Autoencoders

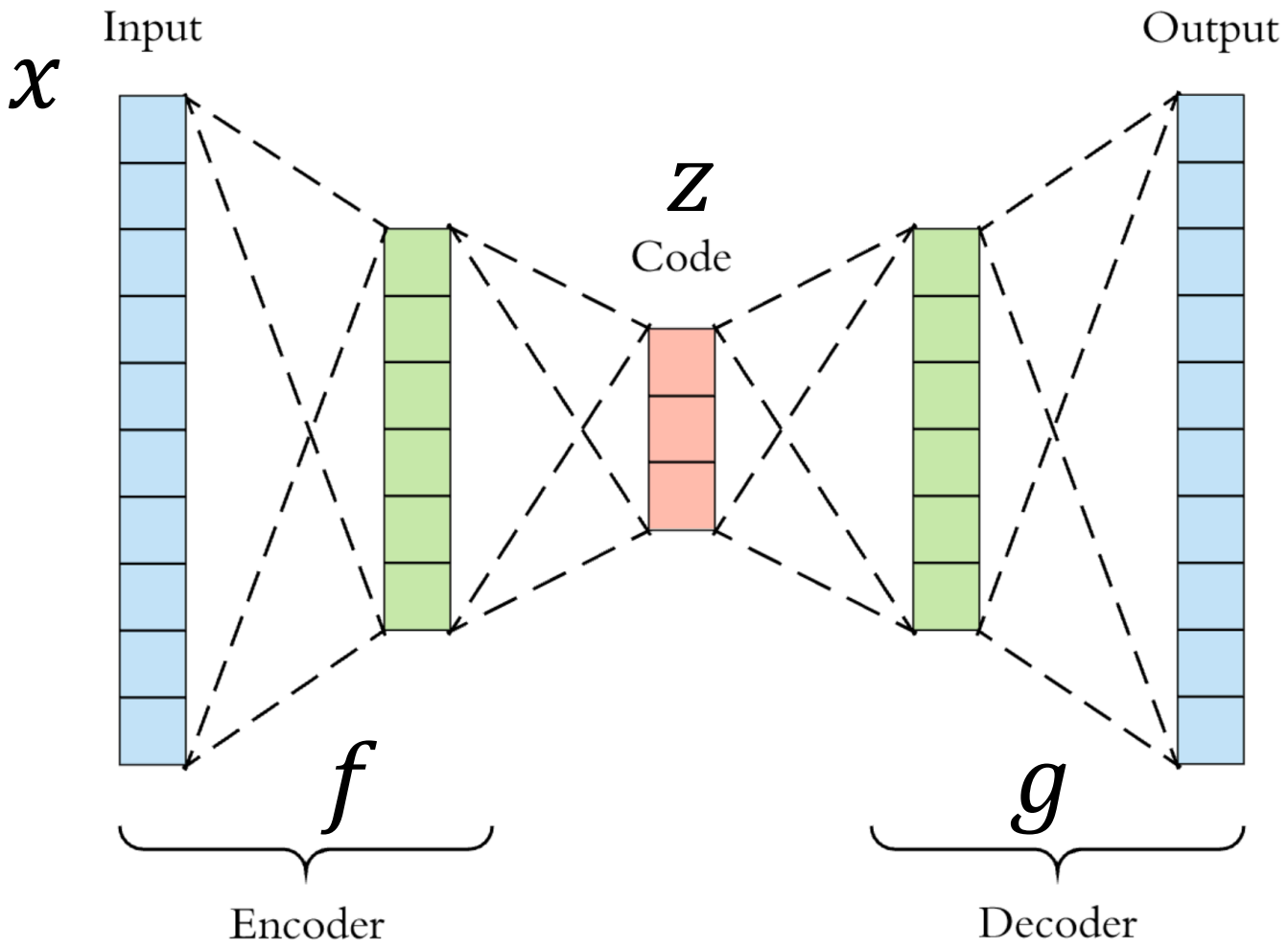
ECE57000: Artificial Intelligence, Fall 2019

