INTRODUCTION TO AI – DEEP LEARNING

Philip Mortimer

AI systems will enable doctors to diagnose diseases and treat people better, so blocking that progress is probably one of the worst things you can do for making the world better.

— Mark Zuckerberg —

AZ QUOTES

RECAP

- Discussed perceptrons – units that take a weighted sum of inputs and a bias to produce a value.
- These units can be combined to draw a straight line that categorises a dataset
- Through the use of a non-linear activation function, these units can draw any line (and thus solve any classification problem).
- Combining these units creates a neural network – which forms the basis of modern ML techniques
- We discussed how we can use labelled datasets to measure how far off the correct value an AI is. We defined this as the loss function. This is sometimes also called the cost function.

RECAP

- Using clever maths, we can calculate the gradient of the loss function with each individual weight and bias of a neural network.
- If we travel a small distance in the opposite direction of this gradient, we will go in the direction that reduces the loss most quickly.
- If we compute this gradient for every weight and bias and alter each weight and bias by a small amount in the direction of the calculated gradient, we will reduce our loss over time.
- The size of the step is normally small and is called the learning rate.
- In supervised learning, we have a large dataset (say of images of handwritten digits). We split this set into a training set, testing set and validation set.

RECAP

- In order to calculate the derivative of the loss with respect to the test data, we must use a batch of images in order to obtain an accurate estimate of the true loss. However, if the batch size is too large, the network will take too much time to train
- We also need to define a neural network architecture. This involves defining the sizes of the input and output layers. Additionally, we need to decide how many neurons should be present for each hidden layer of the network
- We also discussed how we use epochs to decide how long the network should train for.

EXAMPLE

- Supervised learning is most definitely the most commonly used AI technique
- It is also very tricky to grasp
- To reinforce some of the concepts, let's train a neural network to classify tumours as either malignant (cancerous) or benign (non-cancerous)
- We will use a dataset of ~3,500 images
- Each of these images is 224 * 224 pixels wide and in colour. This means that each pixel has three values that denote it's rgb components. Each image thus can be represented using 150, 000 input neurons.

CANCER AI

- That's a lot of parameters. In practice, we would probably employ more advanced techniques to reduce this.
- For example, we may only choose to look at the image as grayscale.
- However, for our example, let's not do this.
- For our output, we know that the image is labelled as either benign or malignant.
- Similar to our MNIST example, let's use two output nodes. Output node #1 can be interpreted as "the probability that the tumour is benign". Similarly, output node #2 is "the probability that the tumour is malignant".
- For our diagnosis model, we'll simply choose the most probable one as our diagnosis.

CANCER AI

- This is a much more complicated problem than MNIST and therefore, it stands to reason that we will need a bigger network.
- Let's use five hidden layers with the following sizes: 170, 800, 800, 500, 100.
- This value has been obtained through a combination of intuition and trial and error.
- This gives us the following architecture:
- 150,528 – 170 – 800 – 800 – 500 – 100 – 2
- That's a total of 26,818,332 weights and biases to be updated. This is a large network and will be expensive to train

CANCER AI

- As the network is so large, we will need a large enough batch size to give a full overview of the network's loss
- However, the batch size should be fairly small in order to ensure the network trains over a reasonable time frame. Let's choose a batch of 64 images
- For the learning rate, we will use 0.001. This often proves to be a good value
- We will also use an optimiser called Adam

ADAM OPTIMISER

- The Adam optimiser is a more advanced way of updating the weights and biases in our network.
- Recall that we are currently using this update method:
- $w_i = w_i - \frac{dl}{dw_i} * r$
- $b_i = b_i - \frac{dl}{db_i} * r$
- The Adam optimiser aims to capture an idea of momentum into updating the weights. Values that have not been changed for a while are likely to be close to optimal, whilst values that were changed more recently are less likely to be.
- It also allows for an adaptive learning rate. Initially, a large learning rate is useful to get us close to the minimum. However, as training goes on, we want to adjust our weights by a smaller amount to help locate the exact local minimum

