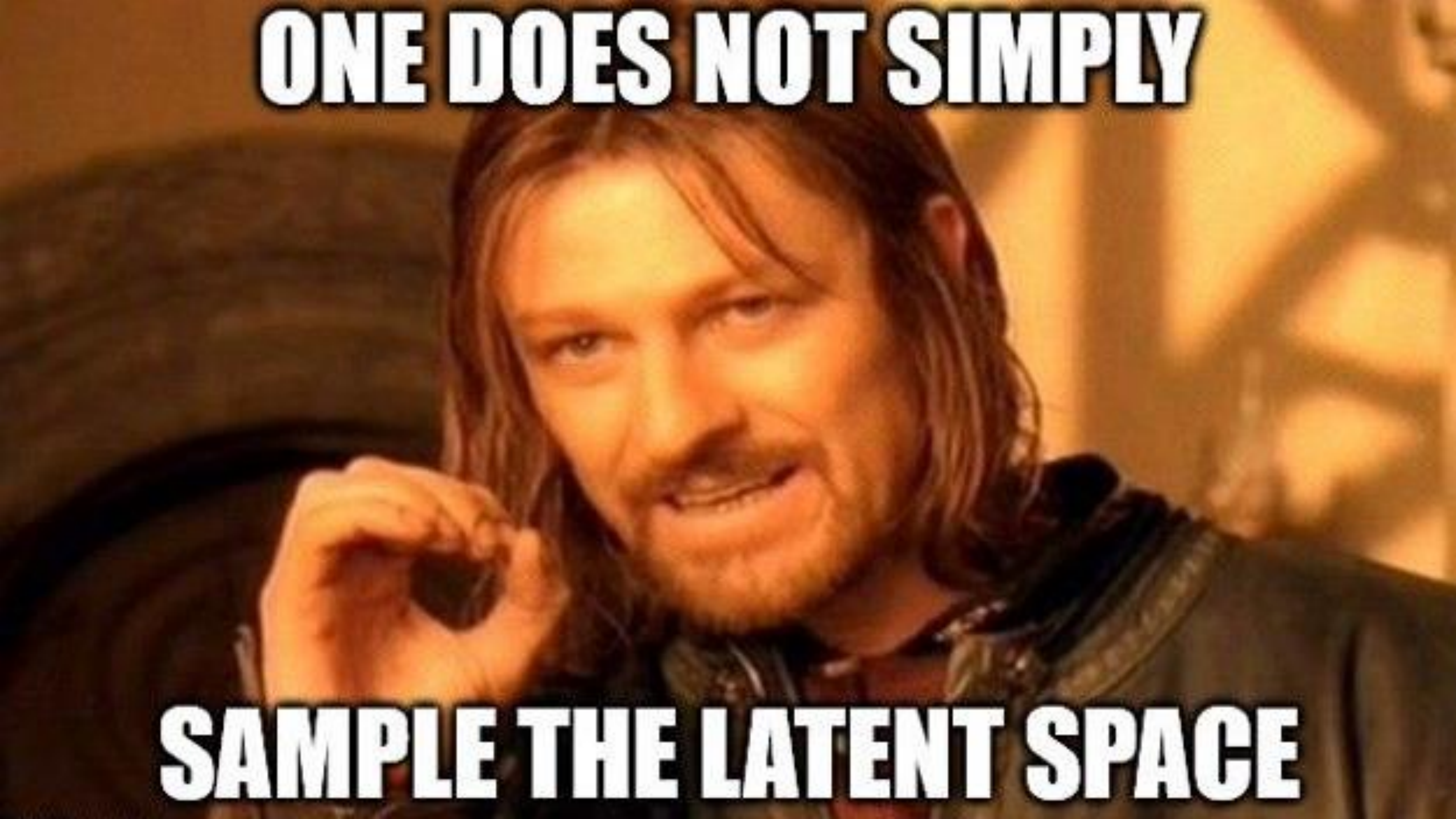

ONE DOES NOT SIMPLY
SAMPLE THE LATENT SPACE

Autoencoding

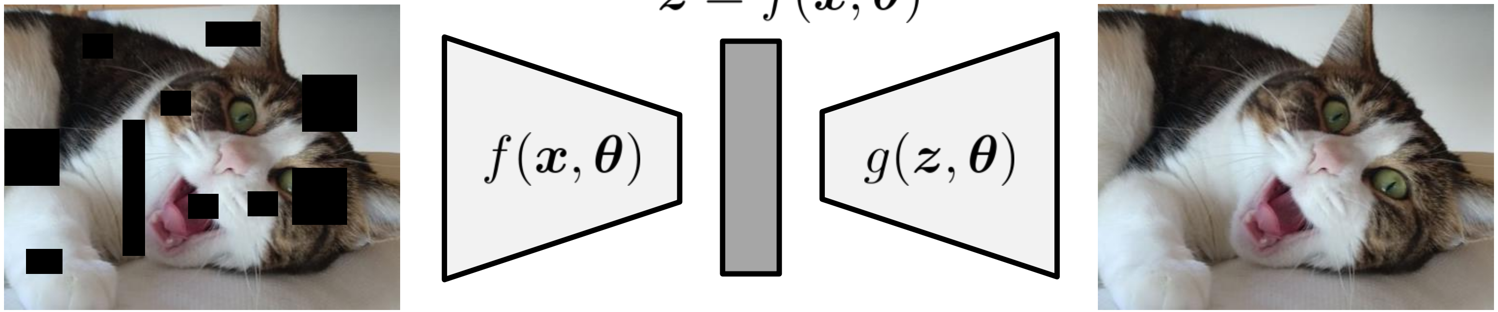
Denoising Autoencoder

$$\boldsymbol{\theta} = \arg\min_{\hat{\boldsymbol{\theta}}} d\left[\boldsymbol{x}, g(f(\boxed{\tilde{\boldsymbol{x}}}, \hat{\boldsymbol{\theta}}), \hat{\boldsymbol{\theta}})\right]$$

- “Encoder”: f()
- “Decoder”: g()
- **Q: What is** $\tilde{\boldsymbol{x}}$?

→ Heavily corrupted versions of x! Corruption: Zero-ing, noise, etc…

$$\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{\theta})$$



[Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, July). Extracting and composing robust features with denoising autoencoders. ICML.](#)
[Vincent, P. et al. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR](#).

Learning Meaningful Feature Representations from Data

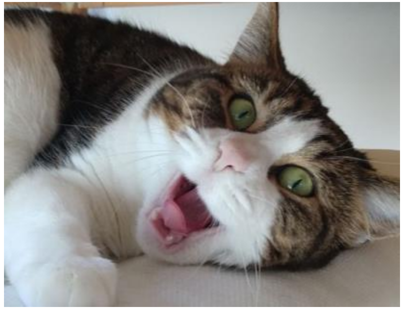
After training: Two sub-modules!

- Image generator
  - Remove encoder
  - Randomly generate latent vector z
  - Generate output sample
  - ??
  - Profit

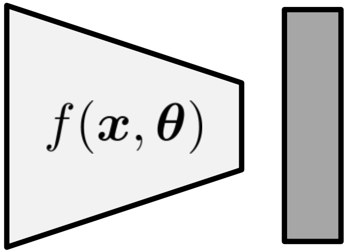
Q: Why?
Embedding shows clusters
→ Makes sense: Distinct encoding for distinct image appearance

But: Embedding is not continuous
→ Also makes sense, because we did not care

Problematic for generative models.
→ What happens **here**? Never seen for decoding!



