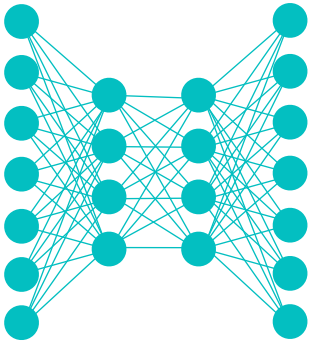
Lecture Notes for

Neural Networks and Machine Learning

Generative Adversarial Networks Overview



Logistics and Agenda

- Logistics
  - None
- Agenda
  - VAE Demo
  - AAE
  - Simple Generative Adversarial Networks

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

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

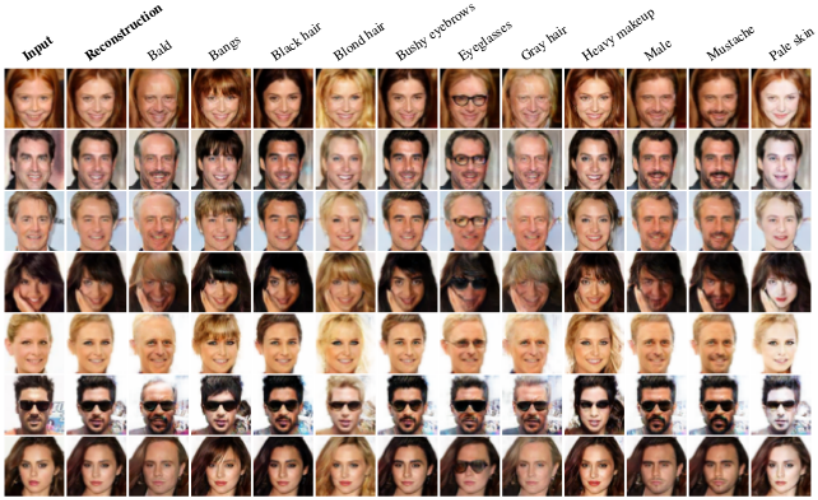
VAEs in Keras

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

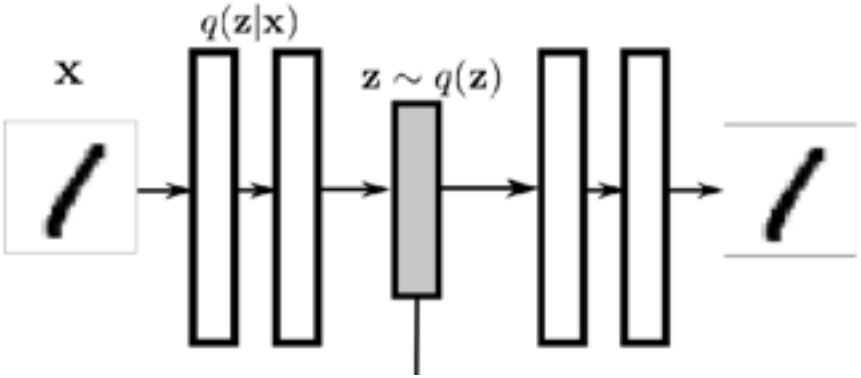
Do we need something more than VAE?

- Arguments for Yes:
  - ELBO is flawed!
  - Assumption of Normal distributions to q(z) is limiting
  - Training tends to be slow
  - Manifold of distributions does 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?

