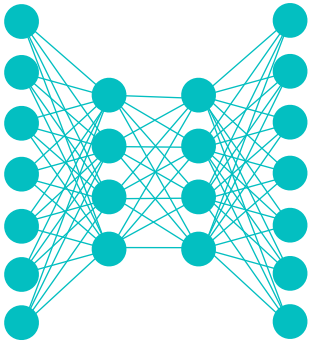
Lecture Notes for

Neural Networks and Machine Learning

Generative Adversarial Networks Overview



Logistics and Agenda

- Logistics
  - None
- Agenda
  - VAE Demo (last time)
  - AAE
  - Simple Generative Adversarial Networks
  - GANs Demo

Last Time: VAEs

```
# 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)
```

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

```
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)
```

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

$$-\sum_{z} 1 + \widehat{\Sigma(x^{(i)})} - \mu(x^{(i)})^2 - \exp(\widehat{\Sigma(x^{(i)})})$$

Note:
Flipped from maximization to minimization

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

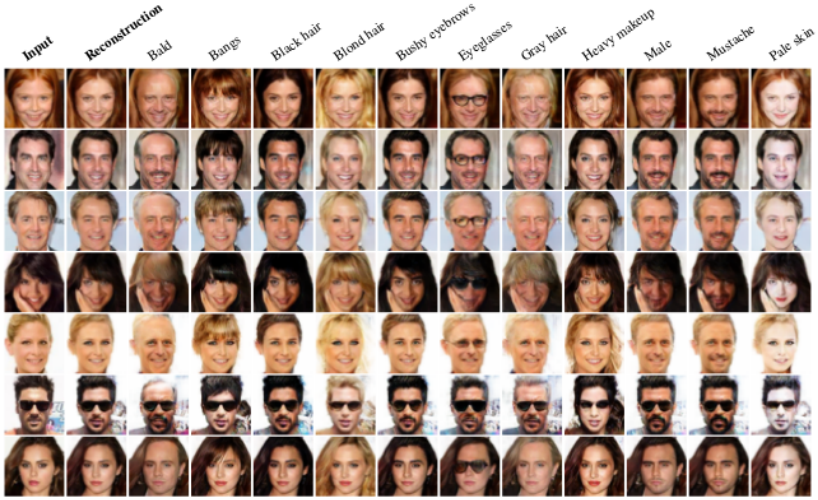
VAEs in Keras

Sampling from variational auto encoder using MNIST

Demo by Francois Chollet

In Master Repo: 07a VAEs in Keras.ipynb

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



Adversarial Auto Encoding

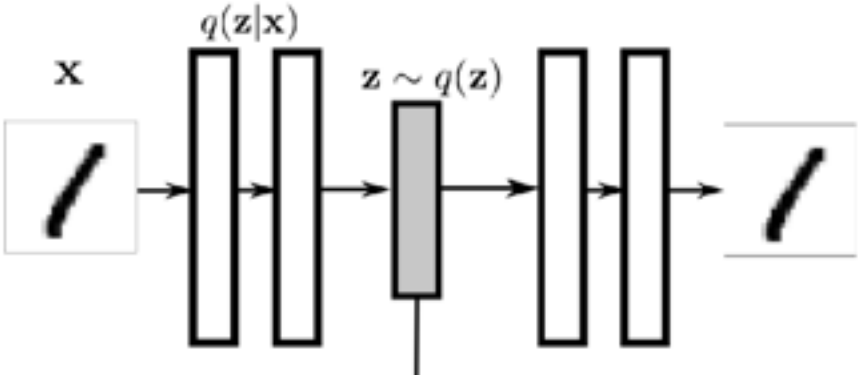
Do we need something more than VAE?

- Arguments for Yes:
  - ELBO is not global optimum! But… provides theory
  - Assumption of Normal distributions to q(z) is limiting
  - Training tends to be slower (…*so do GANs…*)
  - Manifold of distributions do not cover the latent space completely (not guaranteed)
  - 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 a pair of networks?

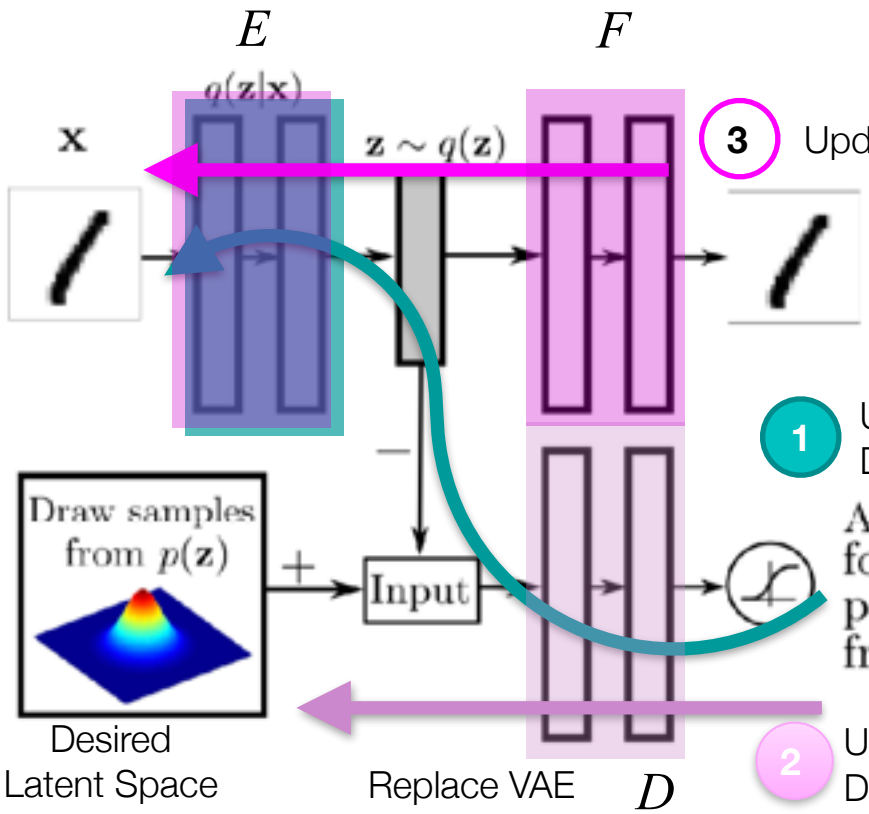
Encoder-Decoder Network



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

The Main Idea

- How can we enforce constraints on the latent space with a pair of networks?