$$z = f(x, \theta)$$



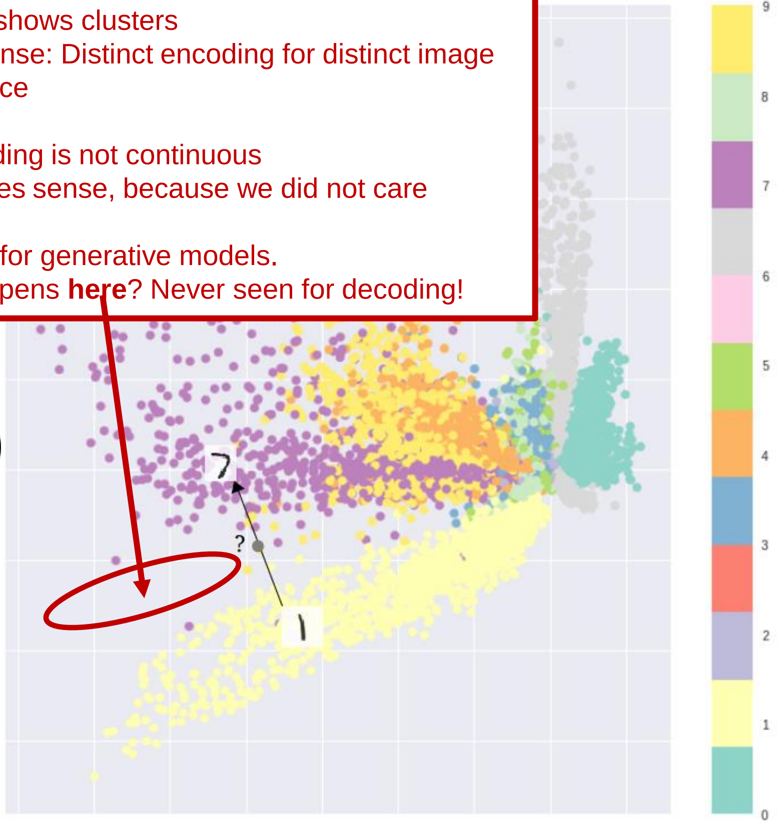
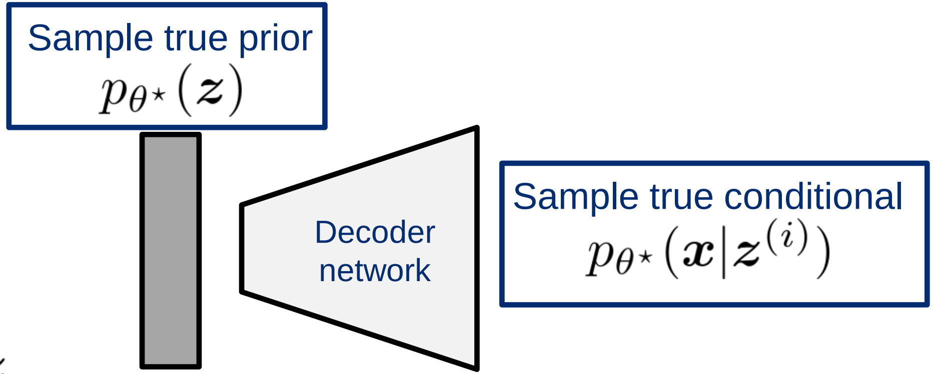


Image from this blog post.

Variational Autoencoders: A Probabilistic Spin

Assume training data $\{\boldsymbol{x}^{(i)}\},\ i = 1, \ldots, N$ is generated from latent representation $\boldsymbol{z}$

Estimate parameters: $\theta^\star$



Q: How to train this model?

Learn parameters that maximize likelihood of training data!

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{z})\, p_\theta(\boldsymbol{x}|\boldsymbol{z})\, \mathrm{d}\boldsymbol{z}$$

Q: Any problem with this?

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. ICLR.
Doersch, C. (2016). Tutorial on variational autoencoders. arXiv preprint arXiv:1606.05908.

