Data Analysis & Machine Learning

Lecture 6:

Intro to Convoluted Neural Networks

Robert Currie

Artificial Neural Networks so far (a Recap)

- A DNN is a network of Neurons which are densely interconnected.
- We have used an ANN to build a generic classifier and train it on data.
- We have built and trained a model which is able to classify.
- Then we train our free *weights/neurons/nodes/parameters* to describe our input dataset.

DNN – Some Caveats

- Doesn’t perform well with noisy inputs or with small signals

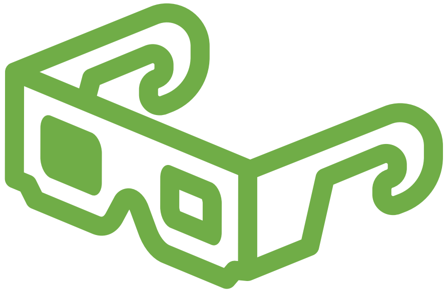
- We have the problem of Vanishing Gradients
  (e.g. DNN of depth >10 are <u>*very*</u> hard to train)

- Doesn’t work well with input changing i.e.
  - Not very good at extracting features if input is translated/transformed

- Requires only single numerical inputs per-neuron
  - Doesn’t scale very well with larger datapoints/datasets

- Effectively a black-box.
  We don’t know what a *single weight* deep inside 2 fully interconnected dense layers might represent in the *real world*...

ANN using 2D inputs



- Take a 2D photographic image as an example.
  (e.g., telescopic pictures, particle tracks, cracks in 2D materials, …)

- For a **4k colour** image we need to use,

  3 * 3840 * 2160 * n = 24,883,200 * neurons weights to use the input with a raw DNN.

- If this is a picture of a cat, we want to extract key features for instance such as *'number of whiskers'*, *'length of fur', or 'size of teeth'.*

- Ideally, we want to extract this automatically without having to label every feature in every image manually.

This can be achieved via Image Processing…

ANN & Image (pre-)processing

If we want to use a DNN to analyse a sample dataset we want to make sure that it’s extracting correlations between **signal** features in the dataset and ***not background or noise***.

An obvious step in this case is to consider pre-processing *all* our input data so that the DNN works well with it.

Doing this manually is time consuming, expensive and in-efficient, and is what humans do to verify that they’re not robots on websites.

Image Filtering – An Aside

Before we start applying image filtering techniques to our data, let's take a step back and look at an example of image filtering.

One of the most famous examples of this is edge detection filtering or applying ***Sobol*** operators to an input image.

This gives us the advantage of extracting out the features (edges) of a scene without having to care about the background of an image.

Image Filtering – Convolutions

Image filters are applied by convolving an operator with an input to give an output.