David I. Inouye

Announcements

- Presentations begin next week!
- First day is volunteers
- Wednesday everyone should be ready to present

Autoencoders map an input to a latent code (encoder) and map this latent code back to the input (decoder)



https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

The optimization problem is to fit the encoder and decoder simultaneously to reconstruct output

- More formally, the autoencoder objective is:

$$\min_{f,g} \mathbb{E}[L(x, \tilde{x})]$$

$$\min_{f,g} \mathbb{E}\left[L\left(x, g\left(f(x)\right)\right)\right]$$

- One example is using Mean Squared Error loss

$$\min_{f,g} \mathbb{E}\left[\left\|x - g\left(f(x)\right)\right\|_2^2\right]$$

If there are no constraints on the encoder and decoder than the identity function works perfectly…

- Suppose $f(x) = x$ and $g(x) = x$

- Then we know that

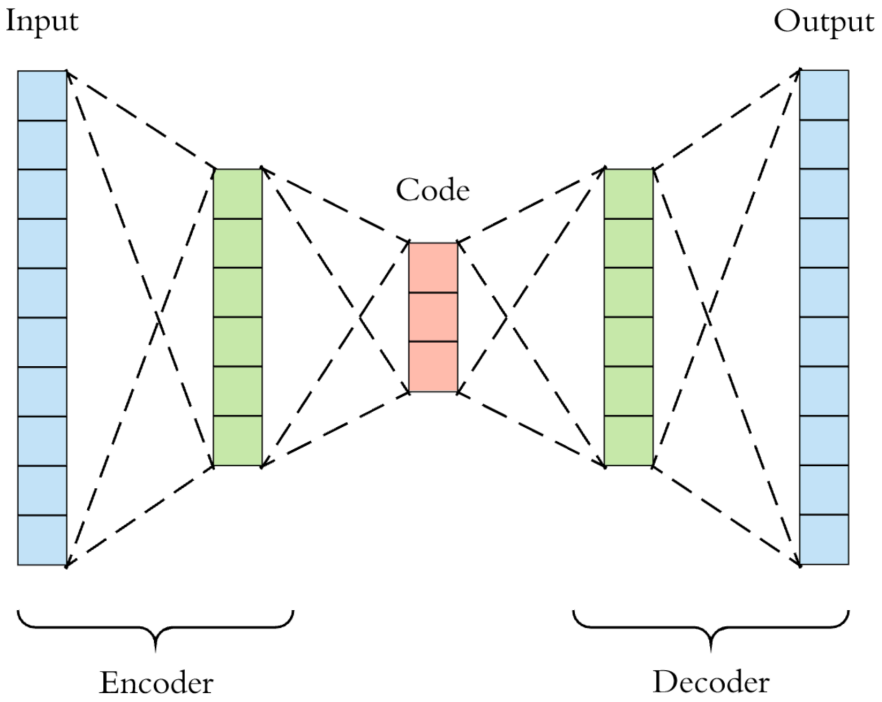
$$\min_{f,g} \mathbb{E}\left[\left\|x - g\left(f(x)\right)\right\|_2^2\right]$$
$$= \min_{f,g} \mathbb{E}[\|x - x\|_2^2] = 0$$

  - And since all terms are positive, this is the global minimum

- Trivial/useless…What can we do?

Adding constraints to f, g or z can often produce interesting properties of z

- Undercomplete autoencoders assume that the latent space has lower dimension, i.e., $k < d$