ADAM

- The maths is complicated (and I don't expect anyone to understand it), but I thought I'd include it for completeness
- All you need to know is that is generally a much more effective way to train a model

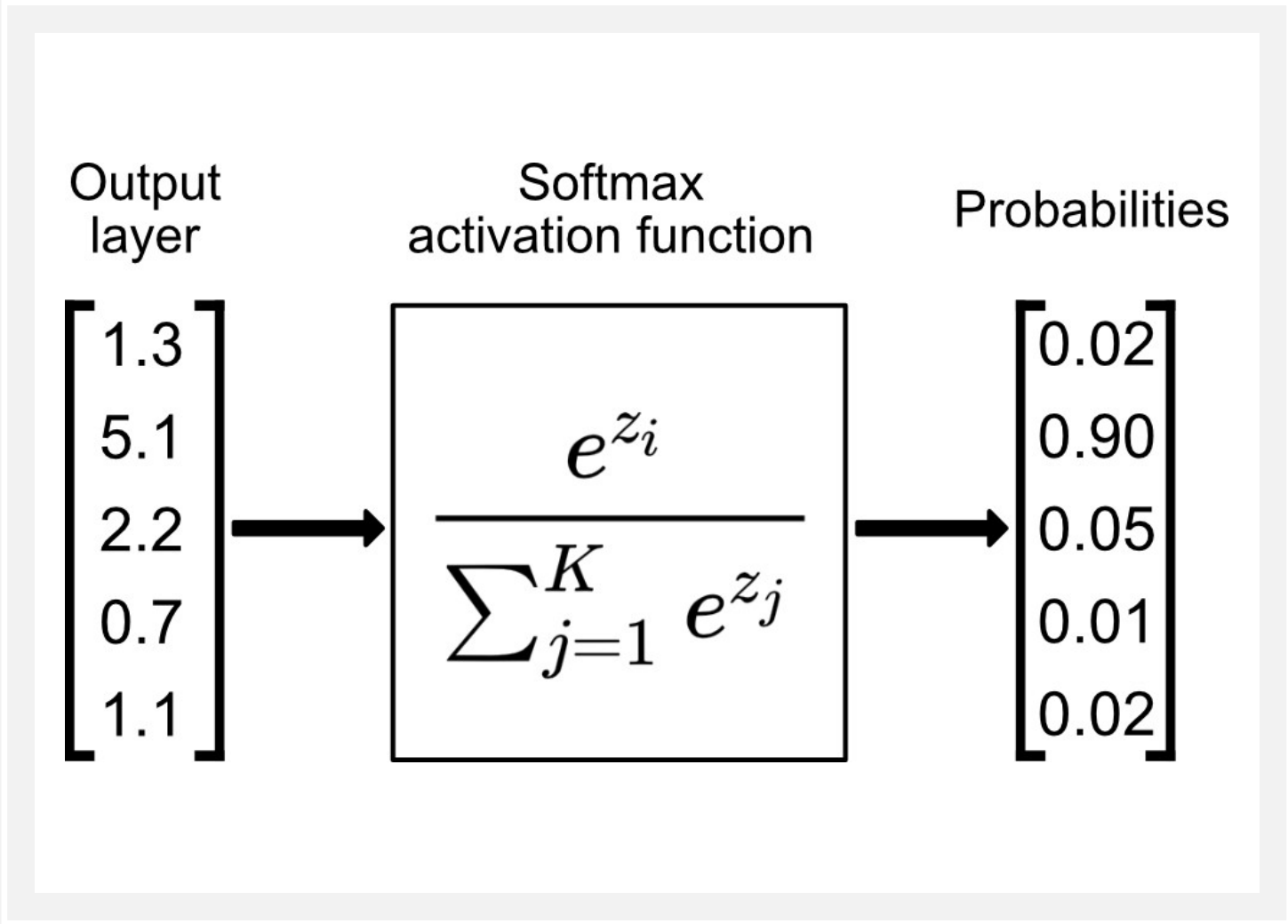
Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

CANCER AI

- We also need to decide what activation functions to use in our neural network.
- For our cancer AI, I have elected to use the reLu function in all layers bar the output layer. For the output layer, I will use the softmax function.
- reLu is used as it has been proven to be effective for general use. It is also really cheap to use in calculations when compared to sigmoid. This is because reLu uses a simple if statement, whilst the sigmoid function uses a number of exponential functions

SOFTMAX

- The SoftMax function is often used in classification tasks (and in conjunction with the sigmoid function)
- It takes in a vector as an argument and produces a vector as an output. The key property of this vector is that the sum of the output elements is 1. This is very useful as we often interpret our model outputs as probabilities.
- When using a softmax activation function, the categorical cross entropy loss function is often used.



