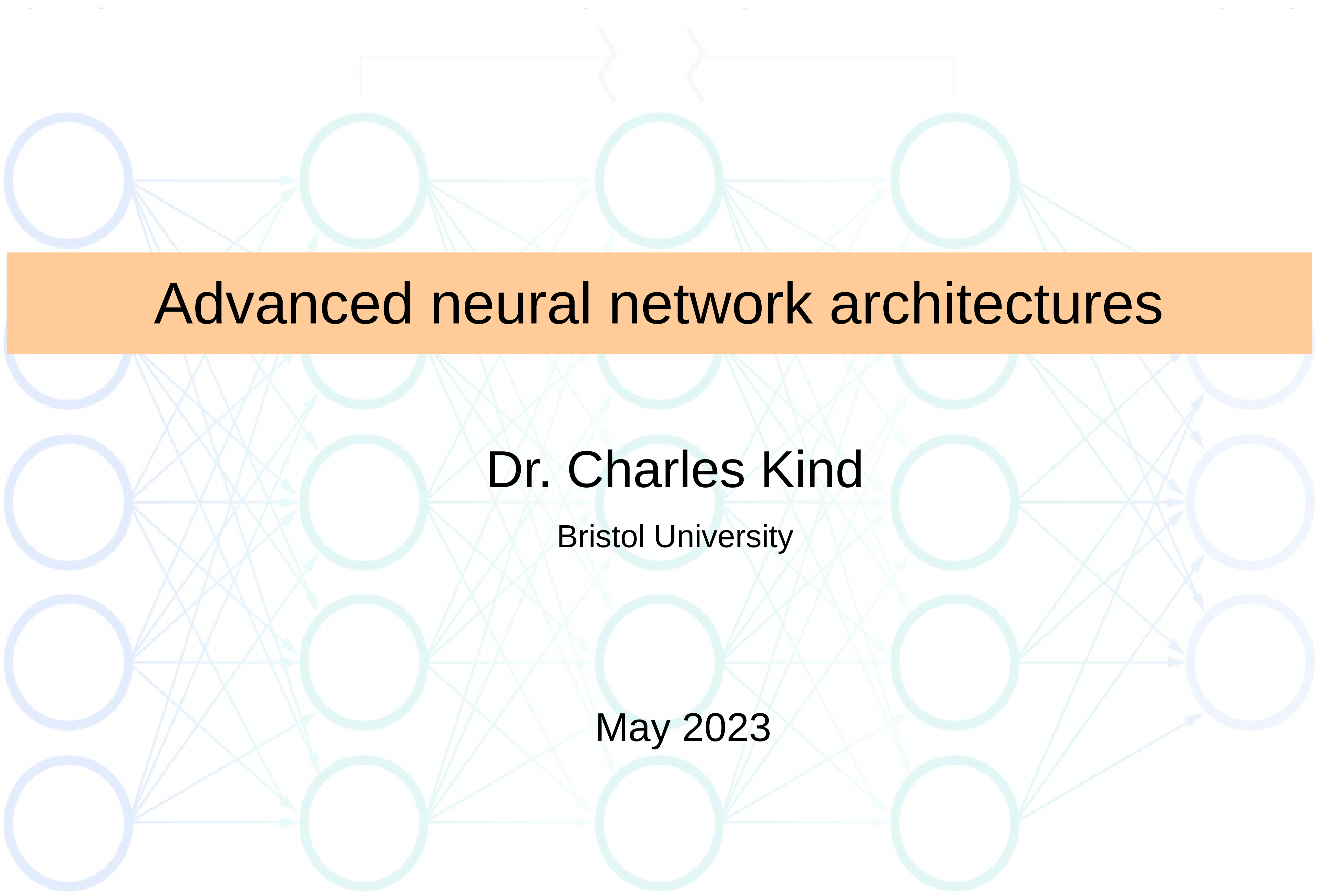


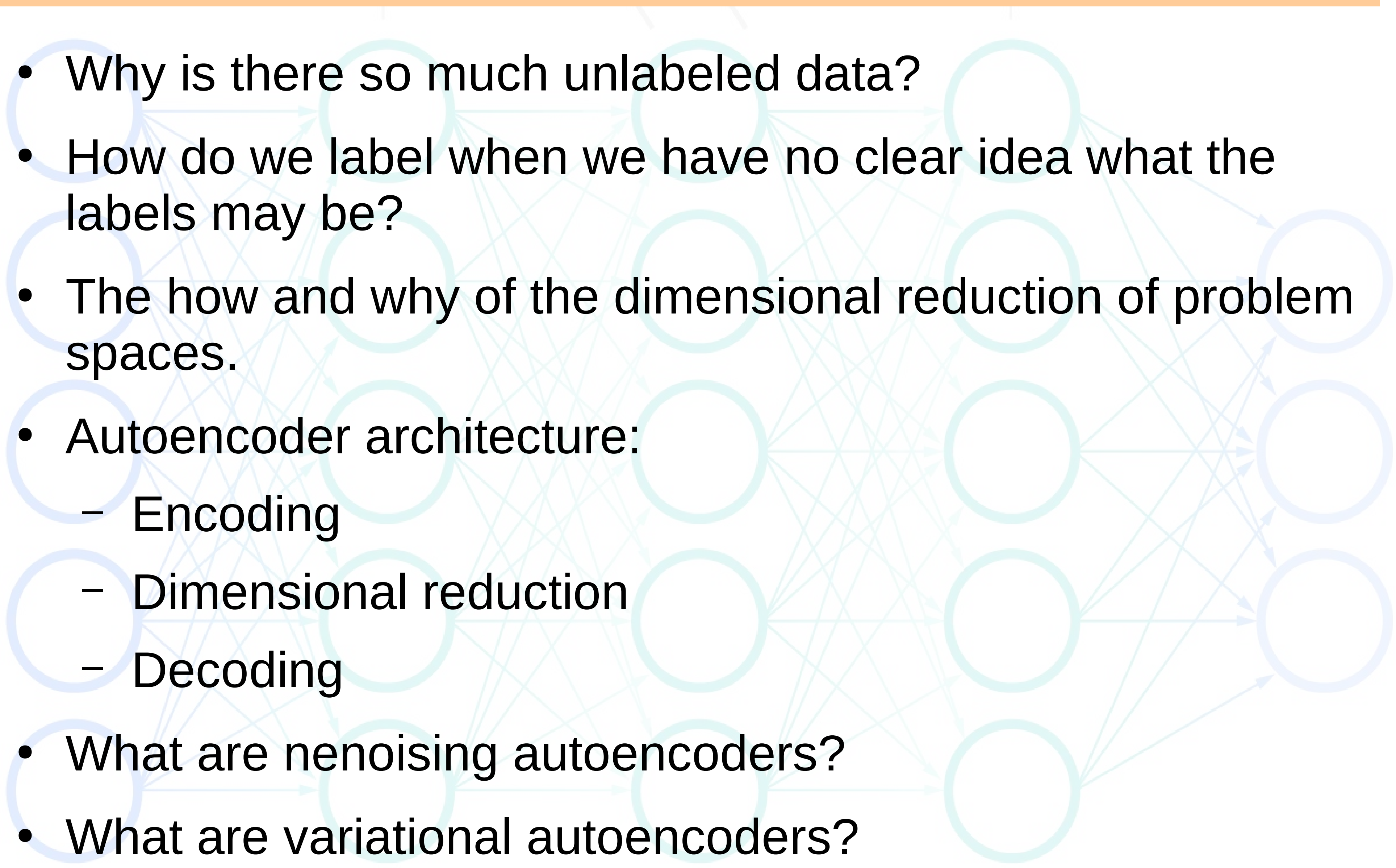
Advanced neural network architectures

Dr. Charles Kind

Bristol University

May 2023

Learning objectives



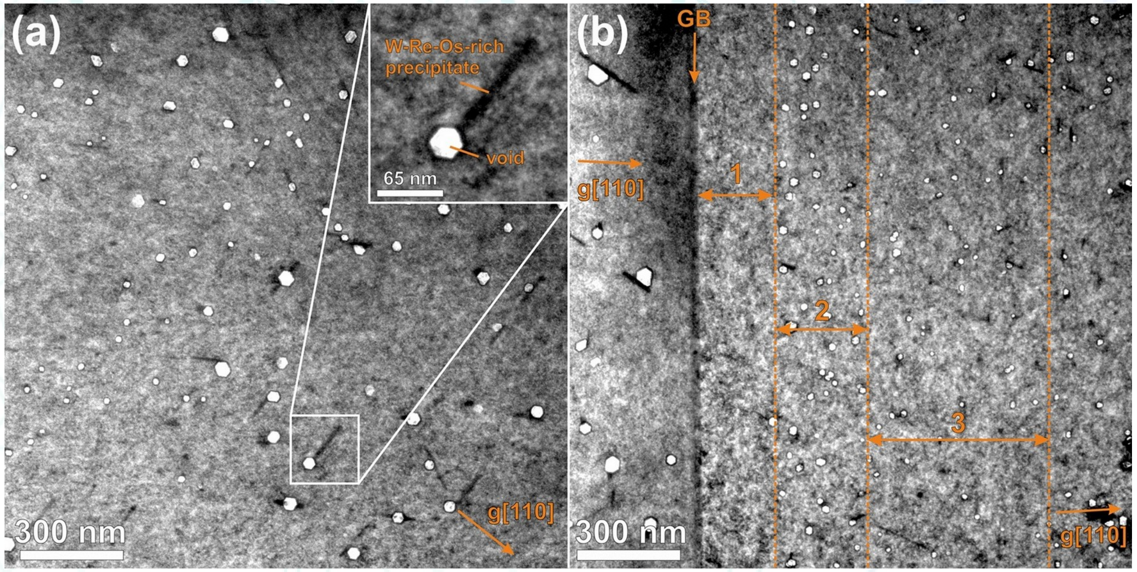
- Why is there so much unlabeled data?
- How do we label when we have no clear idea what the labels may be?
- The how and why of the dimensional reduction of problem spaces.
- Autoencoder architecture:
  - Encoding
  - Dimensional reduction
  - Decoding
- What are nenoising autoencoders?
- What are variational autoencoders?
- What are generative models?

Neural networks need data!

- It seems an obvious statement that neural networks need data!
- If we want to train a DNN to analyse stock market price changes we need historic and current data eg:
  - Tick data for prices
  - Short and long term data
  - Variability analysis
  - Fundamentals such as earnings reports or current events
  - Graphs if we want to use convolutional networks
- If we want to train a DNN to recognise defects in structural materials
  - Many labelled images of various types of defect in the materials we are interested in
  - Many labelled images of “perfect” materials
  - Information on load, stress, strain, decay, radiation etc
- What do we do if we have unlabelled data?

Neural networks need data!

- What do we do if we are not sure what features of our data are important?
- What can we do if we have unlabelled data and we want to somehow classify it?
- There are vast numbers of huge datasets available.
- There is a strong movement to incorporate DNN into the physical sciences to extract new information and model poorly understood mechanisms.



Neutron irradiated tungsten, there are huge datasets enumerating the various physical effects of neutron radiation. What hidden understanding may lie within?