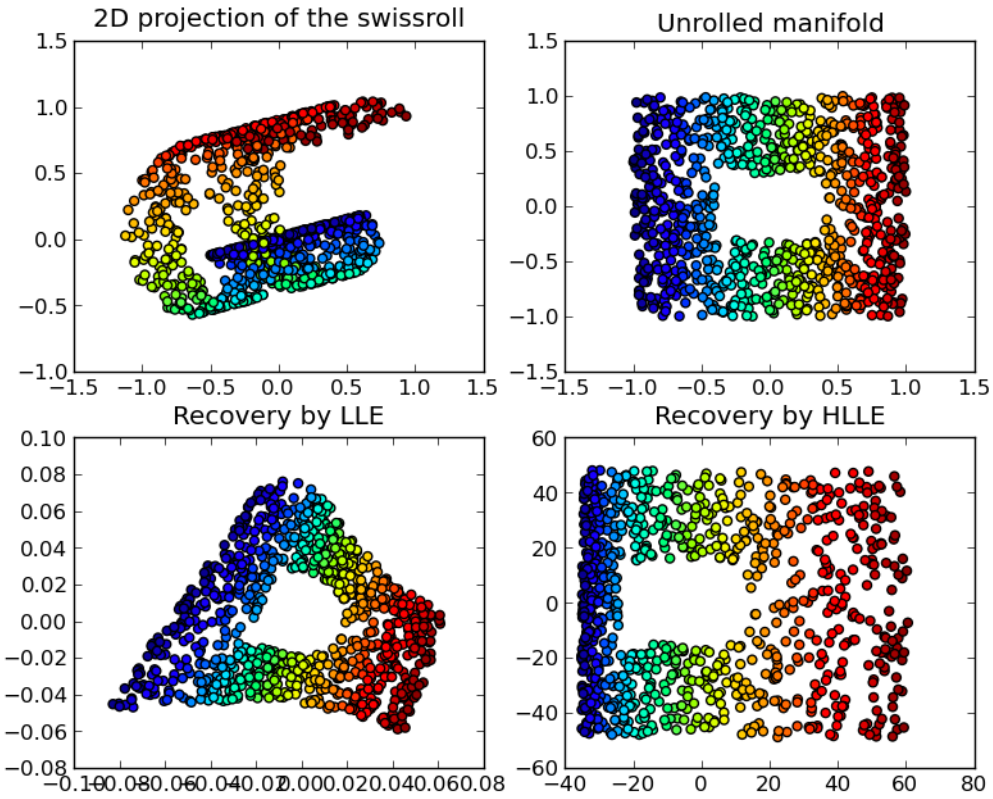


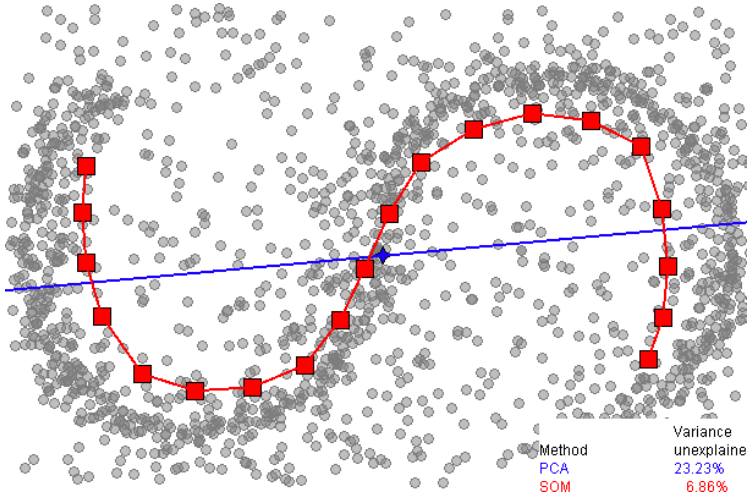
The **undercomplete** and **linear** autoencoder is closely related to PCA

- Formally
  - Let $\mathrm{z} = f(x) = Ax + b, z \in \mathbb{R}^k$
  - Let $\tilde{x} = g(z) = Bz + c$
  - Let $L(x, \tilde{x}) = \mathbb{E}[\|x - \tilde{x}\|_2^2]$

- One particular solution can be derived from PCA though other (closely-related) solutions exist

Why might we want a **non-linear** autoencoder? Non-linear dimensionality reduction





https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction

Even undercomplete autoencoders can be uninteresting if **non-linear**

- With no other constraints, again a uninteresting solution exists where x_i is a training instance
  - $z = f(x) = \begin{cases} i, & x = x_i \\ 0, & \text{otherwise} \end{cases}$
  - $g(z) = \begin{cases} x_i, & z = i \\ 0, & \text{otherwise} \end{cases}$