$$\mathscr{L}^{E,F}_{error} = BCE(x, F \circ E(x))$$

3 Update Encoder and Decoder Pair

Impose Reconstruction Loss

$$\mathscr{L}^{E}_{error} = BCE(1, D \circ E(x))$$

1 Update Encoder to fool Discriminator
Do **not** update Discriminator

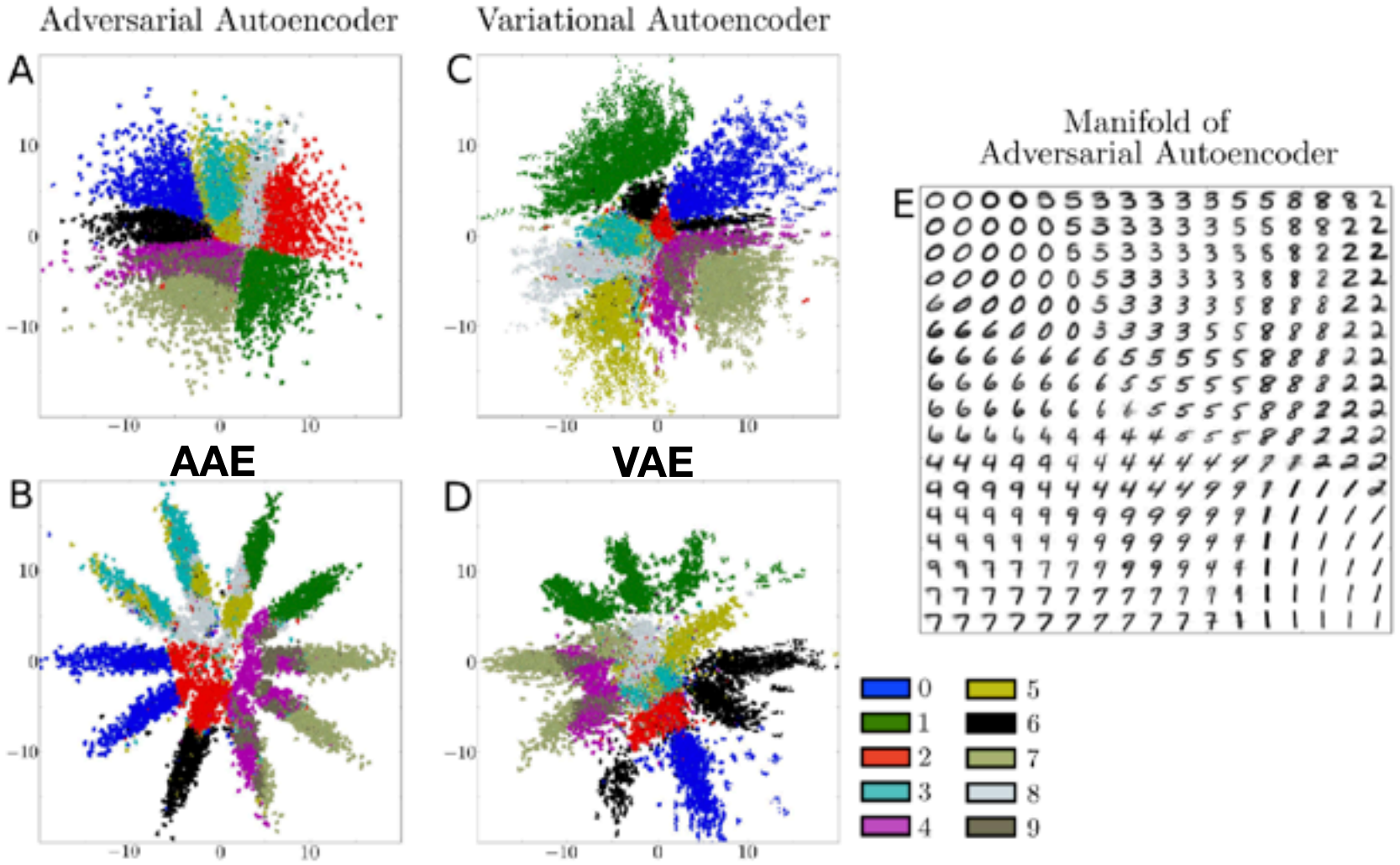
Adversarial cost for distinguishing positive samples $p(\mathbf{z})$ from negative samples $q(\mathbf{z})$