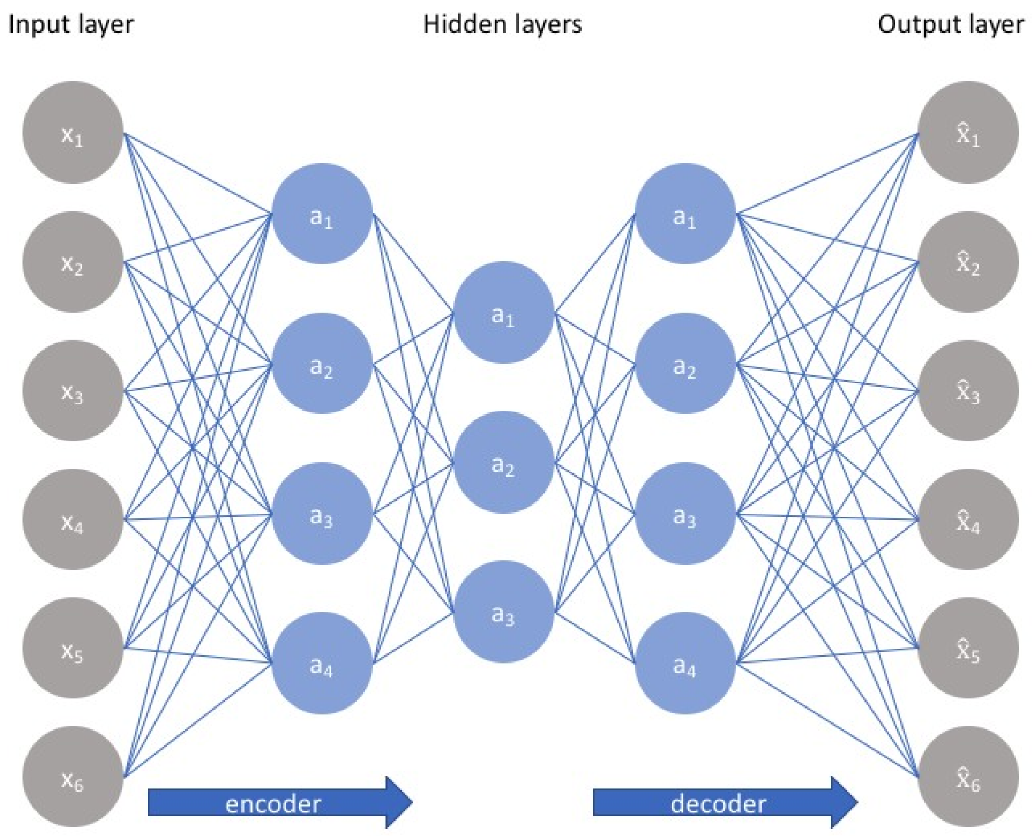
Autoencoders

- An autoencoder does essentially two things:

1) Compresses it’s input data into a lower dimension.

2) Attempts to use the lower dimensional representation to reconstruct the original input.

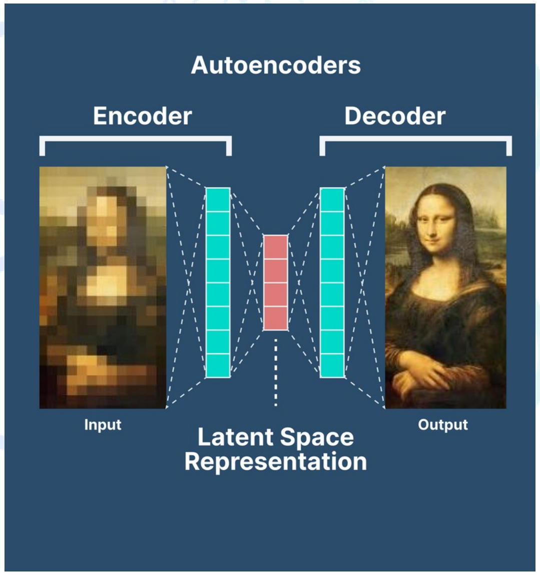
- The difference between the two is called the reconstruction error.
- This error is the cost function of this network and the autoencoder is trained to minimise it via back-prop.



- What is happening? What is the network learning?

Autoencoders, how they work.

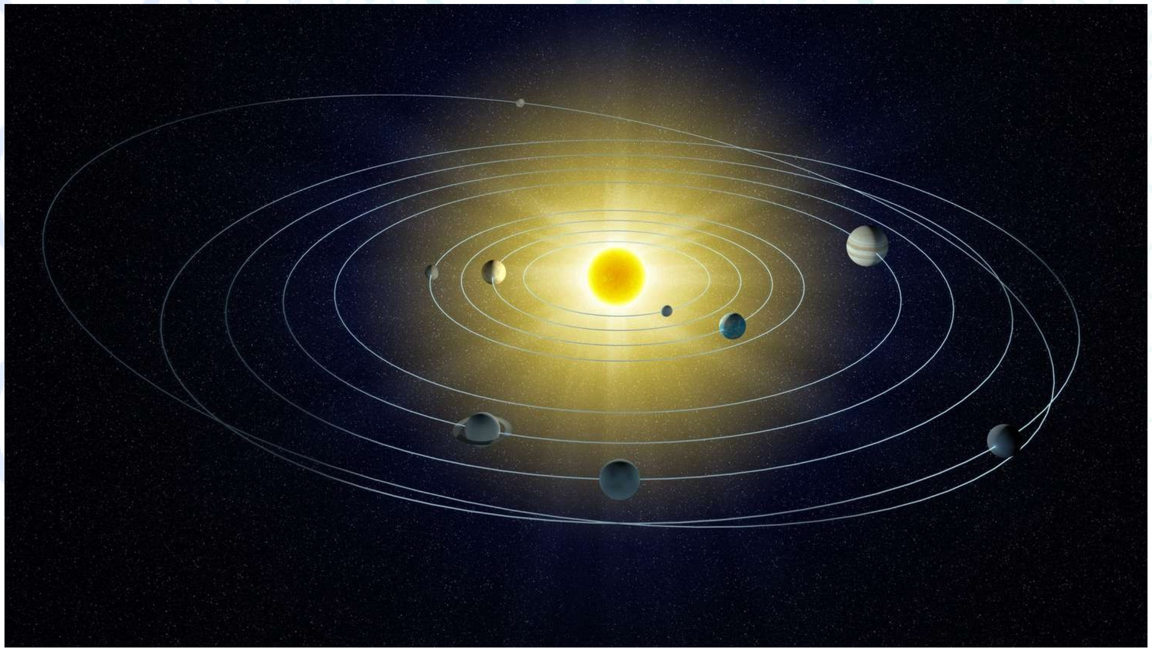
- If we reduce to the absurd, imagine our middle hidden layer is a single neuron.
- If we can reconstruct our original data to within our pre-determined error tolerance we are saying that our data is essentially one dimensional.
- If, on the other hand, we try all reductions of our hidden layers and find we require at least the same number of neurons we can say that our data cannot be dimensionally reduced in any meaningful way.