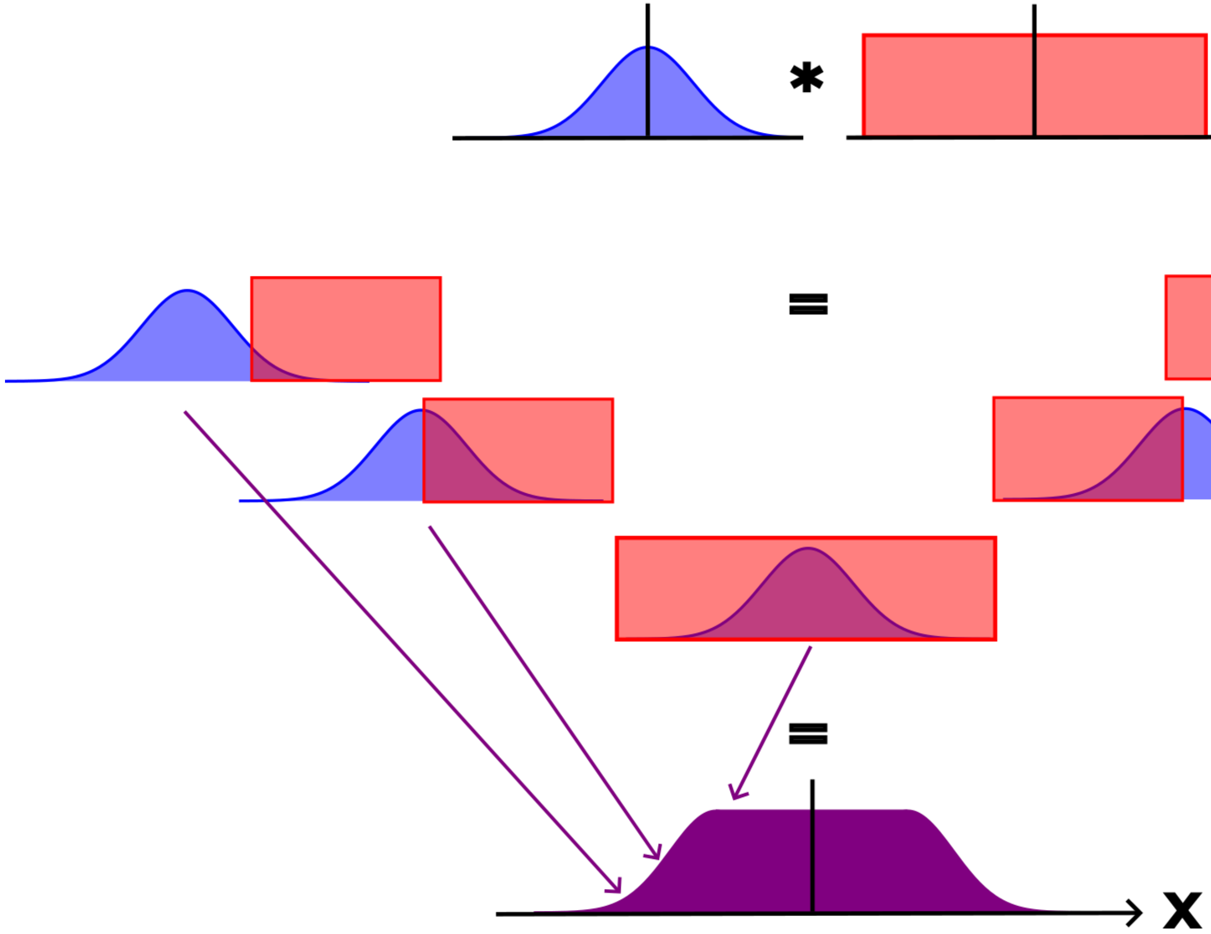
i.e. $$Input * \ Operator = Output$$

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t)g(t-\tau)d\tau$$

These operators can be used to sharpen, blur, up/down-sample data.

Convolutions – 1D example

(f * g)(x)



Convolutions – 2D

The 2nd dimension!

Now we want to convolve 2D functions/operators with 2D data.

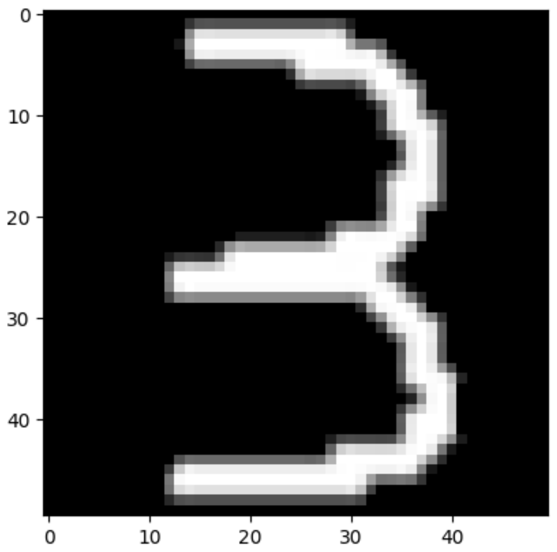
One of the most common 2D functions to demonstrate as I've mentioned is the Sobel Operator*(s)*.

$$\mathrm{G_x} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \mathrm{G_y} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

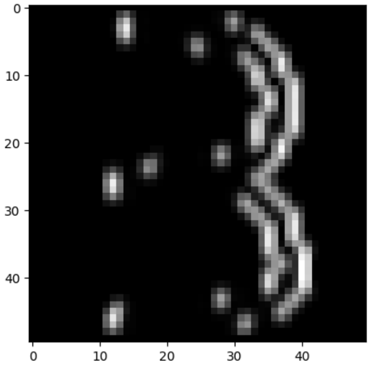
$$\mathbf{G} = \sqrt{\mathrm{G_x^2} + \mathrm{G_y^2}} \qquad \Theta = \mathrm{atan}(G_y, G_x)$$

