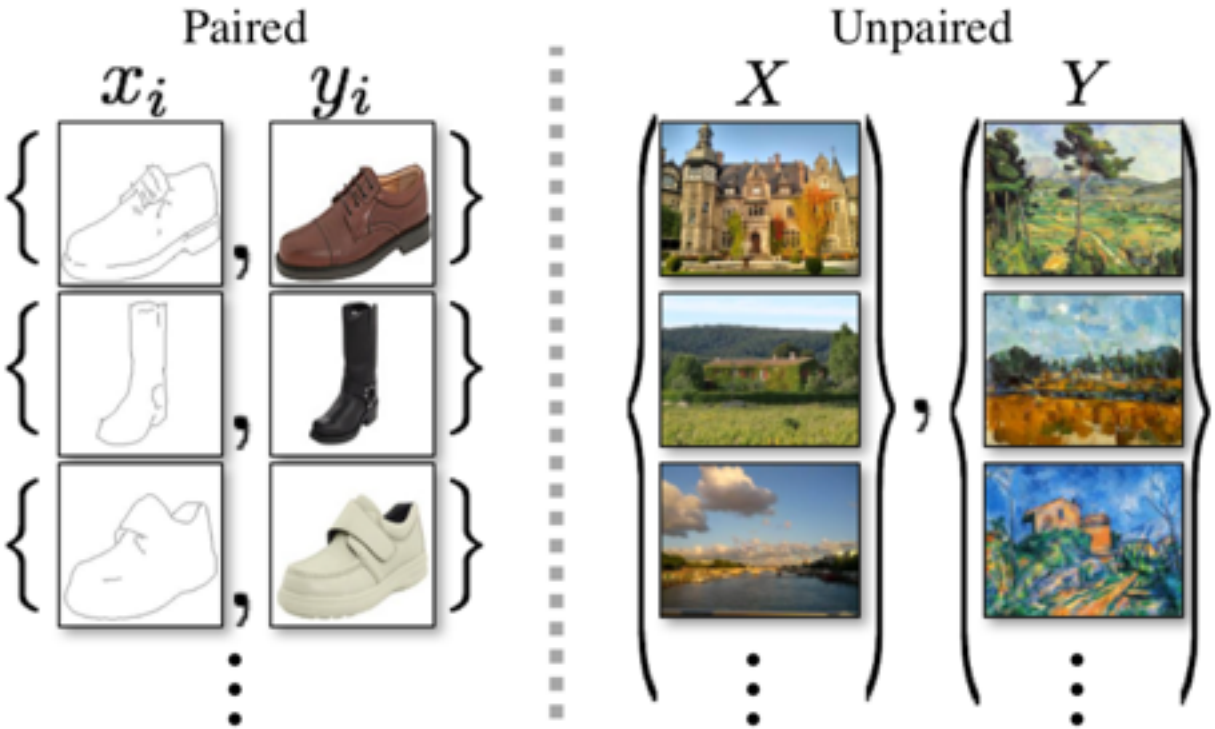
CycleGAN

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

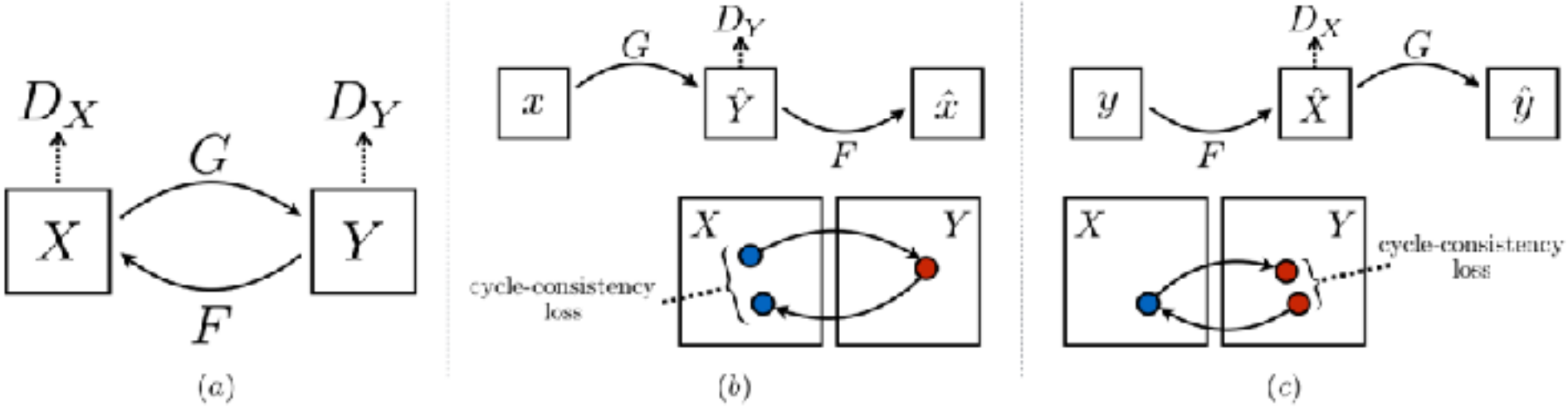
Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros
Berkeley AI Research (BAIR) laboratory, UC Berkeley

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:

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

- 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!