- If we find a reduced representation then we can use the trained network to reconstruct similar datasets that are, for some reason, sparse or missing elements.
- We could try using the network on data from a different domain and see what pops out of the other end! What could this tell us?

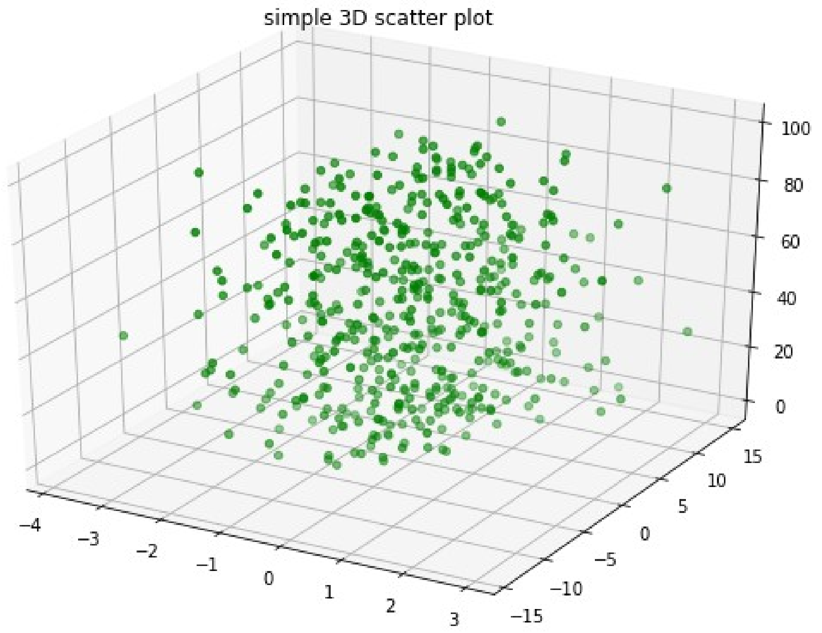
Dimensional reduction

- Many physical systems can be projected down into a lower dimensional space without losing their salient features.
- Consider the orbital mechanics of the solar system.
- We could consider each celestial body in some detail including: physical composition, actual 3d shape, magnetic field, spin etc
- We know, however, that to within a very small margin of error we can just use the centre of mass of the body and calculate from that.



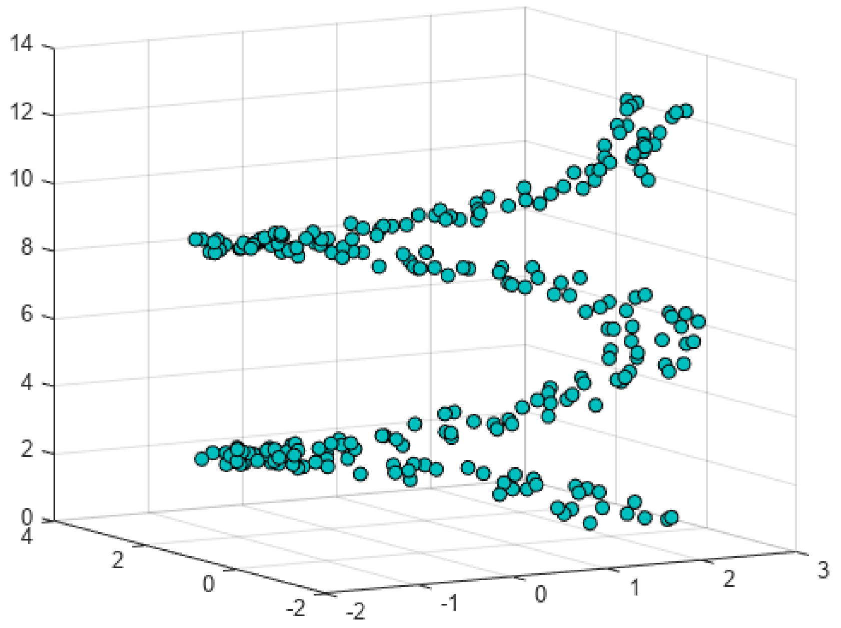
- Extreme accuracy may require some of these additional dimensions
- If there are exceptions in the data these may also require more dimensions.

Dimensional reduction



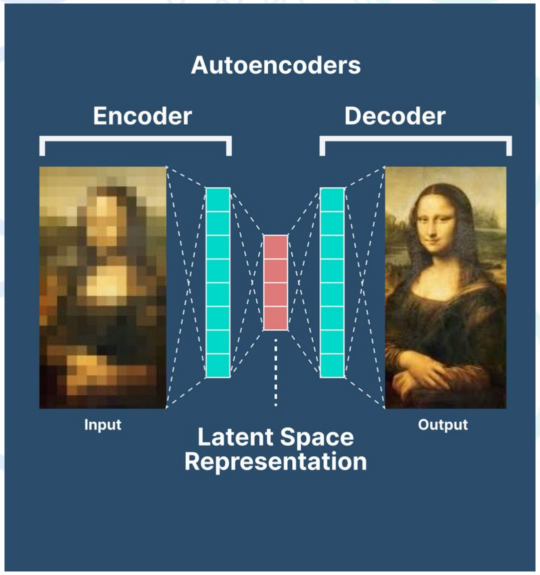
- A random point cloud in three dimensions.
- This data is close to incompressible.
- There is no structure (that we can see).

