R250 Advanced topics in machine learning

Topic 5: autoencoders

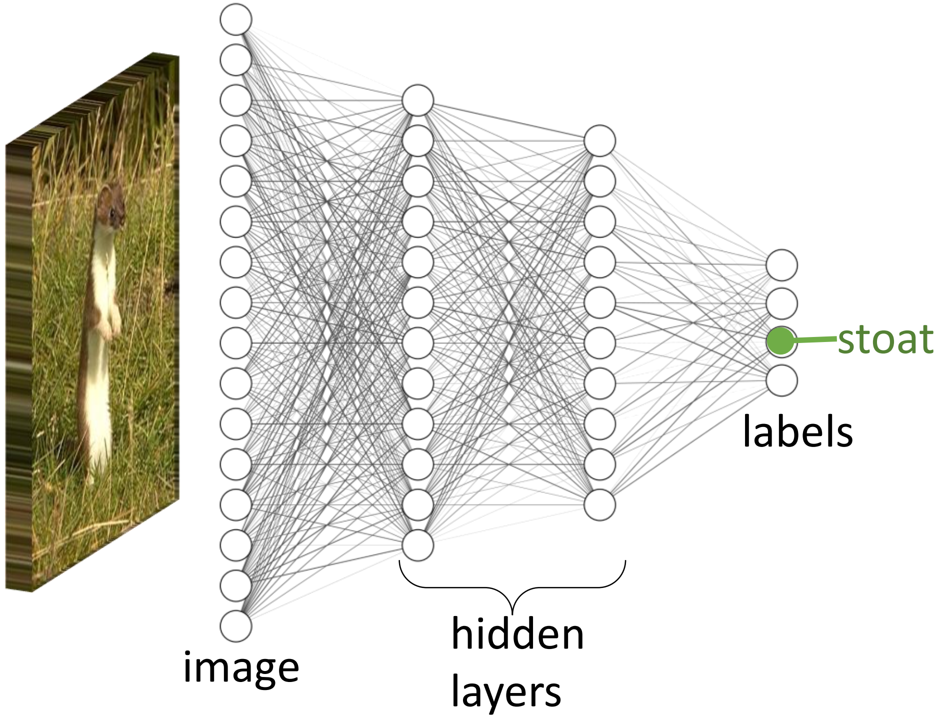
Damon Wischik

What is an autoencoder?

