Imperial College London

ULIS

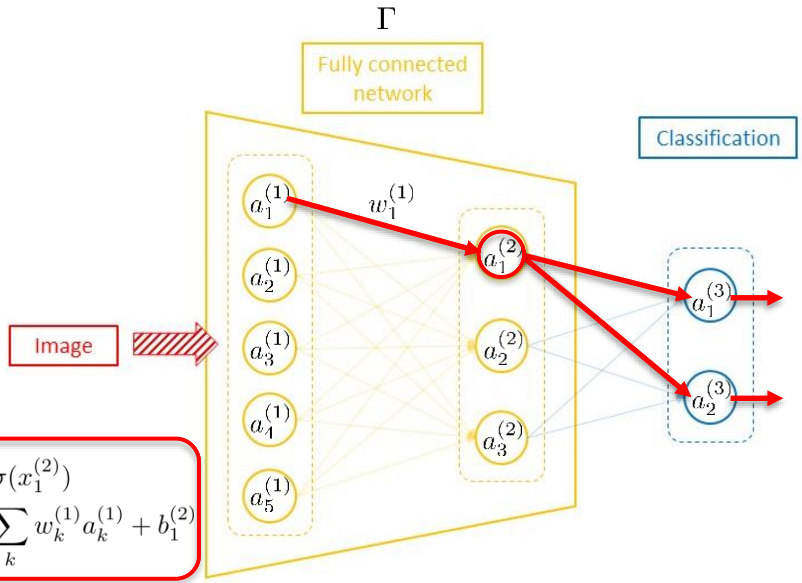
Variational Inference

+

Autoencoders

EPSRC
Centre for Doctoral Training in
NEUROTECHNOLOGY
for life and health

Recap



$$a_1^{(2)} = \sigma(x_1^{(2)})$$
$$x_1^{(2)} = \sum_k w_k^{(1)} a_k^{(1)} + b_1^{(2)}$$

$$\boxed{\frac{\partial L}{\partial w_1^{(1)}}} = \boxed{\frac{\partial L}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial x_1^{(3)}} \frac{\partial x_1^{(3)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial x_1^{(2)}} \frac{\partial x_1^{(2)}}{\partial w_1^{(1)}}} + \boxed{\frac{\partial L}{\partial a_2^{(3)}} \frac{\partial a_2^{(3)}}{\partial x_2^{(3)}} \frac{\partial x_2^{(3)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial x_1^{(2)}} \frac{\partial x_1^{(2)}}{\partial w_1^{(1)}}}$$

- We can classify an image using a fully connected neural network architecture
- Each layer has a non-linear connection to each node in the previous layer
- We train neural networks with back-propagation (which is the chain rule)

Objectives

- What are generative models? Understand the role of **observed variables** and **latent variables** (Terms are not always rigorous)
- Understand Autoencoder architecture (a particular type of generative model) and how to train one using SGD
- Discover Variational Autoencoders, which constrain the latent space of Autoencoders
- Use Variational Inference to train a Variational Autoencoder
- Blur the supervised/unsupervised line with Conditional Variational Autoencoders

Generative Models

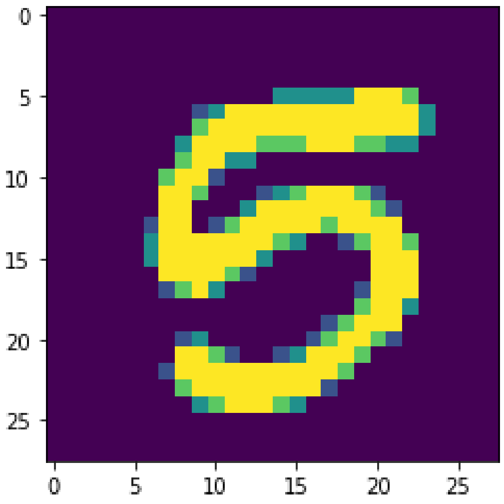
Observed Variables

- Observed variables are variables which we can measure

- For a die, the observed variable is the face of the die which points upwards when we roll it

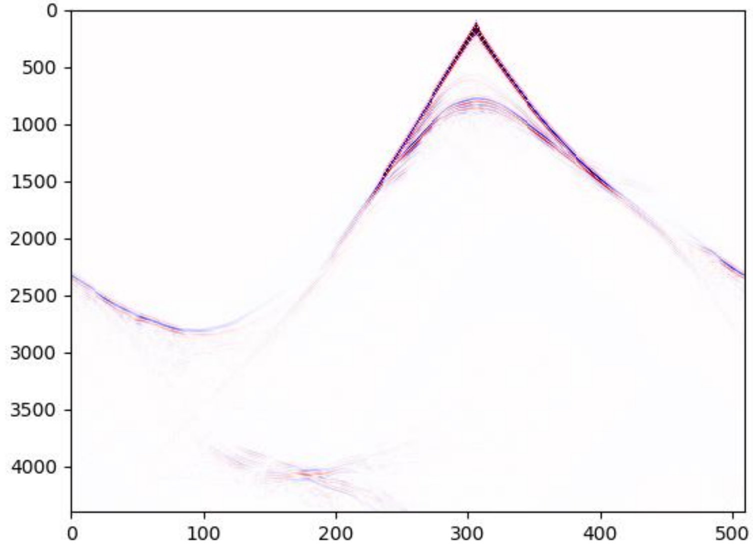
Observed Variables

- Observed variables are variables which we can measure
- In MNIST, the observed variable is an image of a handwritten digit



Observed Variables

- Observed variables are variables which we can measure
- In FWI, the observed variable is the wavefield data



- Can anyone give me an example of another observed variable?

Latent Variables

- Latent variables are variables which explain observed variables

- For a die, the latent variable is the biasedness of the die

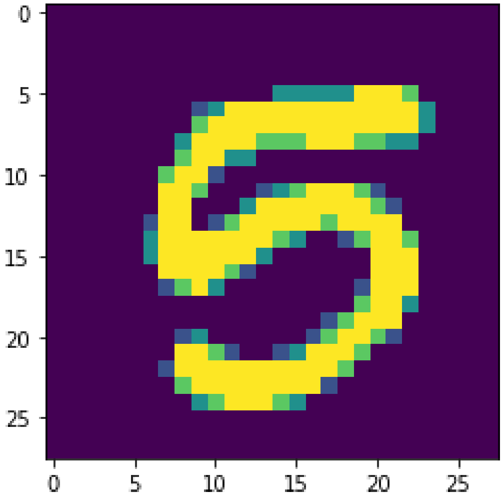
Roll 1 Roll 2 Roll 3 Roll 4 Roll 5

Latent Variable

p(4) = 100%

Latent Variables

- Latent variables are variables which explain observed variables
- In MNIST, the latent variable is the digit class

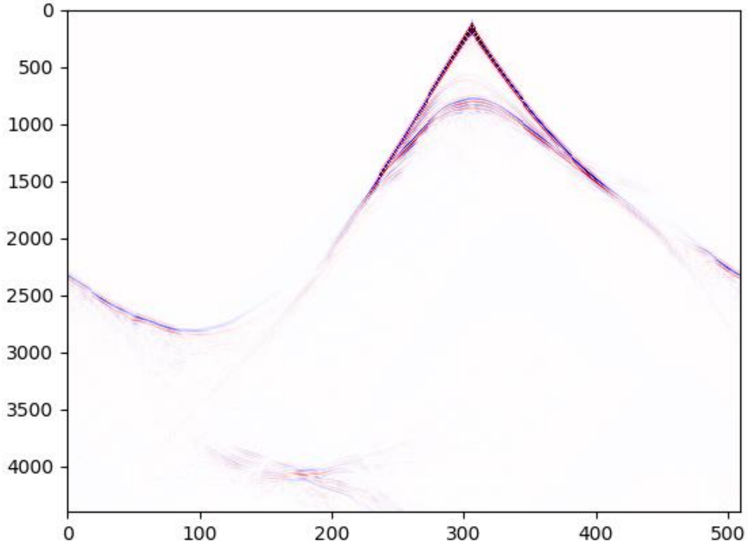


Latent Variable

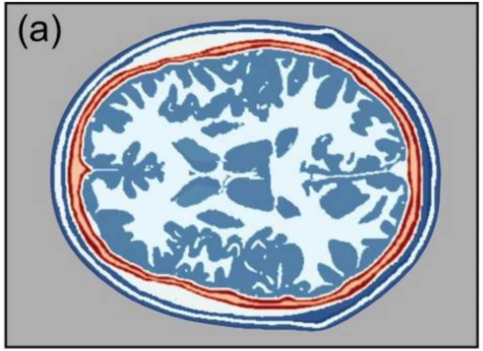
Digit class = 5

Latent Variables

- Latent variables are variables which explain observed variables
- In FWI, the latent variable is the acoustic sound speed



Latent Variable



- Can anyone give me an example of another latent variable?

