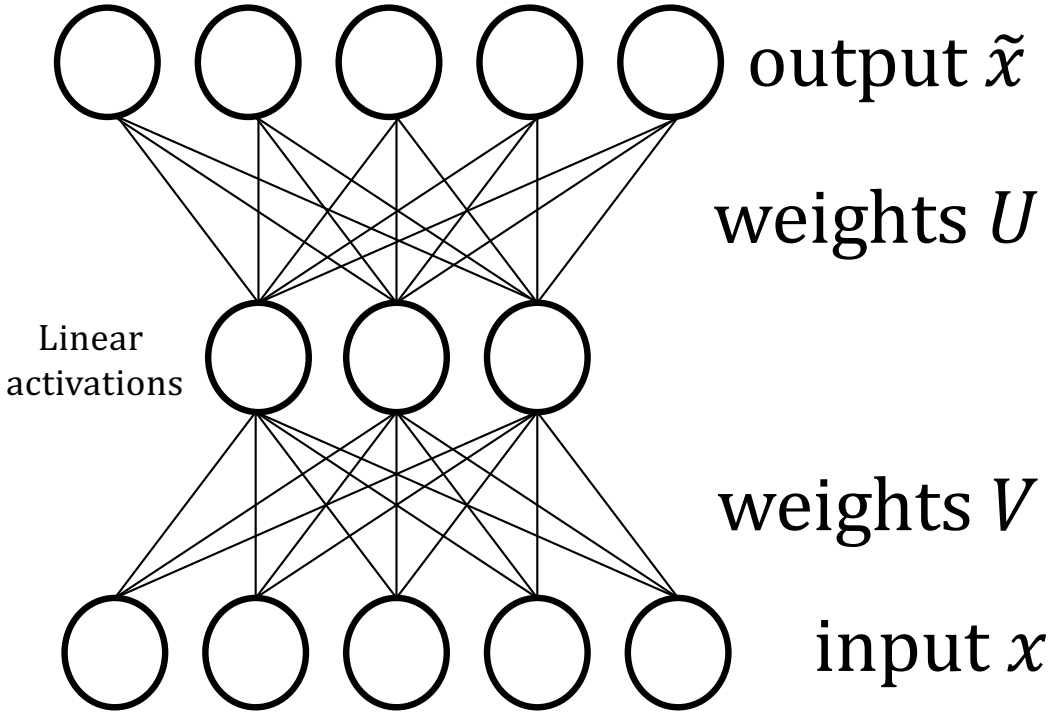
Autoencoders

Bryan Pardo

Northwestern University

(updated winter 2026)