A classifier

Input: labelled data $(X_n, Y_n)_{n=1..N}$

Task: predict the output Y given input X

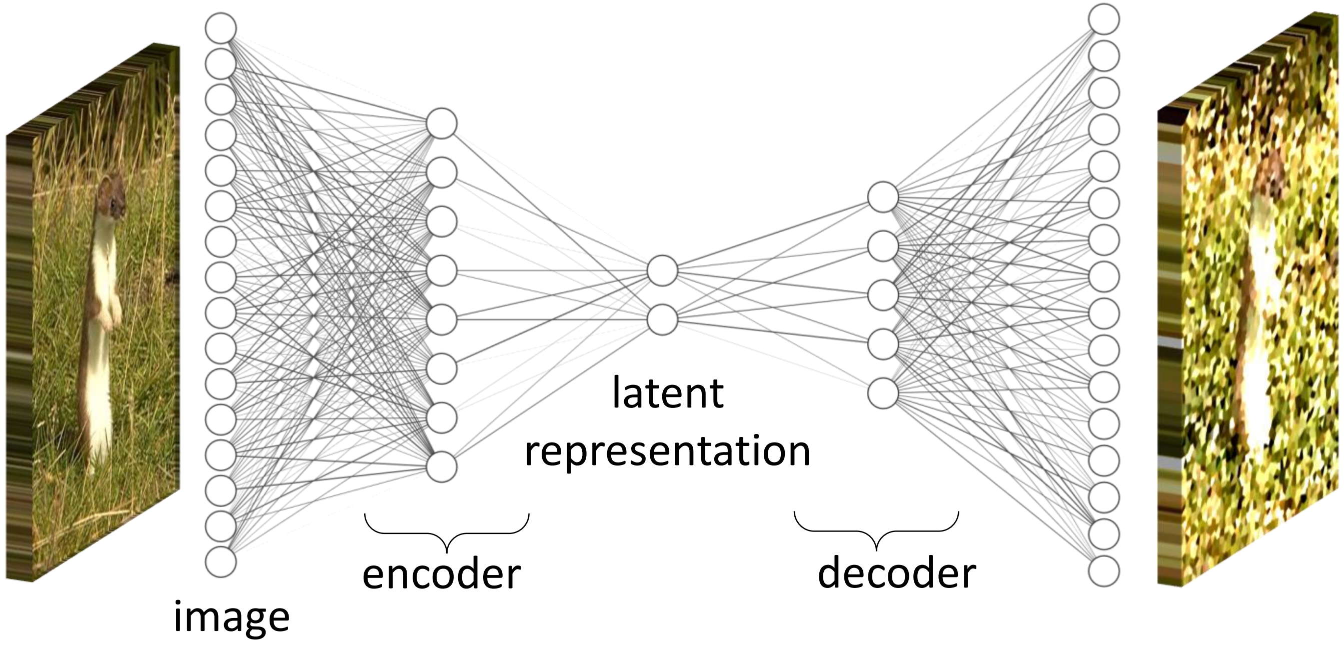


An autoencoder

Input: unlabelled data $(X_n)_{n=1..N}$

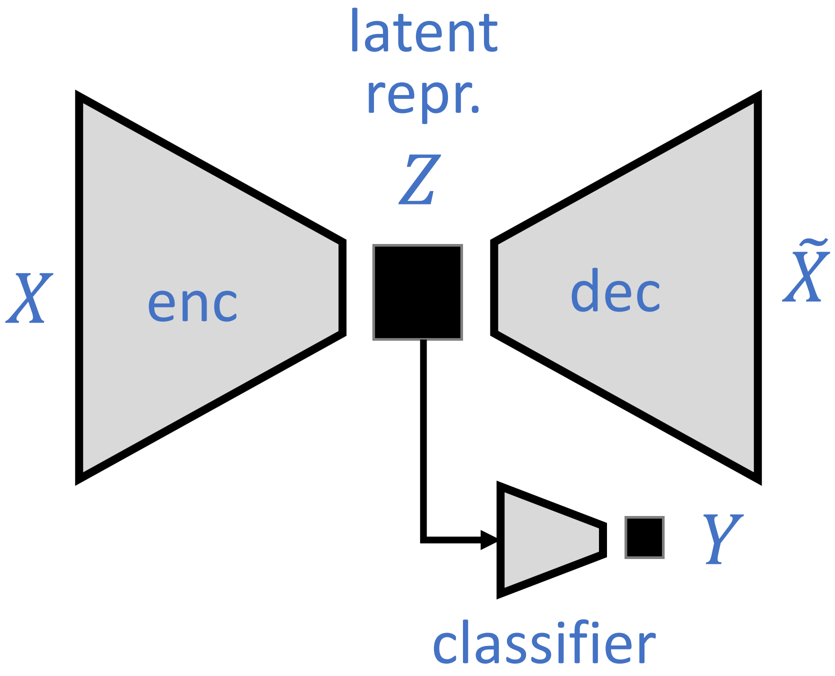
Task: given an input, reconstruct it

Challenge: squeeze the data through a “bottleneck”



What's the point in learning to recreate the input?

It can help with multitask / transfer / semi-supervised learning.