Generative Models

- Generative models capture the joint probability distribution of an observed and latent variable

- If we sample the latent variable, we can generate samples of the observed variable

For example

- A single die $p(\text{face}, \text{bias})$
- MNIST digits $p(\text{image}, \text{class})$
- FWI $p(\text{wavefield}, \text{sound speed})$

Generative Models

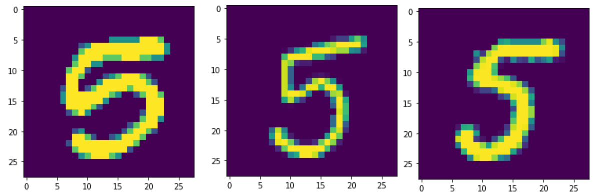
- If we sample the latent variable, we can generate samples of the observed variable
- The generative model contains all the possible biases of a die
- We sample the latent variable by making a die which is biased p(4) = 100%
- Now we can take some samples…

Roll 1 Roll 2 Roll 3 Roll 4 Roll 5

Generative Models

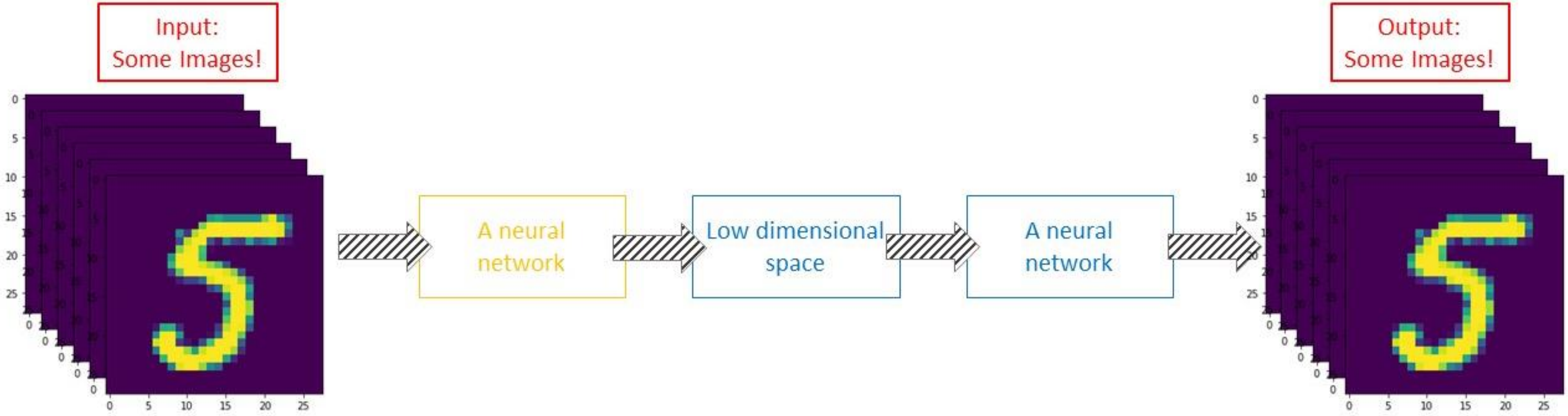
- If we sample the latent variable, we can generate samples of the observed variable
- The generative model contains all the possible handwritten digits
- We choose the digit class as 5
- Now we can take some samples…