Convolutions – 2D example

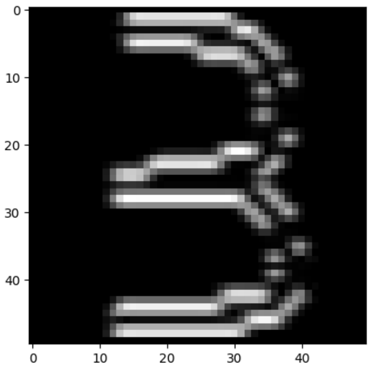
Input=



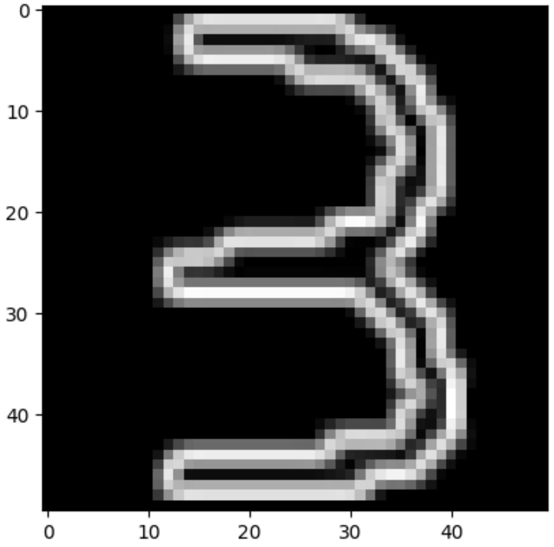
$G_x * Input =$



$G_y * Input =$



$\mathbf{G} * Input = Output =$



ANN – Image Processing

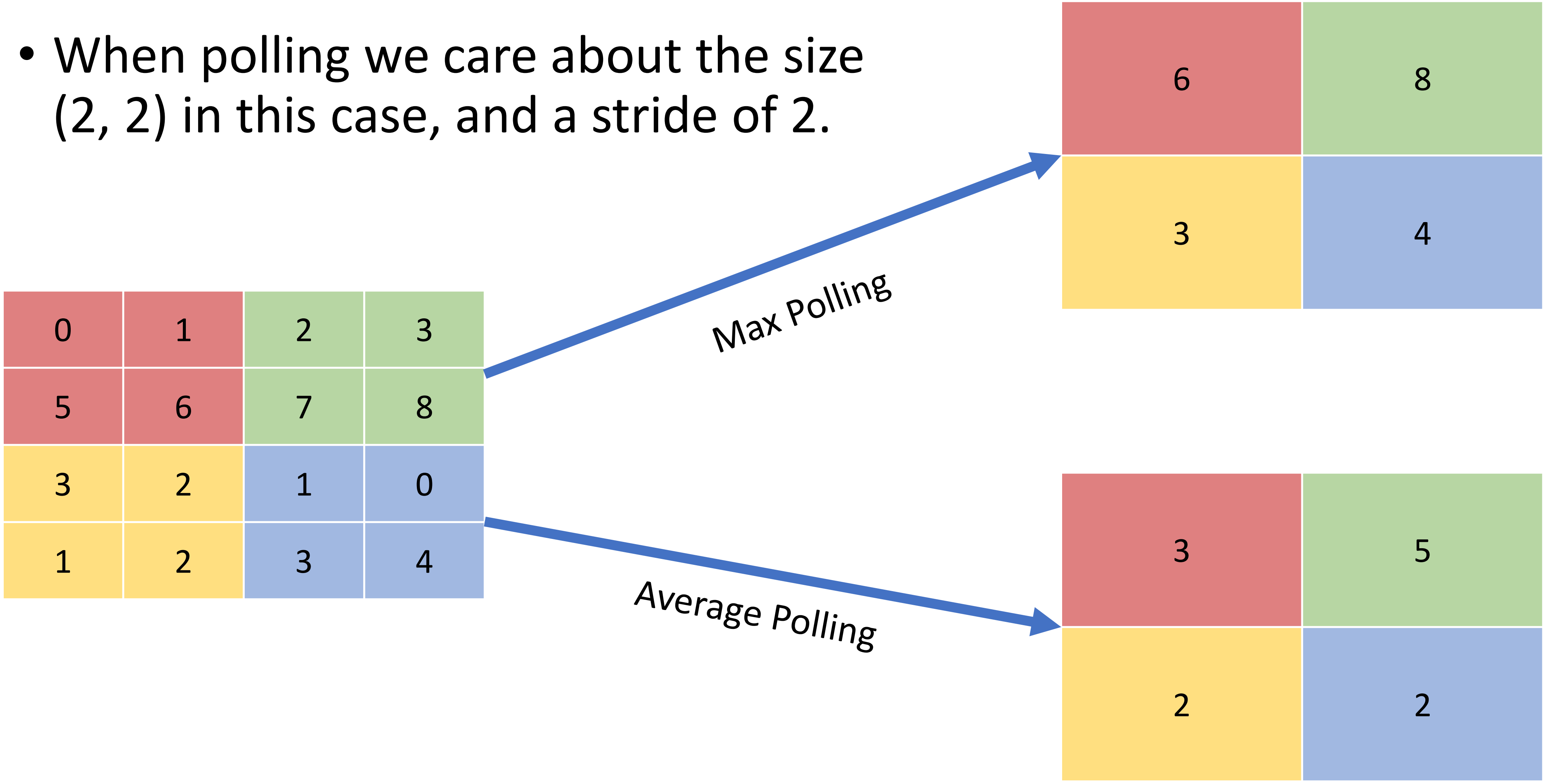
- Sobol operators are a specific example of a using a matrix of dimension (3,3) to extract features from the input.

- Other common image manipulation filters are; MaxPolling, AveragePolling, Flatten, Bounding-Box, Sharpening, Skew, …

- Let’s go over MaxPolling & Flatten

Image Manipulation – Other filters

- When polling we care about the size (2, 2) in this case, and a stride of 2.



CNN neurons

- By taking what we’ve just learned about Image Processing algorithms how do we now build our own neuron?
- Each neuron is now a ***Convolution*** not a summation.
- This means instead of 1 numerical weight per neuron we now have a matrix of weights meaning multiple model parameters per-neuron.
- We still want to use *Activation functions* and *biases* as these are independent of our neuron’s internals.