Simplest Autoencoder: Linear model, 1 hidden layer

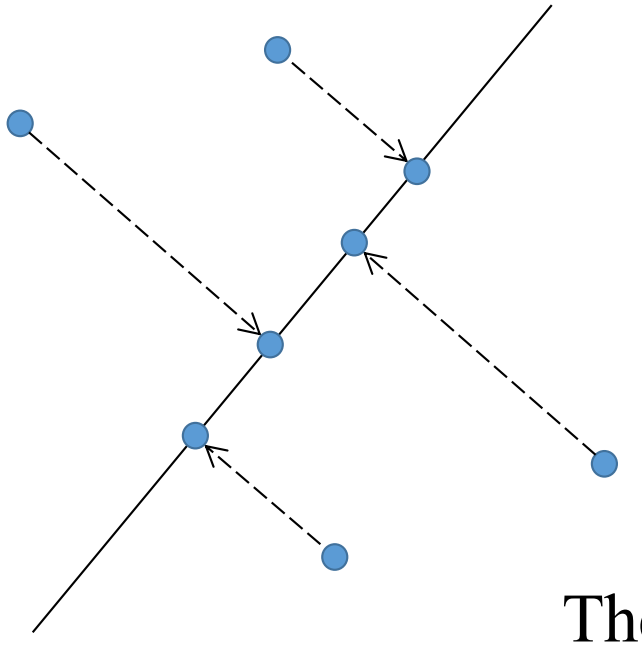


$$\text{loss } \mathcal{L} = \|x - \tilde{x}\|^2$$

$$\tilde{x} = UVx$$

Maps d dimensional input x to k dimensional embedding subspace S

Simplest Autoencoder: Linear model, 1 hidden layer



$$\text{loss } \mathcal{L} = \|x - \tilde{x}\|^2$$
$$\tilde{x} = UVx$$

Maps d dimensional input x to k dimensional embedding subspace S

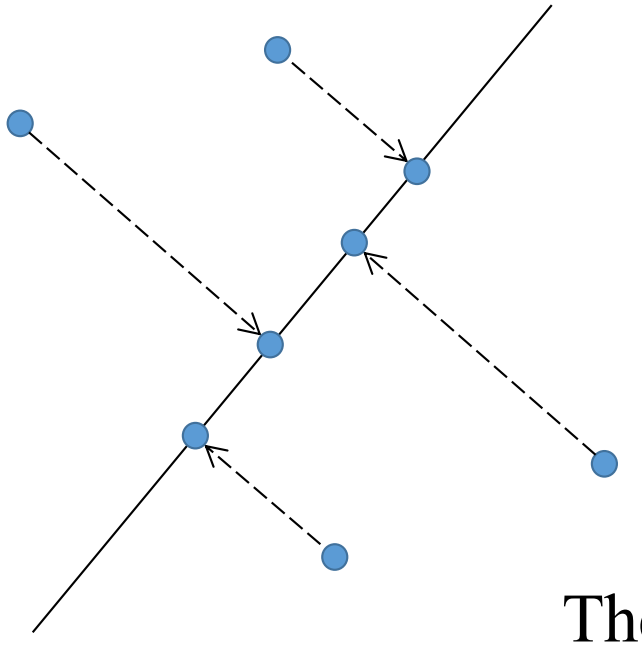
The projection of x onto S is the point in S which minimizes the 2-norm distance to x.

Simplest Autoencoder: Principal Component Analysis

The linear autoencoder learns $U = Q$ and $V = Q^T$, where Q is an orthonormal basis for S.

$$\text{loss } \mathcal{L} = \|x - \tilde{x}\|^2$$

$$\tilde{x} = UVx$$



Maps d dimensional input x to k dimensional embedding subspace S

The projection of x onto S is the point in S which minimizes the 2-norm distance to x.

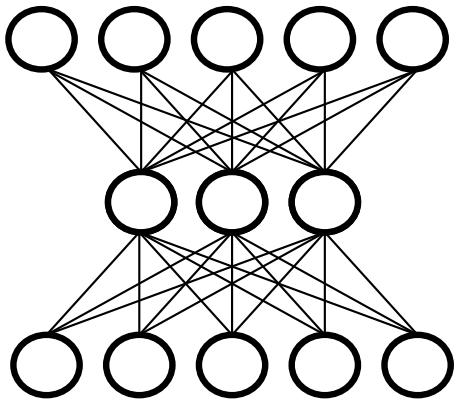
This was just to get the idea…

- You wouldn't actually do this by training a neural net.

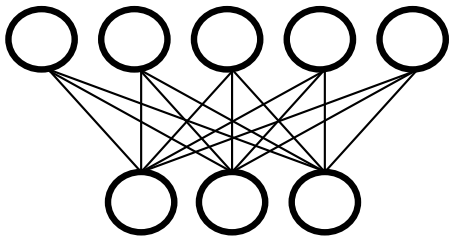
- The standard algorithm is principal component analysis (PCA).