Sample 1 Sample 2 Sample 3

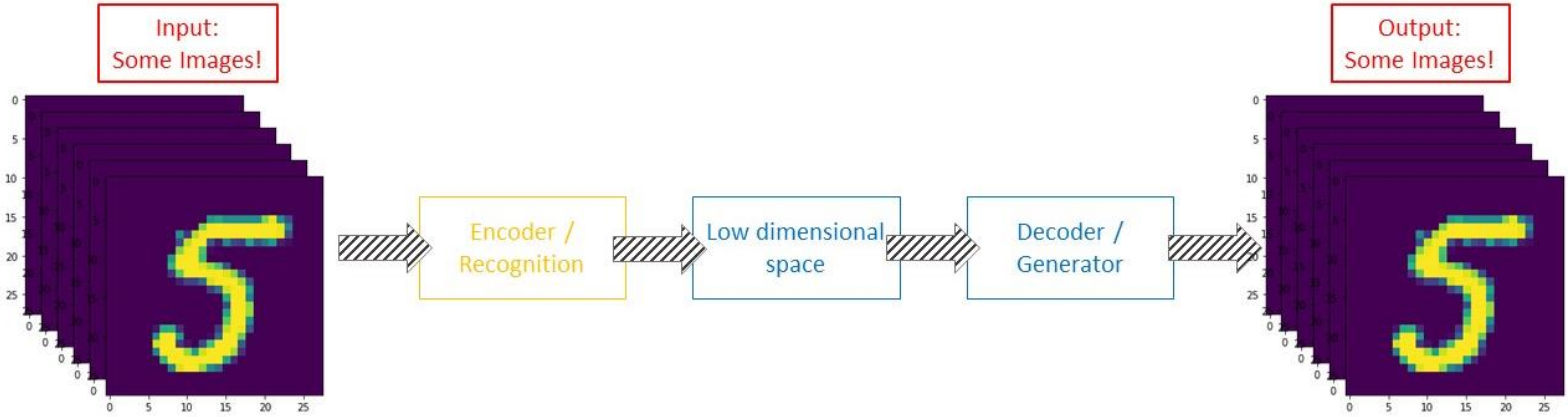


Conceptual study of Auto-encoders and Variational Auto-encoders

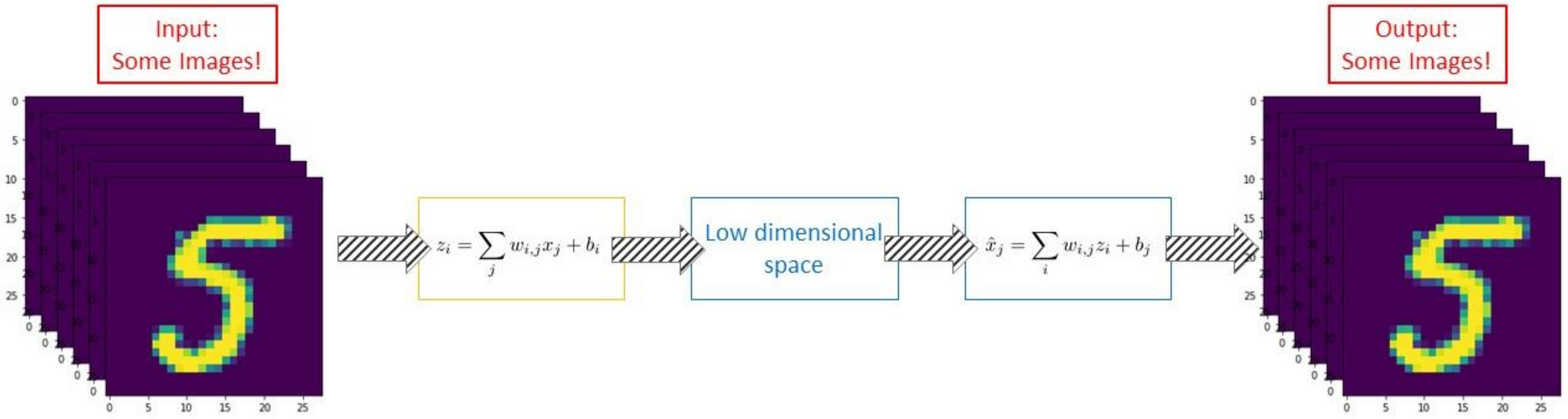
Autoencoder Workflow



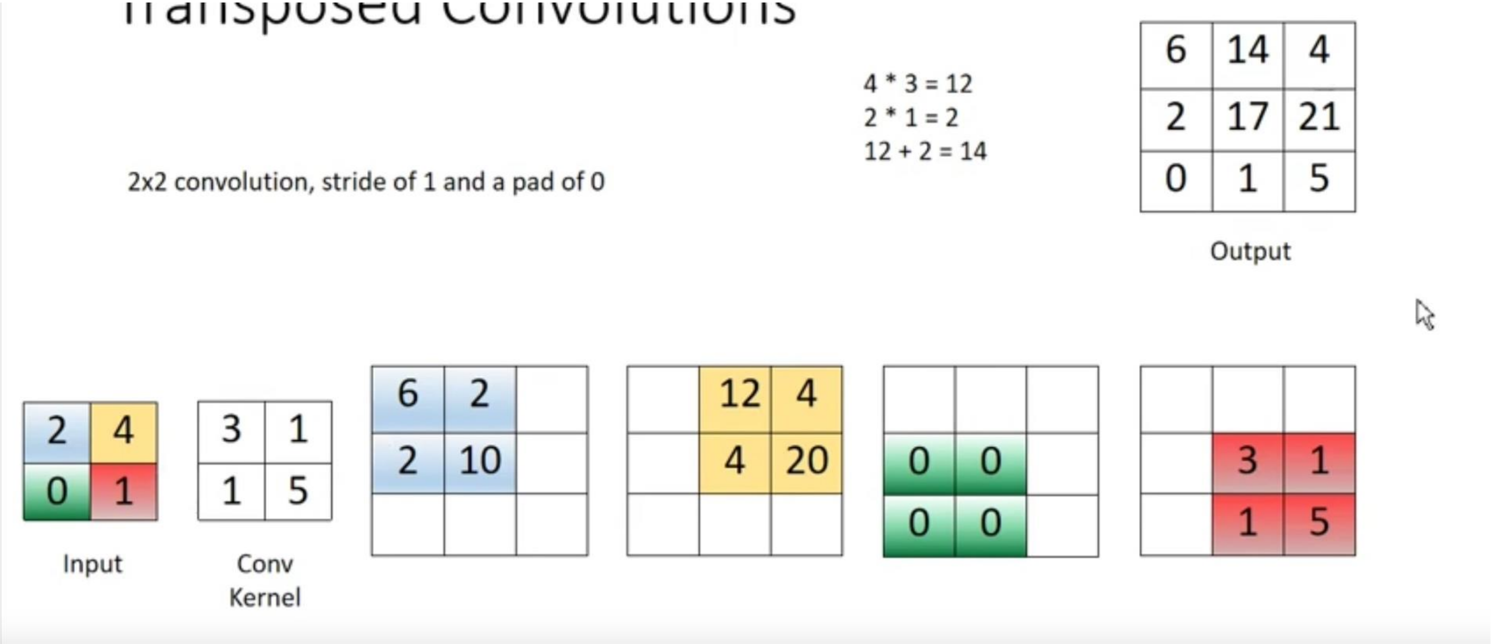
Autoencoder Names



A linear autoencoder is PCA

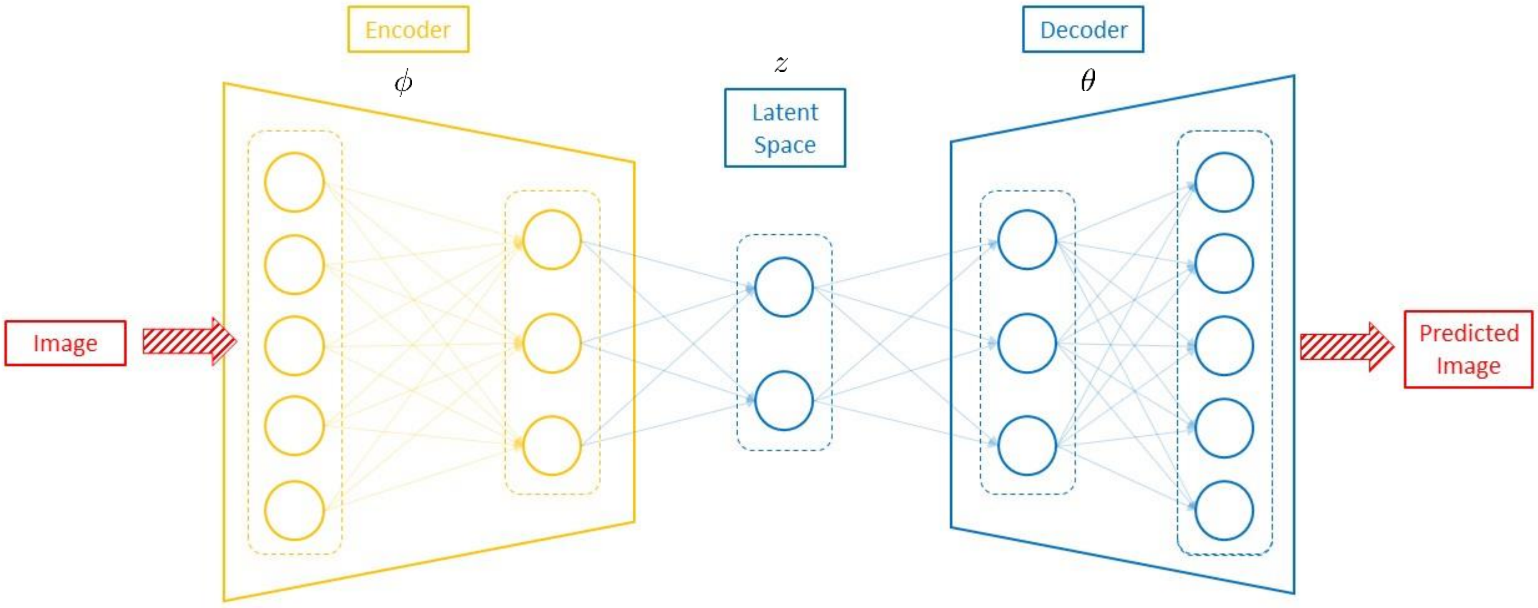


Transposed Convolution (Increasing dimensionality)