2 Update Discriminator to Detect Fake z
Do **not** update Encoder

$$\mathscr{L}^{D}_{error} = BCE(0, D \circ E(x)) + BCE(1, D(z))$$

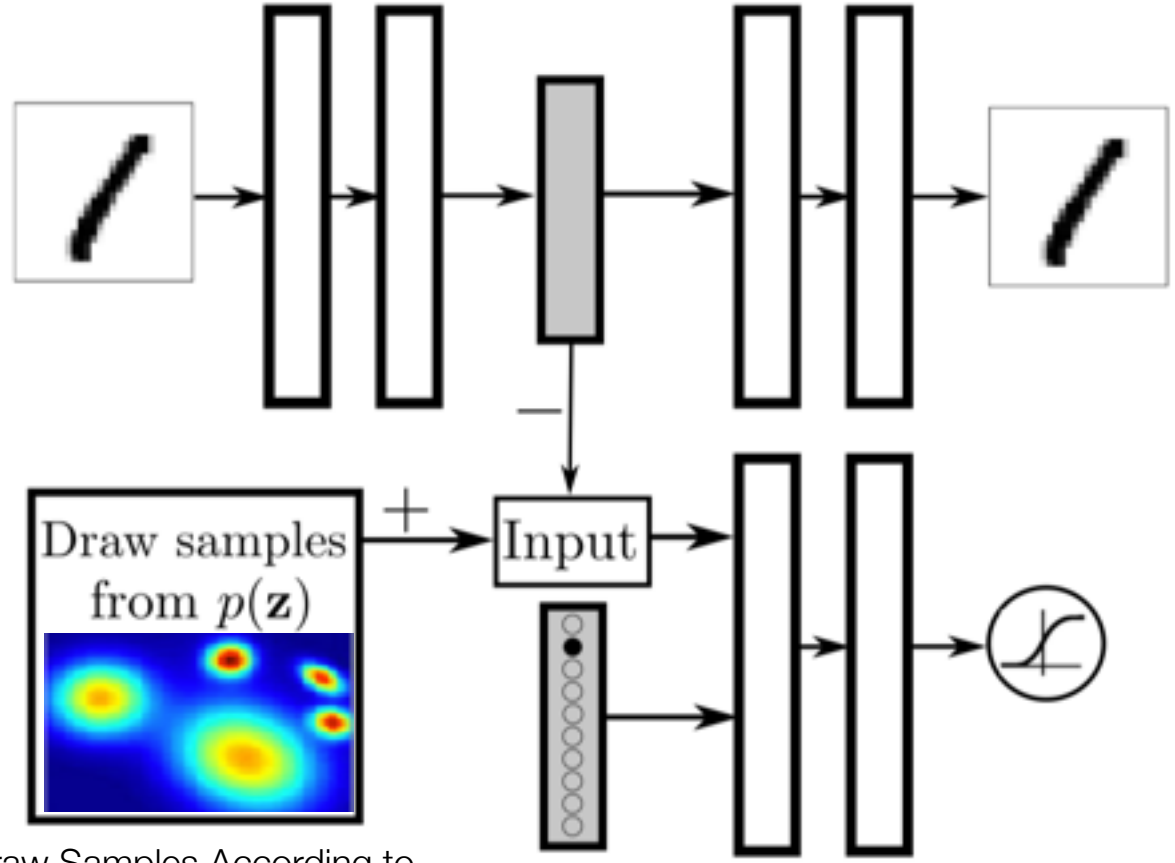
Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. "Adversarial autoencoders." arXiv preprint arXiv:1511.05644 (2015).

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

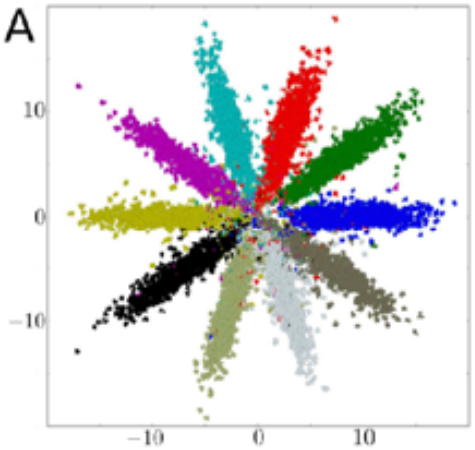


Draw Samples According to
Specific Class Prior Distributions

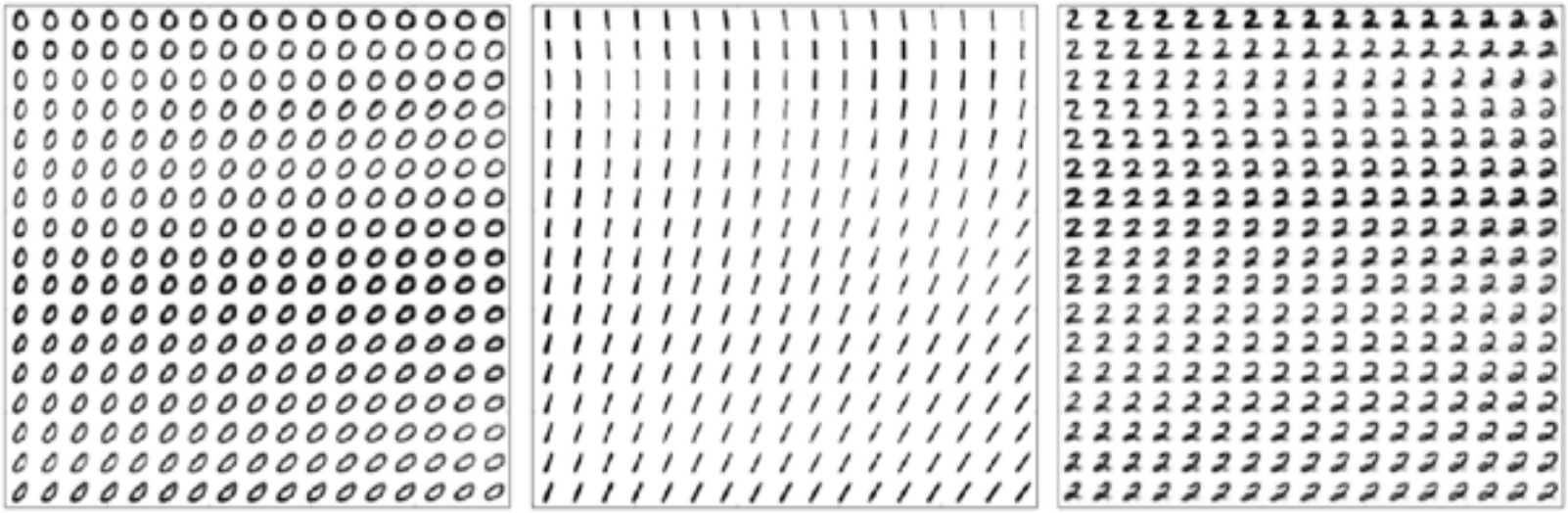
Provide Class Chosen
as One Hot Vector in Discriminator