Variational Autoencoders

Deriving the log data likelihood

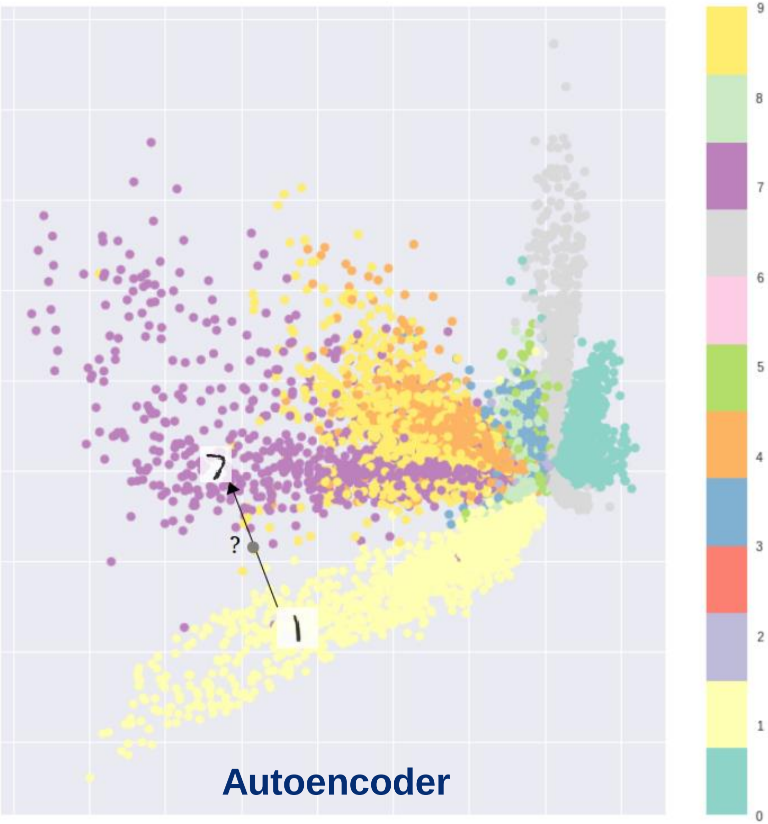
$$
\begin{aligned}
\log p_\theta(\boldsymbol{x}^{(i)}) &= \boldsymbol{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\left[\log p_\theta(\boldsymbol{x}^{(i)})\right] \quad (p_\theta(\boldsymbol{x}^{(i)}) \text{ does not depend on } \boldsymbol{z}) \\
&= \boldsymbol{E}_{\boldsymbol{z}}\left[\log \frac{p_\theta(\boldsymbol{x}^{(i)}|\boldsymbol{z})\, p_\theta(\boldsymbol{z})}{p_\theta(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\right] \quad (\text{Bayes' rule}) \\
&= \boldsymbol{E}_{\boldsymbol{z}}\left[\log \frac{p_\theta(\boldsymbol{x}^{(i)}|\boldsymbol{z})\, p_\theta(\boldsymbol{z})}{p_\theta(\boldsymbol{z}|\boldsymbol{x}^{(i)})} \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})}{q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\right] \quad (\text{Multiply with constant}) \\
&= \boldsymbol{E}_{\boldsymbol{z}}\left[\log p_\theta(\boldsymbol{x}^{(i)}|\boldsymbol{z})\right] - \boldsymbol{E}_{\boldsymbol{z}}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})}{p_\theta(\boldsymbol{z})}\right] + \boldsymbol{E}_{\boldsymbol{z}}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})}{p_\theta(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\right] \quad (\text{Logarithms}) \\
&= \boldsymbol{E}_{\boldsymbol{z}}\left[\log p_\theta(\boldsymbol{x}^{(i)}|\boldsymbol{z})\right] - D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)}), p_\theta(\boldsymbol{z})) + D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)}), p_\theta(\boldsymbol{z}|\boldsymbol{x}^{(i)}))
\end{aligned}
$$

Decoder network, estimates this term through sampling. (Differentiable via re-parametrization, see paper)