- Read about it here:
https://en.wikipedia.org/wiki/Principal_component_analysis

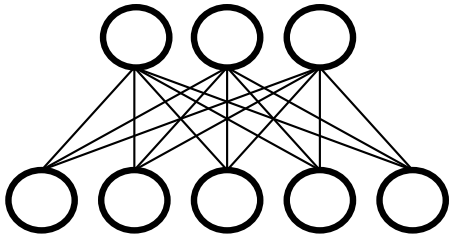
Back to the autoencoder…



Let’s split it into 2 parts



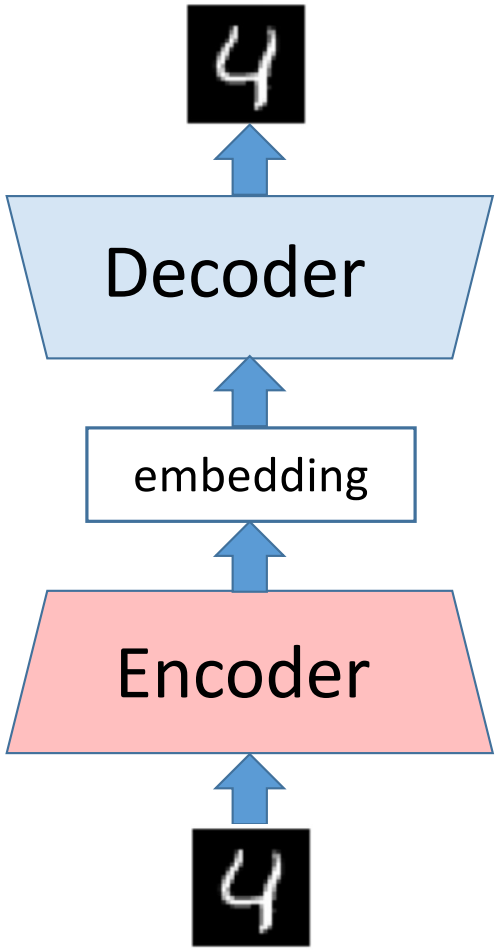
Decoder



Encoder

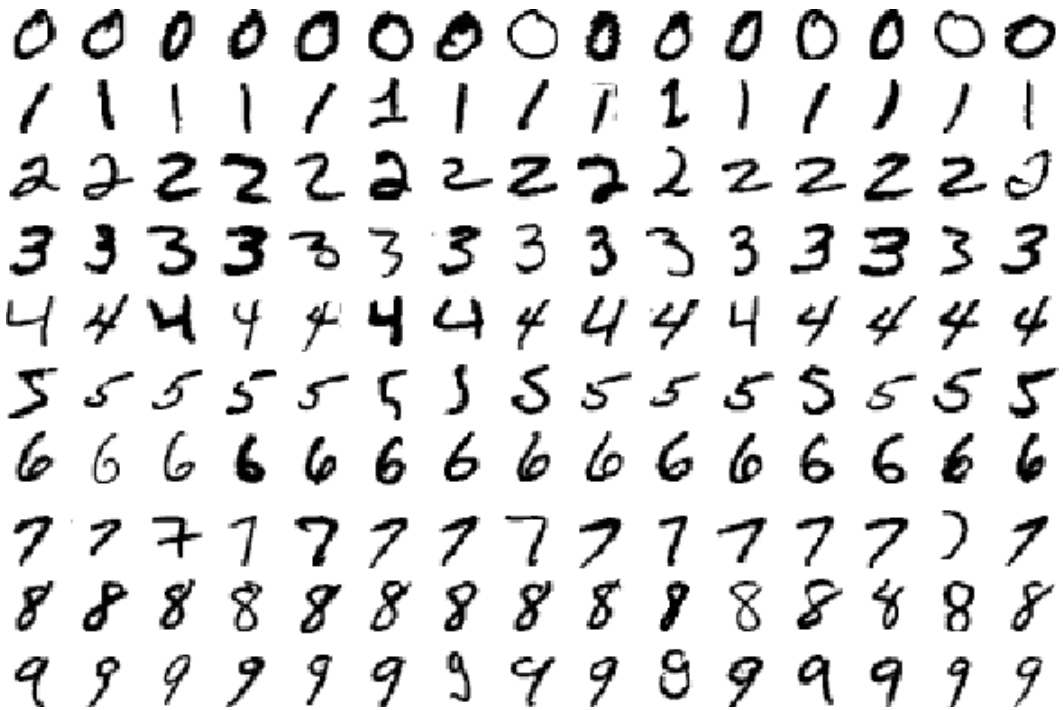
More generally

- With multiple layers & nonlinear activations we can map on to a nonlinear embedding space
- We can represent complex data this way and use the encoder as the input to a supervised network
- We can call the output of the encoder an “embedding”