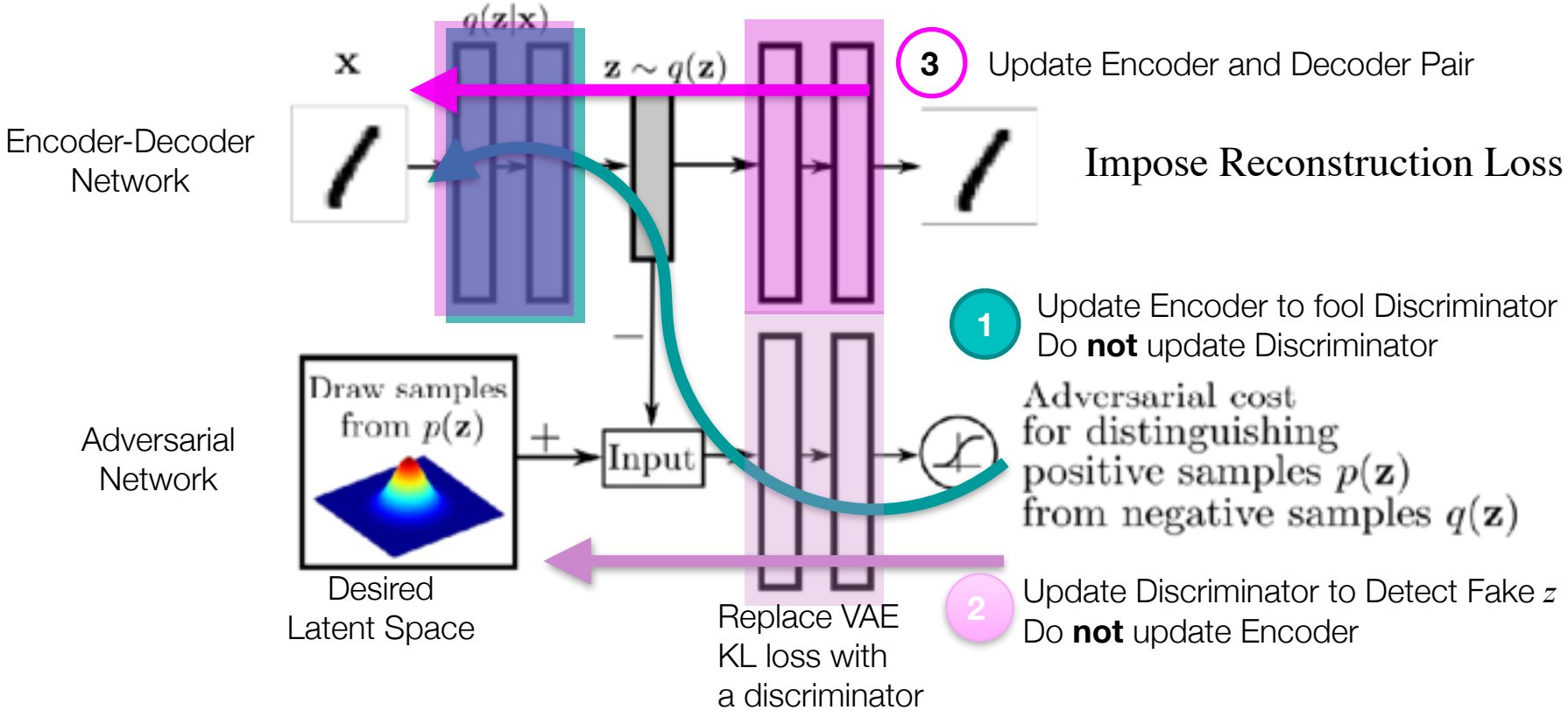
Encoder-Decoder Network



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

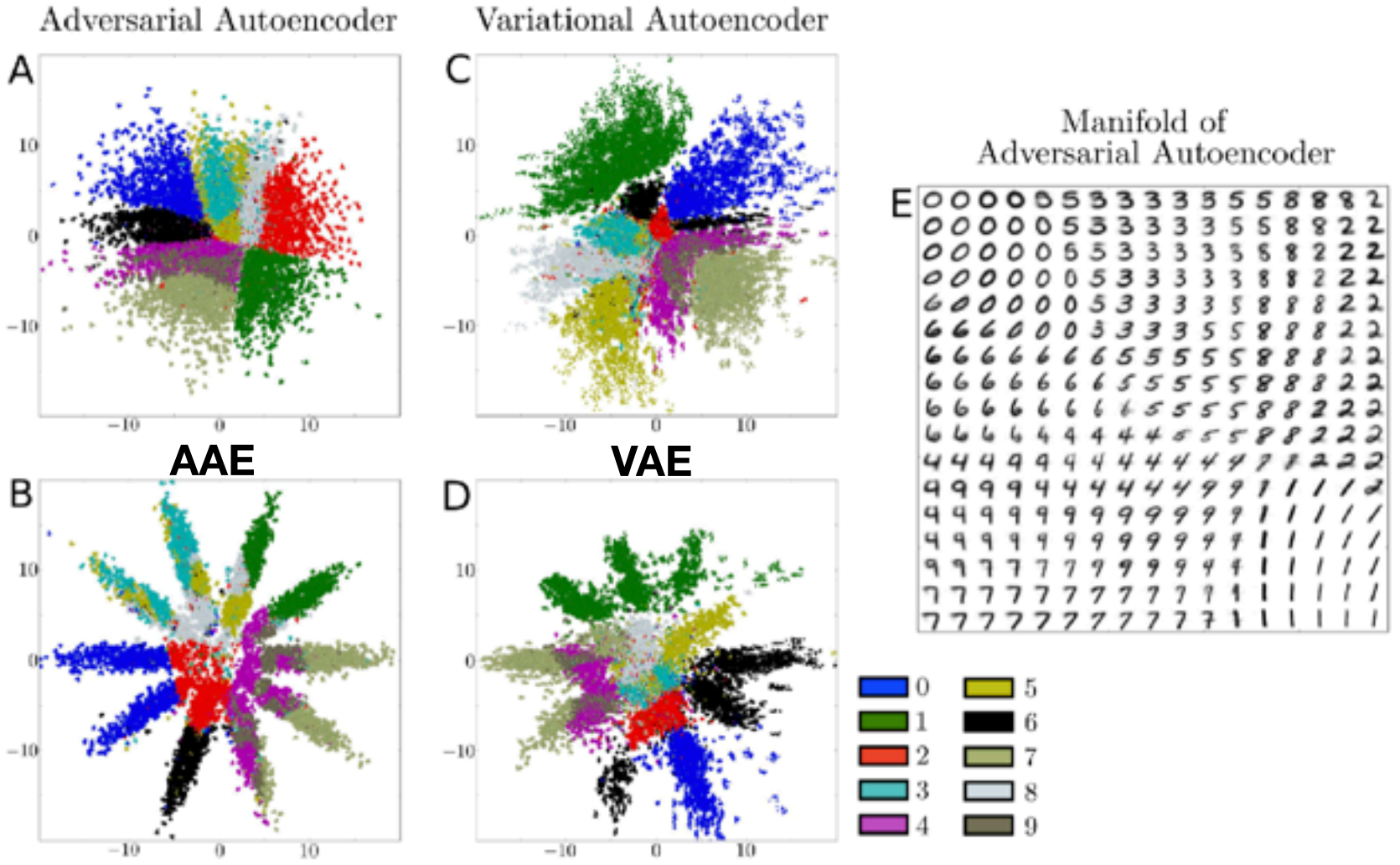
The Main Idea