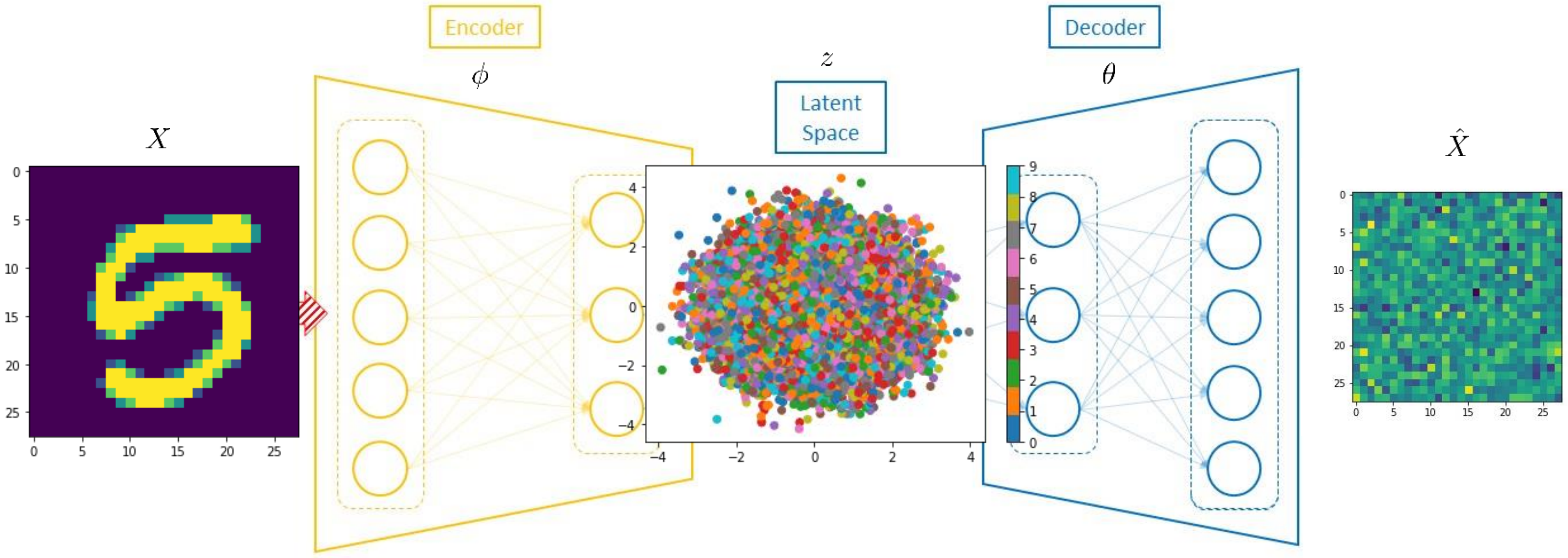


- Shamelessly borrowed from the source https://www.youtube.com/watch?v=96_oGE8WyPg

Lets generate an image



Lets generate an image

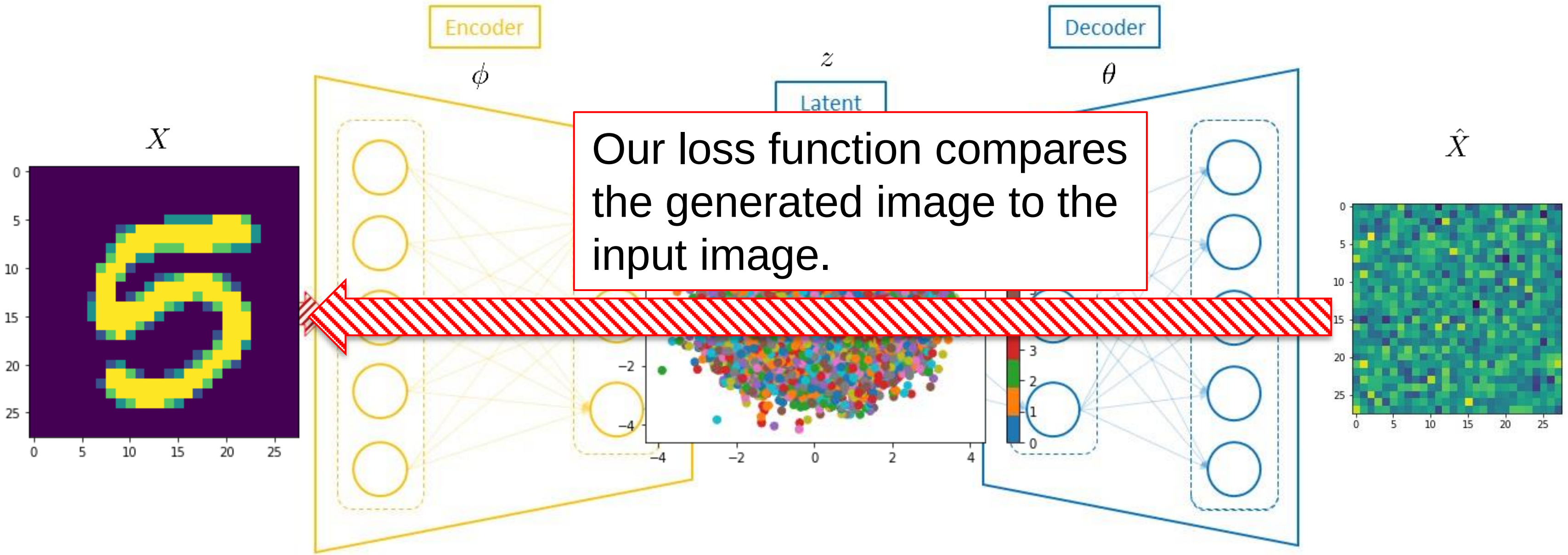


The Encoder (also know as a Recognition network) takes an example image

The layers of the encoder are activated and eventually activate a latent space. Initially the latent space is random

Decoder (also known as Generator) takes the latent activations and generates an image.

How to we train an Autoencoder?



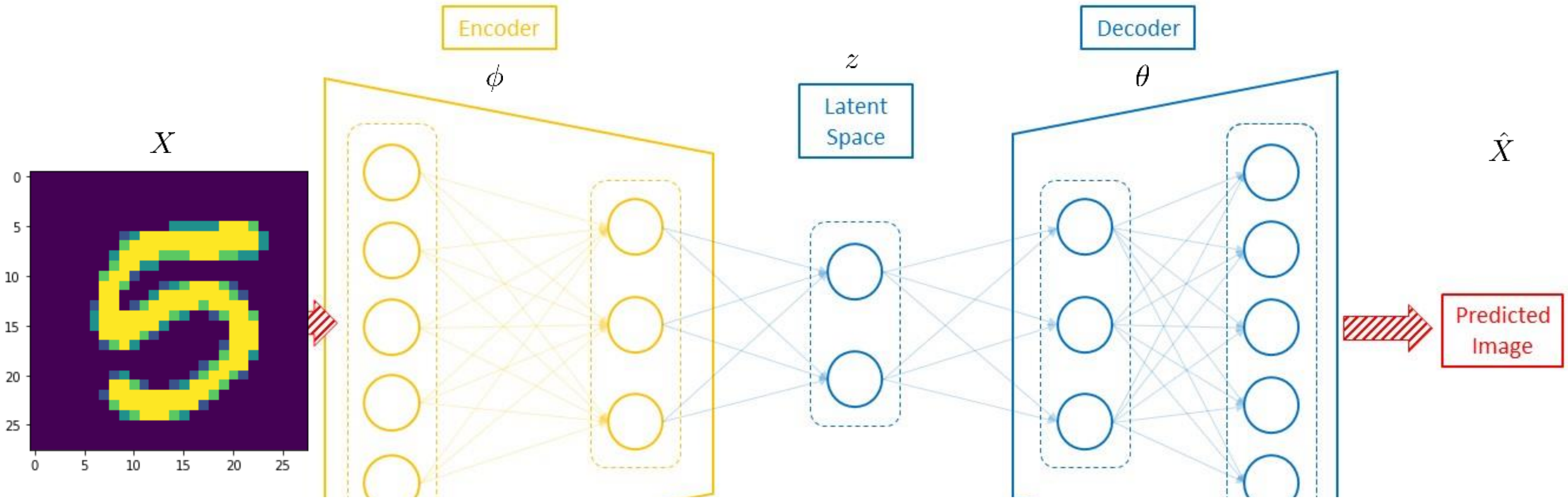
Our loss function compares the generated image to the input image.

The Encoder (also know as a Recognition network) takes an example image

The layers of the encoder are activated and eventually activate a latent space. Initially the latent space is random

Decoder (also known as Generator) takes the latent activations and generates an image.