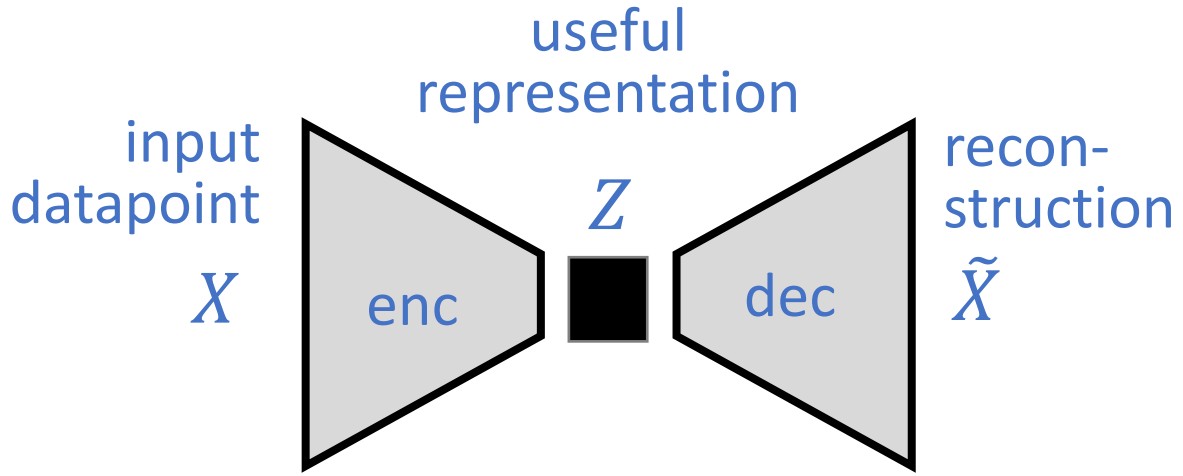


Train a neural network with two objectives:
(a) output the target label Y
(b) reproduce the input

- This is useful if labels are low entropy
  e.g. sentiment classification of text.
  The "reproduce the input" objective (b) gives extra feedback, which helps backpropagation learn useful features.
- It's also useful if you have lots of unlabelled data and only a little labelled data.

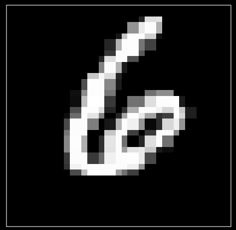
The heart of autoencoding

We hope it will learn a useful / meaningful latent representation.



Surely, if it didn't learn a good representation, it'd have no chance of reconstructing the input from just a few variables!

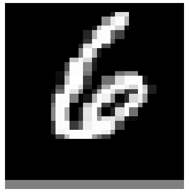
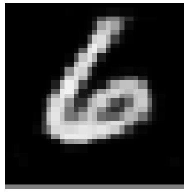
MNIST image



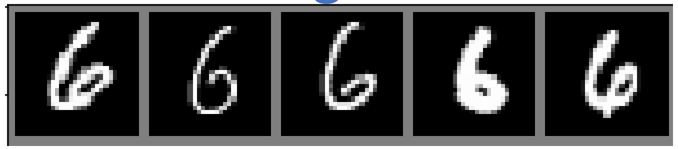
A 4-dimensional representation

```
{'digit': 6,
 'slant': UPRIGHT,
 'weight': MEDIUM,
 'style': LOOSE}
```

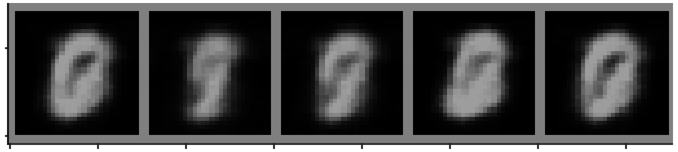
What sort of representations does it actually learn?

MNIST image	4-dimensional representation	recon-struction
	$\begin{bmatrix}1.4400\\1.5164\\0.3757\\3.2569\end{bmatrix}$	