```
# 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'])
```

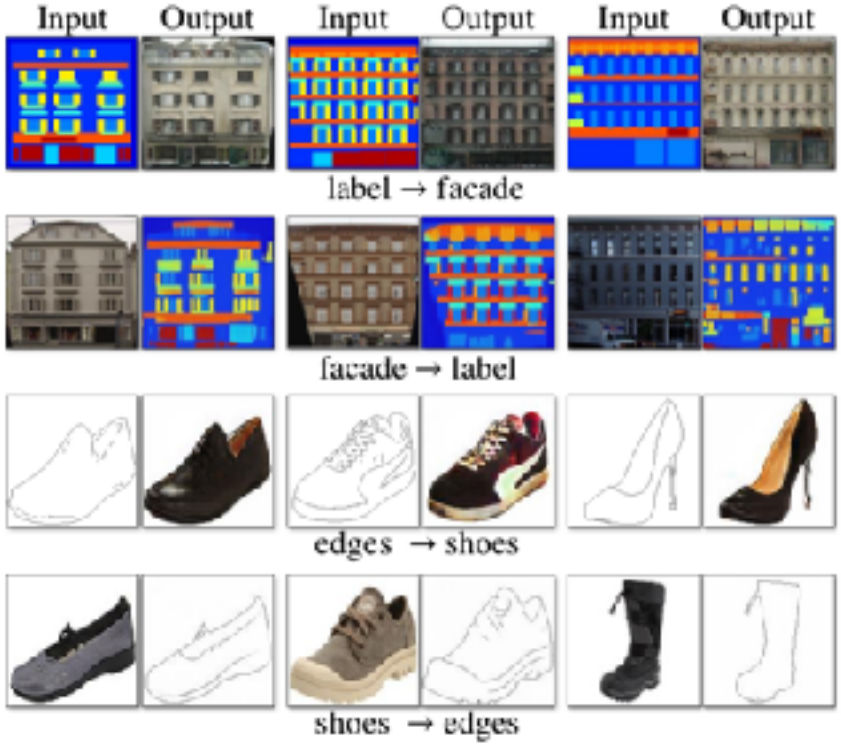
```
# Translate images to the other domain
fake_B = self.g_AB(img_A)
fake_A = self.g_BA(img_B)
# Translate images back to original domain
reconstr_A = self.g_BA(fake_B)
reconstr_B = self.g_AB(fake_A)
# Identity mapping of images
img_A_id = self.g_BA(img_A)
img_B_id = self.g_AB(img_B)
```

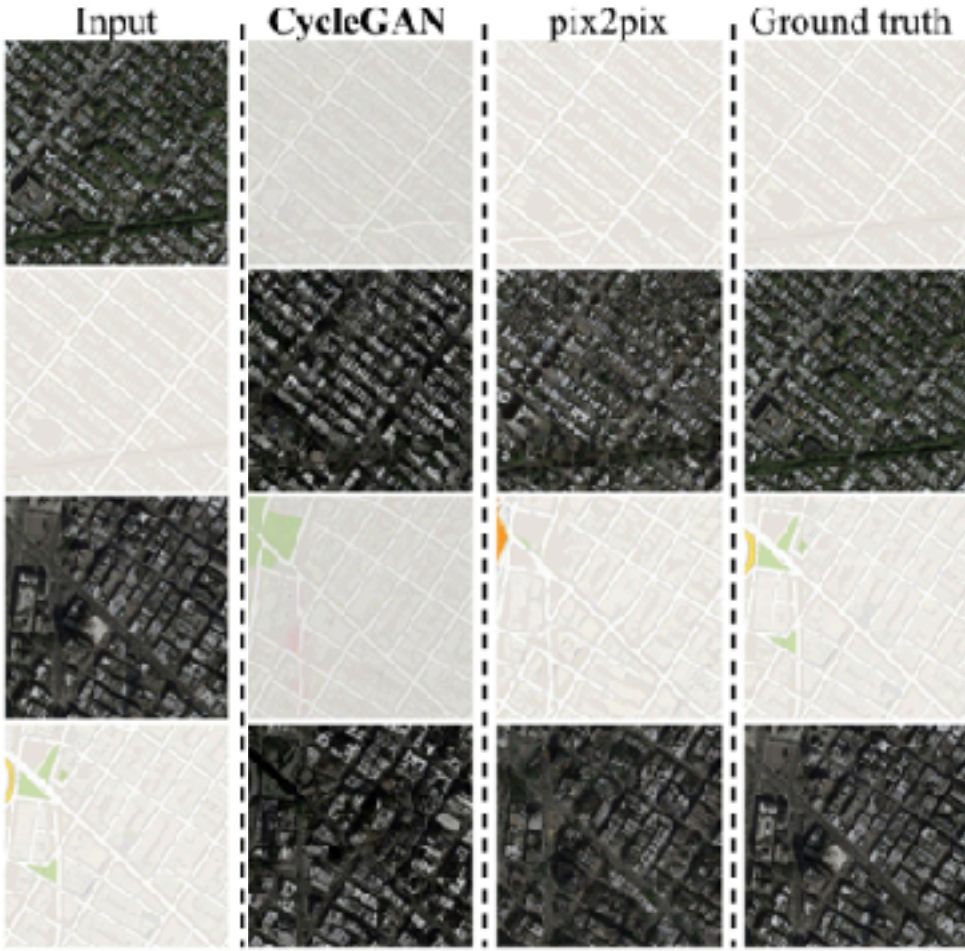
```
# Combined model trains generators to fool discriminators
self.combined = Model(inputs=[img_A, img_B],
                      outputs=[ valid_A, valid_B,
                                reconstr_A, reconstr_B,