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



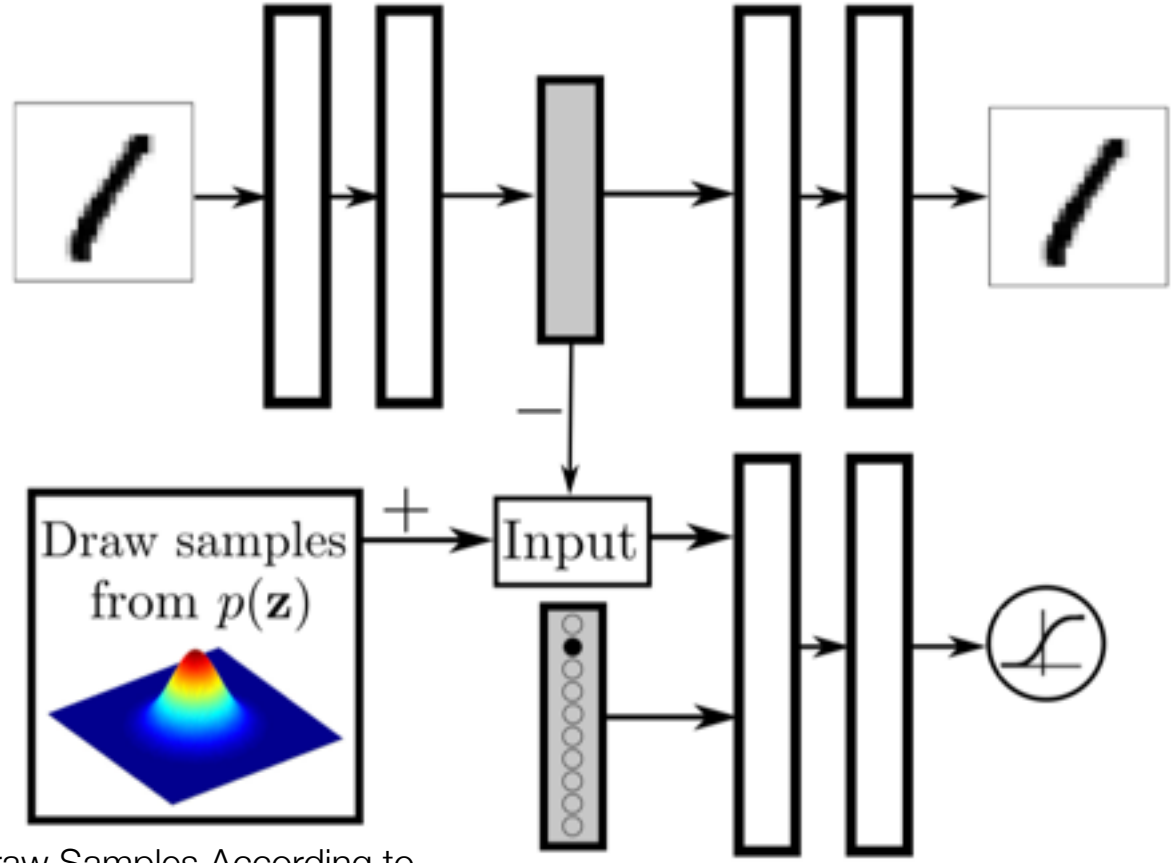
Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. "Adversarial autoencoders." arXiv preprint arXiv:1511.05644 (2015).