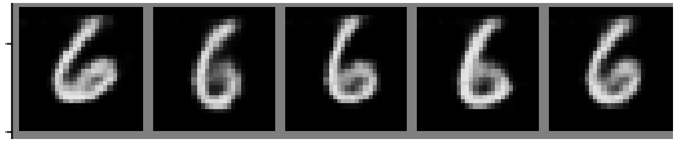
Source images



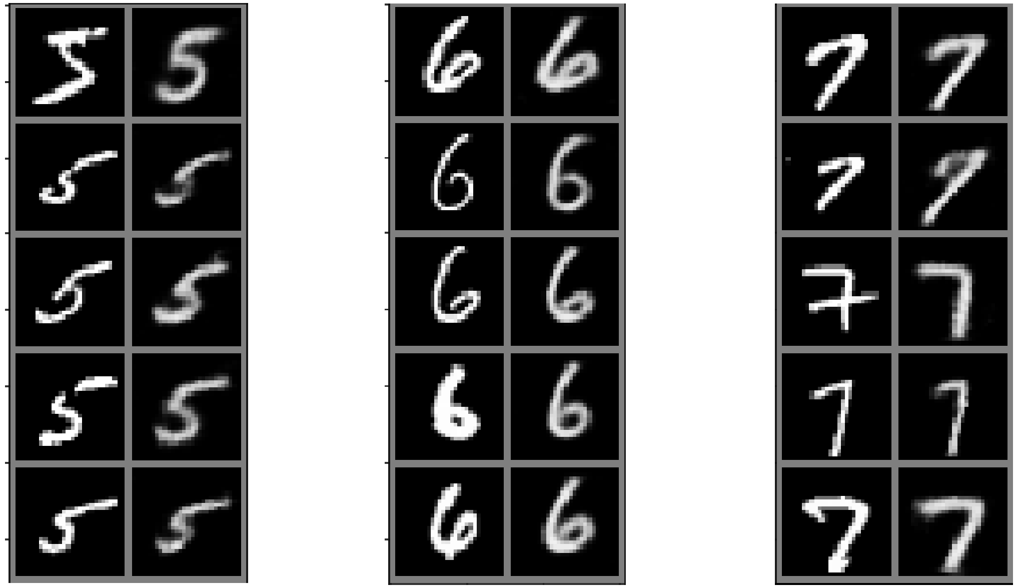
Reconstructions after 0.1 epochs

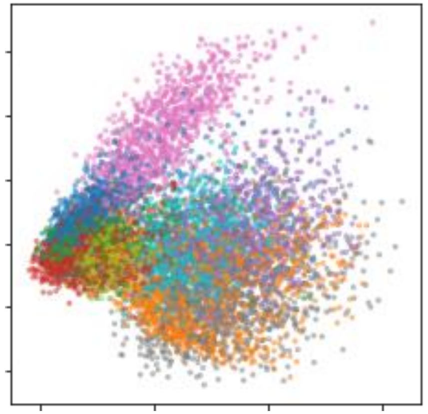


Reconstructions after 2 epochs



Reconstructions after 3 epochs



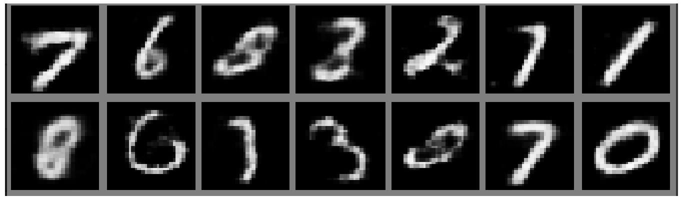


PCA plot showing the latent representations

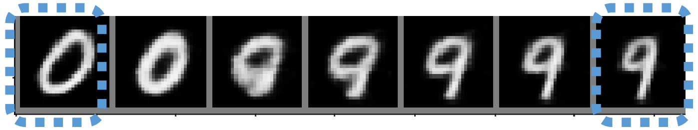
colour = true digit

If we had a good representation, we could ...

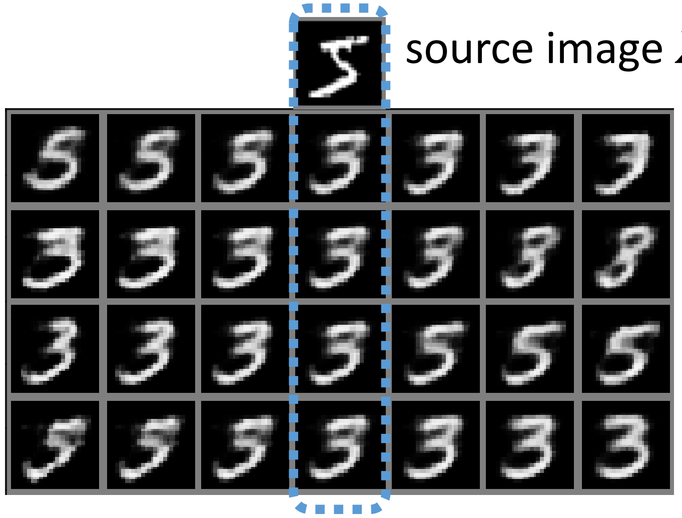
- Pick a random Z, and decode.
  This should let us synthesize entirely new images.



- Take two source images X_1 and X_2, encode to get Z_1 and Z_2, let $Z = (1-\lambda)Z_1 + \lambda Z_2$, and decode Z.
  This should generate a smooth interpolation between the two inputs, where each intermediate looks "nice".



- Take a source image X, encode it to get Z, then vary the "digit" field of Z and decode.
  This should give a family of digits with the same handwriting.