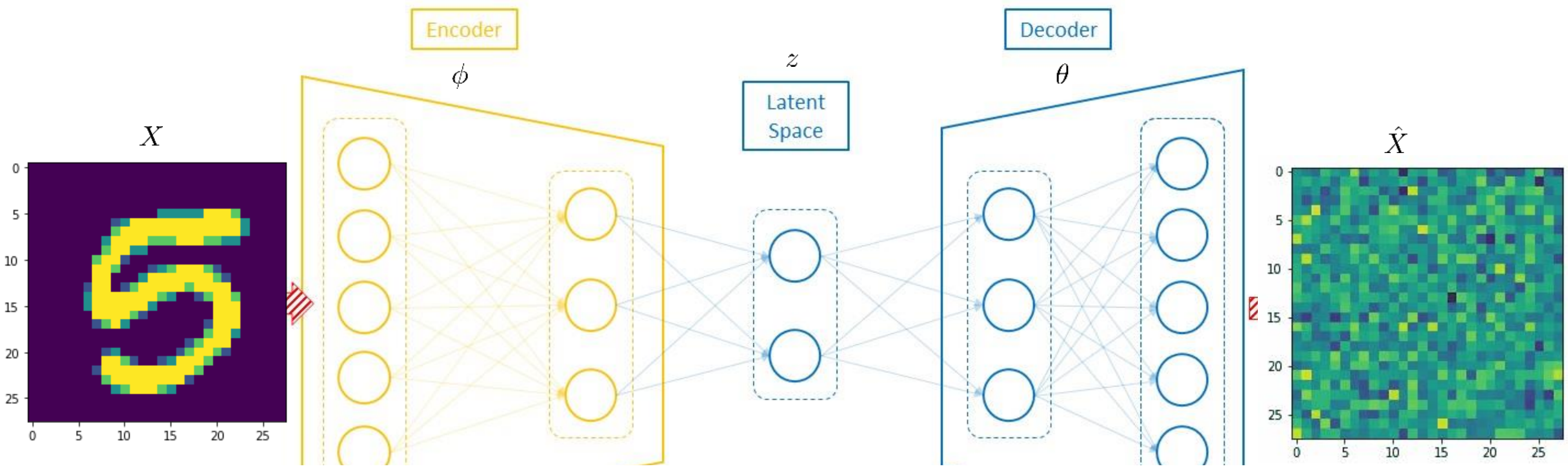
How to we train an Autoencoder?



Steps

1. Compare the input image and the generated image using the L2 norm
2. Take the sum of the pixelwise differences
3. Use the sum of the difference to run a stochastic gradient descent (to train the parameters of the encoder and decoder

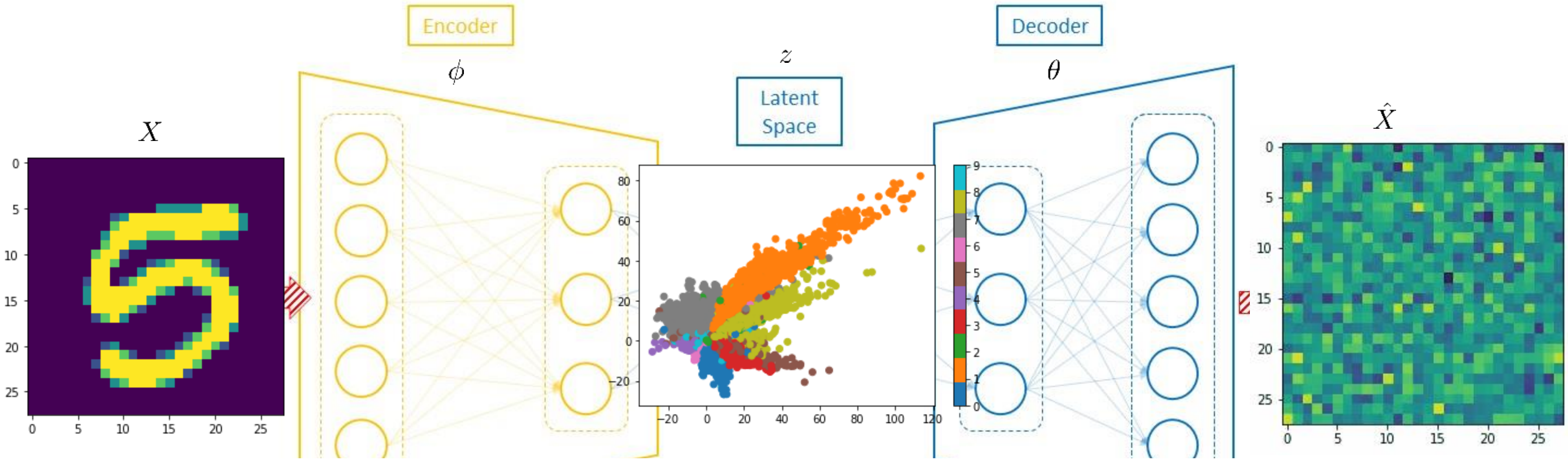
How to we train an Autoencoder?



Steps

1. Compare the input image and the generated image using the L2 norm
2. Take the sum of the pixelwise differences
3. Use the sum of the difference to run a stochastic gradient descent (to train the parameters of the encoder and decoder

How to we train an Autoencoder?



Steps

1. Compare the input image and the generated image using the L2 norm
2. Take the sum of the pixelwise differences
3. Use the sum of the difference to run a stochastic gradient descent (to train the parameters of the encoder and decoder

Mathematics of Training Autoencoders

- This is relatively straight-forward, as usual we just use the chain rule

$$\frac{\partial L}{\partial w_1^{(1)}} = \frac{\partial L}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial x_1^{(3)}} \frac{\partial x_1^{(3)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial x_1^{(2)}} \frac{\partial x_1^{(2)}}{\partial w_1^{(1)}} + \frac{\partial L}{\partial a_2^{(3)}} \frac{\partial a_2^{(3)}}{\partial x_2^{(3)}} \frac{\partial x_1^{(3)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial x_1^{(2)}} \frac{\partial x_1^{(2)}}{\partial w_1^{(1)}}$$

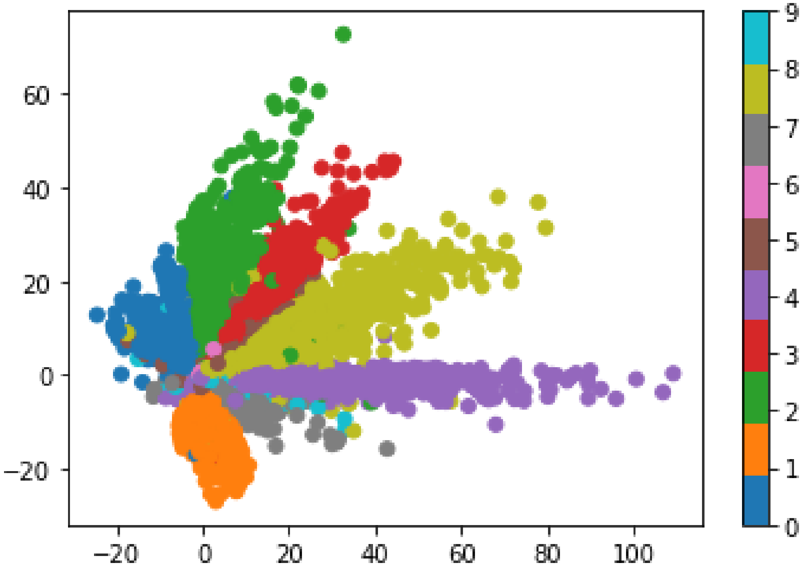
- Now the loss is the difference between the target (observed) image and the predicted image