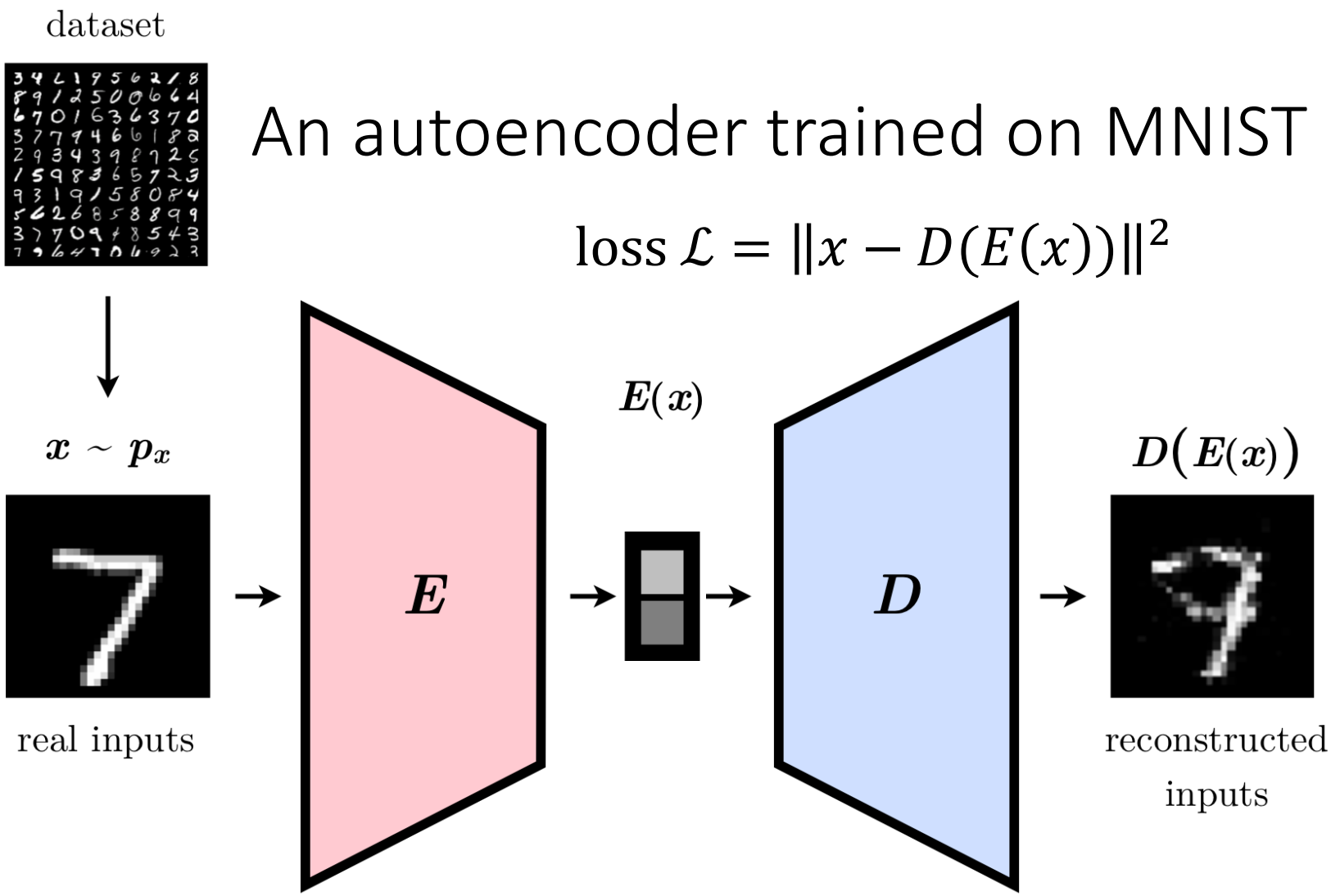
The MNIST dataset

- Famous dataset of handwritten digits
- They trained an autoencoder with a 2-dimensional bottleneck
- This makes visualizing the embedding space easy