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

Conditional Class Latent Spaces

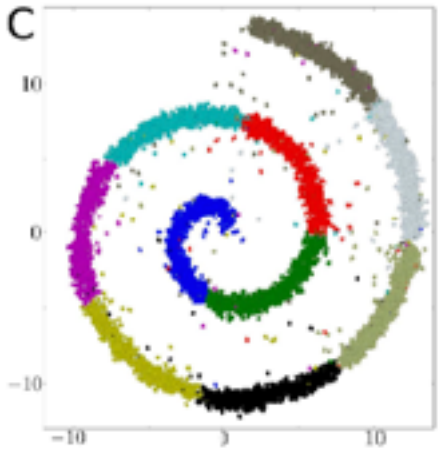


Sample Along Main Axis of the Gaussian Component for Each Digit

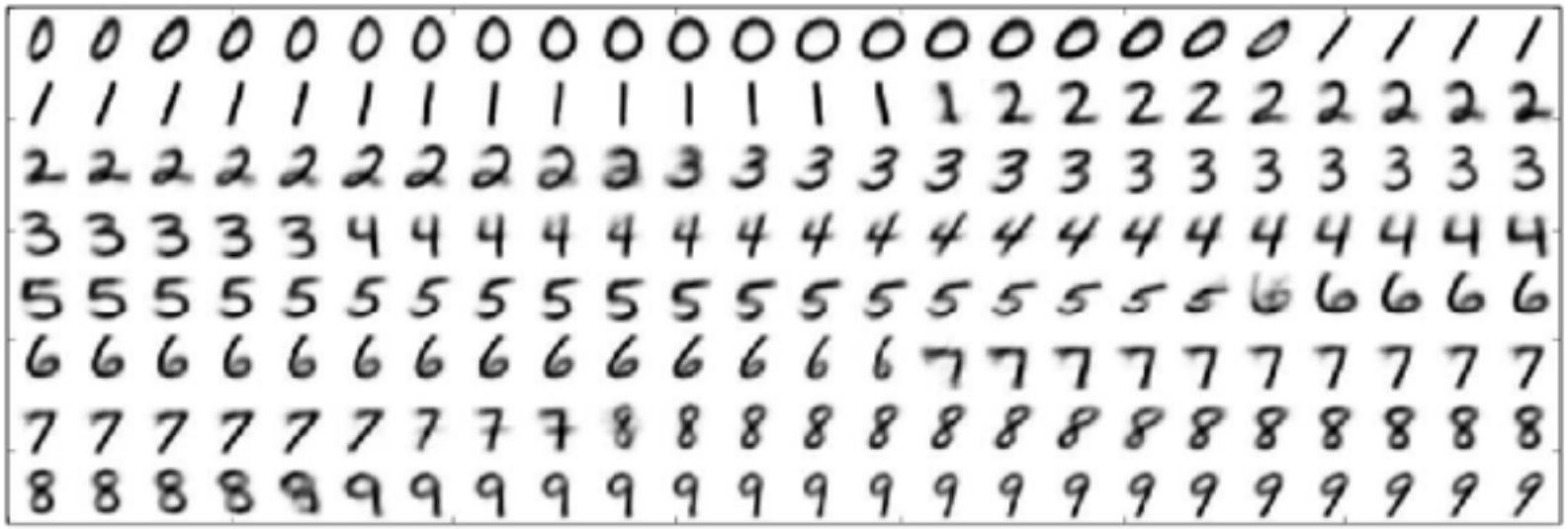


Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, **Ian Goodfellow**, and Brendan Frey. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Conditional Class Latent Spaces