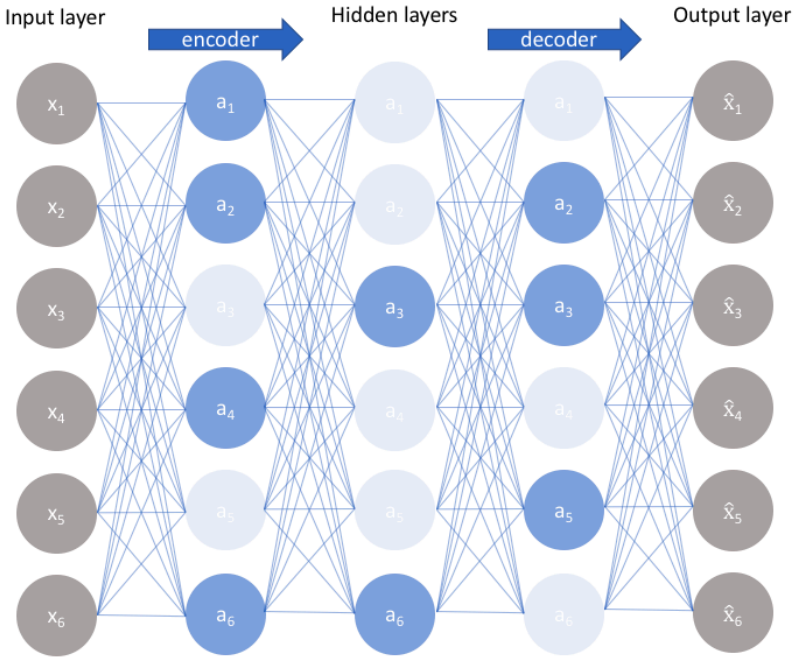
- While this doesn't necessarily happen in practice, it demonstrates that more constraints are usually needed