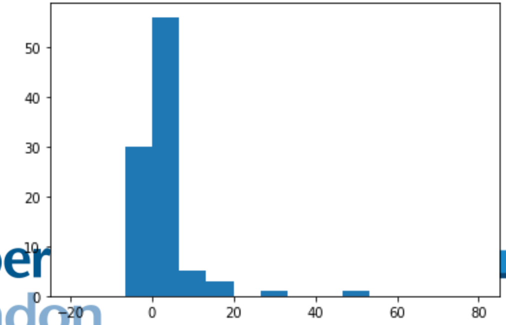
$$L = \mathrm{fn}(X, \hat{X})$$

- For example the L2-norm, which is the pixel-wise square difference between the target and the prediction

$$L = ||X - \hat{X}||^2$$

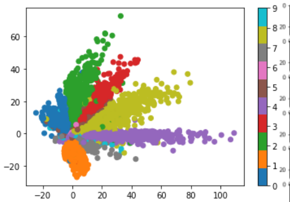
The problem with Autoencoders

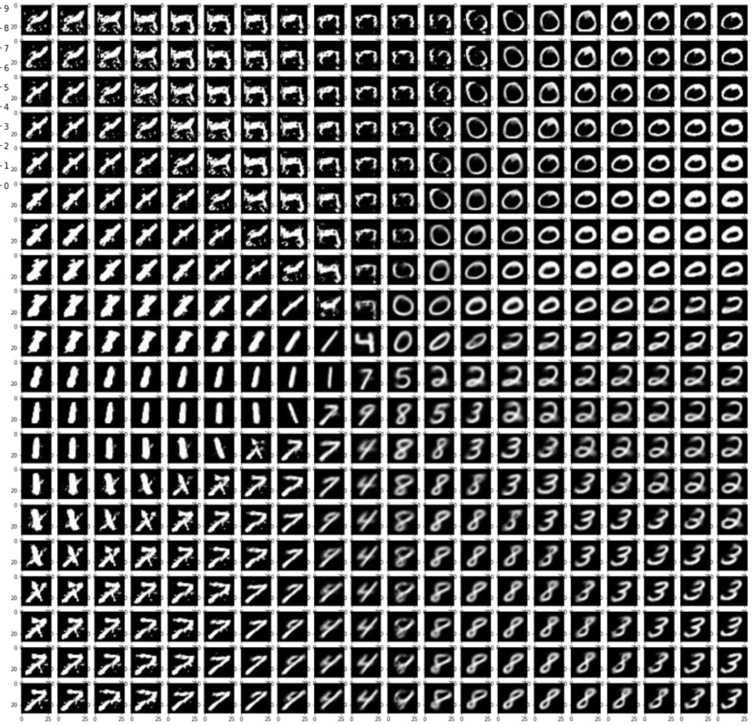




- When we visualise our latent space we see that the range of the latent vector is not limited
- This means the shape of the trained latent space is hard to predict
- *Note that the latent-space looks a bit like a probability distribution*

The problem with Autoencoders





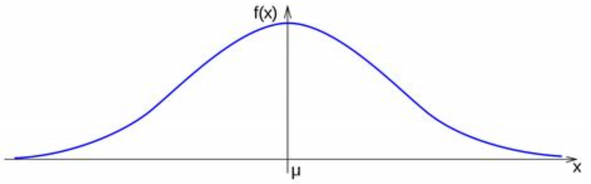
- This is a problem when we try to generate new samples from the latent space
- We either limit the range of values that we generate by sticking close to (0,0)
- Or we generate bogus values by sampling parts of the latent space which don't contain any useful information

Variational autoencoders

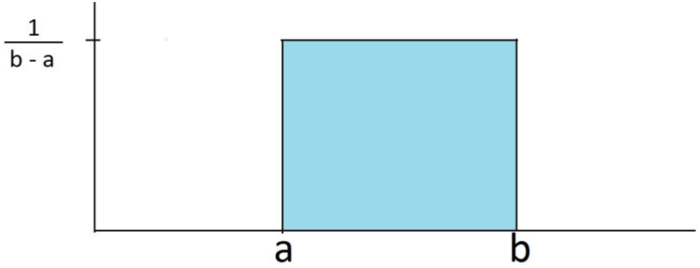
- We can force the latent space to be a specific probability distribution