CATEGORICAL CROSS ENTROPY

- Like MSE, this is an example of a loss function
- The idea is that for one hot encoded outputs, the loss is only judged for the node that should have the value of one.
- Again, this is a complicated formula, so no need to fully understand it

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

CANCER AI

- As this problem is rather complex, I will give the network lots of training time. I will train the network for 30 epochs
- This gives us the following parameters:
- **Learning rate = 0.001**
- **Optimiser = Adam**
- **Batch size = 64**
- **Epochs = 30**
- **Architecture = 150,528 – 170 – 800 – 800 – 500 – 100 – 2**
- **Activation for non-output layers = reLu**
- **Activation for output layer = SoftMax**
- **Loss function = categorical cross entropy**

CANCER AI

- This network took many hours to train on my laptop using my machine learning library.
- Typically neural networks are trained using specialist software and on very powerful computers that have graphics cards.
- However, after training the model overnight, I did produce a neural network to address the task.
- You can find my training code and model here: https://github.com/philipmortimer/AI-Course/tree/main/Programs/Part%205/Tumour-Classification-Using-Machine-Learning
- To open the model, simply open the ".jar" file

CANCER AI

- Our AI is slow and took a long time to train
- However, it does outperform dermatologists at cancer diagnosis
- This clearly demonstrates the power of machine learning models
- However, unlike the MNIST example, our neural network is far less confident when making predictions
- There are a few clear flaws in our network.
- Firstly, the input space is too large for the dataset
- We should reduce this through compression and Gray scaling

CANCER AI

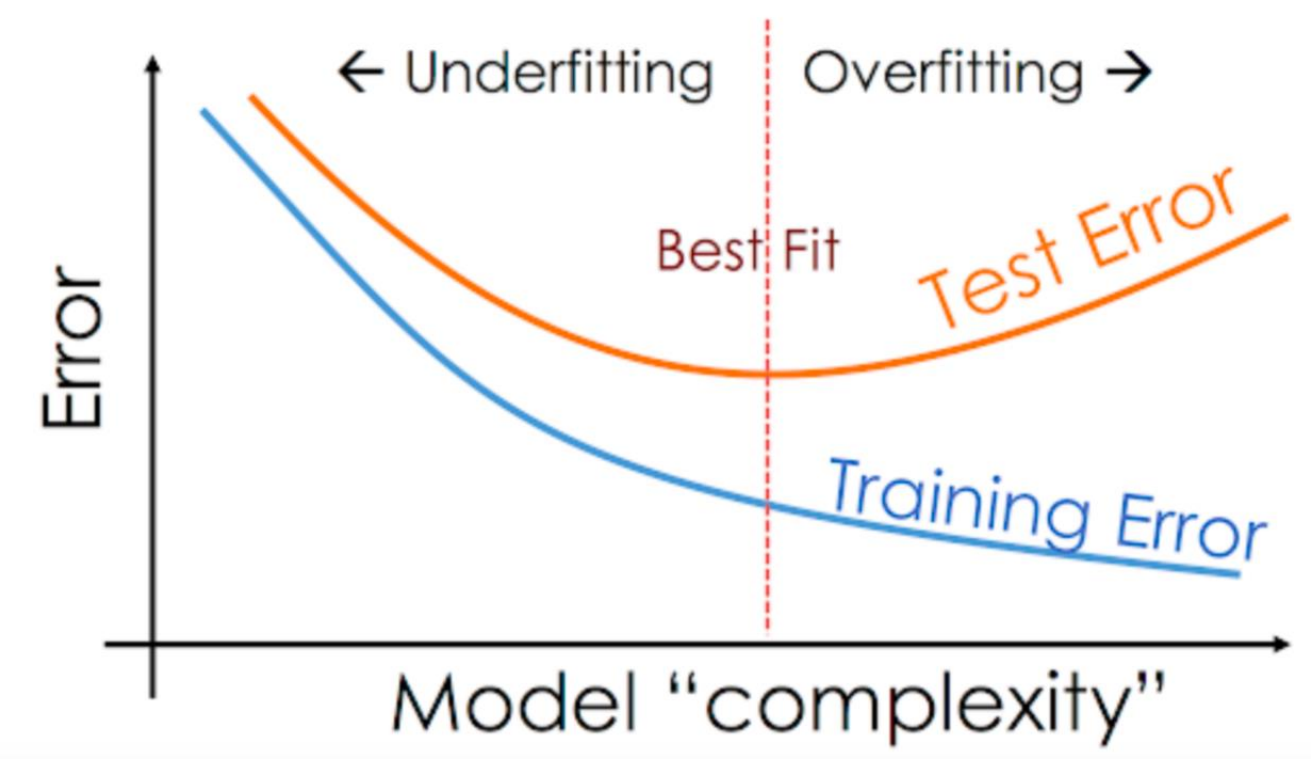
- Our neural network architecture is referred to as a feedforward neural network
- Using this type of architecture for images is not entirely appropriate.
- Normally, each input node should represent a consistent metric
- In our case, a given input node represents a single pixel
- However, pixels don't consistently represent something
- This is because images of tumours aren't all perfectly centred
- We would like a neural network that is able to extract **meaningful** features and use these to produce an AI that is less dependent on perfectly centred images