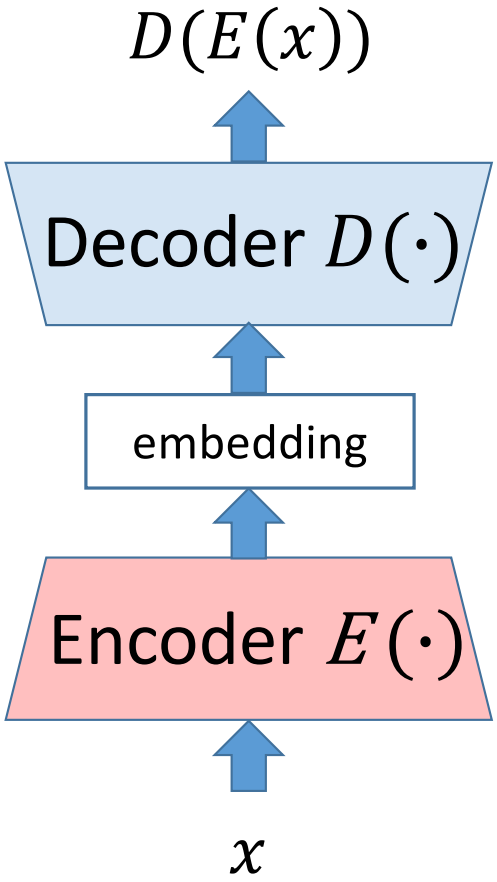
https://emkademy.medium.com/1-first-step-to-generative-deep-learning-with-autoencoders-22bd41e56d18

An autoencoder trained on MNIST



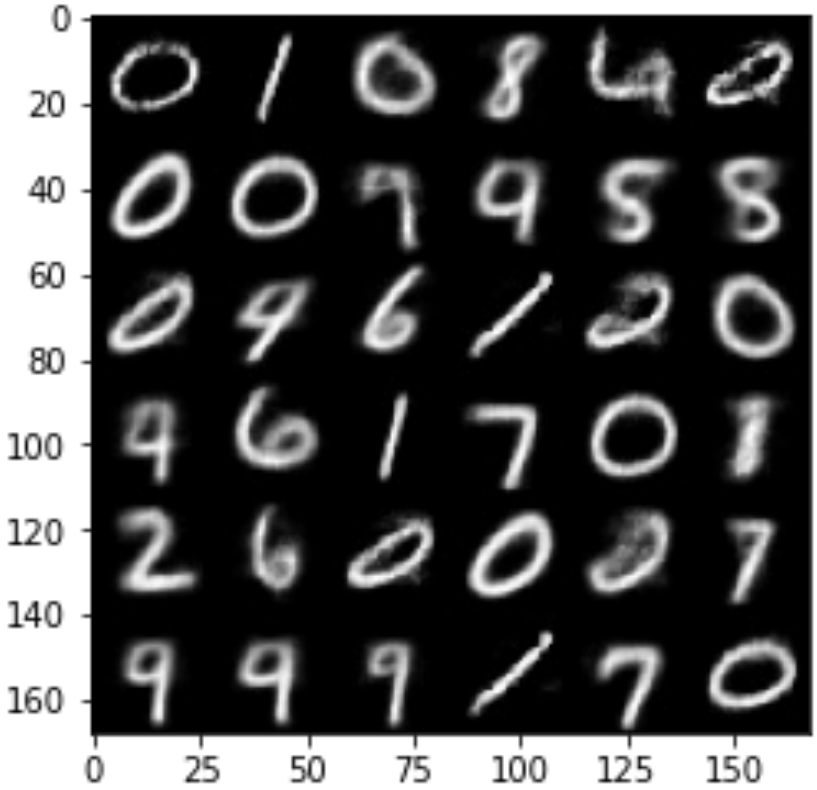
Generating new digits…

- Train the whole system,
- Dump the encoder
- Pick a vector of values to hand the decoder.
- See what you generate