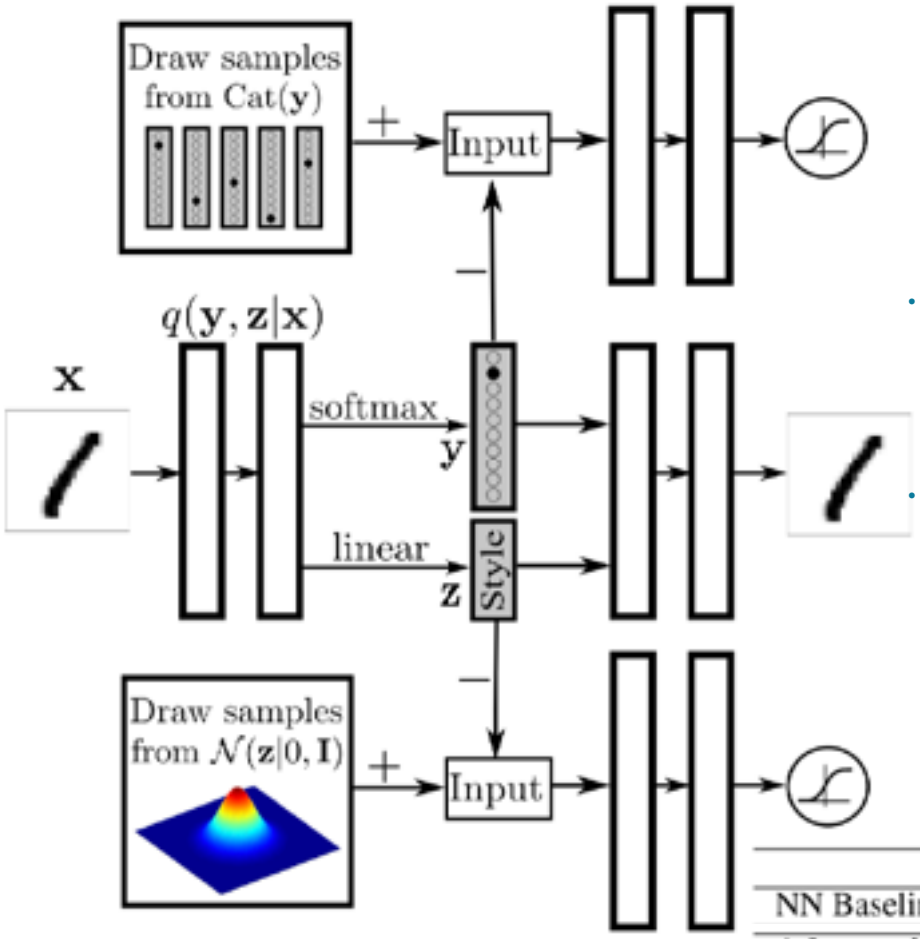


Sample Along Swiss Roll Axis



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

Semi-Supervised Classification



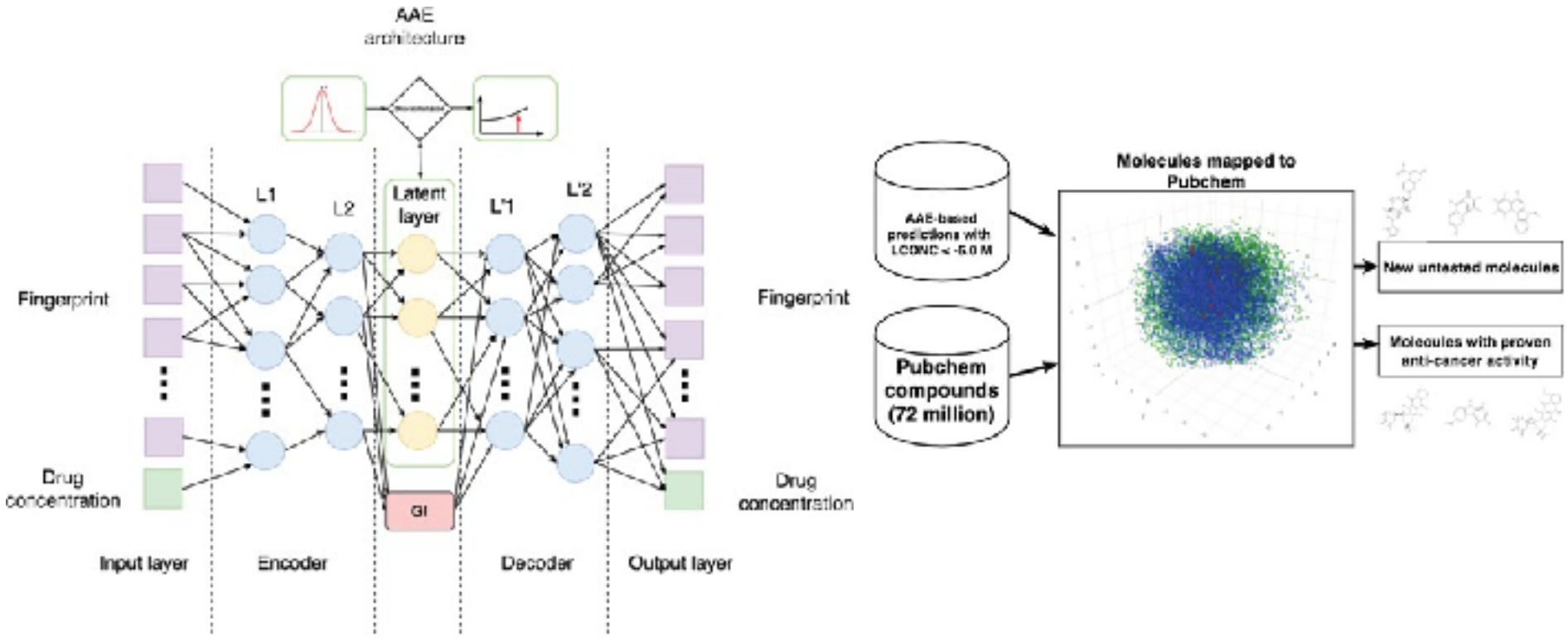
- Top Discriminator
  - Draw samples from one hot encoded labels, Cat(y)
  - Ensures created vector is categorical, softmax(z_{prev}) → "one hot"
- Bottom Discriminator
  - Same as previous
  - Constrains latent representation
- Supervised Training (disentangled?)
  - Update AAE networks with a few batches
  - Use small labeled mini-batches to update $q(y|x)$ with binary cross entropy
  - Repeat

	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).

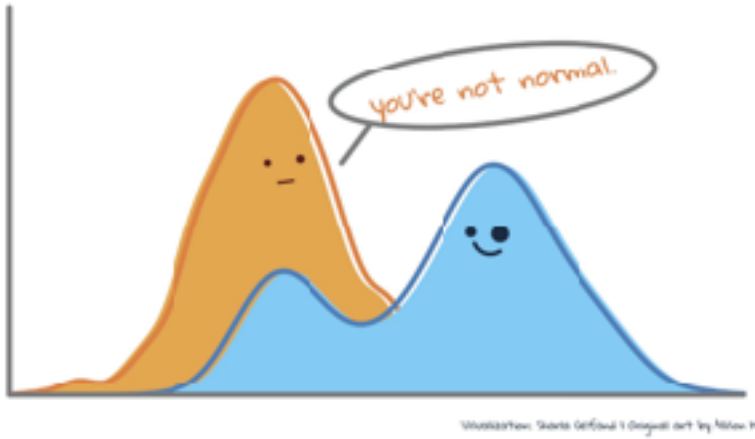
Other Uses For AAE

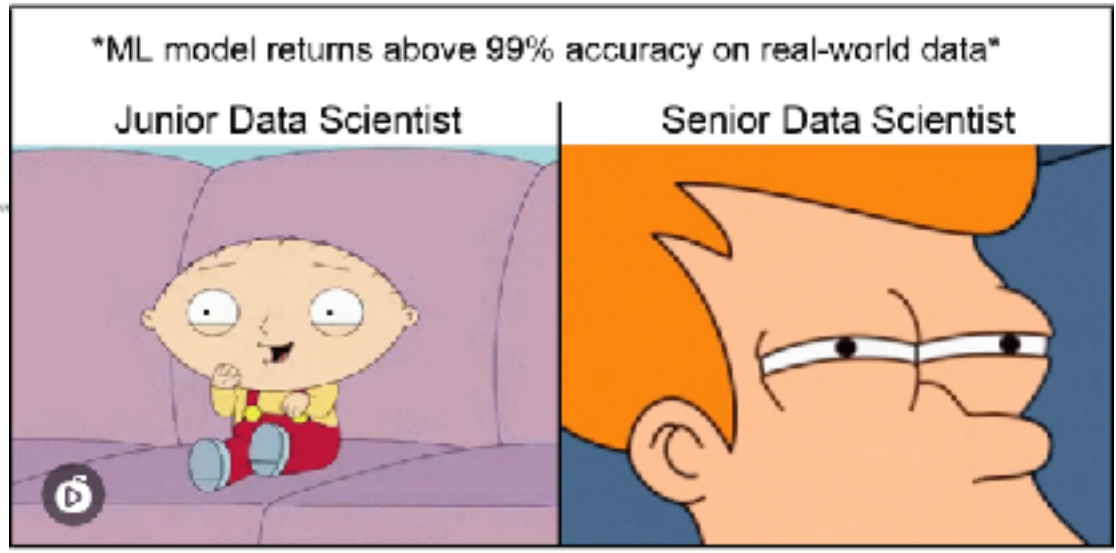
- Molecular Fingerprinting



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.