OVERFITTING

- Additionally, neural networks often face an issue when it comes to training on large datasets
- When training initially, our network learns general features about data points. For example, it may learn that tumours that are red are more likely to be cancerous
- However, neural networks often have more parameters than strictly needed.
- So once they have learned all the features that can be extracted from a network, they will start to use their remaining capacity to memorise data points from the training set. This leads to the training loss continuing to decrease
- However, this also makes the model less accurate when being deployed on real world data (when this memorisation is of no help to it)
- This is called **overfitting**

OVERFITTING

- When the test loss stops decreasing, we know that our model has stopped learning meaningfully
- We should stop training at this point



REGULARISATION

- To prevent overfitting, regularisation techniques are used to reduce the complexity of models
- There are three commonly used techniques: L1, L2 and dropout regularisation
- L1 and L2 are rather mathematically complicated but both essentially involve reducing the value of weights over time. This leads to some neurons being negligible to the network output and thus reduces complexity

DROPOUT REGULARISATION

- The idea behind drop out regularisation is that for each batch, a percentage of all neurons are deactivated (i.e. always return 0).
- This forces the network to use the whole neural network infrastructure when producing an output, instead of relying on one specific output path
- This prevents memorisation as it is much harder to memorise paths when only a random subset of the network functions at a given point in time