Autoencoders are a tool for dimension reduction

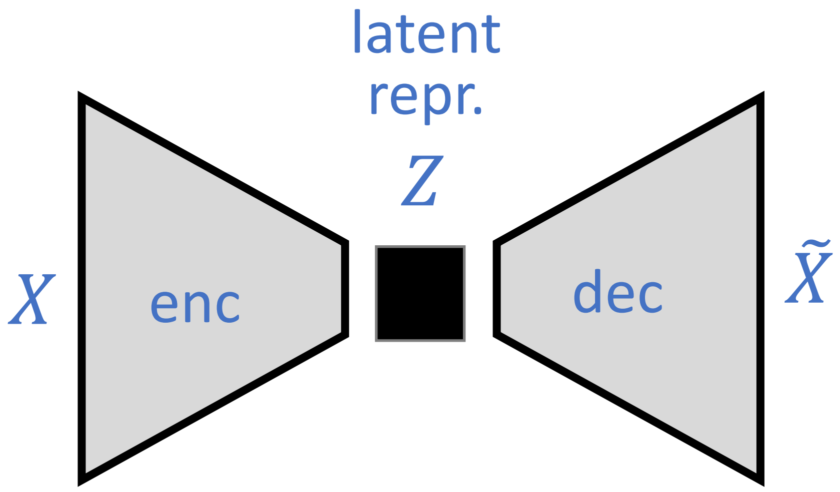
- It’s **easier to train** a supervised learner from dimension-reduced features than from the raw dataset
- The reduced dimensions are **meaningful axes** for our dataset; this is useful for interpolation etc.
- We can **synthesize new data**, by sampling randomly in the reduced-dimension space.

None of this works well off-the-shelf
(hence the papers we will study).

And in fact the entire premise is dodgy.

We haven’t specified a proper evaluation criterion. Without this we can’t compare models, or tune hyperparameters; we’re just blindly hacking.

How should we validate an autoencoder? A thought experiment...



Input: unlabelled data $(X_n)_{n=1..N}$
Reconstruction loss metric: $L(X, \tilde{X})$

- In training, the aim is to minimize the reconstruction loss $\mathbb{E}_{X\sim\text{train}}\ L(X, \tilde{X})$
- The obvious way to validate is to run the network on unseen data (the holdout / validation dataset), and measure the reconstruction loss $\mathbb{E}_{X\sim\text{test}}\ L(X, \tilde{X})$
- But consider a super-intelligent autoencoder, which has learnt to encode input pixel i into bit i of the latent variable $Z \in \mathbb{R}$. This autoencoder is surely not what we want — but it will score perfectly.

Autoencoders are a tool for dimension reduction

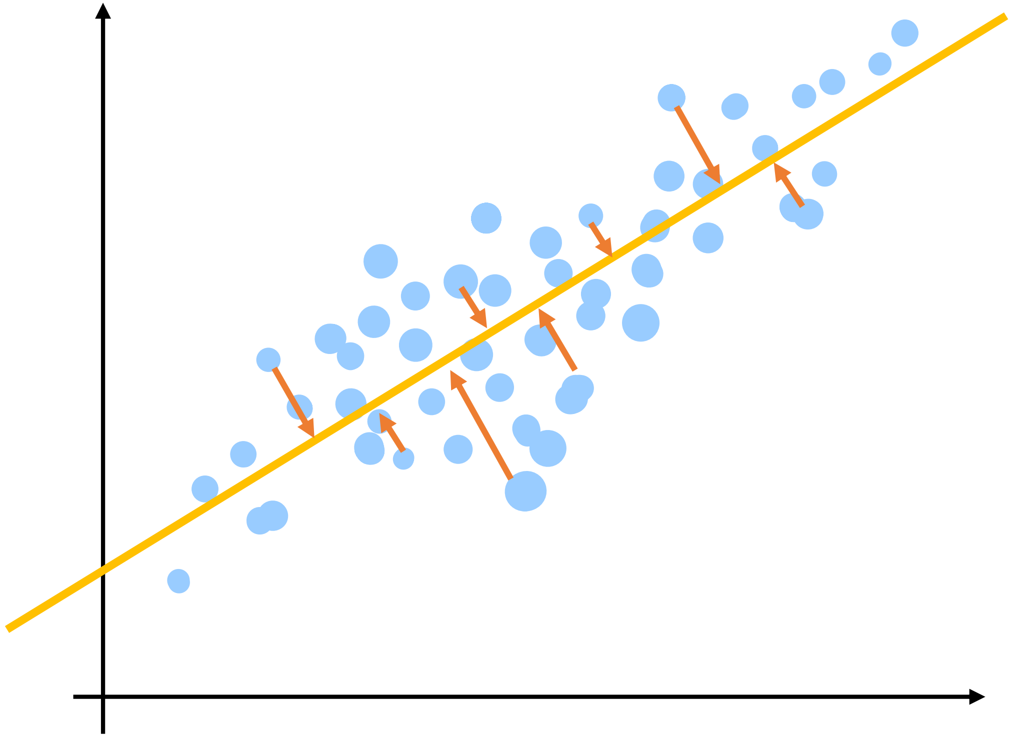
- It's **easier to train** a supervised learner from dimension-reduced features than from the raw dataset
- The reduced dimensions are **meaningful axes** for our dataset; this is useful for interpolation etc.
- We can **synthesize new data**, by sampling randomly in the reduced-dimension space.