Sparse autoencoders add a penalty that the latent space is sparse

- Add a regularization term to latent variables

$$\min_{f,g} L\left(x, g\left(f(x)\right)\right) + \lambda \|f(x)\|_1$$

- This creates **data-dependent** sparsity

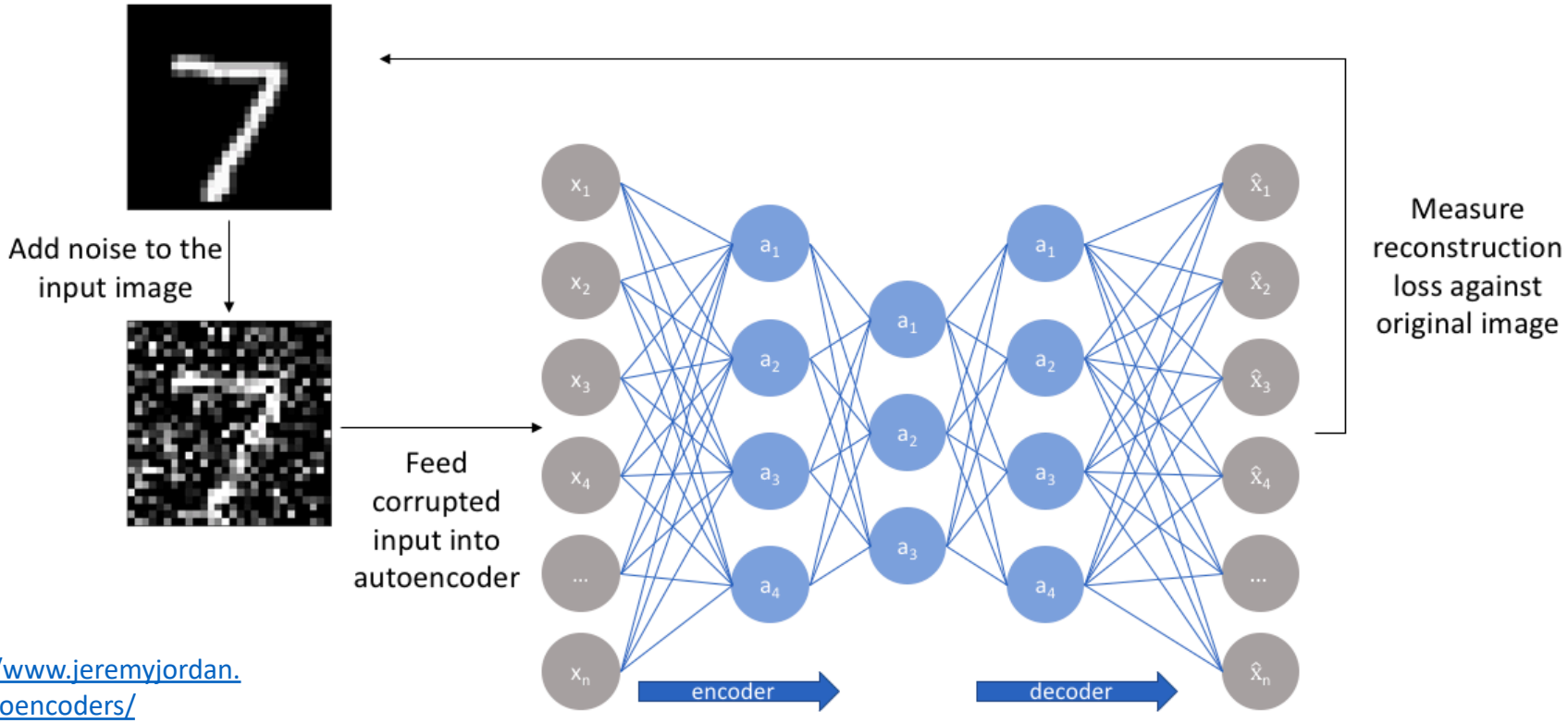


https://www.jeremyjordan.me/autoencoders/

Denoising autoencoders force functions to learn to remove noise rather than copy the input

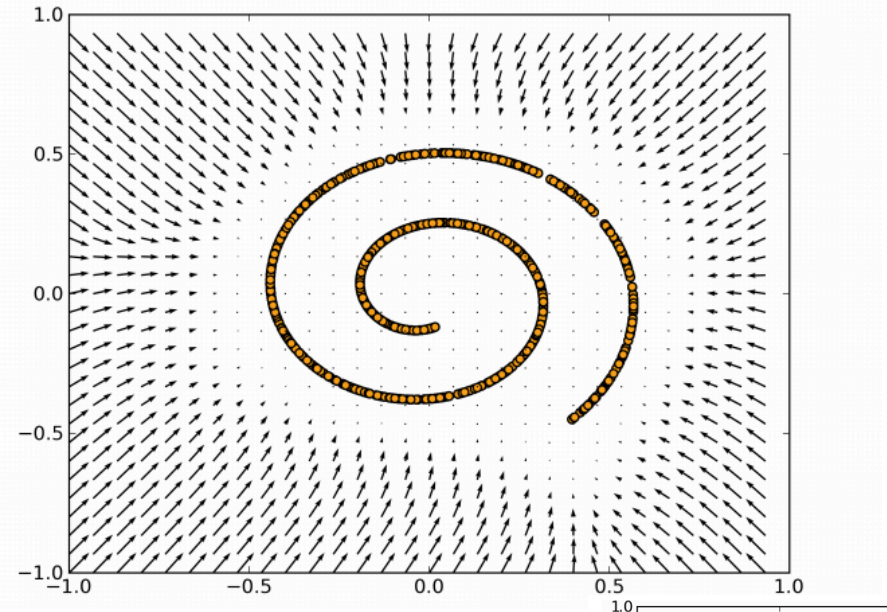
- Add noise to the input so that copying input is not possible

$$\min_{f,g} \mathbb{E}\left[L\left(x, g\left(f(x+\epsilon)\right)\right)\right], \text{where } \epsilon \sim \mathcal{N}(0,1)$$