CNN Classifier network design

- One of the most famous CNN designs is from Yann LeCun et al. 1998

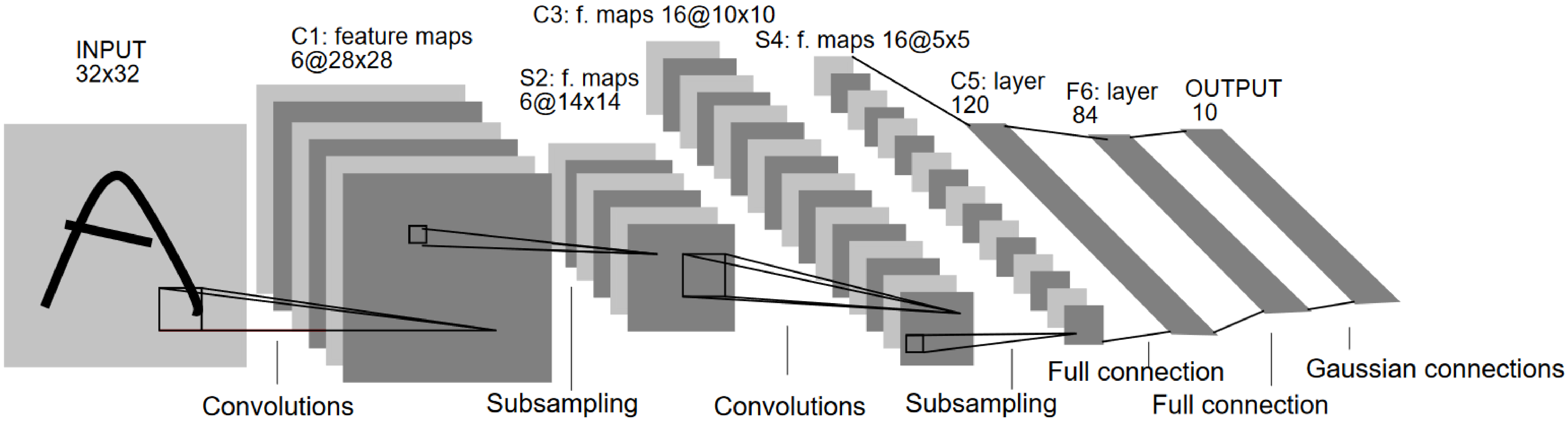
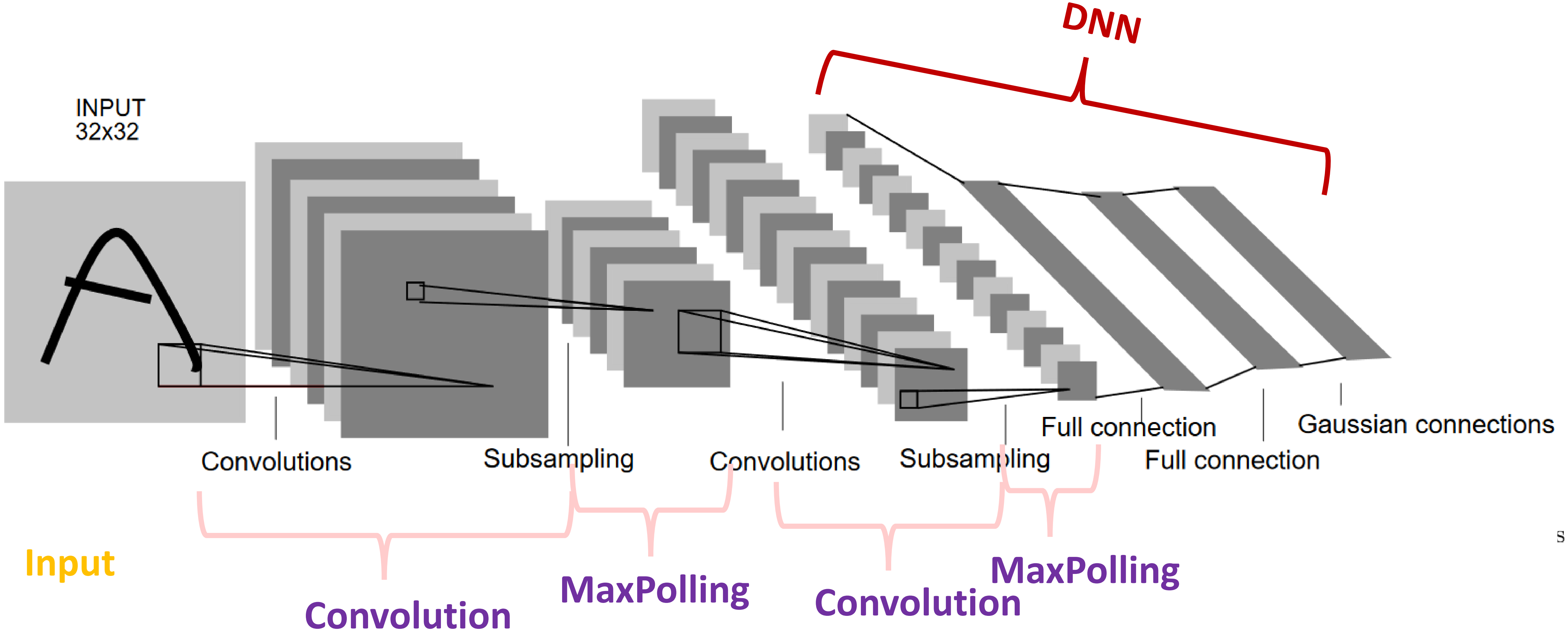