KL between encoder and z-prior → Two Gaussians, closed-form solution

p(z|x) intractable, this cannot be computed. But KL ≥ 0

Embedding



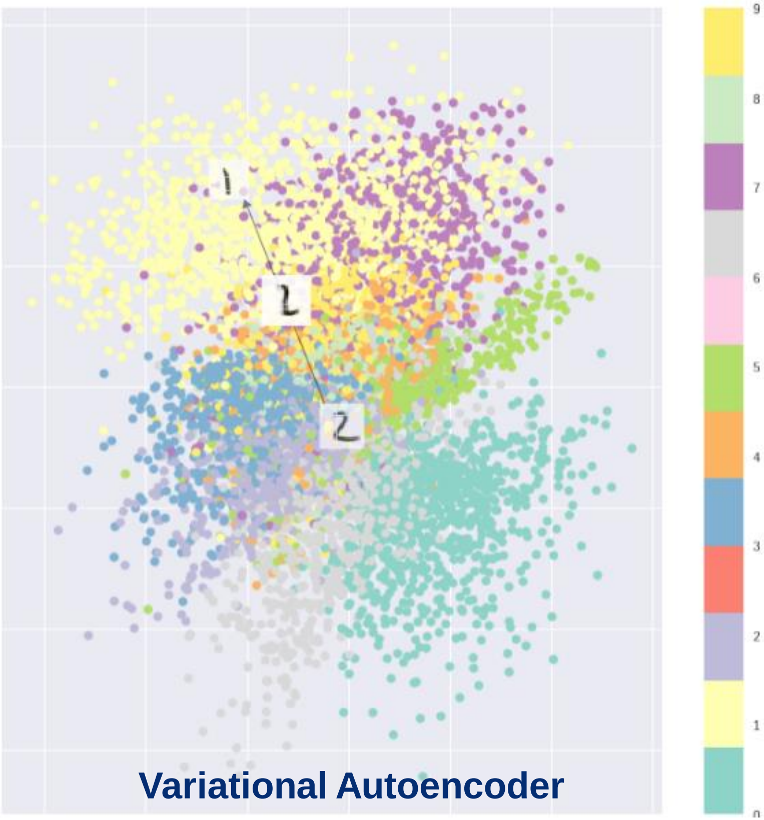


Image from this blog post.

Unsupervised and Representation Learning

Re-introducing the split in feature extraction and semantic mapping,
but trained end-to-end.

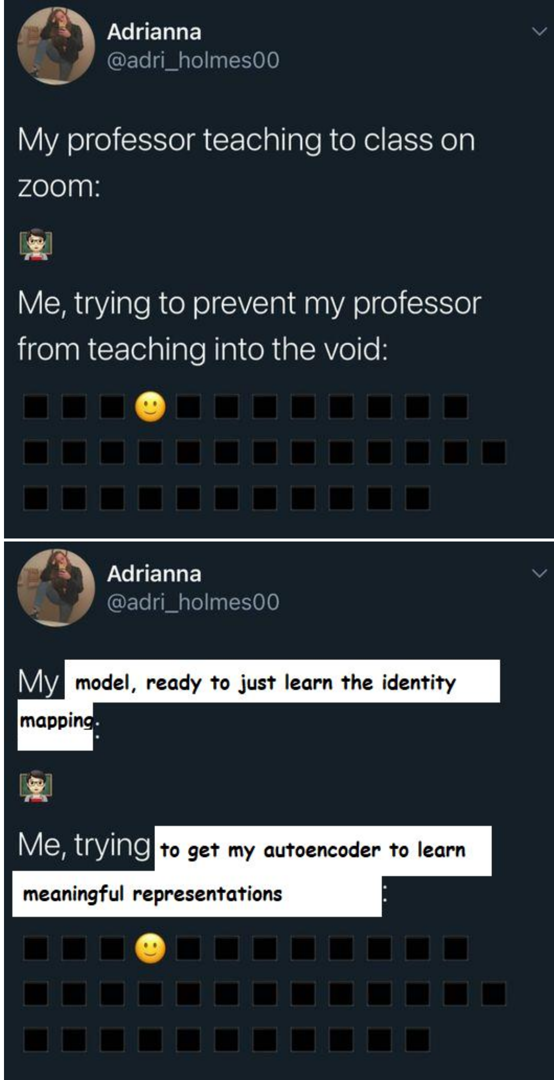
$$\boxed{\min_{\theta,W}} \frac{1}{N} \sum_i L\left(\boxed{g_W\left(\boxed{f_\theta\left(\boxed{x_i}\right)}\right)}, \boxed{y_i}\right)$$

→ Learn generalizable, semantic representations of samples solely from data.