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

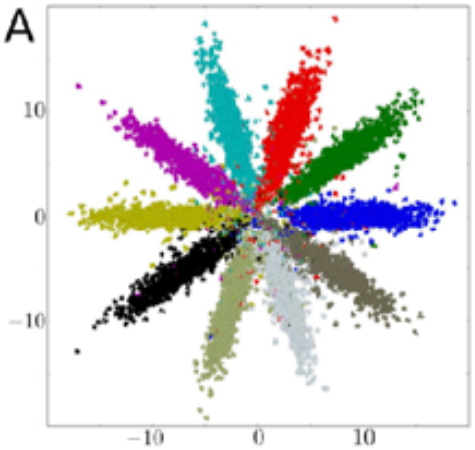


Draw Samples According to
Specific Class Prior Distributions

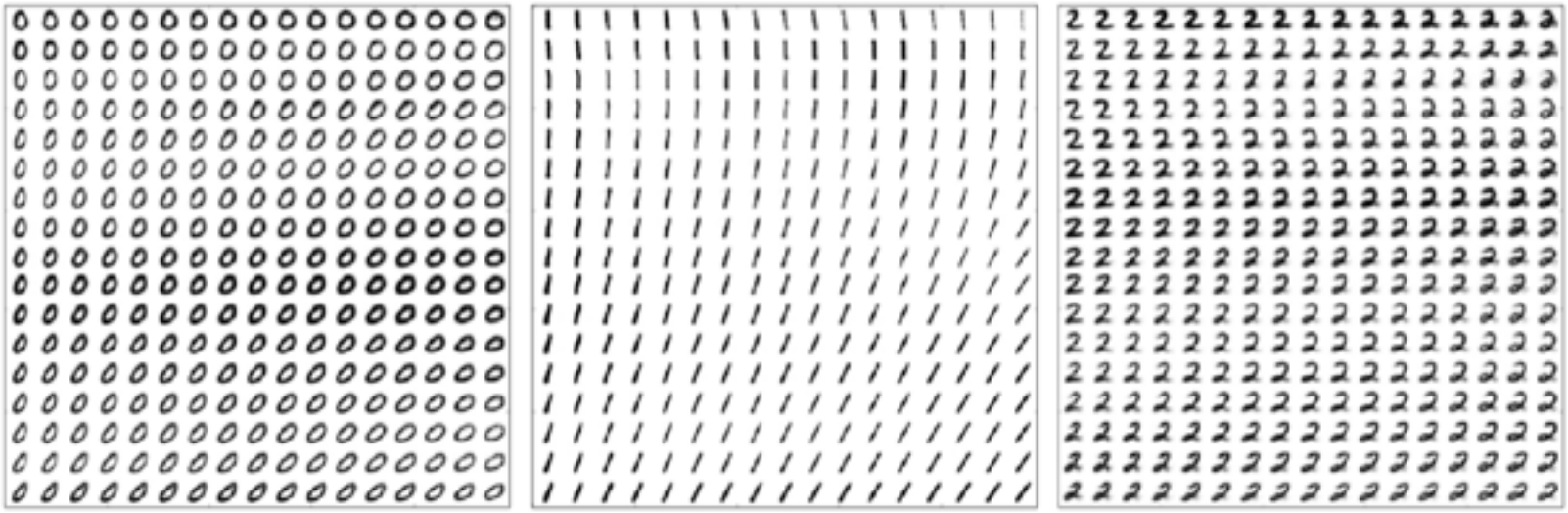
Provide Class Chosen
as One Hot Vector in Discriminator

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

Conditional Class Latent Spaces

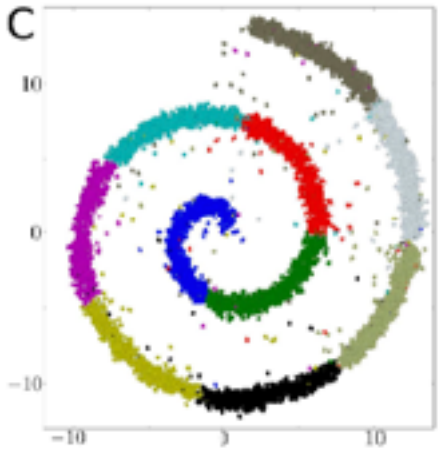


Sample Along Main Axis of the Gaussian Component for Each Digit

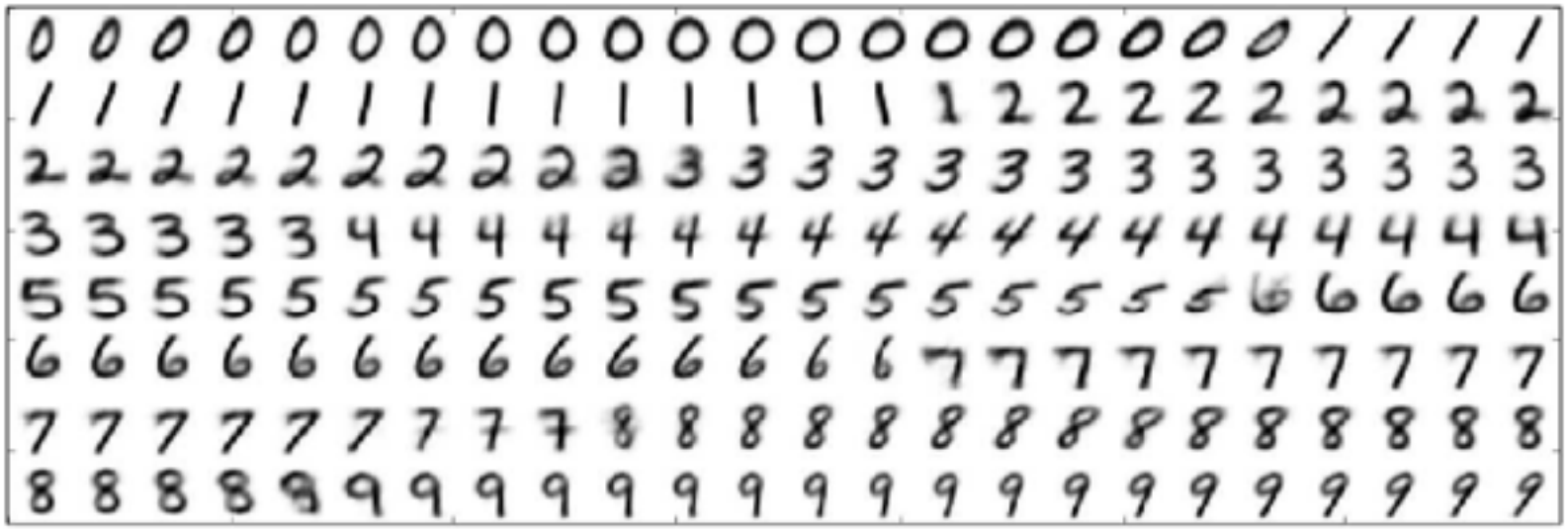


Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, **Ian Goodfellow**, and Brendan Frey. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Conditional Class Latent Spaces