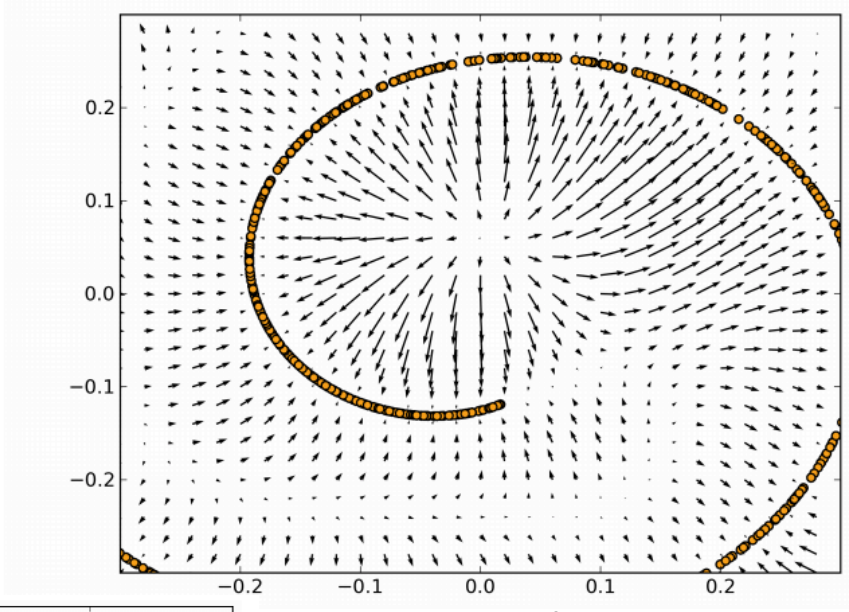


https://www.jeremyjordan.me/autoencoders/

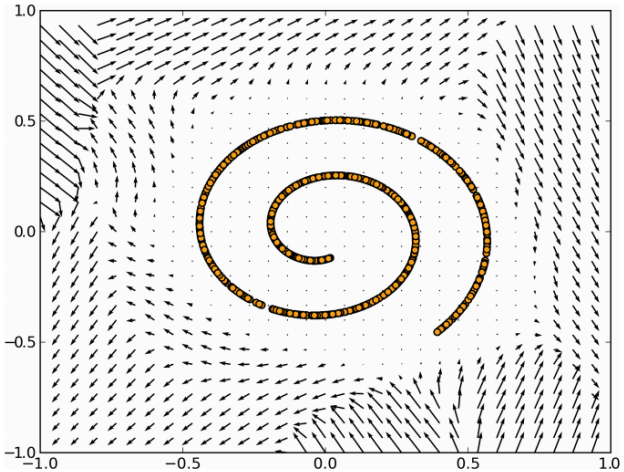
Denoising autoencoders can be shown to learn the structure of the distribution