                                img_A_id, img_B_id ])
self.combined.compile(loss=['mse',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

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

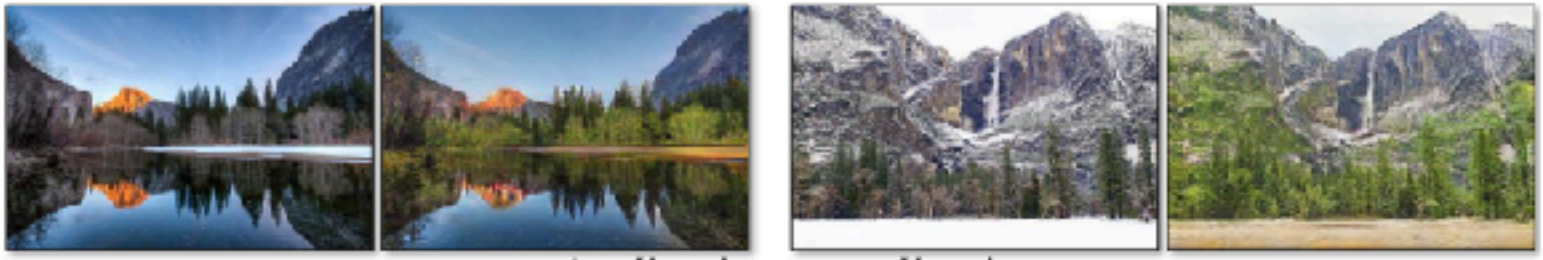
Results


Input
CycleGAN
pix2pix
Ground truth


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


Input
CycleGAN
pix2pix
Ground truth

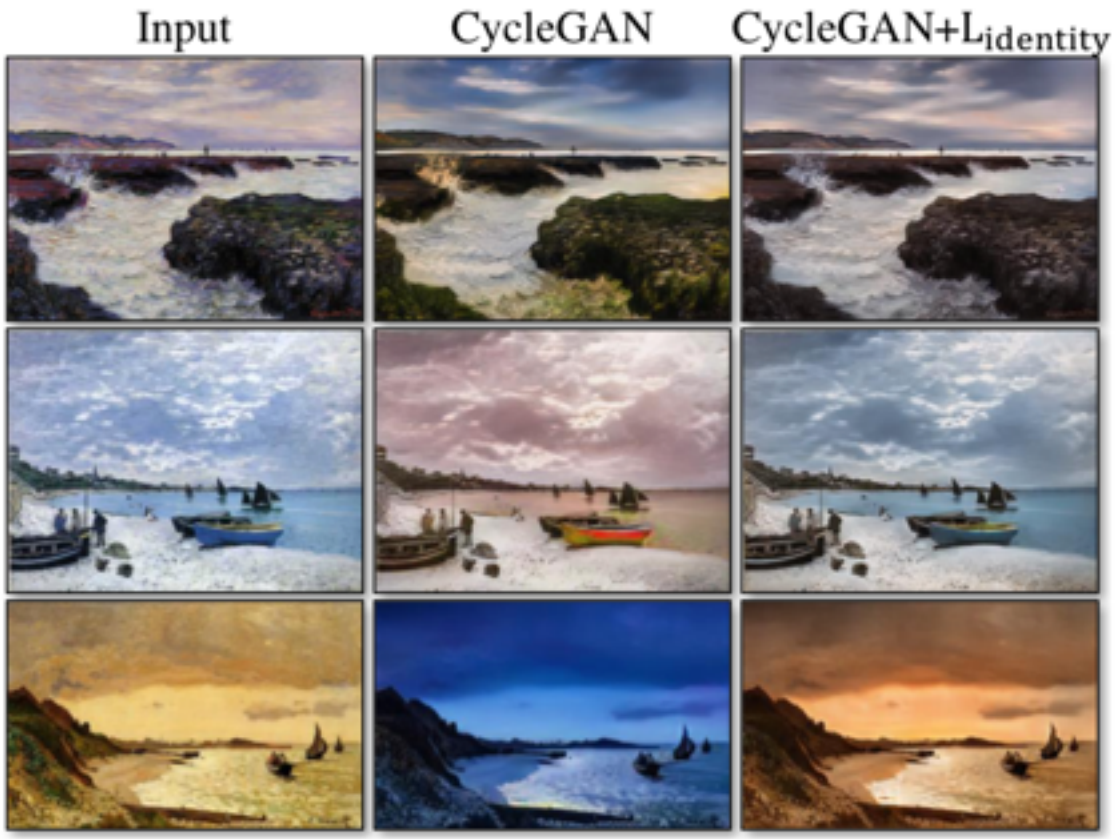
Fun Results


winter Yosemite → summer Yosemite
summer Yosemite → winter Yosemite


apple → orange
orange → apple

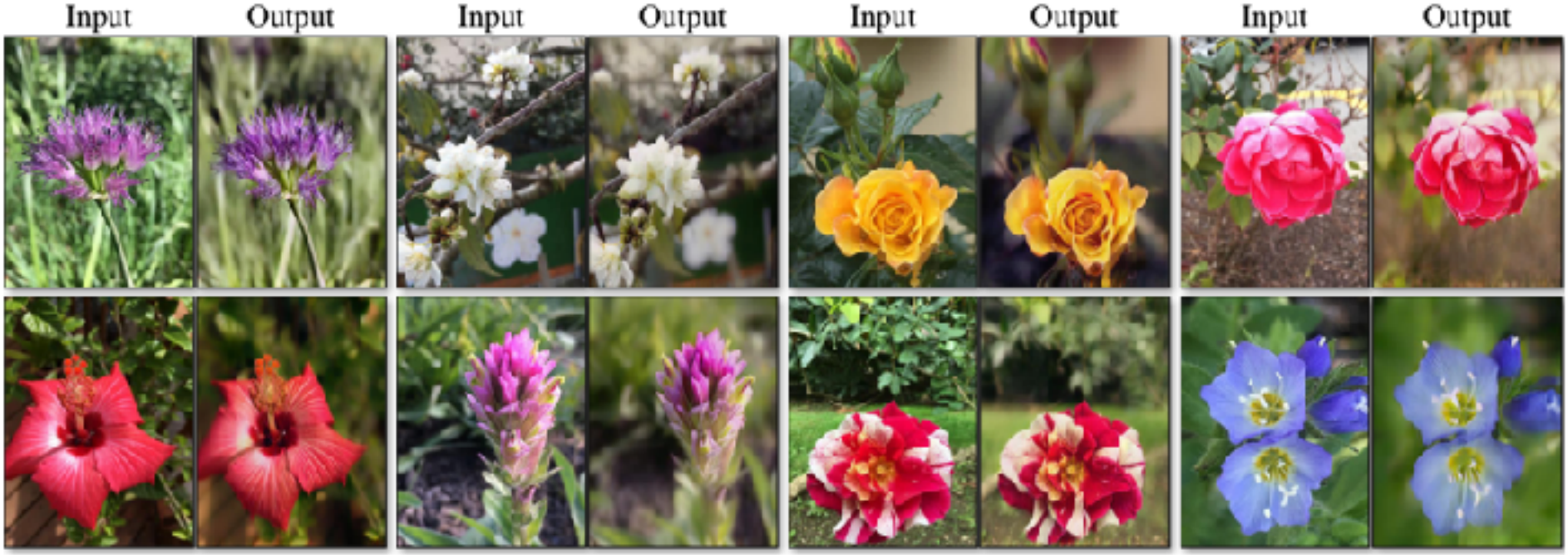
Results: Fixing Hue Changes