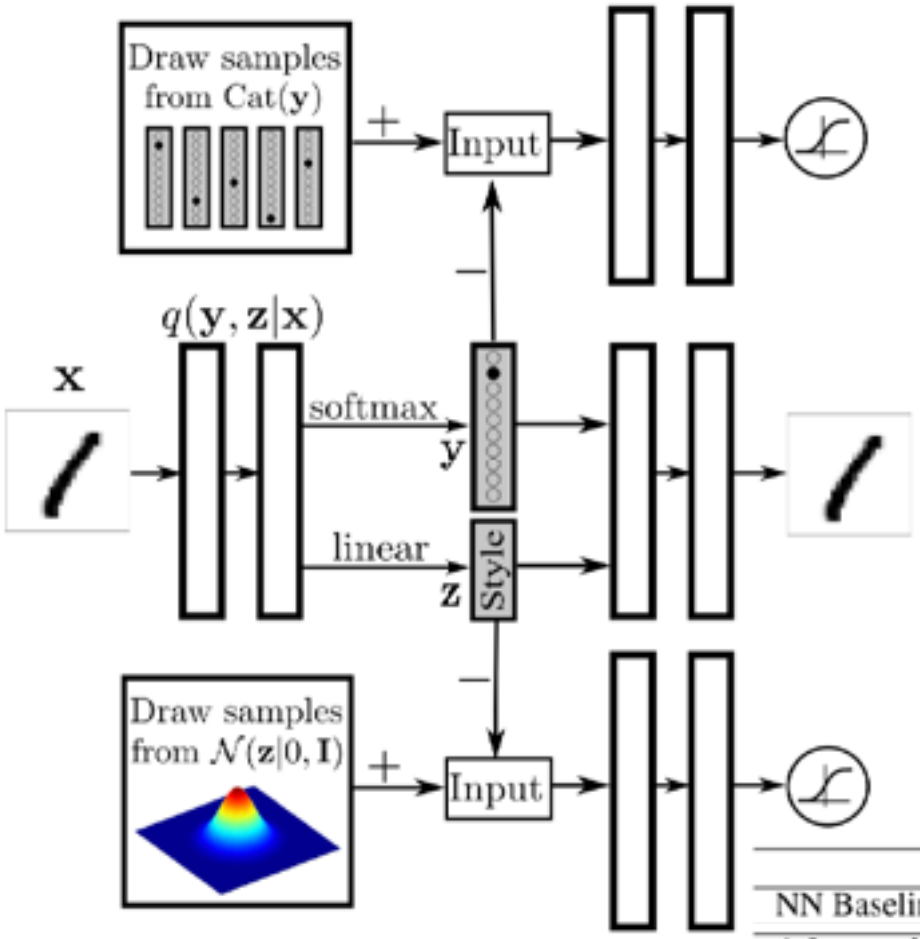


Sample Along Swiss Roll Axis



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

Semi-Supervised Classification



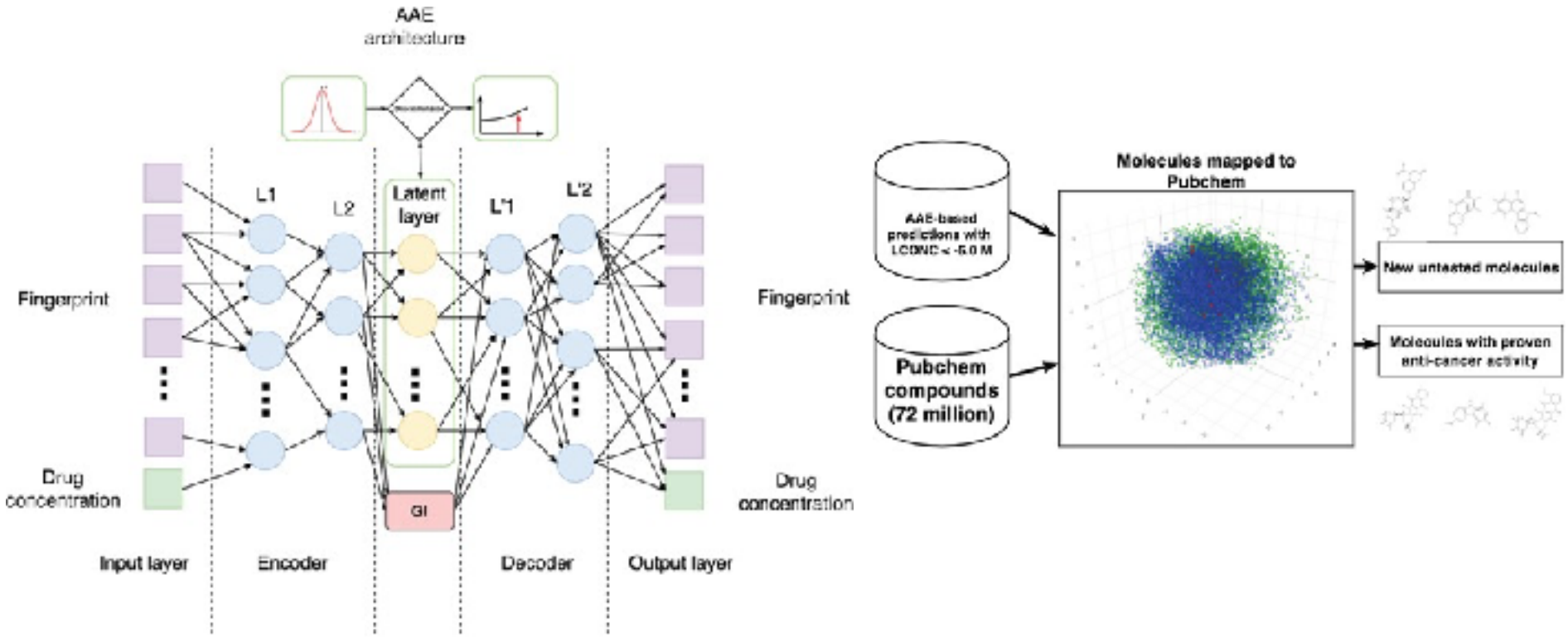
- Top Discriminator
  - Draw samples from one hot encoded labels
  - Ensures aggregated posterior matches class distributions
- Bottom Discriminator
  - Same as previous
  - Constrains latent representation