None of this works well off-the-shelf
(hence the papers we will study).

Just like PCA!

Does PCA give us any insight into the problem of validation?

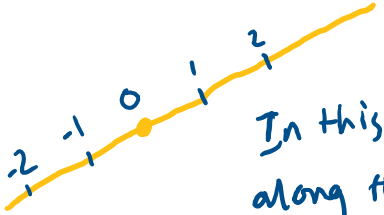
Principle Components Analysis



Given a collection of points $X_1, \dots, X_N \in \mathbb{R}^d$
PCA looks for a linear subspace of dimension $e < d$ to represent the data.

PCA *is* an autoencoder.

- It encodes $X \in \mathbb{R}^d$ into $Z \in \mathbb{R}^e$
- The decoder positions the linear subspace $\mathbb{R}^e$ within $\mathbb{R}^d$
- PCA seeks to minimize mean square error

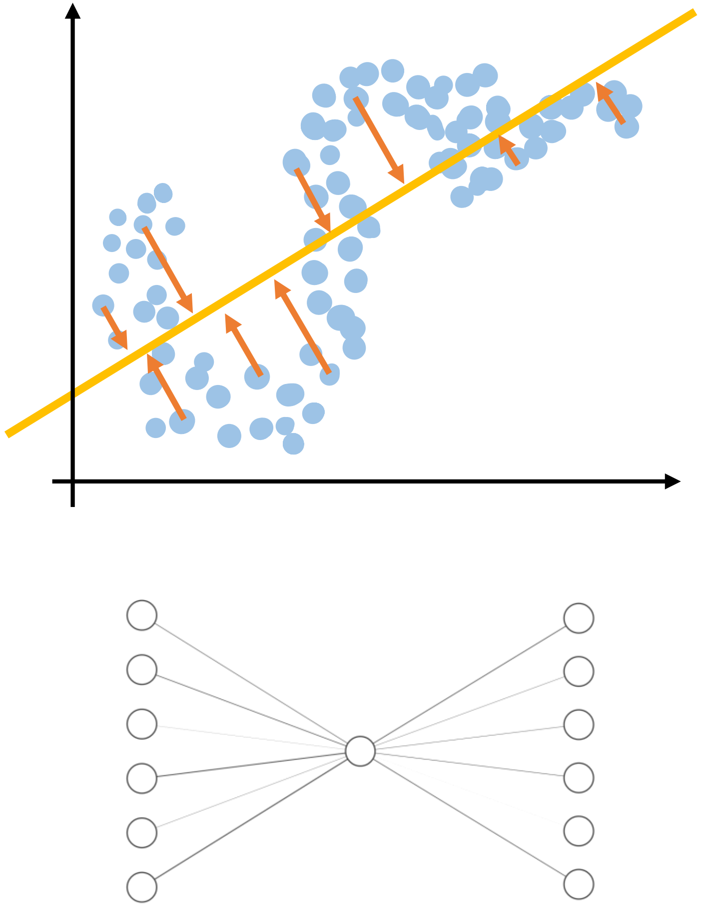


In this picture, $z \in \mathbb{R}$ measures how far along the line to go, from some fixed reference point.