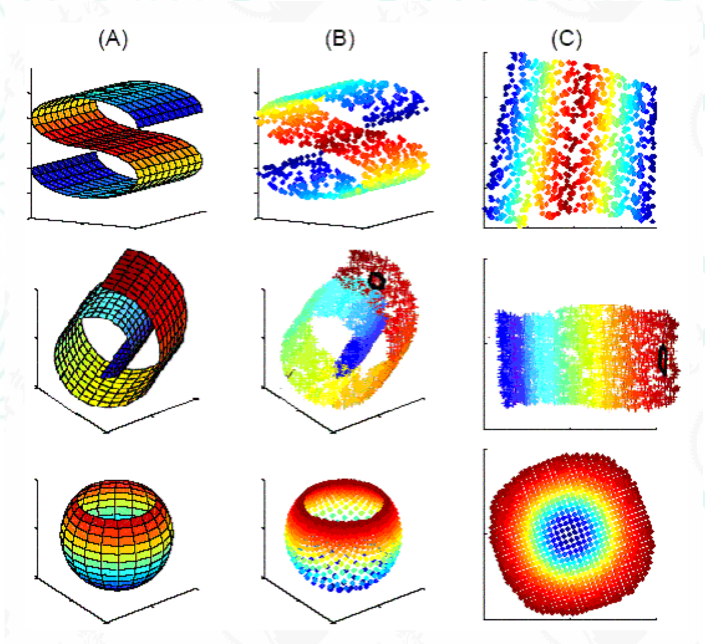
- Structured data has identifiable order, even if you or me cannot see it!
- This data could be represented by a helical function and a small probability distribution.



Autoencoders, how they work.

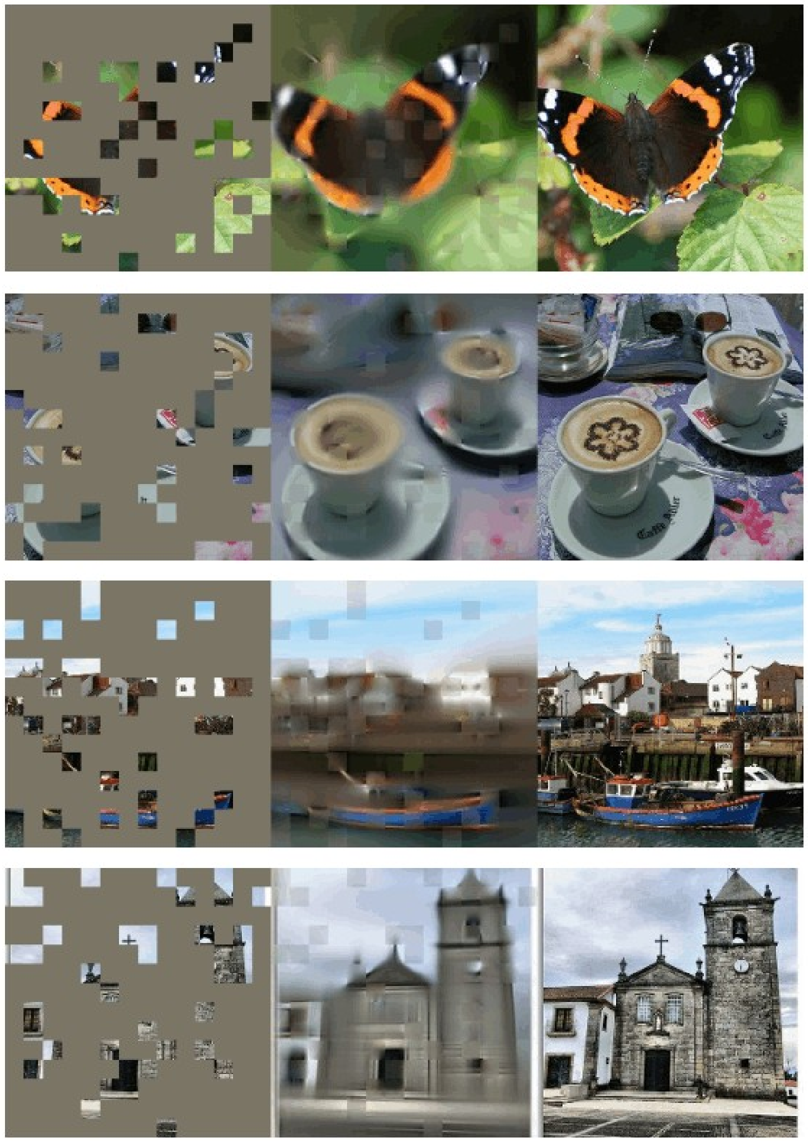
- Real data is usually not random.
- If we have structure then we do not require the entire input space to represent our data.
- Recall the object recognition task and it’s vast high dimensional space. If we where just looking in a very small part of that space, say monotone images we could easily imagine a single (or perhaps three) neuron hidden layer encoding just that colour.