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

CNN network design



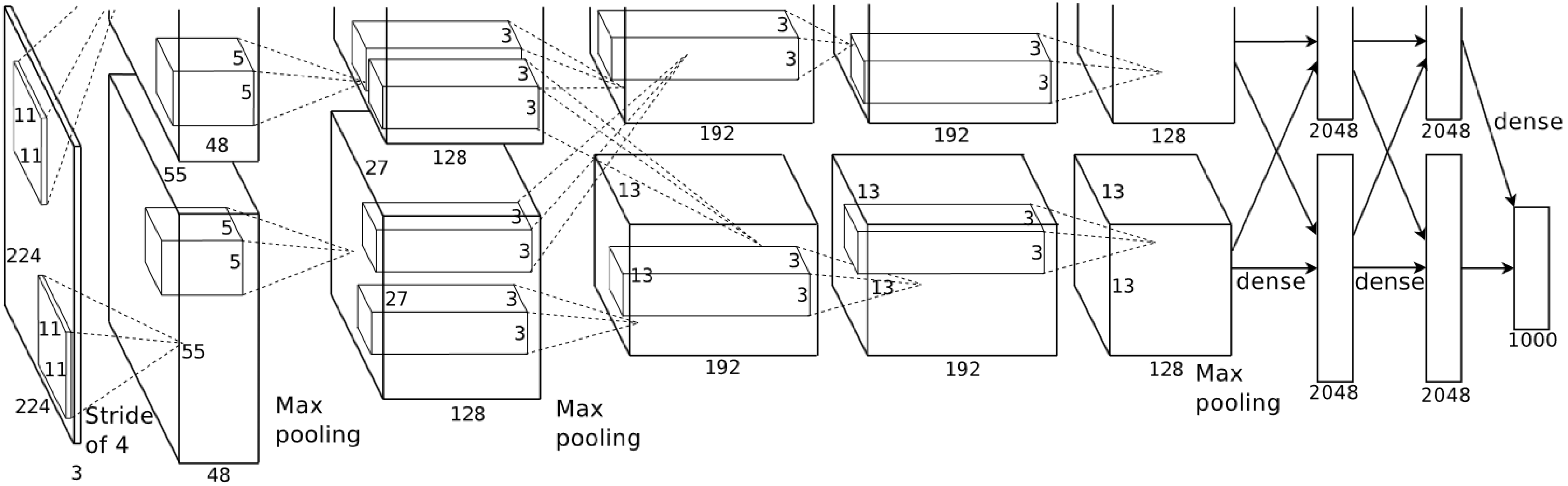
CNN – network design

- CNN networks begin with image pre-processing steps (**convolutions** and **max-polling**).
- The output from this layer is a reduced image format which is passed through successive pre-processing layers with potentially many kernels.
- Eventually for a classifier the output is flattened in some manner and the reduced flattened data is then passed into a **DNN** which we've already seen.
- This reduces the amount of data needing to be passed to a **DNN** and for the **DNN** to more easily separate ***key features*** from background in the data.

  E.g. 4k image may be reduced to 256x256x1 inputs based on extracted features…

CNN Classifiers – AlexNet

- AlexNet developed by Alex Krizhevsky et al, and published in 2012 represents what is now taken to be the basis of modern CNN classifier designs.
- This design was much more complex than LeNet5:



CNN / CDNN – network design

- Convolutional Neural Networks or Convolutional Deep Neural Networks are a combination of various pieces of technology.

- This design allows us to extract information from a given set of labelled inputs using supervised learning.

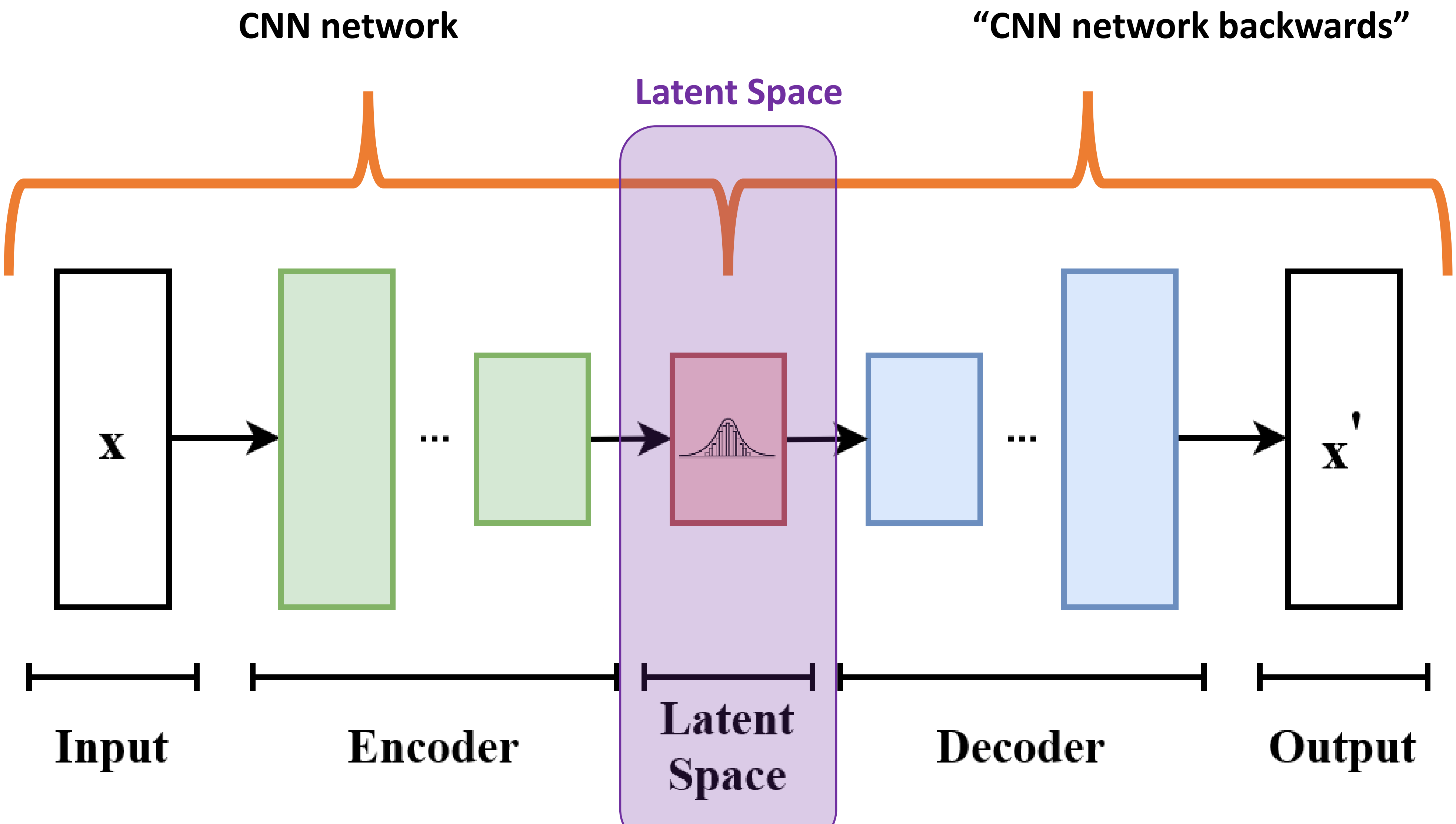
- However, this model design allows us to consider the use-case of un-supervised learning.

  i.e. using the model to *automatically* extract common features.