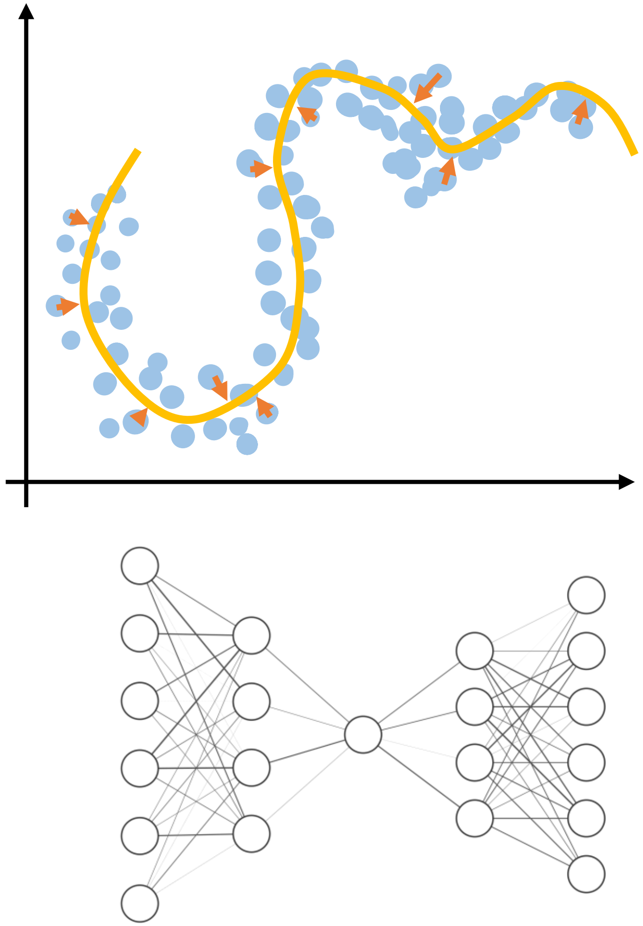
This picture depicts dimension reduction from $\mathbb{R}^2$ to $\mathbb{R}^1$.

- *With $e = d$ we'd get perfect reconstruction (but no dimension reduction)*
- *There are hacks to pick a useful $e < d$ …*

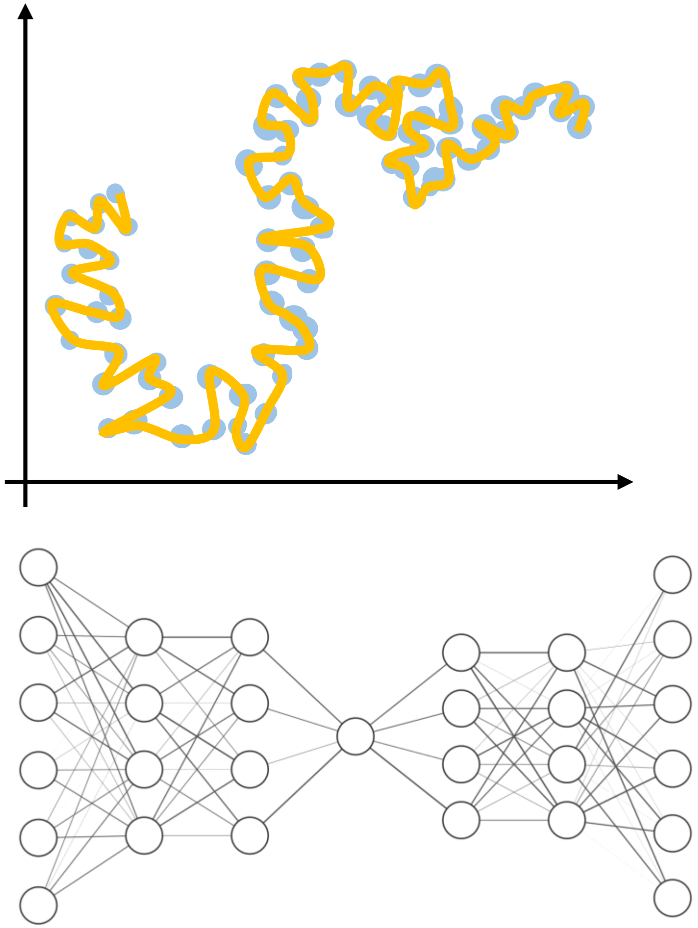
The Goldilocks problem



PCA only looks for linear subspaces. It is incapable of overfitting (as long as $e < d$).