- Supervised Training
  - After GAN training:
  - Use small mini-batches on $q(y|z)$

	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

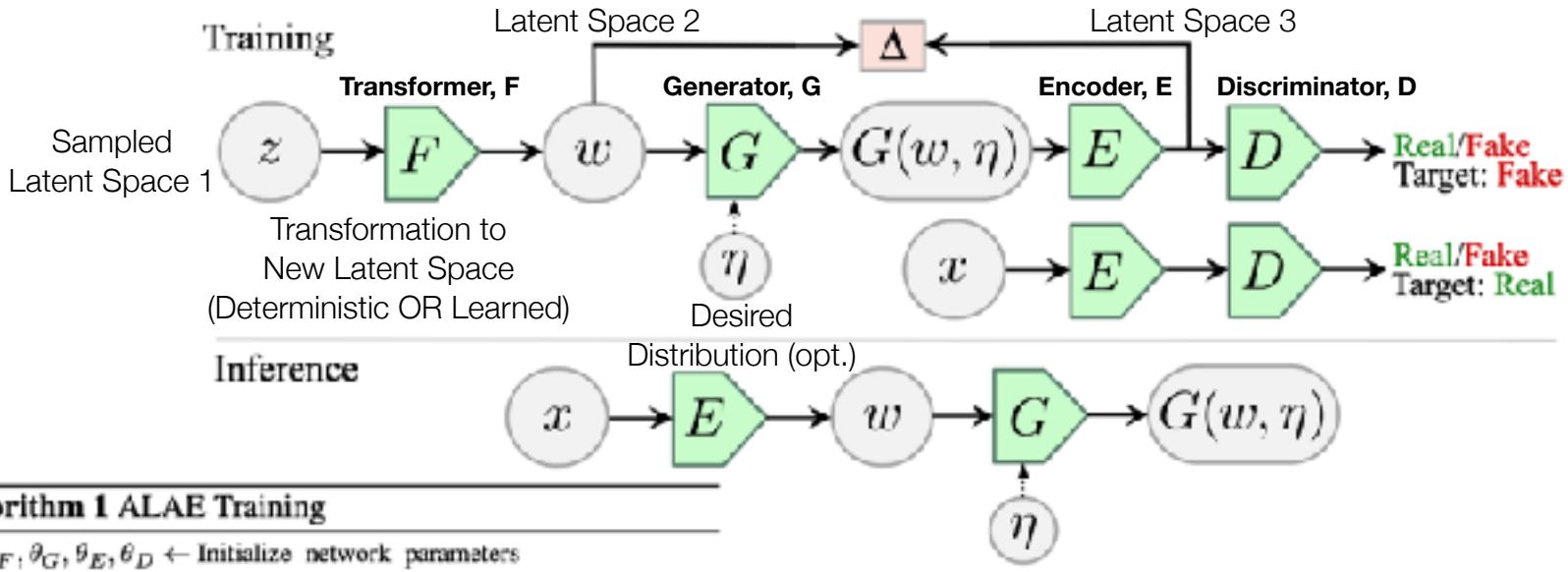


Kadurin, Artur, Alexander Aliper, Andrey Kazennov, Polina Mamoshina, Quentin Vanhaelen, Kuzma Khrabrov, and Alex Zhavoronkov. "The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology." *Oncotarget* 8, no. 7 (2017): 10883.