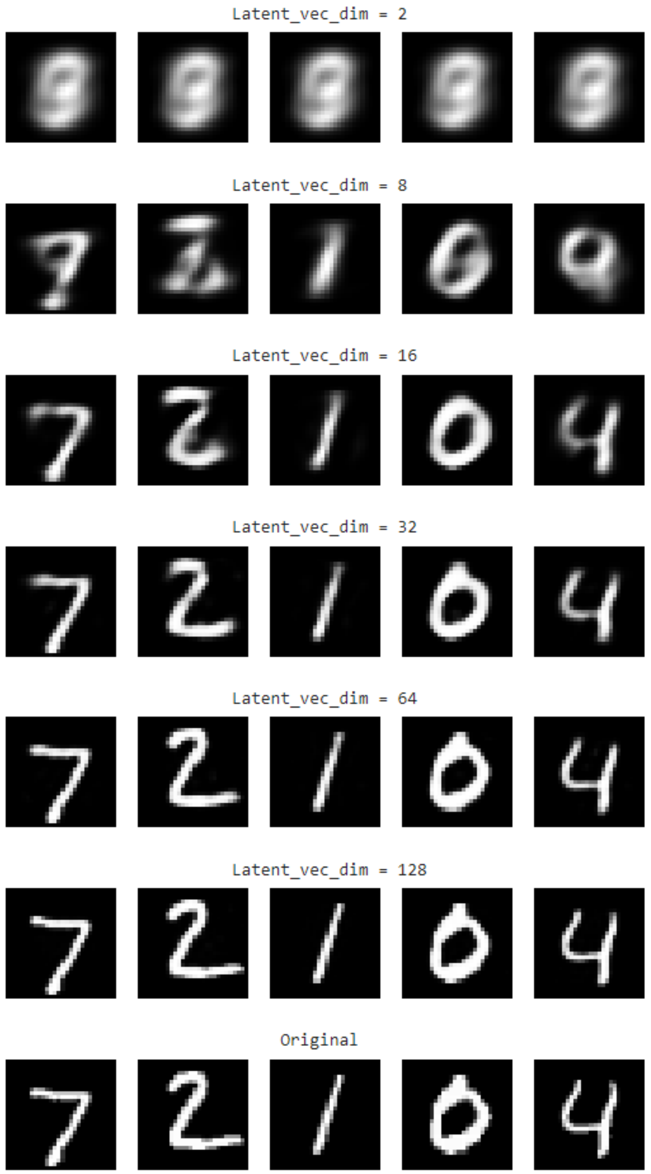
- Low dimensional data.
- These point clouds are points on simply describable manifolds.

Autoencoders, dimension and masking



A trained autoencoder fills masked data and reconstructs images

Different quality reproductions based upon dimensions of smallest hidden layer

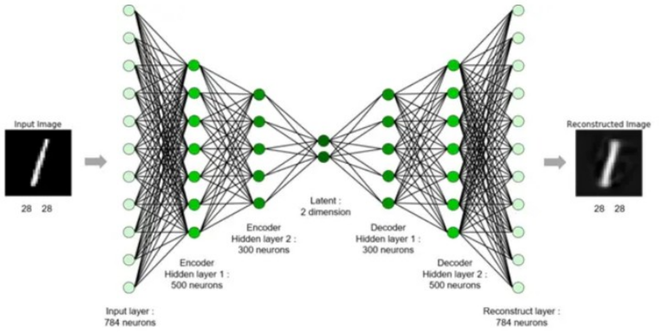


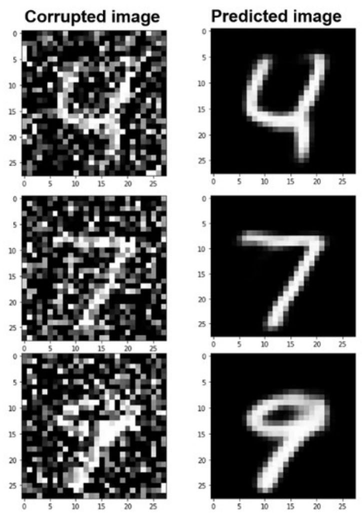
Autoencoders, decoding and denoising.

- Optimally we reduce the dimensions of the data enough to enforce information loss (another rule of thumb or trial end error process).
- The decoder attempts to reconstruct the data from the reduced representation, or latent space as it is known.
- In training the encoder and decoder ‘work together’ to reconstruct the original data to within an acceptable error.
- We need to avoid the architecture that is essentially the identity.
- How can we know that the reason we have an effective reconstruction is due to dimensional reduction with information loss and not just an identity transformation?

Autoencoders, decoding.

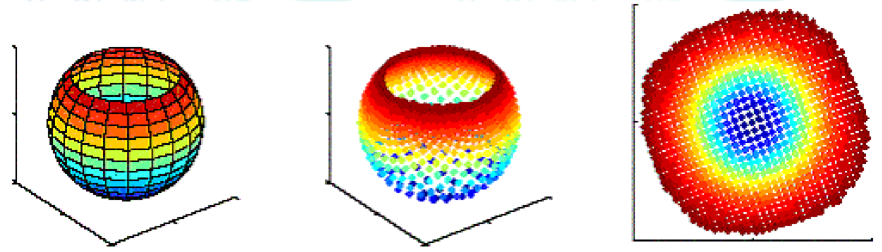
- Optimally we reduce the dimensions of the data enough to enforce information loss (another rule of thumb or trial end error process).
- The decoder attempts to reconstruct the data from the reduced representation, or latent space as it is known.
- In training the encoder and decoder ‘work together’ to reconstruct the original data to within an acceptable error.
- We need to avoid the architecture that is essentially the identity.
- How can we know that the reason we have an effective reconstruction is due to dimensional reduction with information loss and not just an identity transformation?
- We can use a denoising autoencoder.
- Here we add noise to the input and train against the original data without noise.


Input Image
28 28
Input layer : 784 neurons
Encoder Hidden layer 1 : 500 neurons
Encoder Hidden layer 2 : 300 neurons
Latent : 2 dimension
Decoder Hidden layer 1 : 300 neurons
Decoder Hidden layer 2 : 500 neurons
Reconstruct layer : 784 neurons
Reconstructed Image
28 28