Vector field that pushes corrupted data back to manifold as learned by DAE



Zoomed in view



May misbehave outside data area in practice

https://arxiv.org/pdf/1211.4246.pdf

Autoencoders can also use **non-deterministic** or **probabilistic mappings**

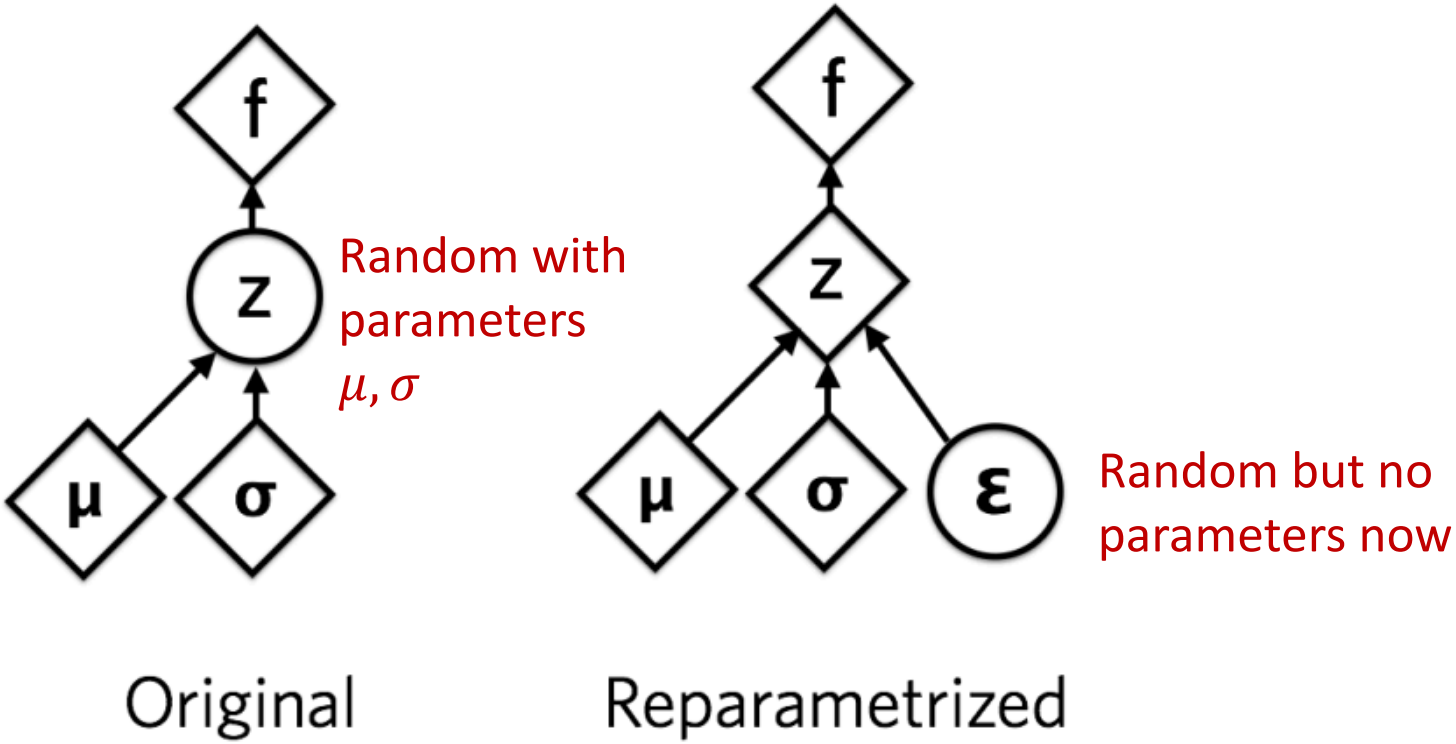
- Instead of $f(x)$, consider probabilistic encoder $q_f(z|x)$
- Instead of $g(z)$, consider probabilistic decoder $p_g(x|z)$
- The output is either the distribution itself or a sample from the distribution
- Illustration: Multivariate Gaussian $p(x, z) = \mathcal{N}(\mu, \Sigma)$ (draw on board)

Variational Autoencoders (VAE) use independent Gaussian for both encoder and decoder

- μ and σ of Gaussian are computed by DNNs
  - $q_f(z|x) \sim \mathcal{N}\left(\mu_f(x), \text{diag}\left(\sigma_f(x)\right)\right)$
  - $p_g(x|z) \sim \mathcal{N}\left(\mu_g(z), \text{diag}\left(\sigma_g(z)\right)\right)$

- How can we compute gradient through probabilistic functions?