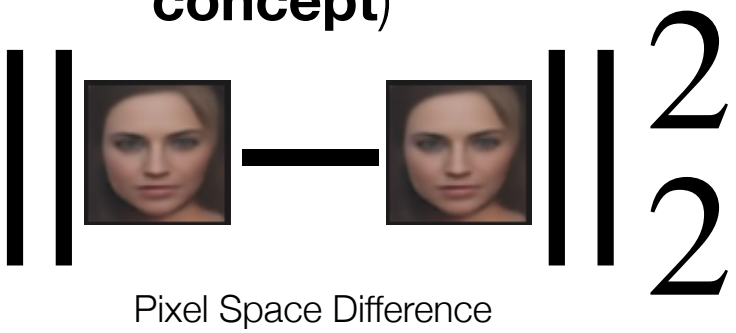
Adversarial Latent Auto-Encoders



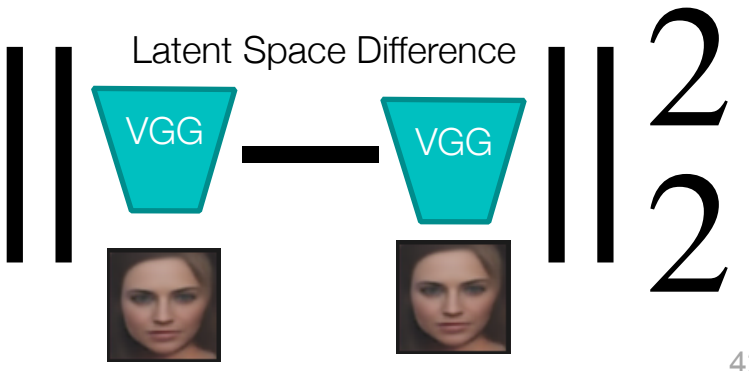


Drawbacks of AAEs

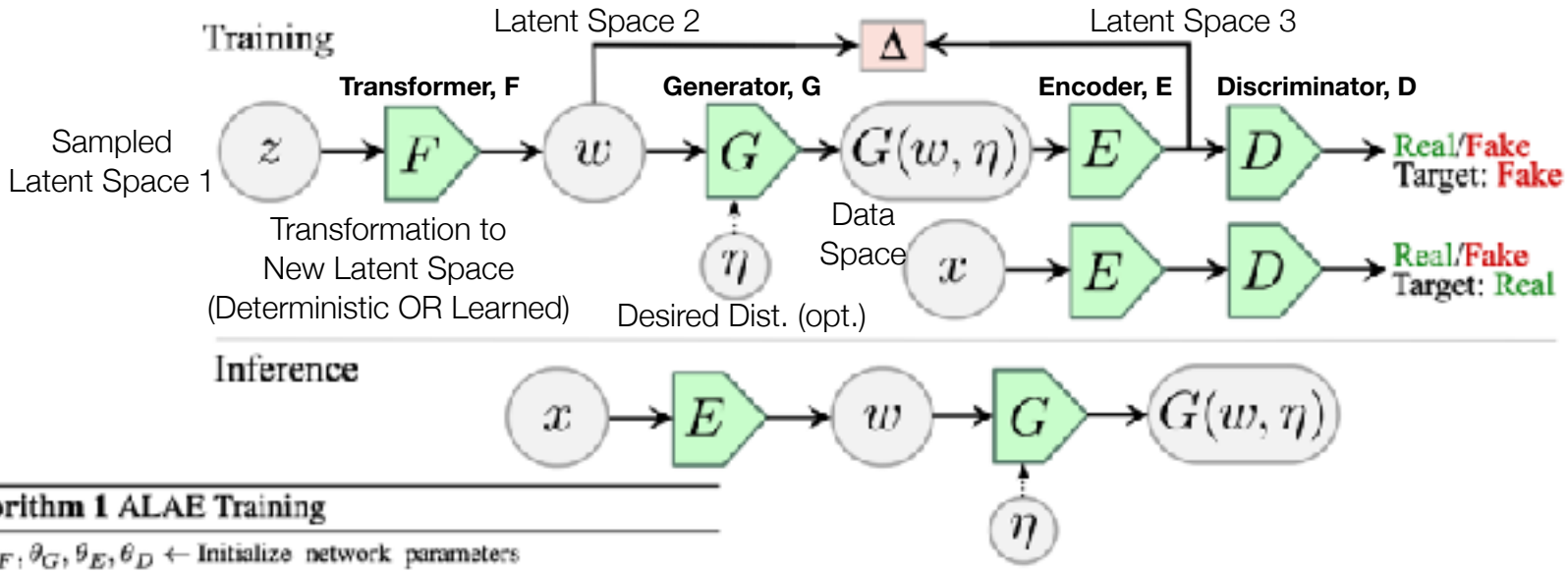
- AAEs capture representational properties of an encoder-decoder pair, with latent space following a specified structure
- Drawbacks:
  - Not entirely generative because we need examples
  - Not guaranteed to create **disentangled** representations (*VAE is better for this*…)
  - Requires that latent space distribution be selected apriori
  - Reconstruction loss calculated in data space, which may not be informative for optimizing latent space (**important concept**)