- Input x_i with label y_i (**Caveat**: y is derived from data itself)
- f_Θ computes high-level representations of x_i
- g_W is a classifier that maps representations to labels

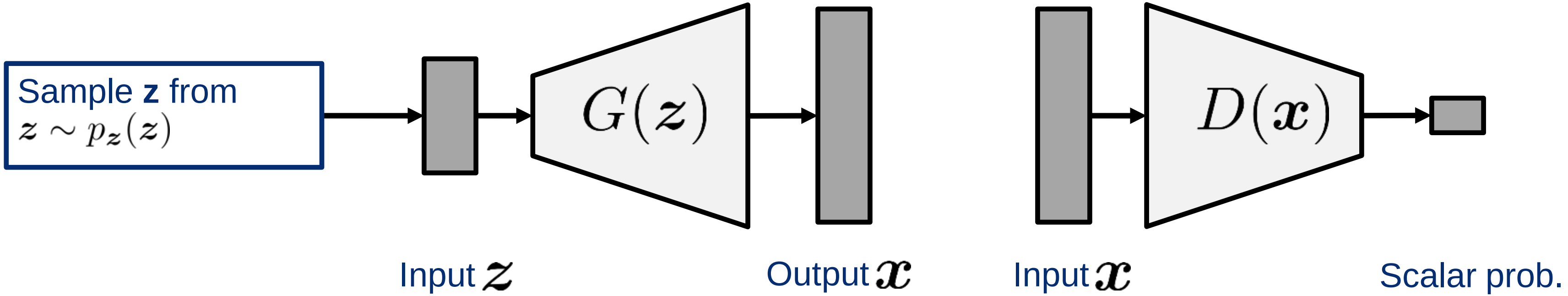
→ **Task: Find parameters Θ,W such that loss is optimal**


THE #1 DATA SCIENTIST EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY MODEL'S TRAINING"
HEY! GET BACK TO WORK!
TRAINING!
OH. CARRY ON.


Adrianna
@adri_holmes00
My professor teaching to class on zoom:
Me, trying to prevent my professor from teaching into the void:
Adrianna
@adri_holmes00
My model, ready to just learn the identity mapping:
Me, trying to get my autoencoder to learn meaningful representations:


When your GAN suffers
from mode collapse

Generative Adversarial Networks



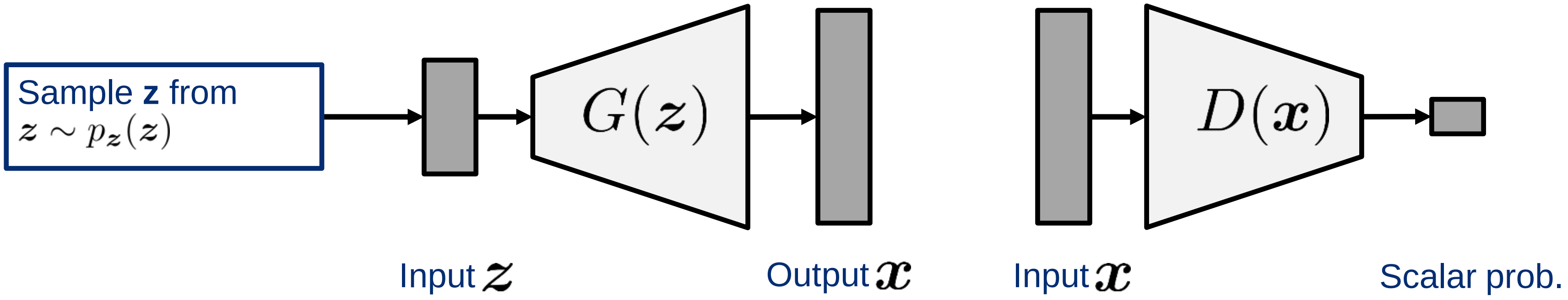
$$\min_G \max_D V(D,G) = \boldsymbol{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \boldsymbol{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G((z))))]$$

The generator

- Accepts an input noise vector with prior $p_{\boldsymbol{z}}(\boldsymbol{z})$
- Represents mapping $G_{\theta_g}(\boldsymbol{z})$ that generates distribution p_g over the data space
- G is a differentiable function

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. NeurIPS (pp. 2672-2680).