Other uses of CNN networks

- CNN networks extracts key features from a dataset in an automated way.
- This means that a CNN can encode information and extract the key features through training on a given dataset without labels.
- This allows us to detect when an input varies significantly from the feature-set it was trained on.
- Also, for a sufficiently large latent phase-space we’re potentially able to generate new datapoints by combining features in a way that is not seen in a given training dataset.

Variational AutoEncoder – Network design



Conv2DTranspose – An Aside

Forward convolution typically decreases-dimensions/downsamples.

This extracts information from the input image.

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 5 & 6 & 7 & 8 \\ 3 & 2 & 1 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix} * \begin{bmatrix} 3 & 5 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 27 & 39 & 51 \\ 55 & 59 & 63 \\ 25 & 21 & 17 \end{bmatrix}$$

Conv2DTranspose – An Aside

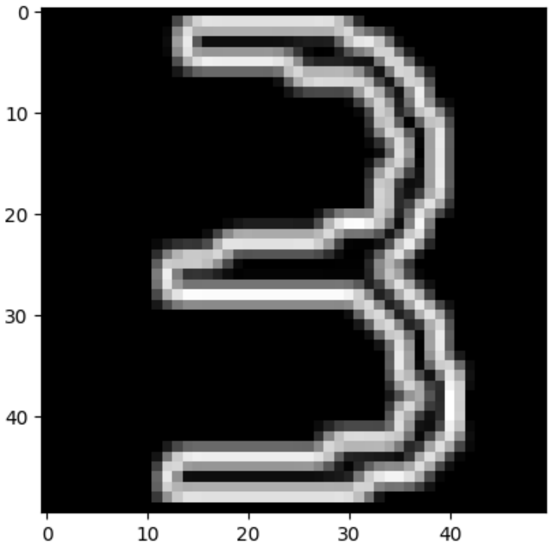
Backward/transposed convolution increases-dimensions/upsamples.

This means using a different kernel to the forward convolution we can recreate the original image.

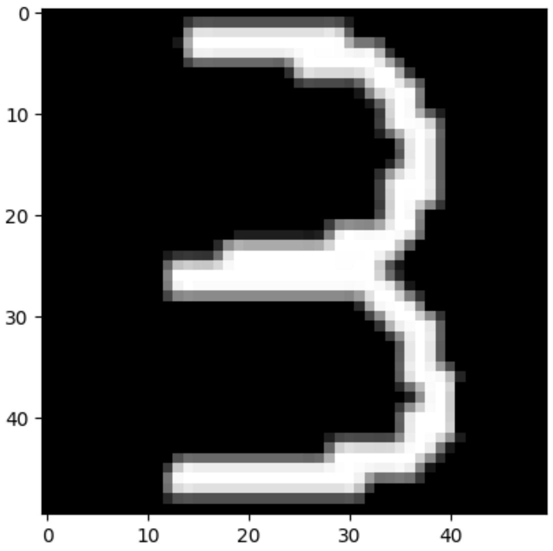
$$\begin{bmatrix} 3 & 5 \\ 2 & 2 \end{bmatrix} ** \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 5 & 6 & 7 & 8 \\ 3 & 2 & 1 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Conv2DTranspose – An Aside

- Not always trivial to know what the form of the kernel is that we need to get from input to output.
- Thankfully, we know our input and output, so we can train this kernel to find the optimal solution using training tools ☺



$$** \begin{bmatrix} ? & \cdots & ?? \\ \vdots & \ddots & \vdots \\ ???? & \cdots & ??? \end{bmatrix} =$$