- We can even force it to be a standard normal gaussian

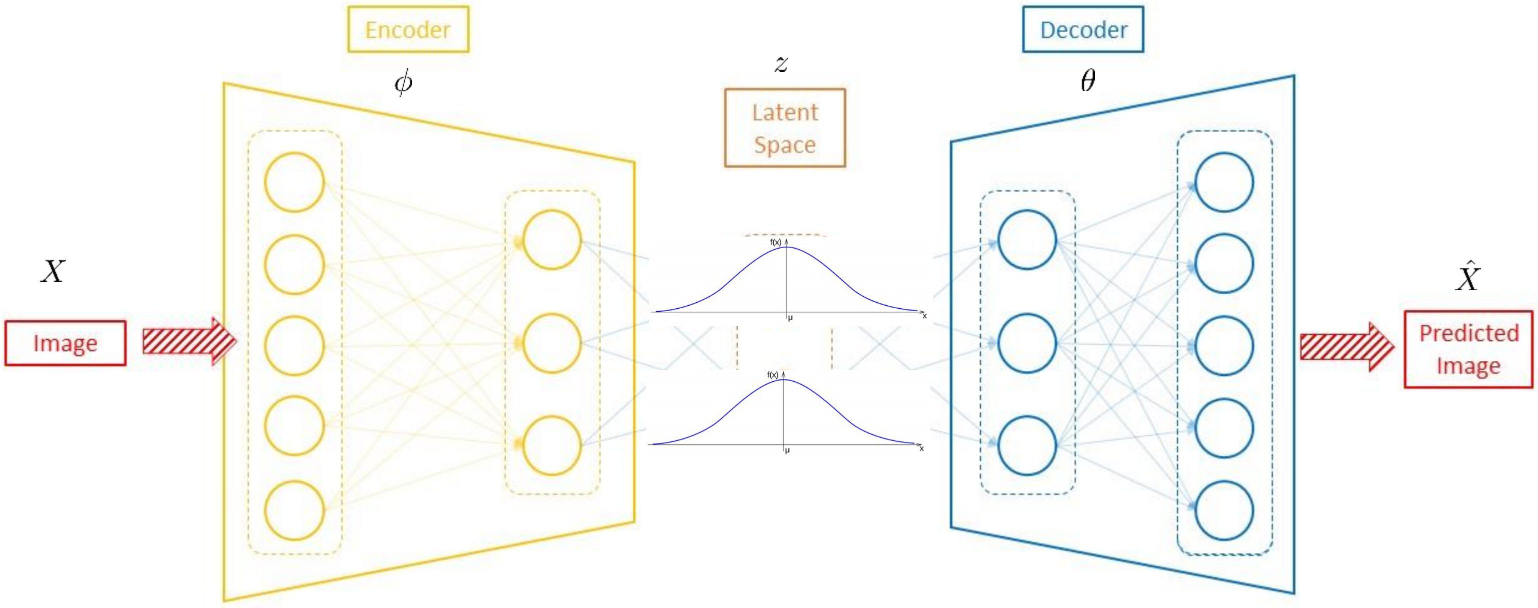
$$z_i \sim \mathcal{N}(0, I)$$



$$z_i \sim \mathcal{U}(1, 6)$$

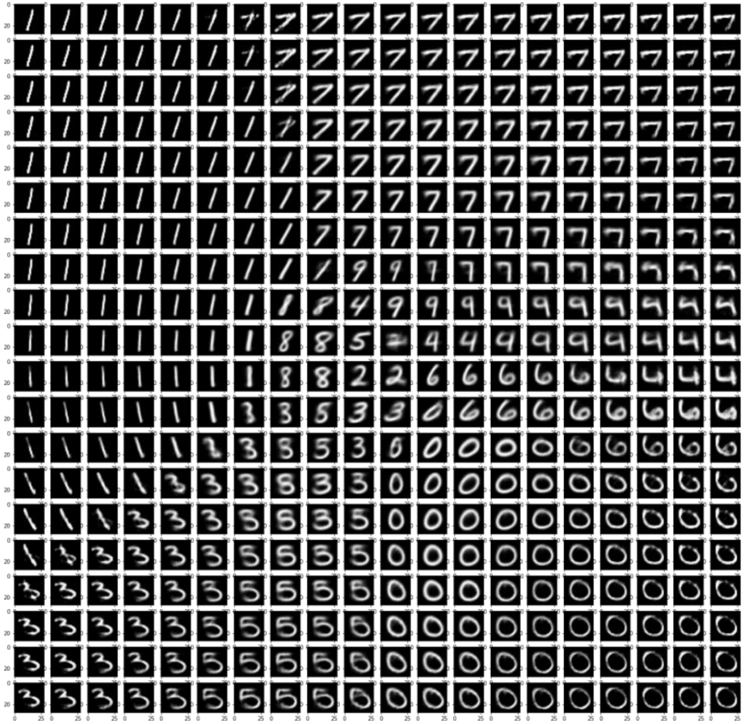


Variational Autoencoder

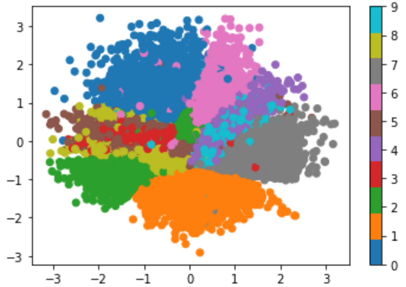


Now our latent space represents a probability distribution *which we have chosen*

Why is this better?

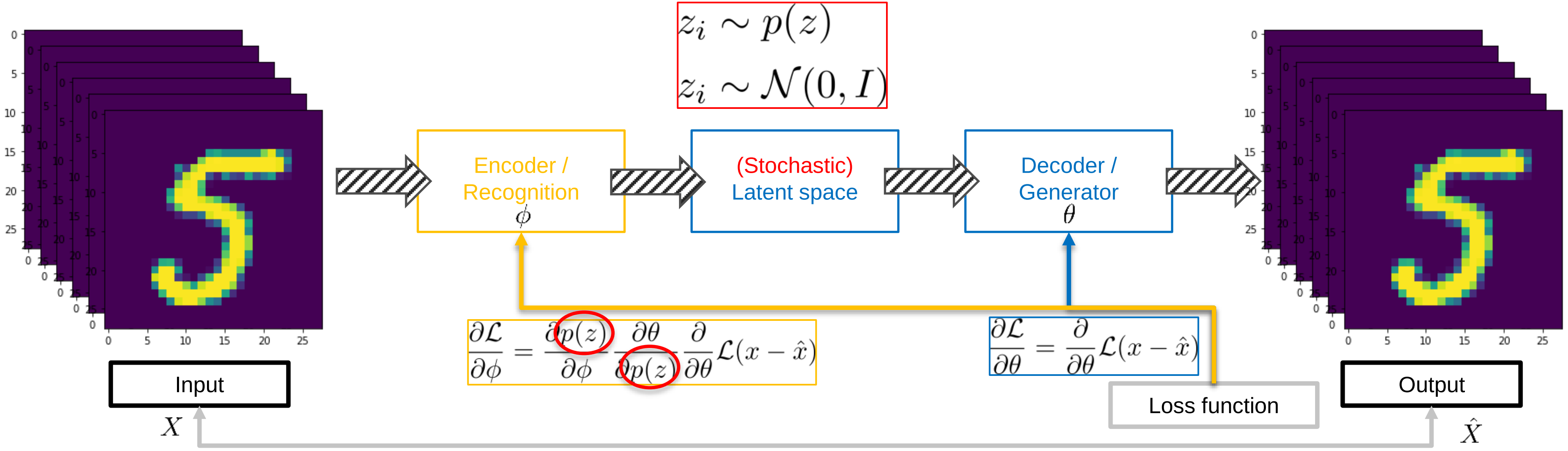


- (SPOILER) Now our latent space has a gaussian shape, sampling is much easier



- … and we no longer generate bogus values

Why can't we train our autoencoder anymore?



Training a
Variational Autoencoder

Bayes equation to the rescue …or maybe not

- *Please see the accompanying notes*
- Bayes equation is a framework for finding the posterior of a probabilistic optimisation problem
- Unfortunately the evidence distribution is intractable so it is challenging to solve directly
- An equivalent formulation for VAE’s is to minimise KL divergence between an approximating distribution and the posterior for the latent space

Evidence Lower Bound

- This KL divergence also can’t be solved directly
- We rearrange the KL divergence to remove the evidence term from the posterior
- This means we can minimise the negative KL divergence between the approximating distribution and the joint distribution (of the latent and observed variables)

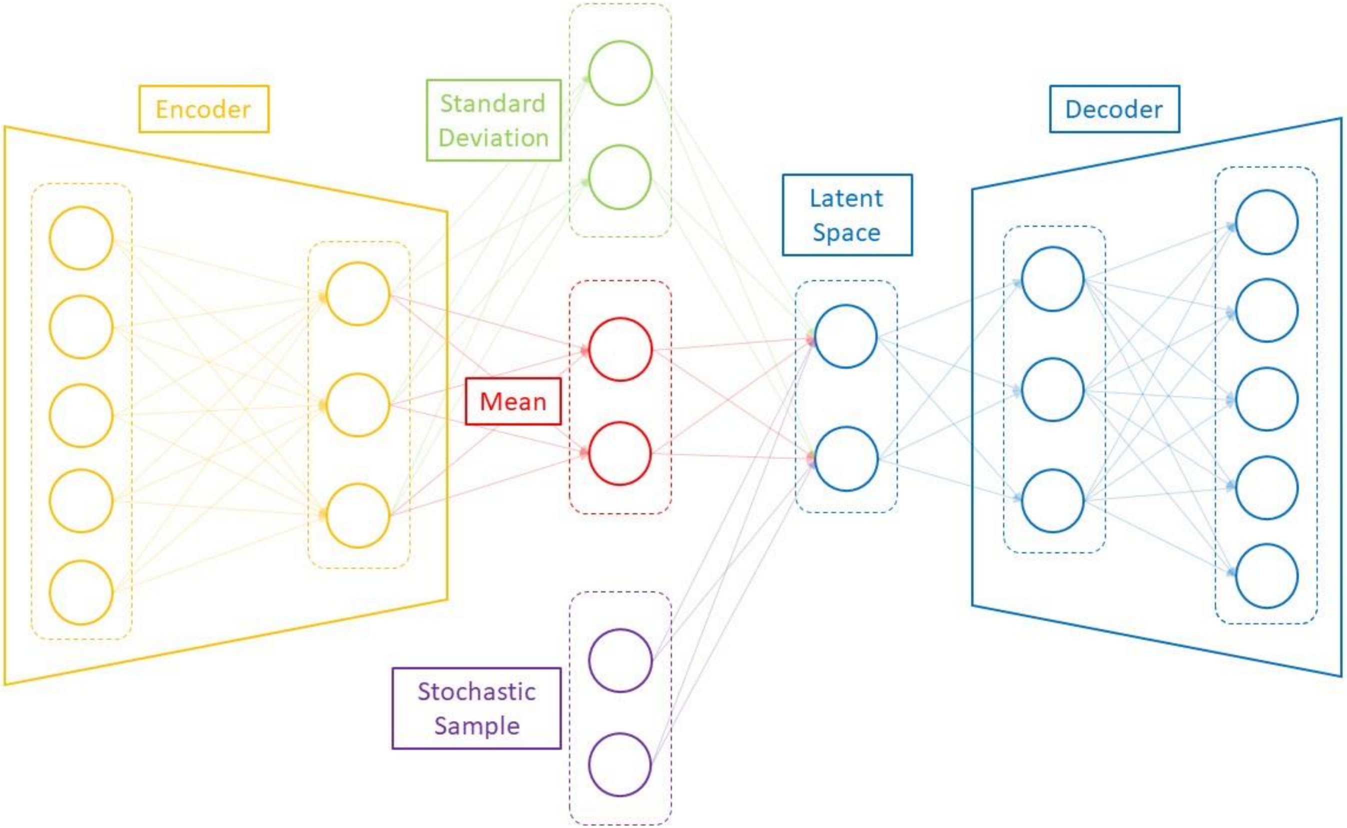
The log-derivative trick and score-function method