Generative Adversarial Networks



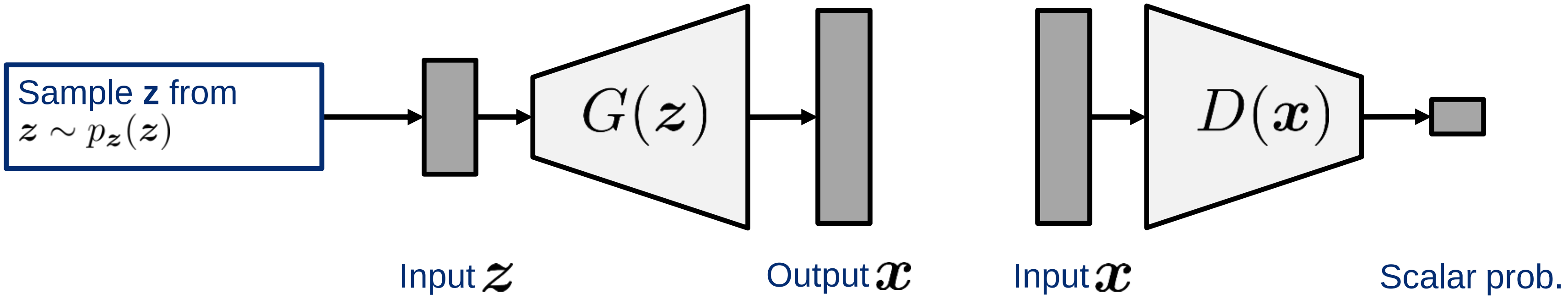
$$\min_G \max_D V(D,G) = \boldsymbol{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \boldsymbol{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G((z))))]$$

The discriminator

- Accepts an input sample $\boldsymbol{x}$
- Represents mapping $D_{\theta_d}(\boldsymbol{x})$ that yields probability of $\boldsymbol{x} \sim p_{\mathrm{data}}$
- D is also a differentiable function

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. NeurIPS (pp. 2672-2680).

Generative Adversarial Networks



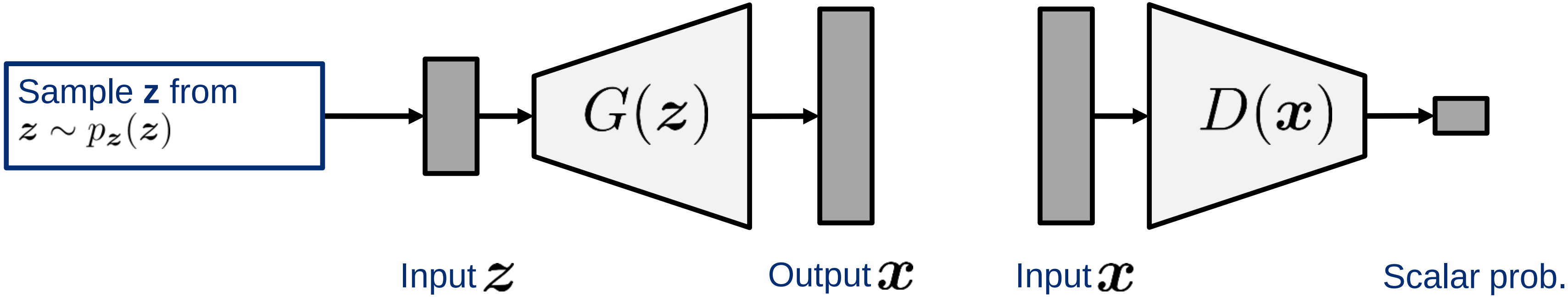
$$\min_{G} \boxed{\max_{D}} V(D, G) = \boxed{\boldsymbol{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]} + \boxed{\boldsymbol{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G((z))))]}$$

The value function

- Discriminator $D_{\theta_d}(\boldsymbol{x})$ assigns correct label to any sample it is presented: real / fake
  → Should be maximal: Very good discrimination

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. NeurIPS (pp. 2672-2680).