- Needed another loss to ensure no changes in the coloring:



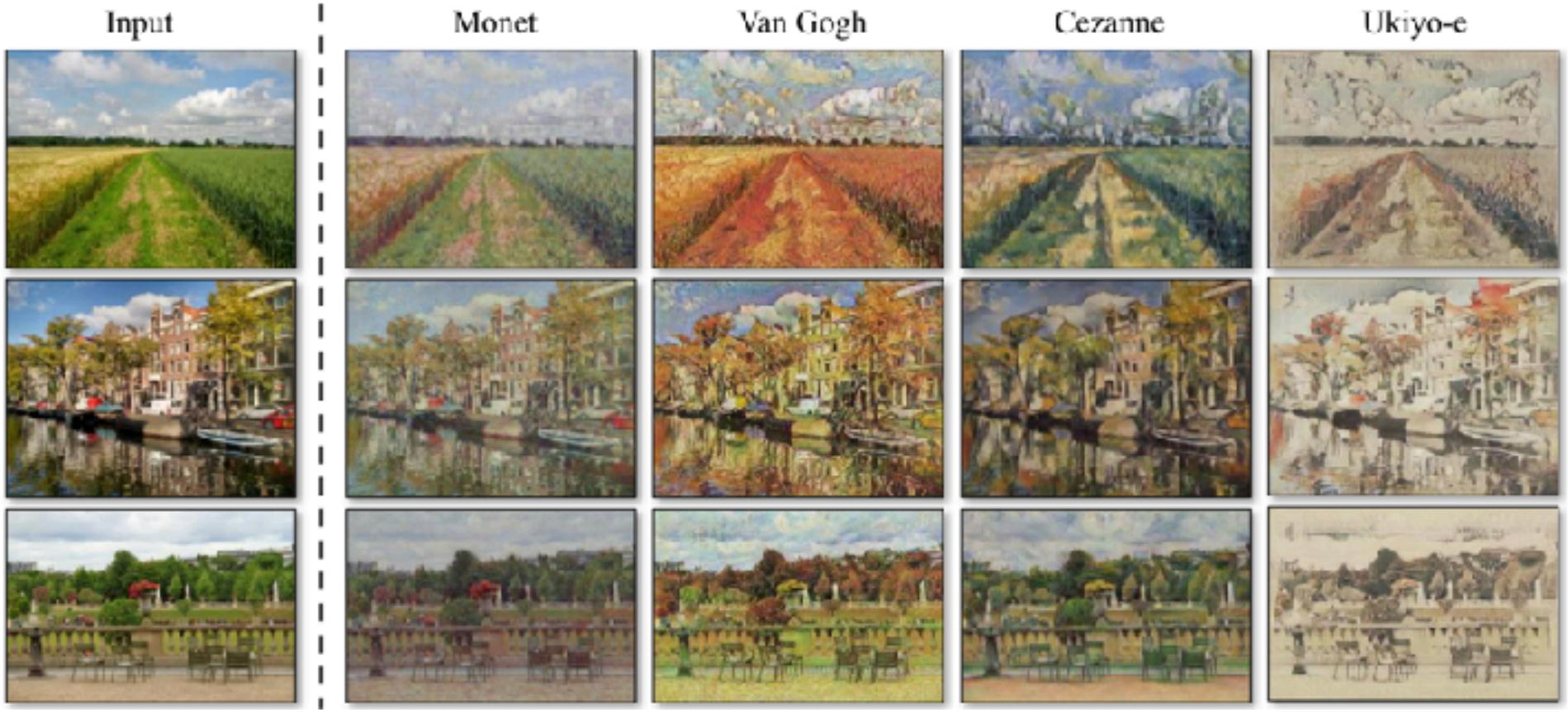
$$\mathcal{L}_{\text{identity}}(G, F) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(x) - x\|_1]$$

Image Enhancement

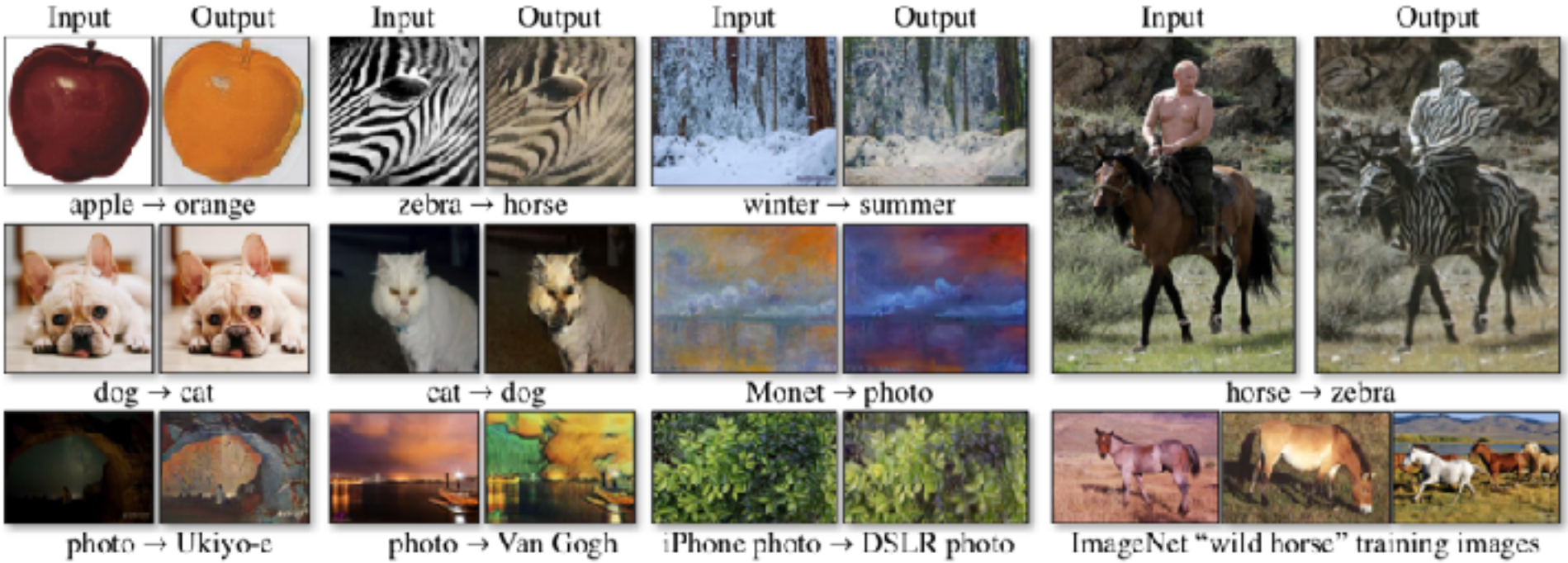


Map from iPhone Images to DSLR Photos

Full Circle: Back to Style Transfer



They can't all be gold



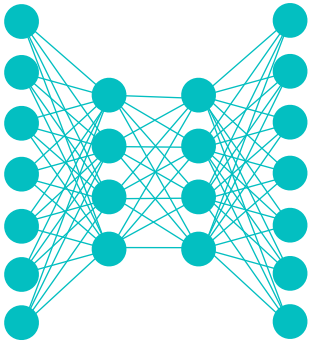
Lecture Notes for

Neural Networks and Machine Learning

Holy GAN Zooks

Next Time:
Continued

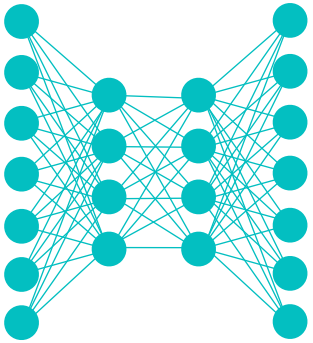
Reading: None



Lecture Notes for

Neural Networks and Machine Learning

Holy GAN-Zooks Continued