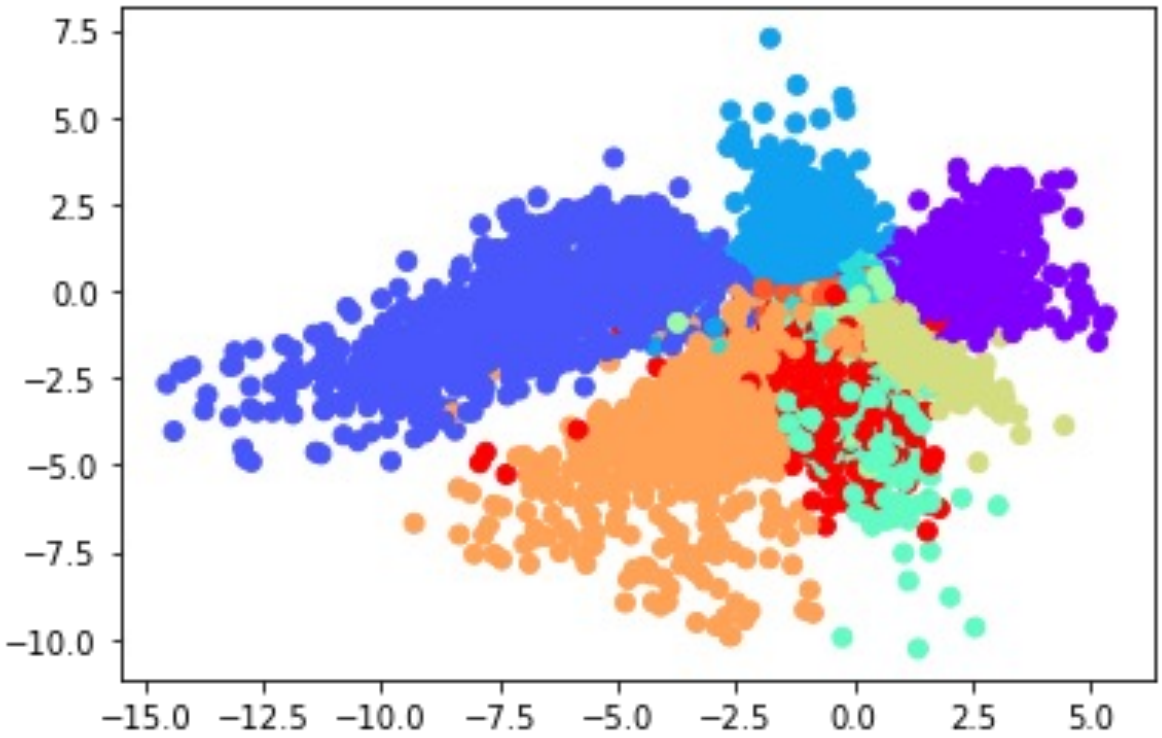
Generating images from embedding vectors

- It is difficult to know what will happen when you move in a direction
- There are spots in the space that are…well…weird.
- Where's the 3?



https://emkademy.medium.com/1-first-step-to-generative-deep-learning-with-autoencoders-22bd41e56d18