Generative Adversarial Networks

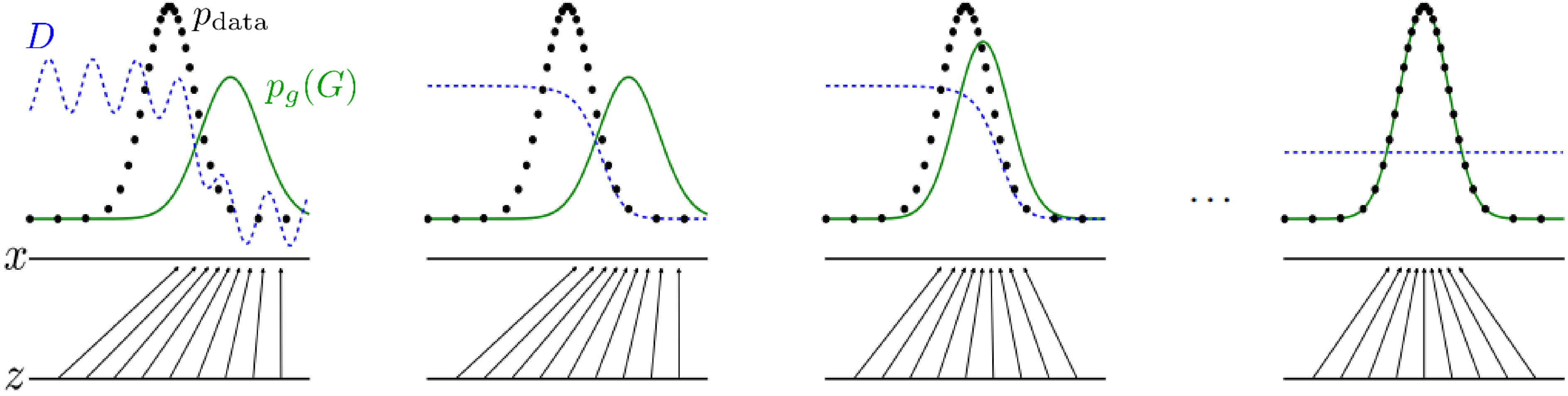


$$\boxed{\min_{G}} \max_{D} V(D, G) = \boldsymbol{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \boxed{\boldsymbol{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G((z))))]}$$

The value function

- Discriminator $D_{\theta_d}(\boldsymbol{x})$ assigns correct label to any sample it is presented: real / fake
  → Should be maximal: Very good discrimination
- Generator $G_{\theta_g}(\boldsymbol{z})$ attempts to "fool" the discriminator
  → Should be minimal: Poor discrimination

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. NeurIPS (pp. 2672-2680).



- Near convergence: $p_g(G)$ is similar to p_{data}, and $D(\boldsymbol{x})$ is partially accurate
- Inner loop: $D(\boldsymbol{x})$ is trained to better discriminate, converging to $D^{\star}(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$
- After G update: Gradient of $D(\boldsymbol{x})$ has guided $G(\boldsymbol{z})$ to be more likely classified as “real”
- After multiple iterations, $p_g(G) = p_{\text{data}}$ and $D(\boldsymbol{x}) = \frac{1}{2}$

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. NeurIPS (pp. 2672-2680).

Challenges

Poor discriminator learning schedule

Poor generator learning schedule

Challenges

Poor discriminator learning schedule

- Discriminator provides gradients for learning
- If discriminator too strong → Saturation, and zero gradient
  - This problem is very real!
  - Beginning: Generator is weak; consider corruption etc. to confuse discriminator

Poor generator learning schedule

Challenges

Poor discriminator learning schedule

- Discriminator provides gradients for learning
- If discriminator too strong → Saturation, and zero gradient
  - This problem is very real!
  - Beginning: Generator is weak; consider corruption etc. to confuse discriminator

Poor generator learning schedule

- Generator updates w.r.t. discriminator gradient
- Will exploit any discriminator weakness → Mode collapse
  - This problem, unfortunately, is equally real
  - NNs are surprisingly brittle (will see later)

The bottom line: GANs are nice – after the fact.