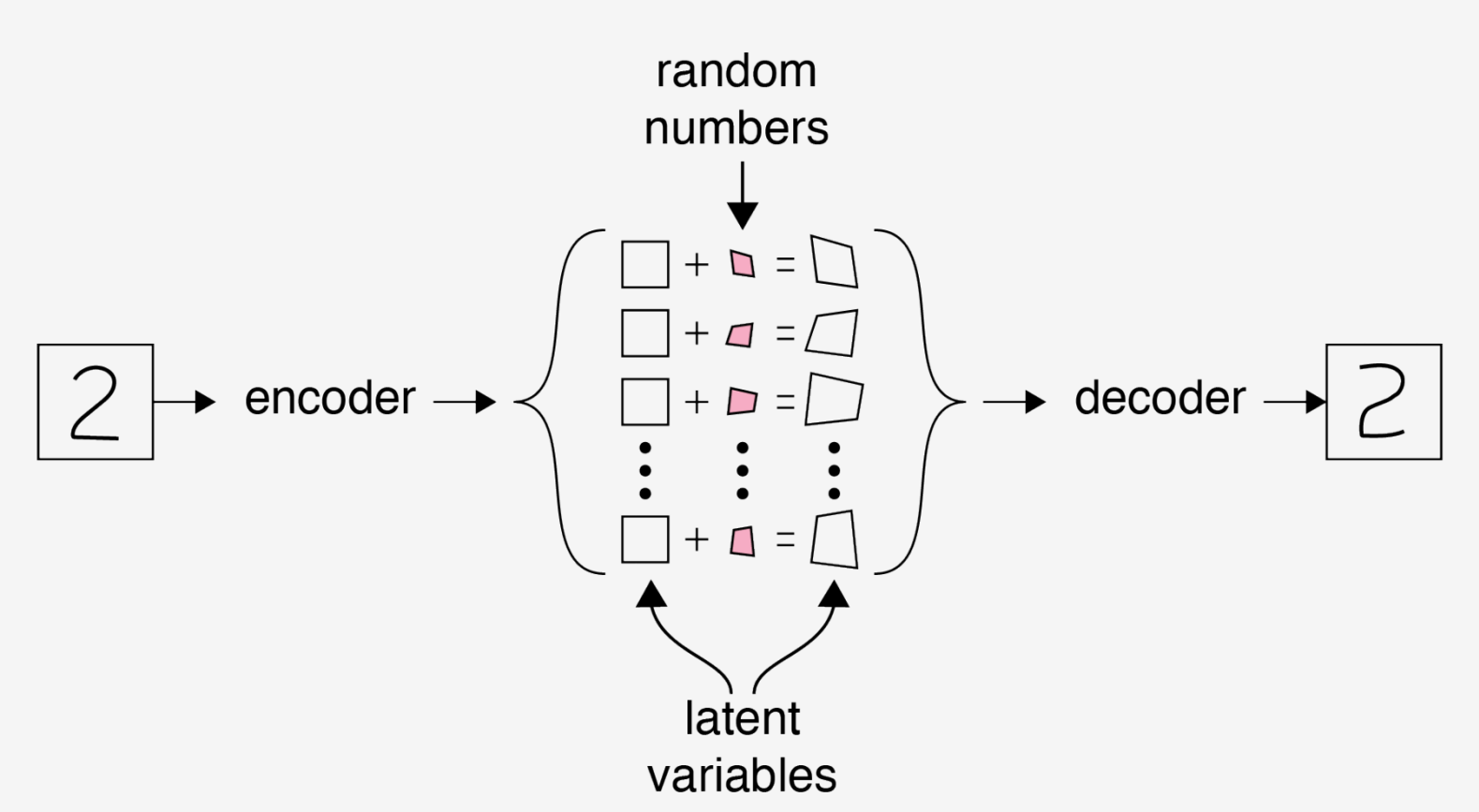
Reparameterization trick allows (stochastic) gradients to be computed



$$\text{Reparametrization Trick}: \; z = \mu + \sigma * \epsilon; \quad \epsilon \sim \mathcal{N}(0,1)$$

https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

How does this look like all together?



https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

A different probabilistic perspective on Variational Autoencoders (VAE): Latent variables

- Variational inference is a way of approximately minimizing the observed likelihood:

$$\min_p \widehat{\mathbb{E}}[\log p(x)]$$

$$\min_p \widehat{\mathbb{E}}[\log \int_Z p(x,z)dz]$$

- We can show that the following is a lower bound for **any distribution** q

$$\log p(x) \geq ELBO(x;q)$$

$$= \int_Z q(z) \log p(x,z)\, dz - \int_Z q(z) \log q(z) dz$$

- If we let $q(z) = q_f(z|x)$, then we have VAE
- Can be seen as minimizing divergence between $KL\left(q_f(z|x), p(z|x)\right)$

Some references and examples

- https://www.jeremyjordan.me/autoencoders/
- http://www.deeplearningbook.org/contents/autoencoders.html
- https://github.com/pytorch/examples/tree/master/vae