- We want to optimise by taking the derivative of the new KL divergence
- We can't take a derivative of the expectation of this KL divergence directly
- Instead we use the log-derivative trick to put this in a tractable form which allows a solution

The reparameterisation trick & Path-wise derivatives

- The score-function/log-derivative approach is prone to instability because the variance of the gradient estimate is high
- Another way to calculate the derivative is to arrange the approximating distribution so it is a deterministic function of another distribution with constant coefficients
- This means we can differentiate through the parameters of the deterministic function to get a lower-variance gradient update

Variational Autoencoders + Reparameterisation Trick

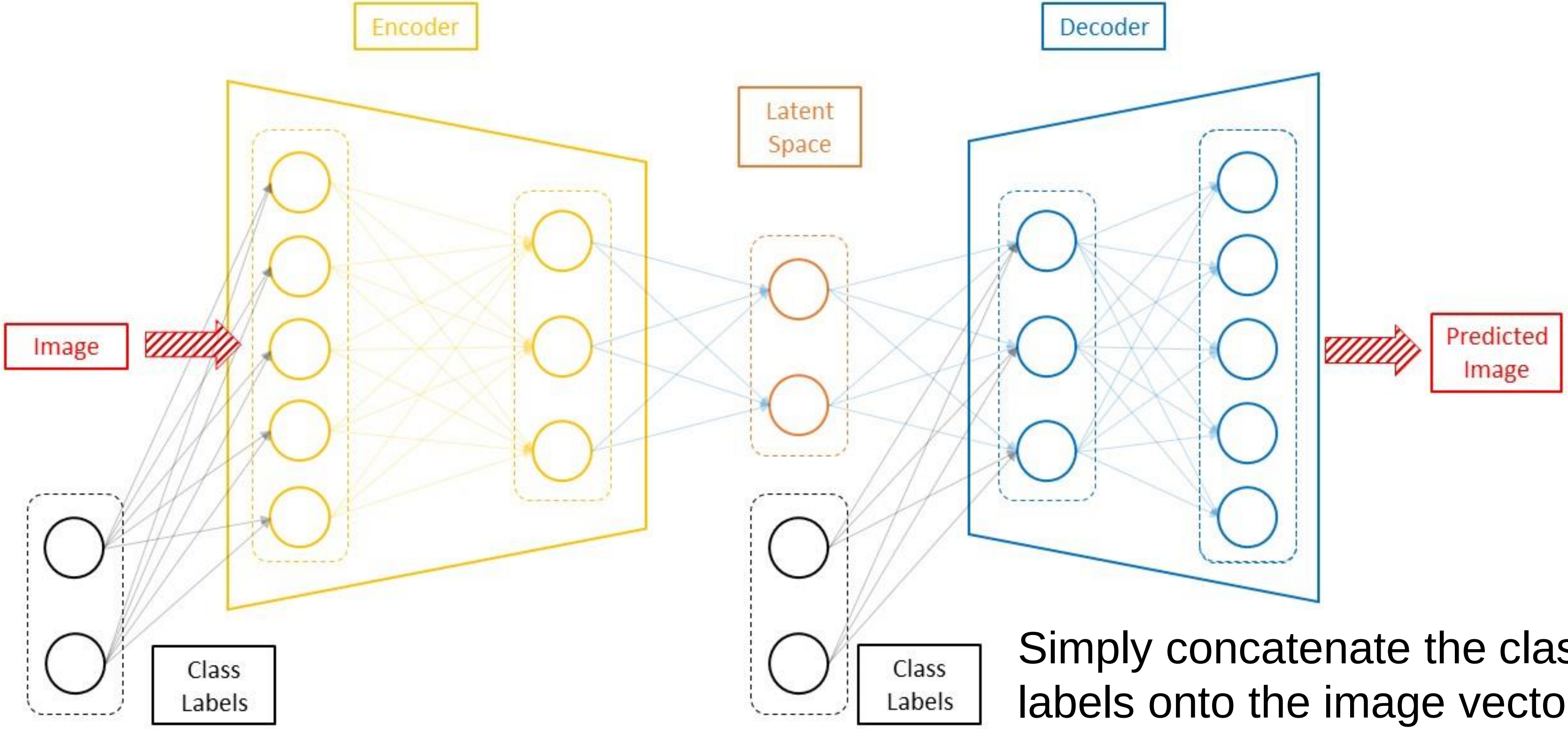


Concluding remarks

Objectives

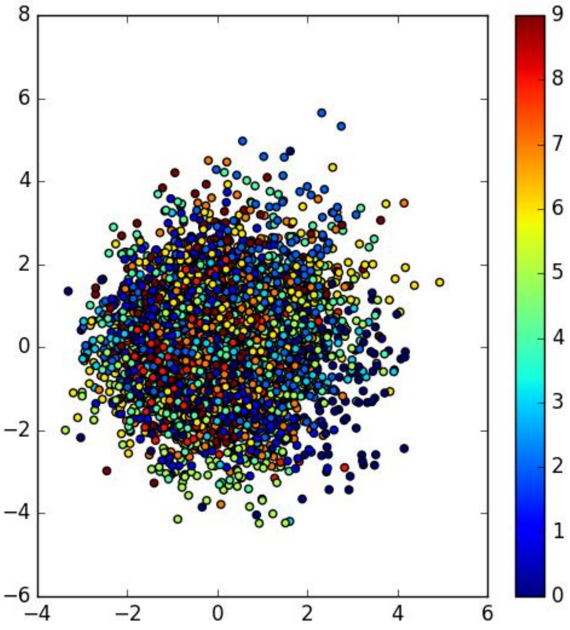
- What are generative models? Understand the role of **observed variables** and **latent variables** (Terms are not always rigorous)
- Understand Autoencoder architecture (a particular type of generative model) and how to train one using SGD
- Discover Variational Autoencoders, which constrain the latent space of Autoencoders
- Use Variational Inference to train a Variational Autoencoder
- Blur the supervised/unsupervised line with Conditional Variational Autoencoders

Conditional variational autoencoders (& Semi-supervised learning)



Simply concatenate the class-labels onto the image vector (before encoding) and the latent vector (before decoding)

Conditional variational autoencoders (& Semi-supervised learning)



- Note that the class labels are additional dimensions, so now each class is normally distributed in the latent space

- This tool is particularly useful for generative modelling in unbalanced datasets

- For a useful blog please see

https://wiseodd.github.io/techblog/2016/12/17/conditional-vae/

Useful links

- Please see the following blog https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html which covers many of the older types of specific architectures

- An in-depth tutorial https://arxiv.org/pdf/1606.05908.pdf

- t-distributed stochastic neighbour embedding is a way of visualising high-dimensional latent spaces
  - https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a
  - https://discuss.pytorch.org/t/t-sne-for-pytorch/44264

Novel Architectures

- Auto-encoder classification
  https://ieeexplore.ieee.org/abstract/document/9424386

- Normalising flow, particularly autoregressive flow
  https://arxiv.org/pdf/1606.04934.pdf

As good as GAN’s?

- Very deep VAE is a new type of architecture which looks roughly like a u-net
- For more information here’s the paper https://arxiv.org/pdf/2011.10650.pdf