2-D embedding space of the trained autoencoder



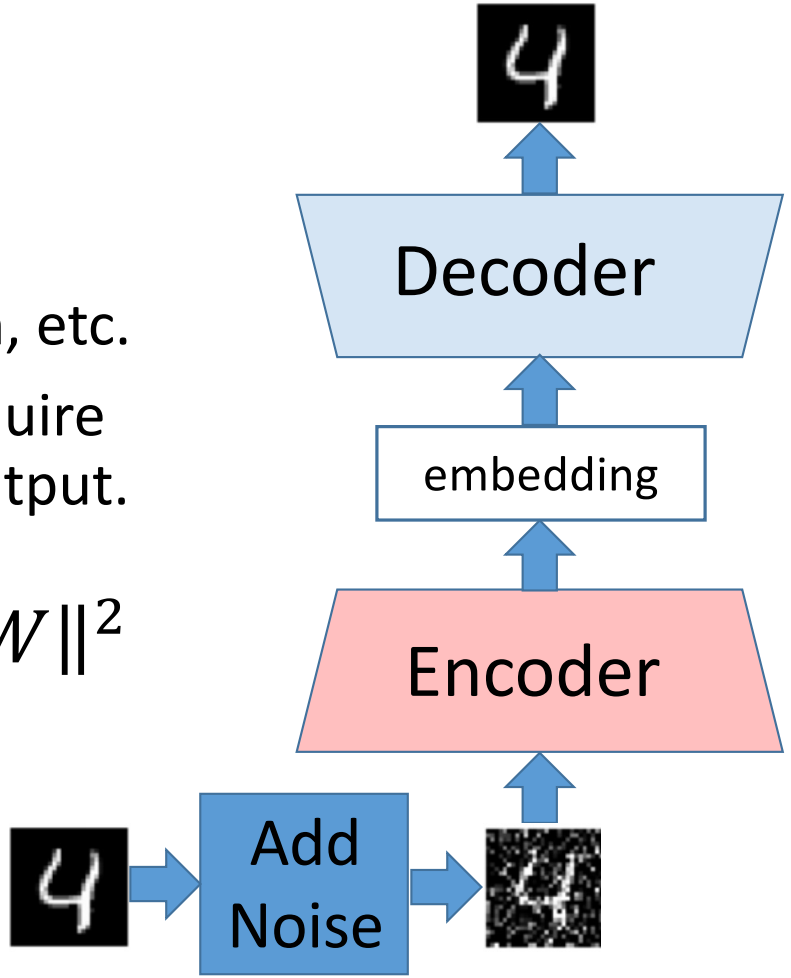
https://emkademy.medium.com/1-first-step-to-generative-deep-learning-with-autoencoders-22bd41e56d18

Good for denoising!

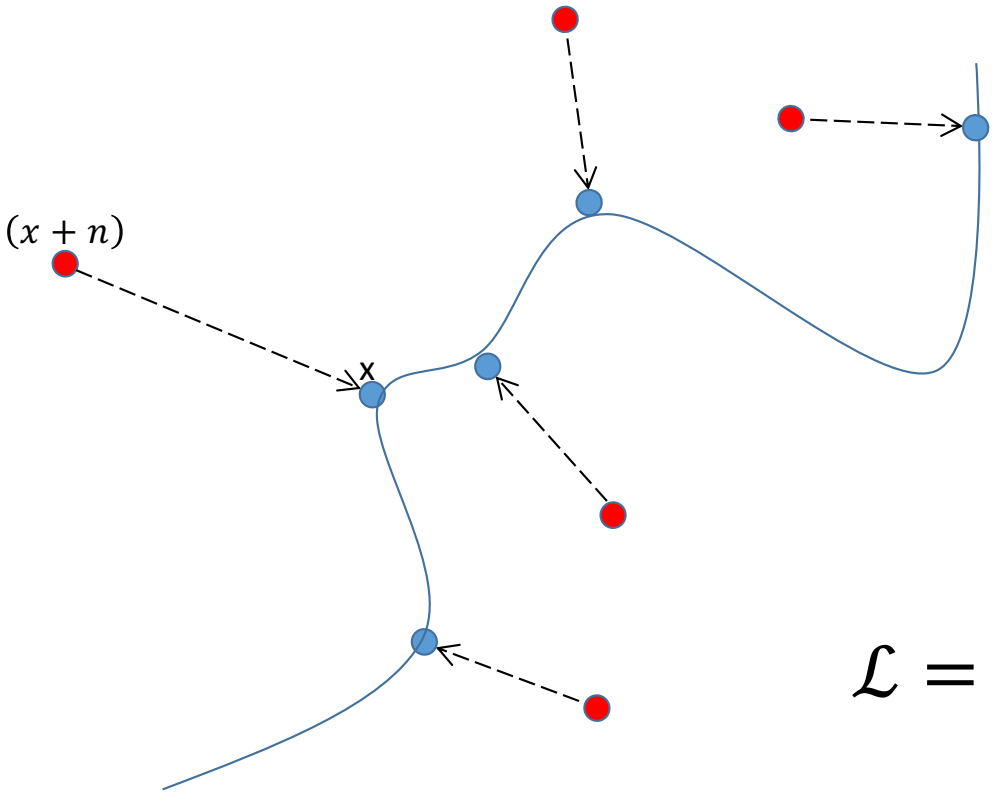
- Autoencoders can be trained to remove noise from images, speech, etc.
- Just add noise to the input and require reconstruction of the non-noisy output.

$$\mathcal{L} = \|x - D(E(x + n))\|^2 + \|W\|^2$$

- This is a denoising autoencoder.