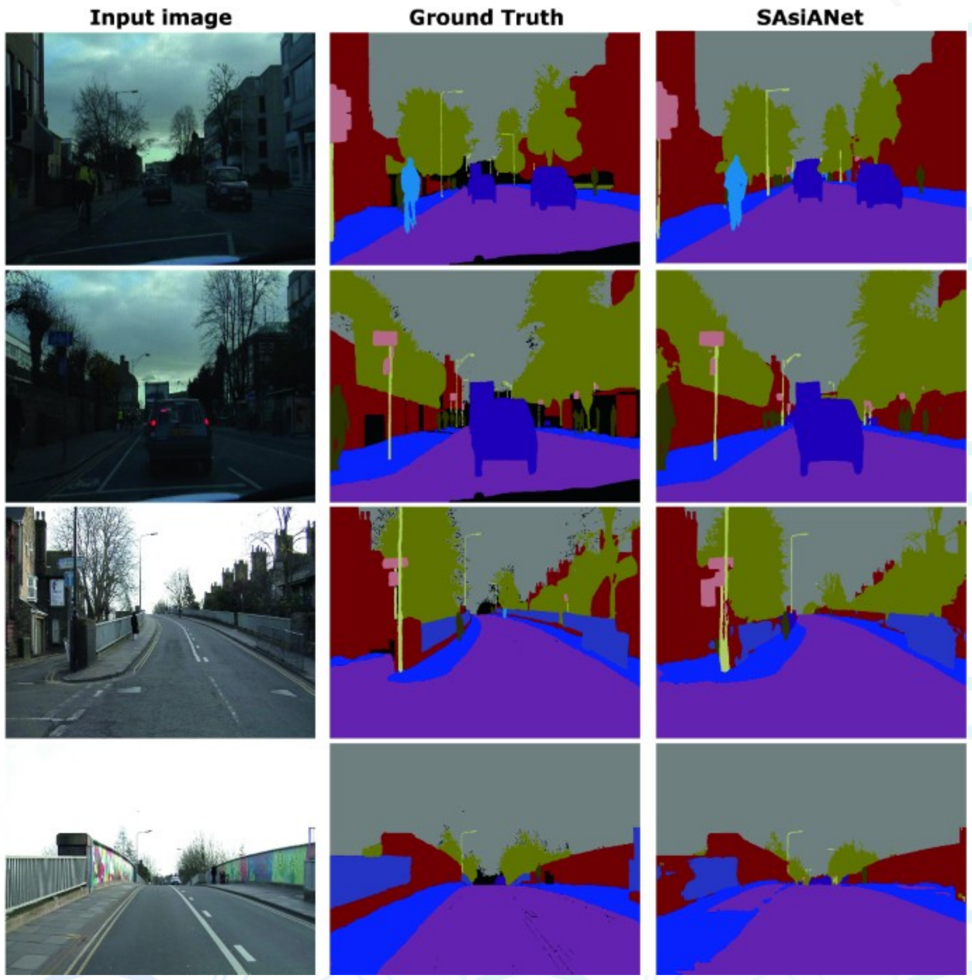
Using autoencoders

- Feature extractor:
  - Remove the decoder after training is complete.
  - Analyse the new data set and you will see some data clustered into distinct areas in the space.
  - You can use clustering or nearest neighbour analysis to extract this information.
  - Particularly useful with categoric data.
- Anomaly detection:
  - Large reconstruction error on anomalous data.

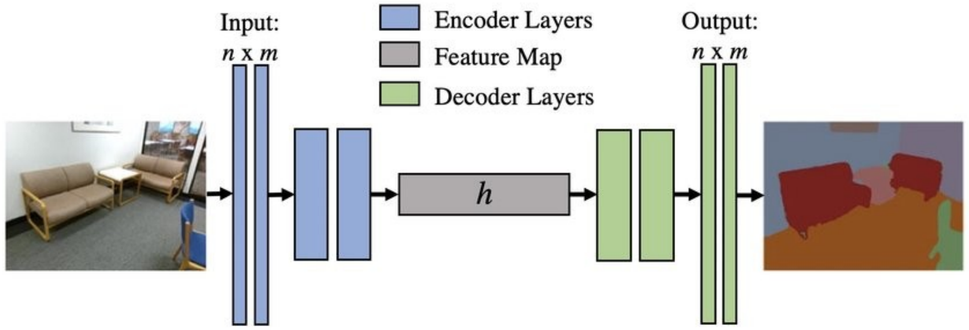


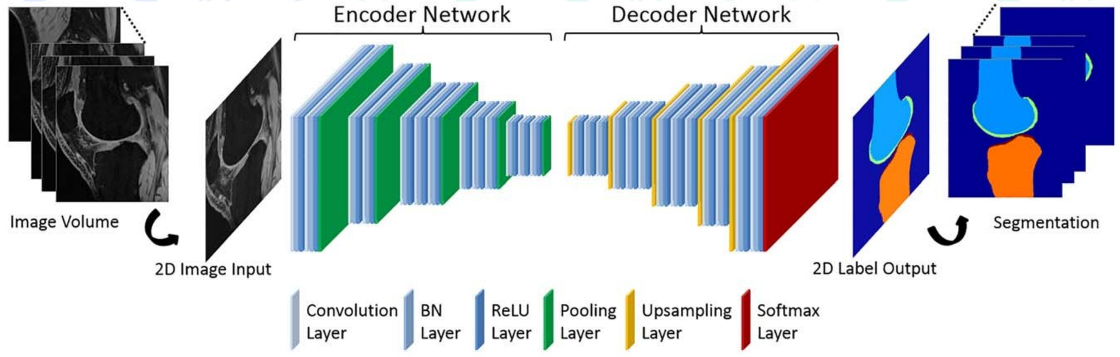
  - Consider a point that is distant from our manifold, this point will have a large error.
- Missing value reconstruction.
  - We can use our trained network to add missing data from appropriate data sets.

Using autoencoders



- An autoencoder trained to classify images into patches.
- This type of segmentation is used for systems such as self driving cars.
- A convolutional architecture is used for encoding and decoding.