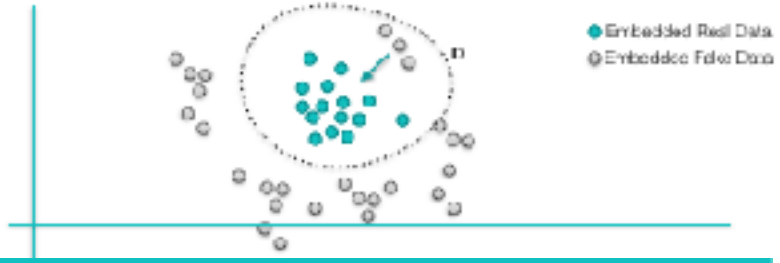
Logistics and Agenda

- Logistics
  - Next Lecture: ??
  - GAN Lab Posted
- Agenda
  - More GANs:
    - GAIA
    - 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)

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

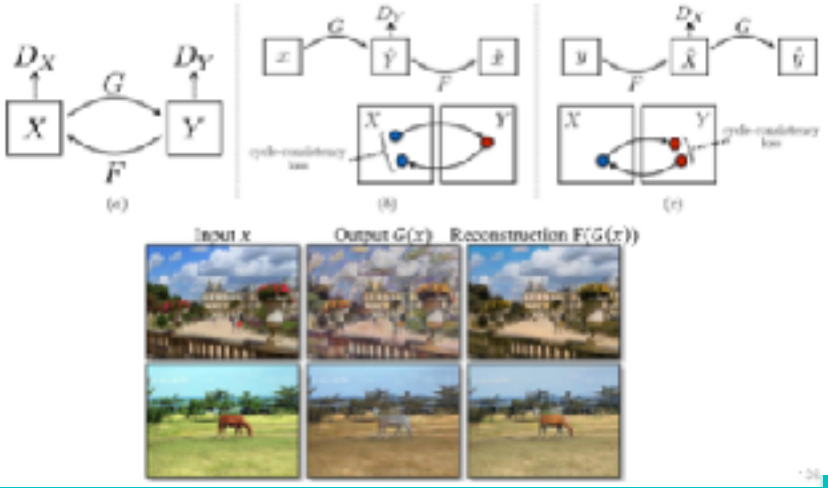
Lemma 5.1 *For random variables X, Y and function $f(x, y)$ under suitable regularity conditions,* $\mathbb{E}_{x\sim X, y\sim Y|x}[f(x,y)] = \mathbb{E}_{x\sim X, y\sim Y|x, x'\sim X|y}[f(x',y)]$

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

Re-parameterize: Make it whatever distribution you want to sample from!