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
  - Requires that latent space distribution be known apriori
  - Reconstruction loss calculated in data space, which may not be informative for optimizing latent space

Adversarial Latent Auto-Encoders, ALAE



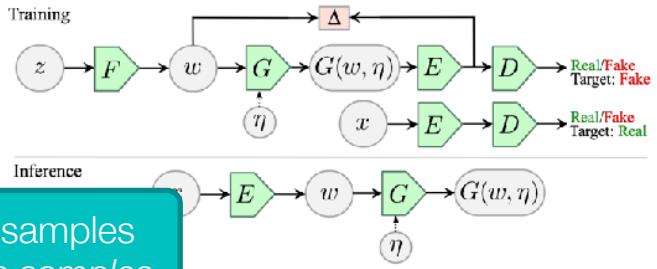
Algorithm 1 ALAE Training

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



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: $L^{E,D}_{adv} = \text{softplus}(D \circ E(x)) + \frac{\gamma}{2} \mathbb{E}_{p_D}$
7: $\theta_E, \theta_D \leftarrow \text{ADAM}(\nabla_{\theta_D, \theta_E} L^{E,D}_{adv}, \theta_D, \theta_E, \alpha, \beta_1, \beta_2)$
8: Step II. Update F, and G
9: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
10: $L^{F,G}_{adv} = \text{softplus}(-D \circ E \circ G \circ F(z))$
11: $\theta_F, \theta_G \leftarrow \text{ADAM}(\nabla_{\theta_F, \theta_G} L^{F,G}_{adv}, \theta_F, \theta_G, \alpha, \beta_1, \beta_2)$
12: Step III. Update E, and G
13: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
14: $L^{E,G}_{error} \leftarrow \|F(z) - E \circ G \circ F(z)\|_2^2$
15: $\theta_E, \theta_G \leftarrow \text{ADAM}(\nabla_{\theta_E, \theta_G} L^{E,G}_{error}, \theta_E, \theta_G, \alpha, \beta_1, \beta_2)$
16: **end while**

Based On Nash Equilibrium

Based On Wasserstein Distance

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.

Need to study GANs

- The previous slide setup some notation and processes that we need to study in order to:
  - Understand why these are advantageous
  - Understand how to do them properly
  - Understand the tradeoffs and computational cost
- Therefore, the Next Lectures:
  - Nash Equilibrium (Vanilla GAN)
  - GAN Training Tricks
  - Least Squares GAN
  - Wasserstein GAN
  - BigGAN

The Simple GAN

Geoff Hinton after writing the paper on backprop in 1986

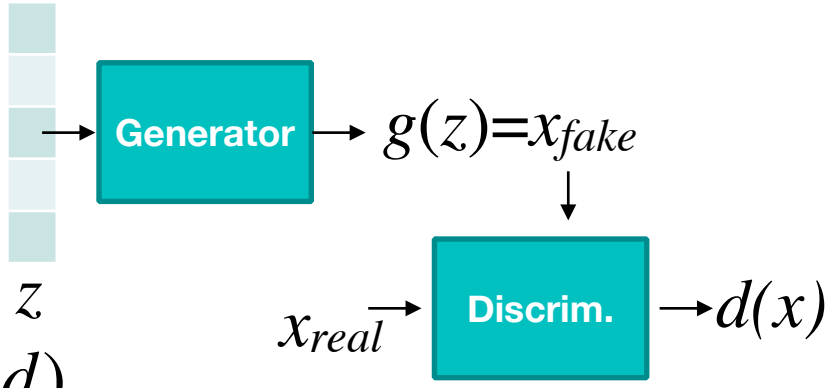
Latent Variable Approximation with GANS

- **Idea**: transform random input data into target of interest
  - Like an image!
- Rather than training an encoder, we need a transformation algorithm
  - 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!
  - … Neural Networks are like the chiropractors of the machine learning…

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

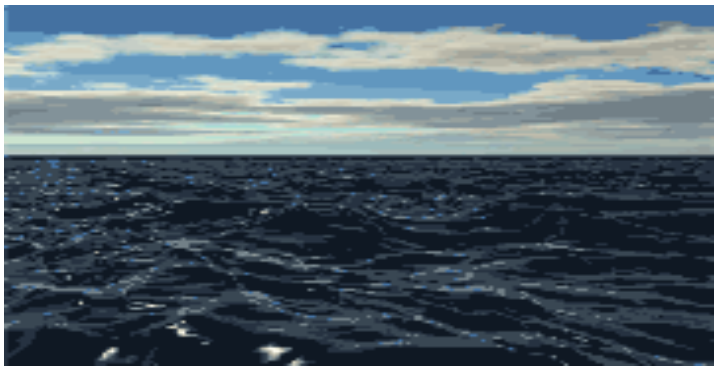
everything dependent on d discriminator maximizes

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
- 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?!**

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) labeled data and (2) generated data!