Pixel Space Difference



Adversarial Latent Auto-Encoders, ALAE



Algorithm 1 ALAE Training

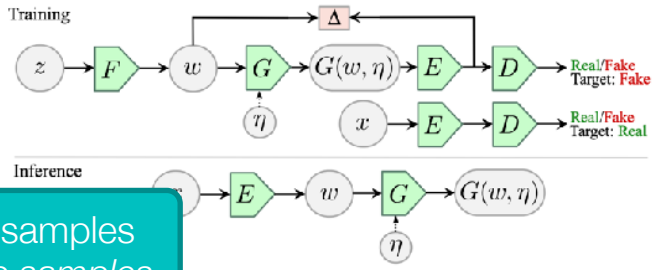
1: $\theta_F, \theta_G, \theta_E, \theta_D \leftarrow$ Initialize network parameters
2: **while** not converged **do**
3: Step I. Update E, and D
4: $x \leftarrow$ Random mini-batch from dataset
5: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
6: $L_{adv}^{E,D} \leftarrow \text{softplus}(D \circ E \circ G \circ F(z))) + \text{softplus}(-D \circ E(x)) + \frac{\gamma}{2} \mathbf{E}_{p_{\mathcal{D}}(x)} \left[\|\nabla D \circ E(x)\|^2 \right]$
7: $\theta_E, \theta_D \leftarrow \text{ADAM}(\nabla_{\theta_D, \theta_E} L_{adv}^{E,D}, \theta_D, \theta_E, \alpha, \beta_1, \beta_2)$
8: Step II. Update F, and G
9: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
10: $L_{adv}^{F,G} \leftarrow \text{softplus}(-D \circ E \circ G \circ F(z)))$
11: $\theta_F, \theta_G \leftarrow \text{ADAM}(\nabla_{\theta_F, \theta_G} L_{adv}^{F,G}, \theta_F, \theta_G, \alpha, \beta_1, \beta_2)$
12: Step III. Update E, and G
13: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
14: $L_{error}^{E,G} \leftarrow \|F(z) - E \circ G \circ F(z)\|_2^2$
15: $\theta_E, \theta_G \leftarrow \text{ADAM}(\nabla_{\theta_E, \theta_G} L_{error}^{E,G}, \theta_E, \theta_G, \alpha, \beta_1, \beta_2)$
16: **end while**

- Train G(F) to fool discriminator
- Minimize difference F and E(G(F))
- Reconstruction loss: x and E(D(x)), *never actually calculated*, just inferred
- Therefore, w~E(G(w)) and x~G(E(x))

Pidhorskyi, Stanislav, Donald A. Adjeroh, and Gianfranco Doretto. "Adversarial latent autoencoders." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14104-14113. 2020.

Adversarial Latent Auto-Encoders, ALAE

D ->1 = Real data, D-> 0 = Sampled Data (Fake)



Algorithm 1 ALAE Training

1: $\theta_F, \theta_G, \theta_E, \theta_D \leftarrow$ Initialize network parameters
2: **while** not converged **do**
3:
4:
5:
6: $L_{adv}^{E,D} \leftarrow \text{softplus}(D \circ E \circ G \circ F(z)) + \text{softplus}(-D \circ E(x)) + \frac{\gamma}{2} \mathrm{E}_{p_{\mathcal{D}}(x)} \left[\|\nabla D \circ E(x)\|^2 \right]$
7: $\theta_E, \theta_D \leftarrow \text{ADAM}(\nabla_{\theta_D, \theta_E} L_{adv}^{E,D}, \theta_D, \theta_E, \alpha, \beta_1, \beta_2)$
8: Step II. Update F, and G
9: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
10: $L_{a}^{F,G}$
11: θ_F , $\beta_1, \beta_2)$
12: Ste
13: $z \leftarrow$
14: $L_{error}^{E,G} \leftarrow \|F(z) - E \circ G \circ F(z)\|_2^2$
15: $\theta_E, \theta_G \leftarrow \text{ADAM}(\nabla_{\theta_E, \theta_G} L_{error}^{E,G}, \theta_E, \theta_G, \alpha, \beta_1, \beta_2)$
16: **end while**

Related to Nash Equilibrium, but shifted…

Based On Wasserstein Distance, … which is really helpful…

E,D: Detect fake samples
minimize D for fake samples

E,D: Detect real samples
minimize -D for real samples

E,D: Gradient Penalty
Keep Gradient Magnitude Small
Keep in mind for later

F,G: Try to fool discriminator,
minimize -D for fake samples

E,G: Keep latent spaces similar

Pidhorskyi, Stanislav, Donald A. Adjeroh, and Gianfranco Doretto. "Adversarial latent autoencoders." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14104-14113. 2020.

Do we really need to learn this?

Reconstructions

Generated Images (Fake)

Pidhorskyi, Stanislav, Donald A. Adjeroh, and Gianfranco Doretto. "Adversarial latent autoencoders." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14104-14113. 2020.

Yep, We need to study GANs

- The ALAE used some notation and processes that we need to study in order to understand:
  - why these are advantageous
  - how to do them properly
  - the tradeoffs and computational cost
- Therefore, the remaining topics:
  - Nash Equilibrium (Vanilla GAN)
  - GAN Training Tricks
  - Least Squares GAN
  - Wasserstein GAN
  - Other Tips and Tricks leading to BigGAN, StyleGAN
  - And Others!

The Simple GAN

Geoff Hinton after writing the paper on backprop in 1986

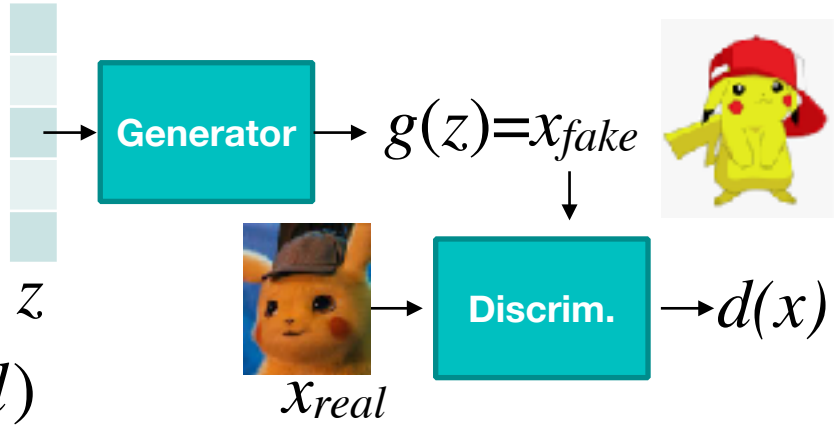
Latent Variable Approximation with GANs

- **Idea**: transform random input data into target of interest
  - Like an image!
- Rather than training an encoder, we need only a transformation algorithm (generator)
  - Transformer will be a…
  - Neural Network (of course!)
- Transformer is equivalent to "complex" sampling method
- How to optimize and provide training examples?
  - Use two Neural Networks!
  - … Deep Learning researchers are like the chiropractors of the machine learning community…