New 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

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

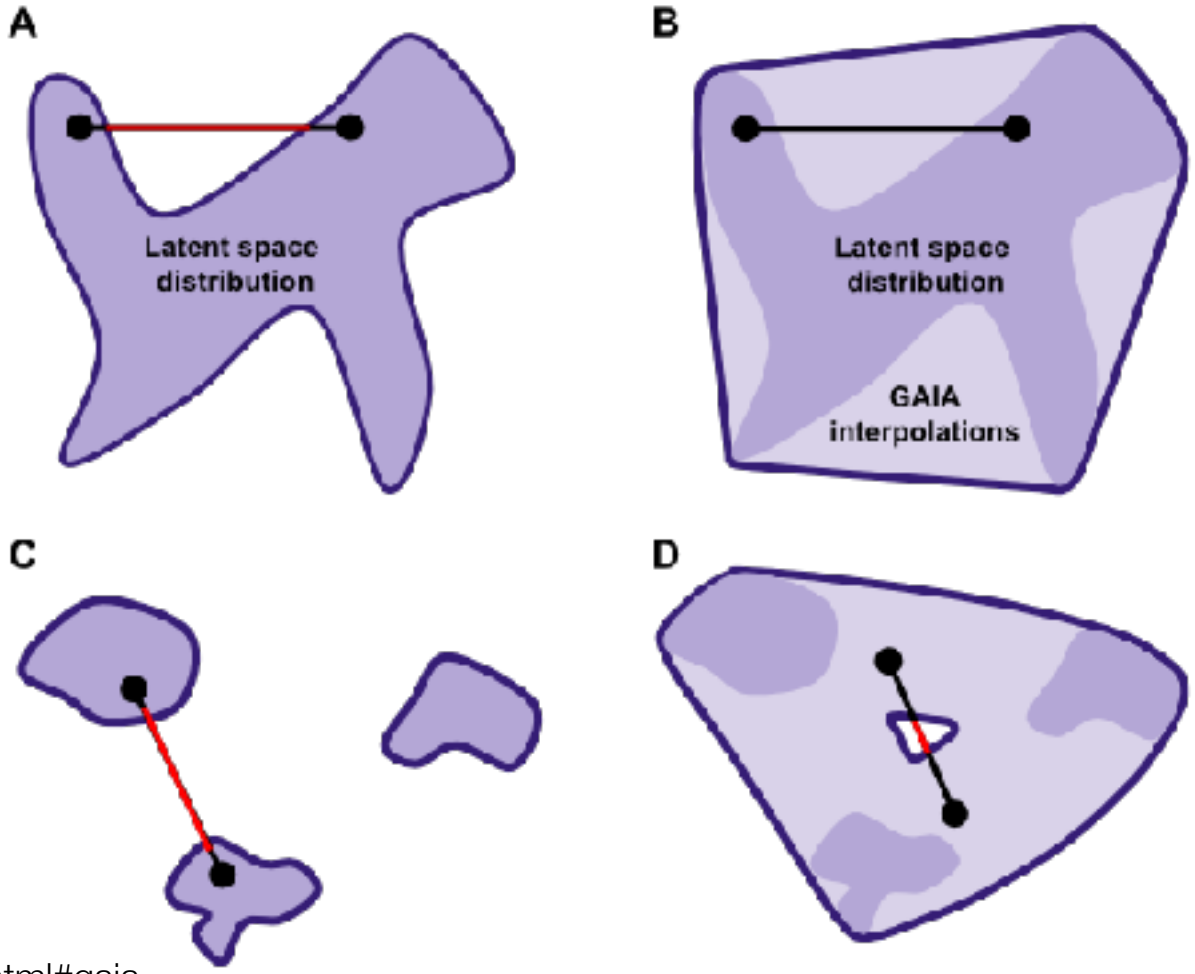
Brad Theilman
UC San Diego
btheilma@ucsd.edu

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

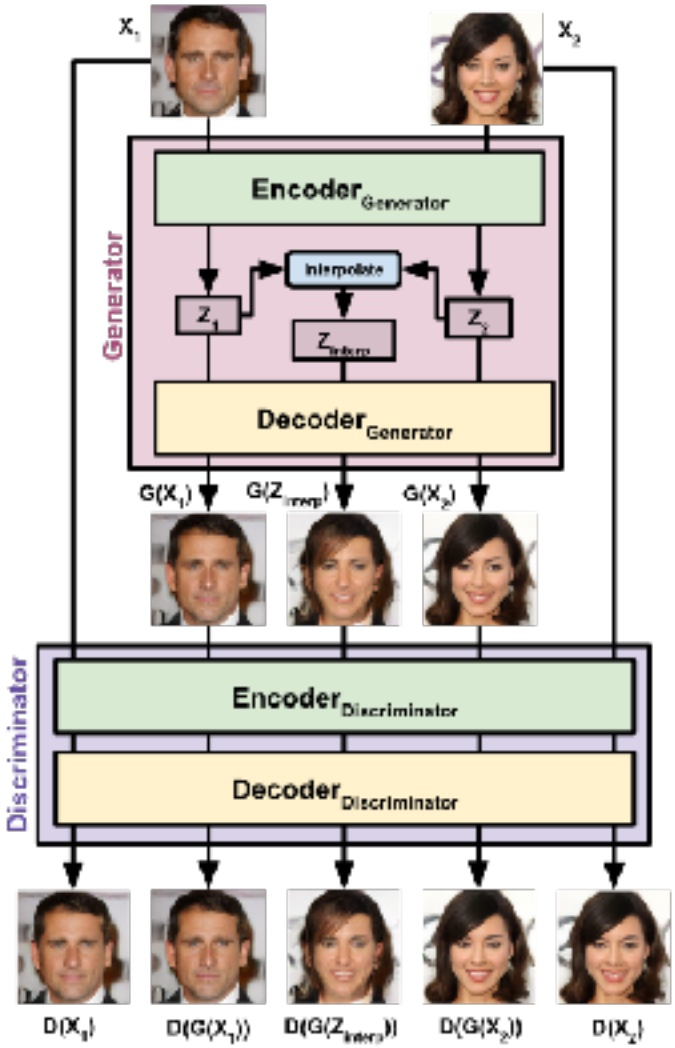


https://timsainburg.com/gaia.html#gaia

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

Text to Image Synthesis with GANs

Generative Adversarial Text to Image Synthesis

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran
Bernt Schiele, Honglak Lee

REEDSCOT[1], AKATA[2], XCYAN[1], LLAJAN[1]
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)

Generate Images from Text