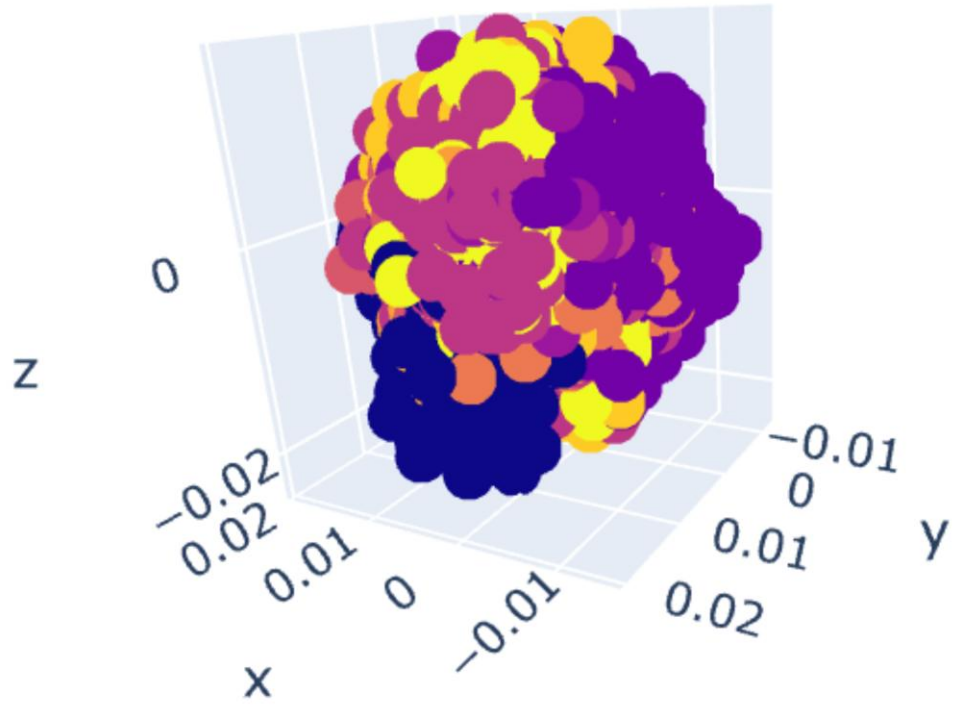


Variational AutoEncoder – Latent Space

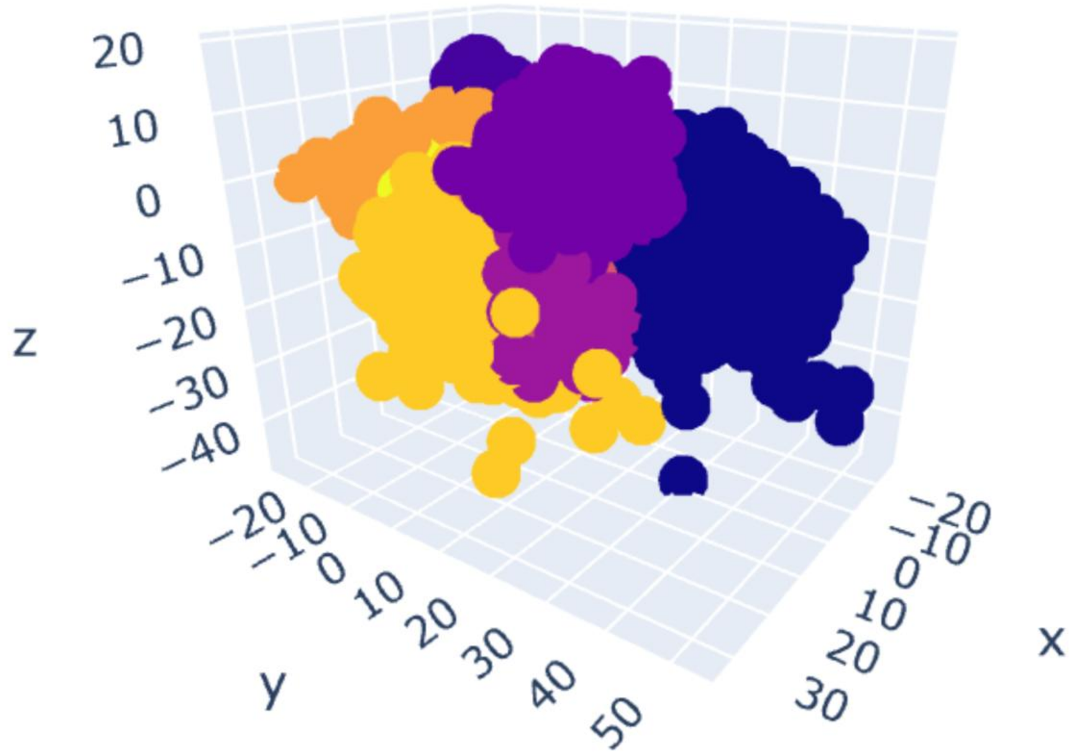
- A VAE is a good example of an unsupervised neural network.
- This network design has a bottleneck which reduces the amount of information which can flow through it.
- The theory behind this is that it forces information to be encoded into the latent-space which represents the dataset being trained on.
- This is still effectively a ‘black-box’ style of network as we don’t have a clear interpretation of what a single weight in the system may represent in the real world.

Variational AutoEncoder – Latent Space

Before training a network, latent space distribution is effectively random



After training the distribution of data within the latent-space is more strongly grouped & ordered