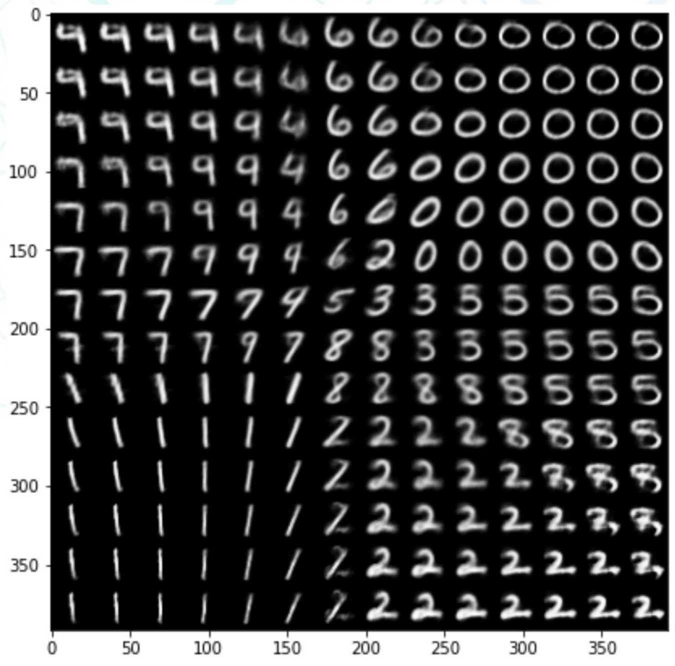




Architecture of a convolutional autoencoder for image segmentation

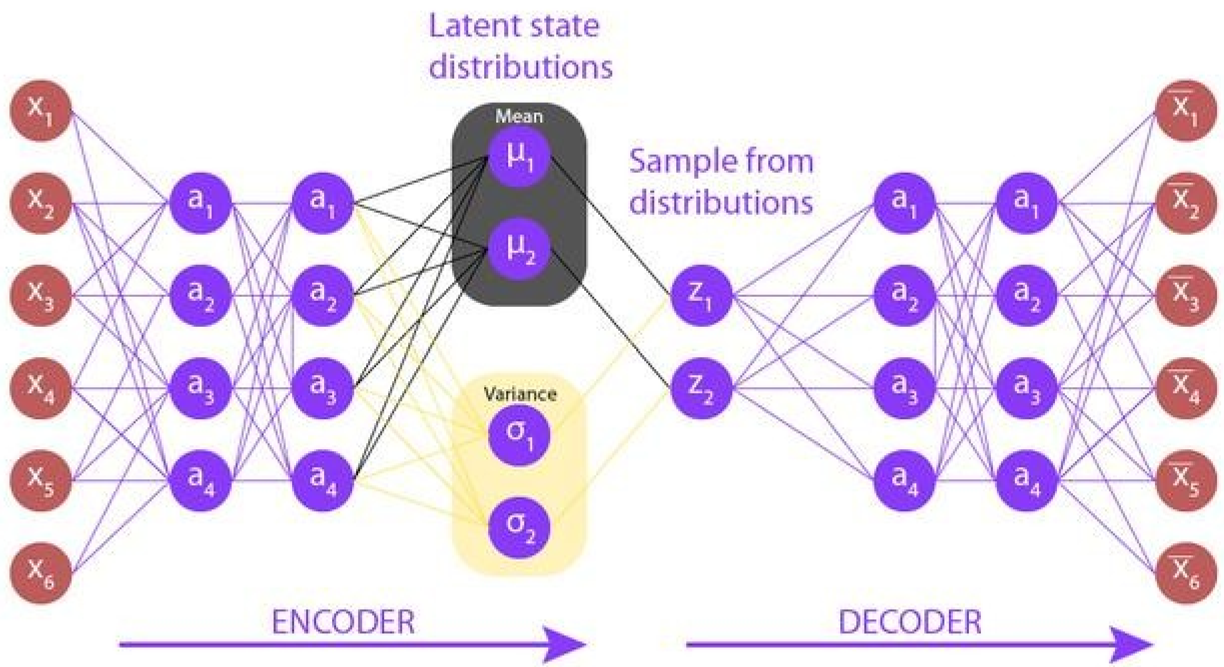
Variational autoencoders

- We would like to be able to feed a vector into the reduced hidden layer of an autoencoder and get meaningful output … but it doesn’t work, why?
- The problem space is, of course, vast, and discontinuous. If we pick a vector at random then we are extremely unlikely to choose one that has any ‘meaning’.
- What if we could just sample from the local distribution of clustered data points within the problem space? If, for example, we could pick from the vectors close to horses and astronauts.
- You can think of the distribution as the ‘bag’ of learned objects of a particular class.
- This then is the idea of a variational autoencoder. When we train the network we somehow learn the local distributions within the problem space.
- Then we can pick vectors close to the distributions we are interested in.
- To the right is a walk through the vector space of the distributions of handwritten numbers from a variational autoencoder.



Variational autoencoders

- The ‘bottleneck’ is replaced with two separate vectors:
  - A vector representing the mean of your distribution
  - Another vector representing the standard deviation of your distribution
- Training changes to reflect this architecture.
- The cost function has two terms:
  - The reconstruction cost based off the expectation (in the probablistic sense as we are sampling from a distribution)
  - The KL divergence, which essentially tries to ensure that the distribution you are learning is close to a normal distribution


Latent state distributions
Mean
μ_1
μ_2
Variance
σ_1
σ_2
Sample from distributions
x_1 x_2 x_3 x_4 x_5 x_6
a_1 a_2 a_3 a_4
z_1 z_2
$\bar{x}_1$ $\bar{x}_2$ $\bar{x}_3$ $\bar{x}_4$ $\bar{x}_5$ $\bar{x}_6$
ENCODER
DECODER

Generative models

- You can think of generative neural networks as ‘backwards’ neural networks!
  - These networks generate NEW data with similar statistics to their training data

Stable diffusion prompt: a photograph of an astronaut riding a horse

- GAN: Generative adversarial networks
  - Two networks compete against each other
  - One networks gain is the others loss
  - These networks are therefore trained to fool their discriminator networks whose purpose is to spot generated data
  - ChatGPT and Stable Diffusion both use this architecture in their training