- Need a correspondence function for text and image representations:

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

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

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

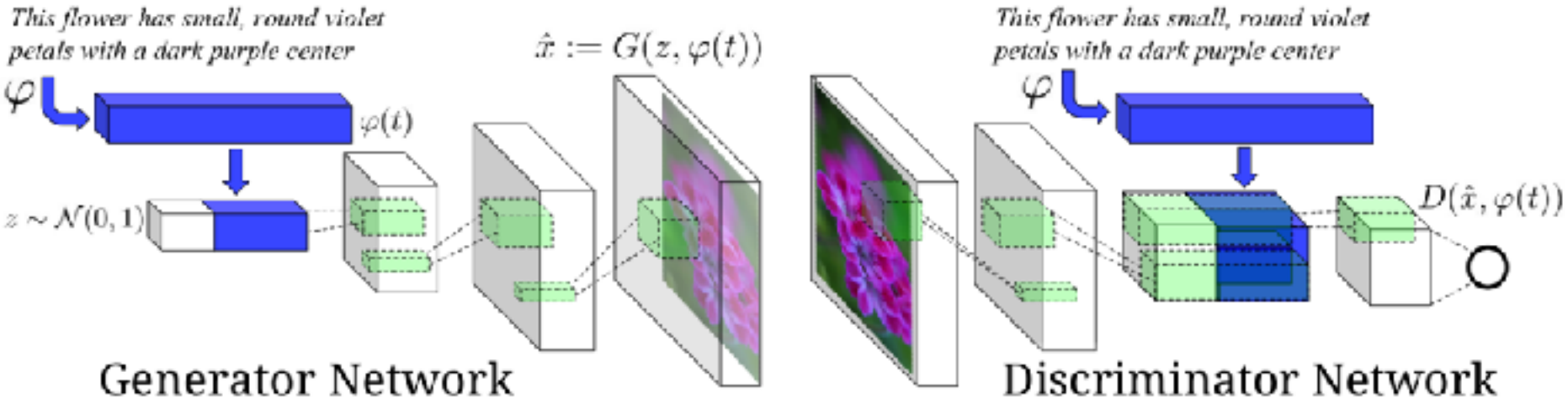


Figure 2. Our text-conditional convolutional GAN architecture. Text encoding $\varphi(t)$ is used by both generator and discriminator. It is projected to a lower-dimensions and depth concatenated with image feature maps for further stages of convolutional processing.

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

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

Results Not Satisfying

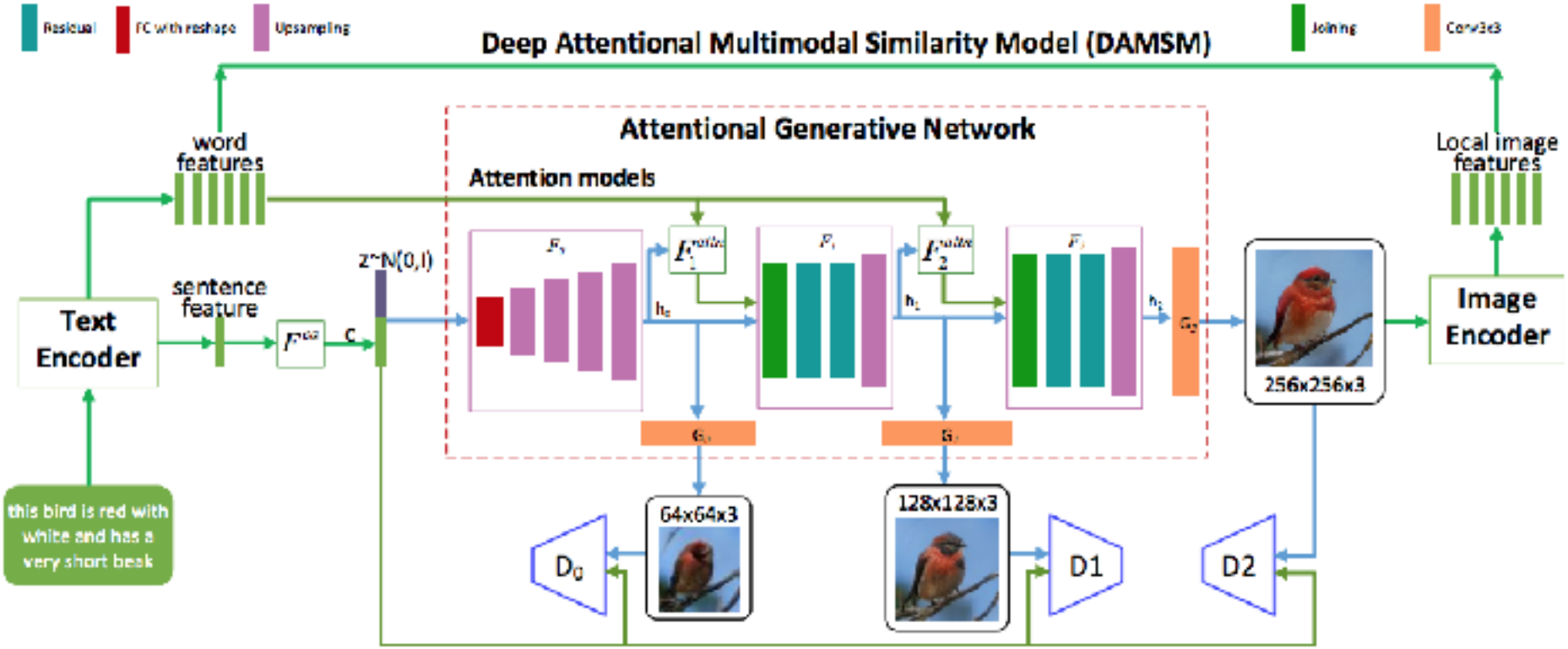
- So we need more processing!