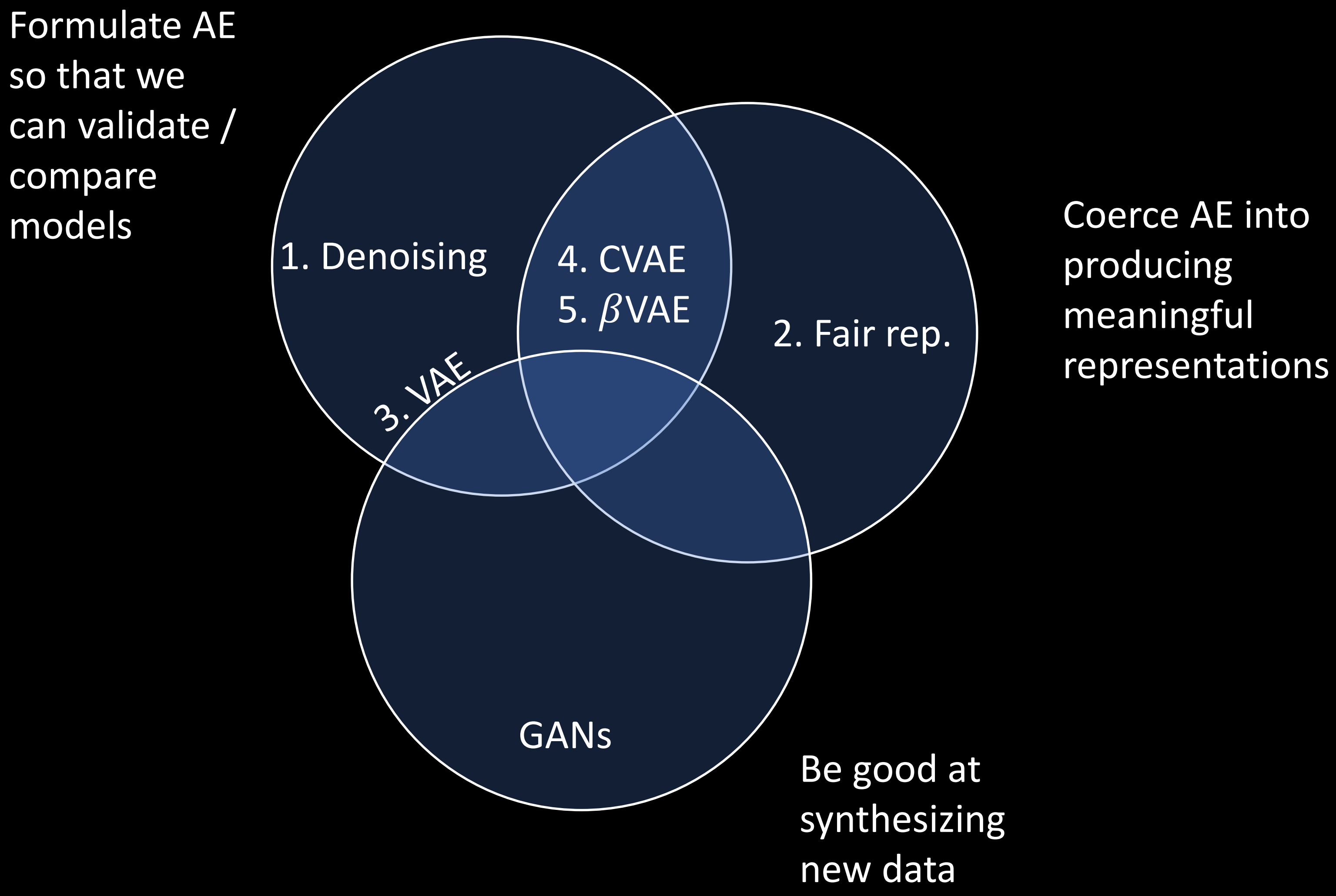


If we allow nonlinear enc and dec, surely we can describe the data better.



Too much capacity → overfitting.

In the story of autoencoders, there are three overlapping challenges.



Schedule

20 January (1 hour)	Introduction
27 January (1 hour)	1. Denoising AEs 3a. VAE
3 February (1 hour)	2. Fair representations 3b. VAE
10 February (2 hours)	4. Conditional VAE 5. β-VAE 6. VAE+RNN ? Project report ideas

Assessment

- participation ≈ 5%
- presentation ≈ 15%
- project report 70%

Presenters, please chat with me the Friday before your presentation.

You should *all* read the papers, try the code, and participate in the discussion.

(Please introduce yourself. I'll record for marking purposes.)

Arrangements

There is a Cambridge Gitlab repository for this course, with a toy MNIST example in Pytorch.

Presenters will contribute working code.

Participants should also contribute issues / pull requests / code. (This gives you participation marks.)