DAE: Maps to a learned non-linear embedding

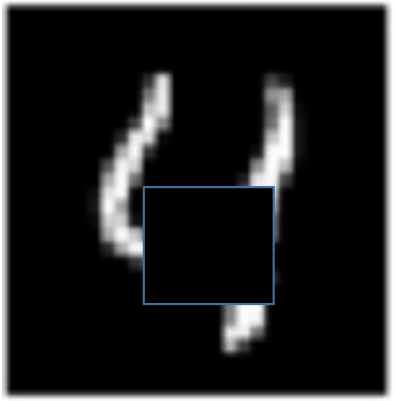


Same idea as PCA, in some sense, but nonlinear….

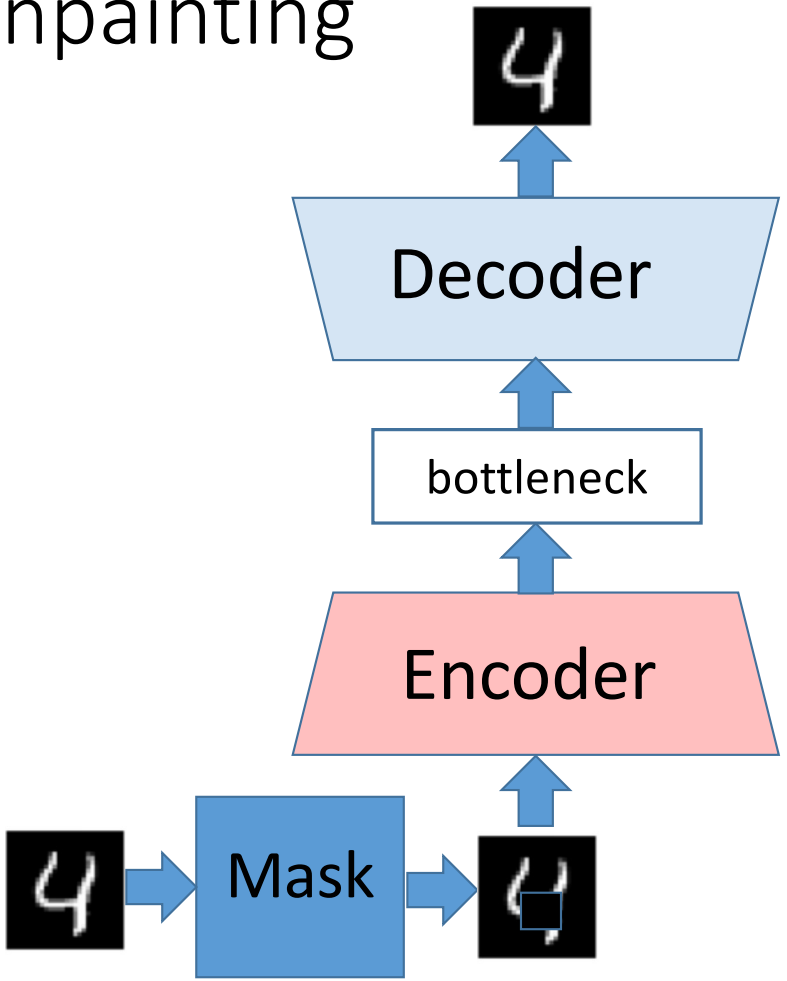
$$\mathcal{L} = \|x - D(E(x + n))\|^2 + \|W\|^2$$

Also great for imputation/inpainting

- If the “noise” we add is masking out large patches….



- We can train it to fill in blanks.



Sparsity constraints are good

- Often better data representations can be gotten by adding a sparsity constraint to the loss function.

$$\mathcal{L} = \|x - D(E(x))\|^2 + \|W\|^2$$

Here W is the network weights.

- This is a sparse autoencoder.