- Add attention!

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



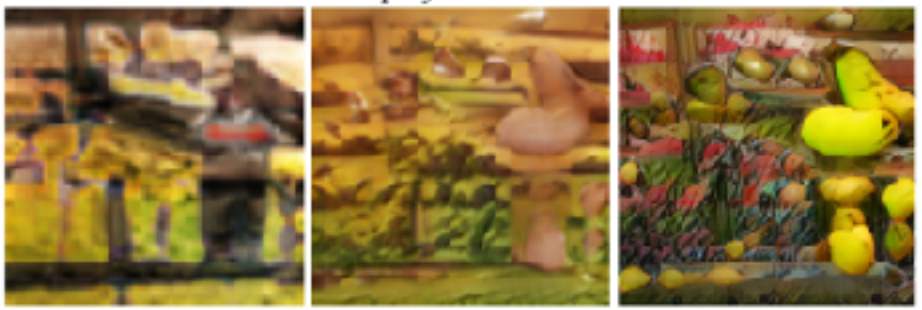
Results are very satisfying!

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

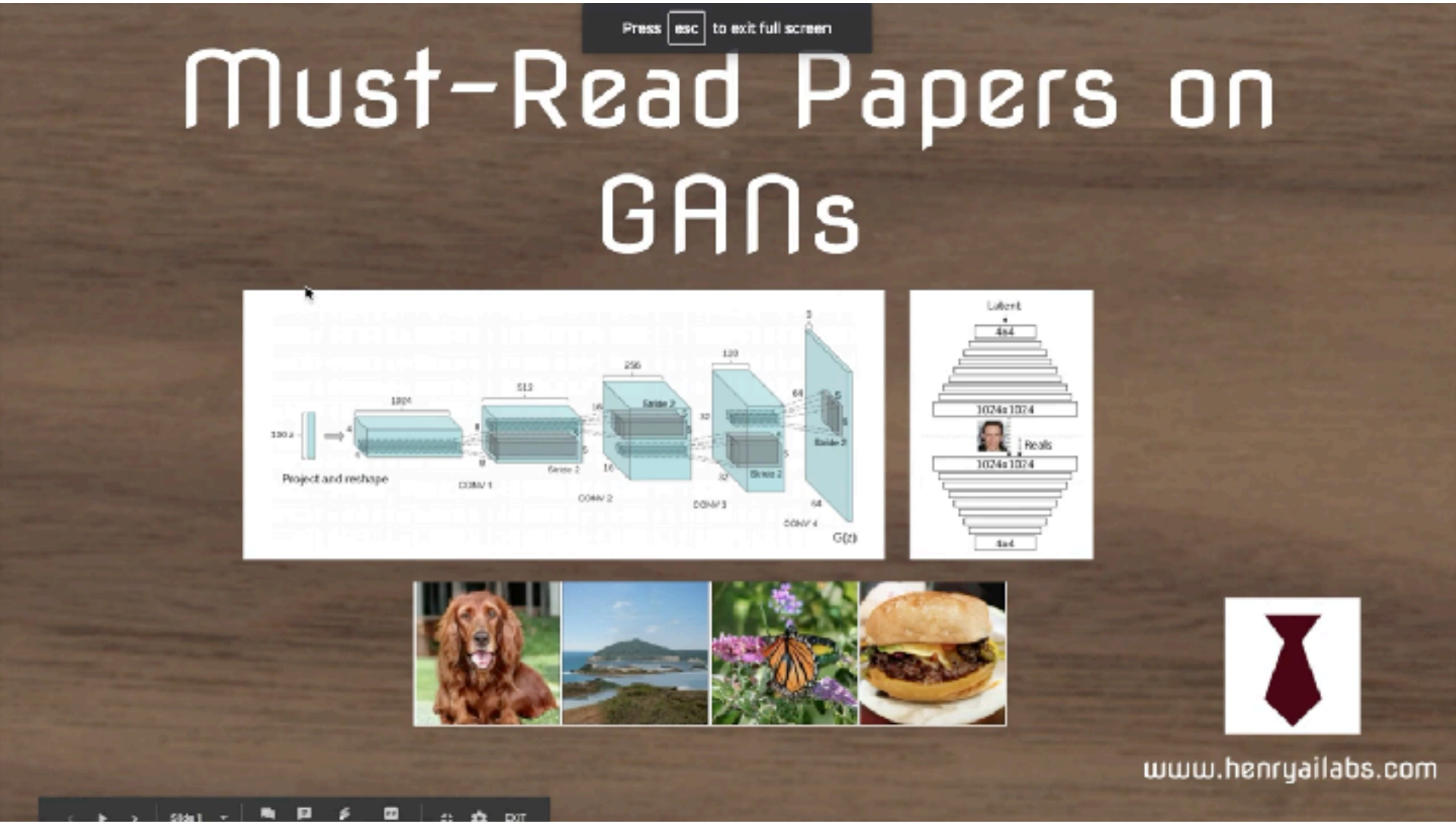
BigGAN

- Insert BigGAN Explanation here

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

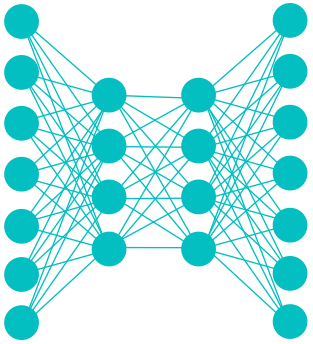
Lecture Notes for

Neural Networks and Machine Learning

Holy GAN-Zooks

Next Time:
Lapan Ch. 1

Reading: None



Backup slides

Title Between Topics

Example Slide

Title

Subtitle

`Follow Along: Notebook Name`