- Generator:

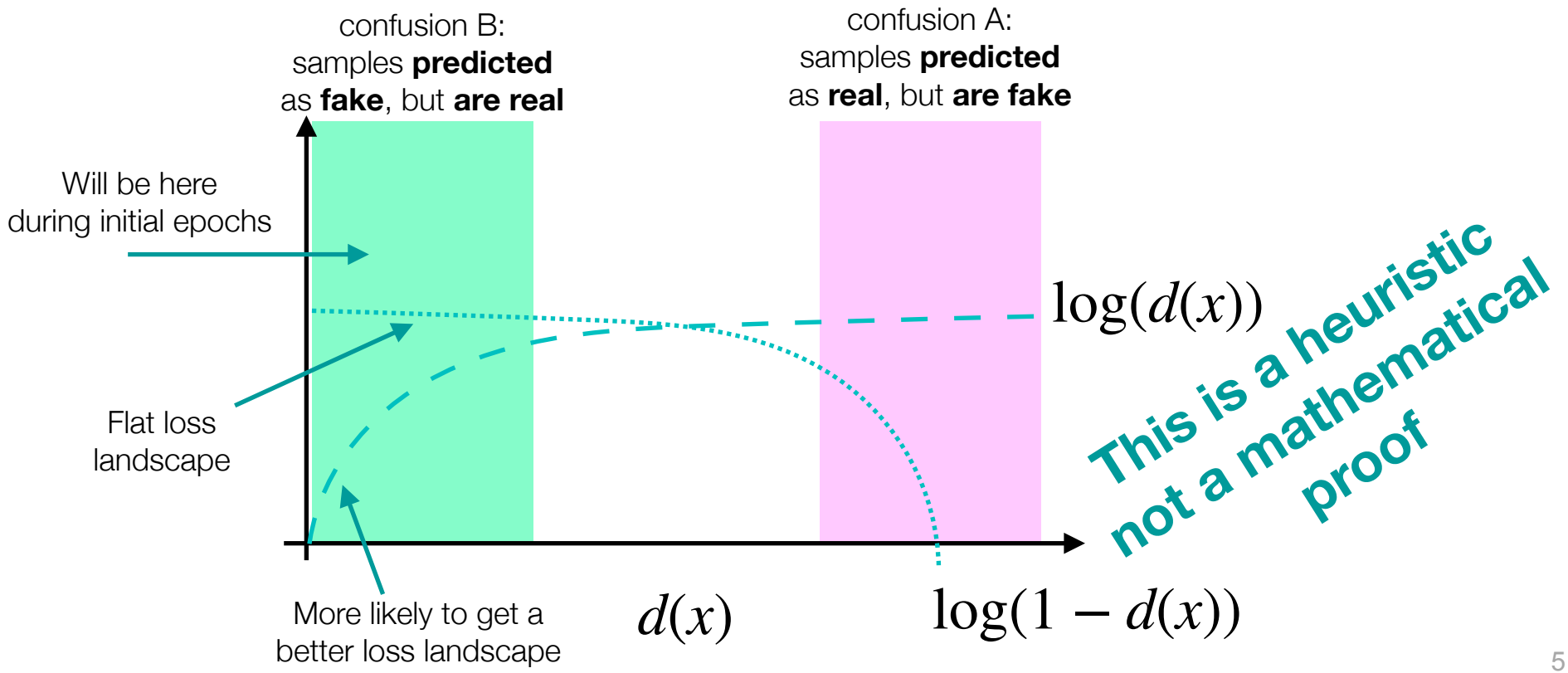
$$\max_{g} \mathbf{E}_{x \leftarrow g(z)} \left[\log(d(x_{fake})) \right]$$

Freeze Discriminator Weights

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

Why was minimizing 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

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?! **Still No! Why?**

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

A bag of tricks from F. Chollet

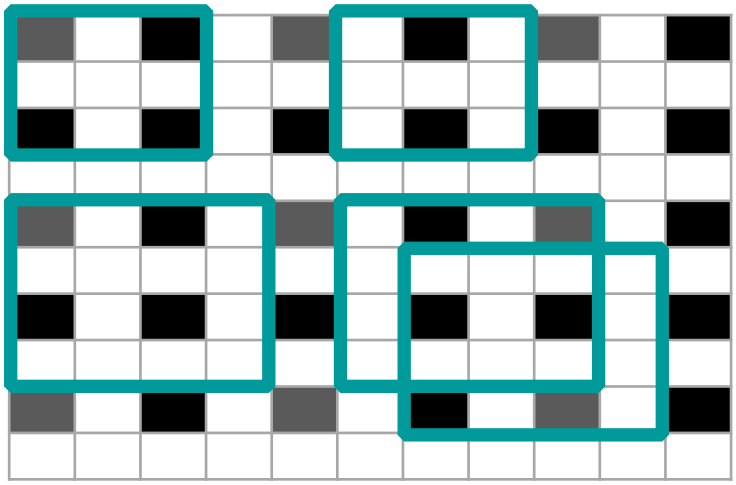
The process of training GANs and tuning GAN implementations is notoriously difficult. There are a number of known tricks you should keep in mind. Like most things in deep learning, it's **more alchemy than science**: **these tricks are heuristics, not theory-backed guidelines**. They're supported by 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 in optimizer and labels
  - GANs get stuck a lot
- Sparse gradients are not your friend here
  - No max pooling, no ReLU 😳
- Make decoder upsampling multiple of stride…
  - Unequal pixel coverage (checkerboards)

What do you mean checkerboard patterns?

- Kernel size should be a symmetric multiple of the stride



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

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

- Option 2: Bilinear resizing of activations, rather than zero insertion or Transpose convolution

Some Results of Generation

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

Highlight: Nearest Training Sample