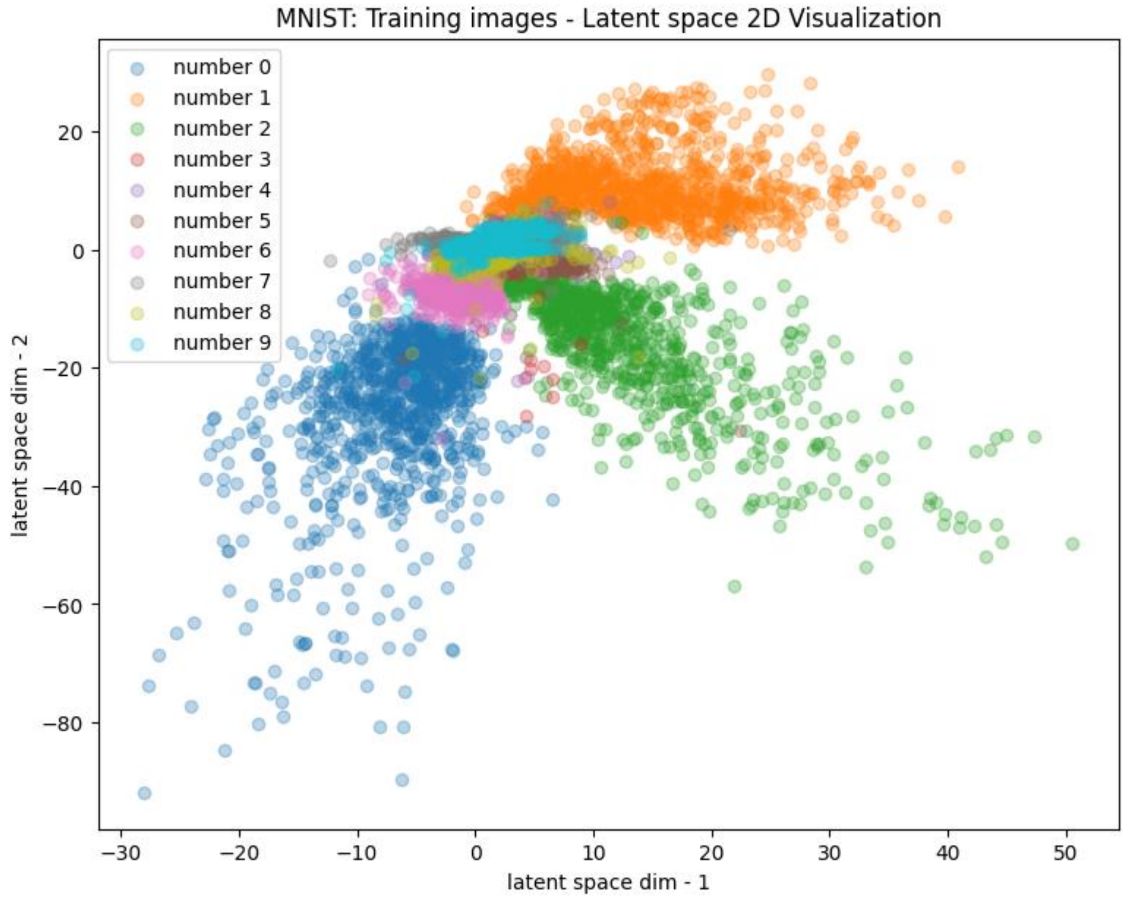
Variational AutoEncoder – Latent Space

We want the latent space in a VAE to allow us to share features between similar species.

e.g.

The numbers **9** and **4** are often ***similar***, so we expect them to potentially be close in the latent-space.
If this is the case our model has encoded most of the ‘shapes’ in a way that extracts out ‘**key features**’.

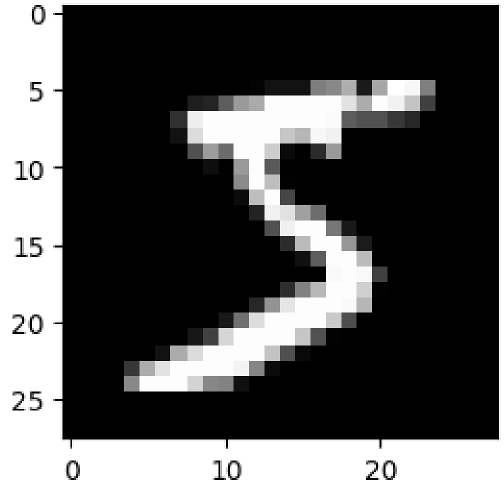
An overly-optimized or overly-constrained model may have these 2 numbers widely separated. In this case the model has learned about the 2 numbers, but not that they look similar.



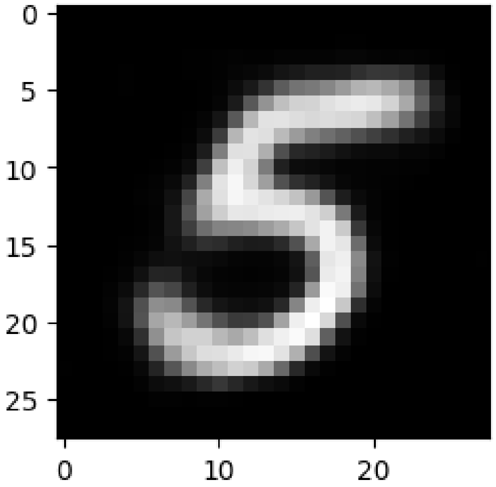
Variational AutoEncoder – Noise Reduction

- One of the main uses of a VAE network design is to perform noise reduction.
- Reducing an image to its key features and re-constructing it allows us to do this.

Input Image



Re-Generated Image using Latent-Space representation



CNN – Other Applications

- Other examples of CNN usage is in anomaly detection.
- This relies on understating what the latent-space of a trained model looks like.
- This relies on the fact that a trained model will give an unexpectedly large value in this space for an anomalous datapoint which isn’t described well by the dataset the model was trained on.

CNN – Model Training (1)

- CNN or CDN(N) models are complex models which still have many free parameters to be optimized when training on a given dataset.
- This means that datasets that give the best results for these models we want the largest datasets possible.
- In the case that the input data is translationally invariant we can apply translations on input images to make sure that our model doesn’t over-optimize.

CNN – Model Training (2)

- Imagine a dataset was composed of people waving.

  People tend to wave with their dominant hand.
  ~90% of people are right-handed.

  Model might assume that only right-handed people wave.
  (or that people waving are right-handed…)

- Given that a person in a mirror still tends to look like a person.

  We can translate our data randomly to avoid this bias.

CNN – Model Training (3)

- In the case of certain models and certain datasets invariance is a good and desirable characteristic.

- However, if we're looking to build automatic number recognition for documents scanned with the *correct* orientation…

$$6 \approx 6 \neq 9$$

- Translations are good & useful when used correctly, but they can also cause problems when not checked.

Workshop – Tomorrow

- Tomorrow's workshop will be applying CNN to situations which we've discussed today.
- I will mention some common mistakes from last week's session at the start.
- There is a bit more to do this week which reflects this is a more difficult part of the course.
- Good Luck, and as always, I'm happy to answer any questions ☺