Generative Adversarial Network

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

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



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

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

only portion dependent on g
generator minimizes

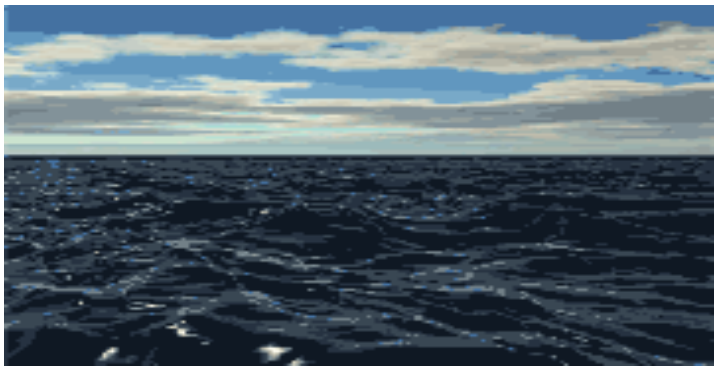
everything dependent on d
discriminator maximizes

Taking turns: Nash Equilibrium

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

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

- If only the problem was convex! This would be easy to solve…
- But $v(g,d)$ is not convex
  - Saddle points everywhere
  - Like optimizing in an Ocean



- Taking turns on the gradient descent may never converge (Nash equilibrium is not convergence)
  - **So we have no convergence guarantee**
  - But practically we like when discriminator == chance

Practical Objectives

- Discriminator objective, gradient **ascent**:

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

- 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
- Ian 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:

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

- 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 it is wrong?
    - not trying to win, just make the other person lose

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

Now we are not guaranteed convergence or equilibrium when training the GAN…
But it trains…

Final Loss Functions

- Discriminator:

Freeze Generator Weights

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

Same as minimizing the binary cross entropy of the model with: (1) real data=1 and (2) generated data=0!

- Generator:

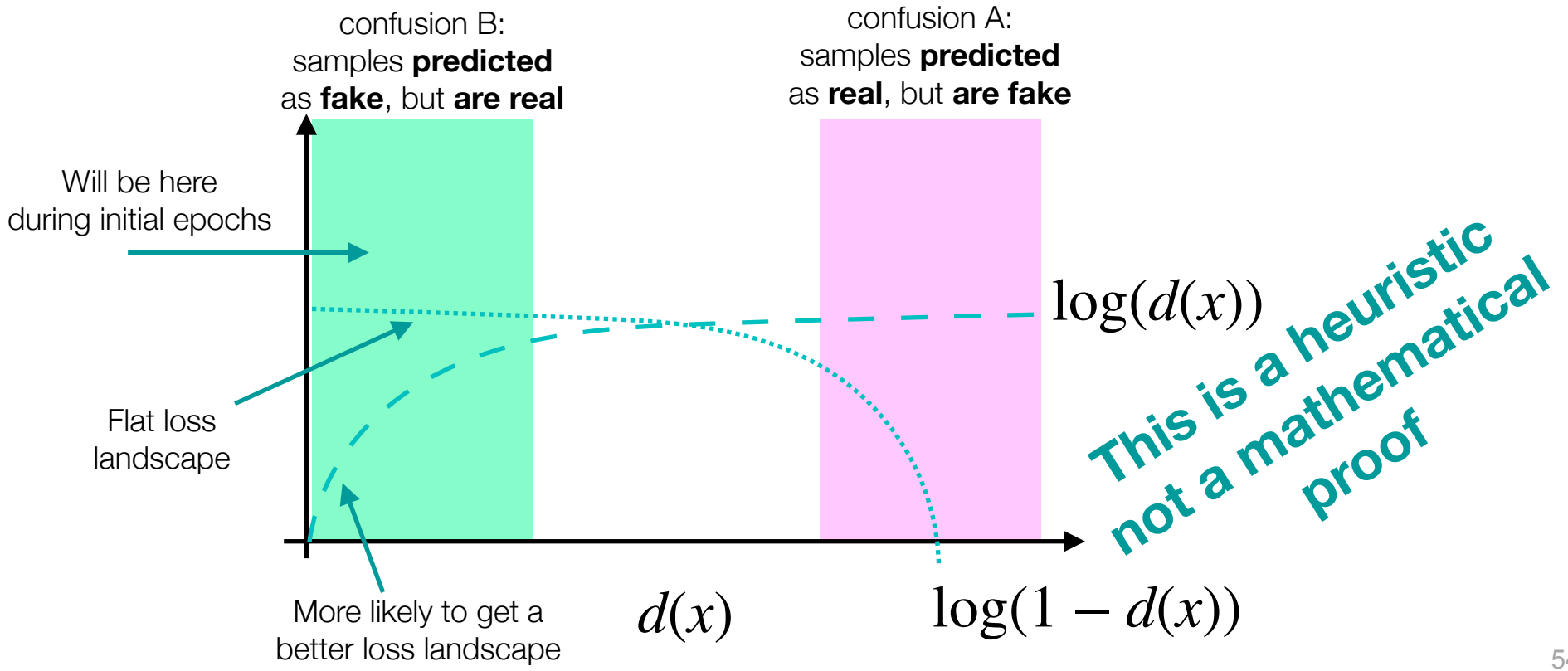
$$\max_{g} \mathbf{E}_{x \leftarrow g(z)} \left[\log(d(x_{fake})) \right]$$

Freeze Discriminator Weights

Same as minimizing the binary cross entropy of "mislabeled" generated data=1!

Intuition for why this really helps…

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



Lecture Notes for

Neural Networks and Machine Learning

GANs

Next Time:
Practical GANs

Reading: Chollet CH8