CONVOLUTIONAL NEURAL NETWORKS

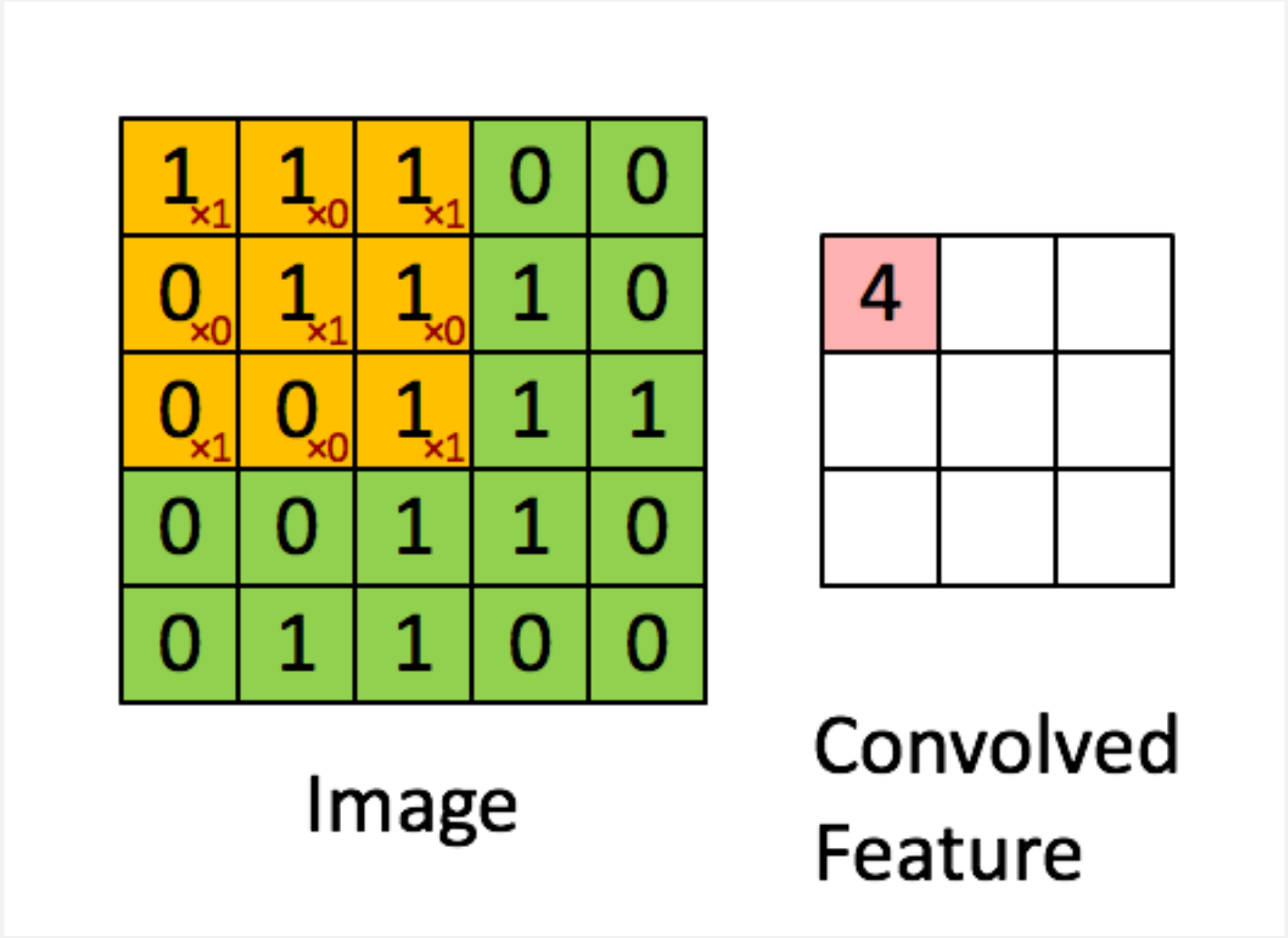
- A Convolutional Neural Network (CNN) is a very common neural network structure.
- CNN's are more commonly used than ANN's (what we've looked at previously) and have several advantages over them
- They extract high level features from datapoints and thus are rotationalyl and positionally invariant. In short, they "learn" more intelligently
- CNN's are commonly used on images

CNN

- CNN's combine artificial neural networks (ANN's) like we've previously discussed with an algorithm that extracts high level features from very complicated data.
- This is very useful for image classification, as images may have millions of individual pixels. By extracting high level features (e.g. edges, gradients etc.), we are able to train AI on large datasets of very large images. This is just not possible with ANN's.
- As CNN's learn from high level features, they do not require all data items to follow the same format.
- In the case of tumour recognition, a CNN would mean that tumours don't have to be right in the centre of the frame.
- In short **CNN = feature extraction + Neural Network**
- We already know exactly how the neural network functions

FEATURE EXTRACTION

- Typically CNN uses a combination of convolutional layers and pooling layers
- Convolutional layers use a sliding window to reduce the dimensionality of an image. This sliding window uses complicated matrix calculations to extract features from an image



Credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

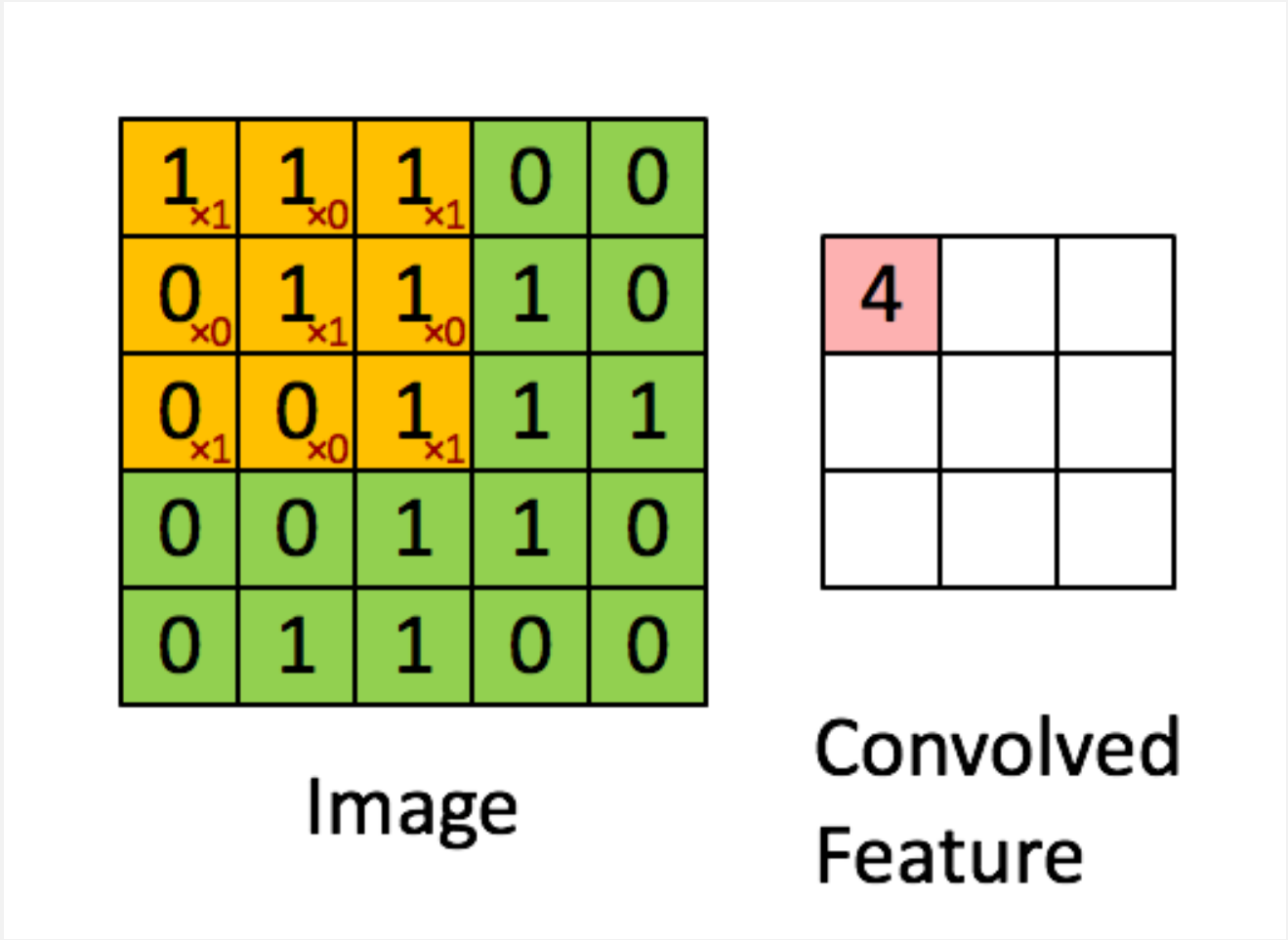
- The idea here is we have a kernel matrix with the following values:

1 0 1

0 1 0

1 0 1

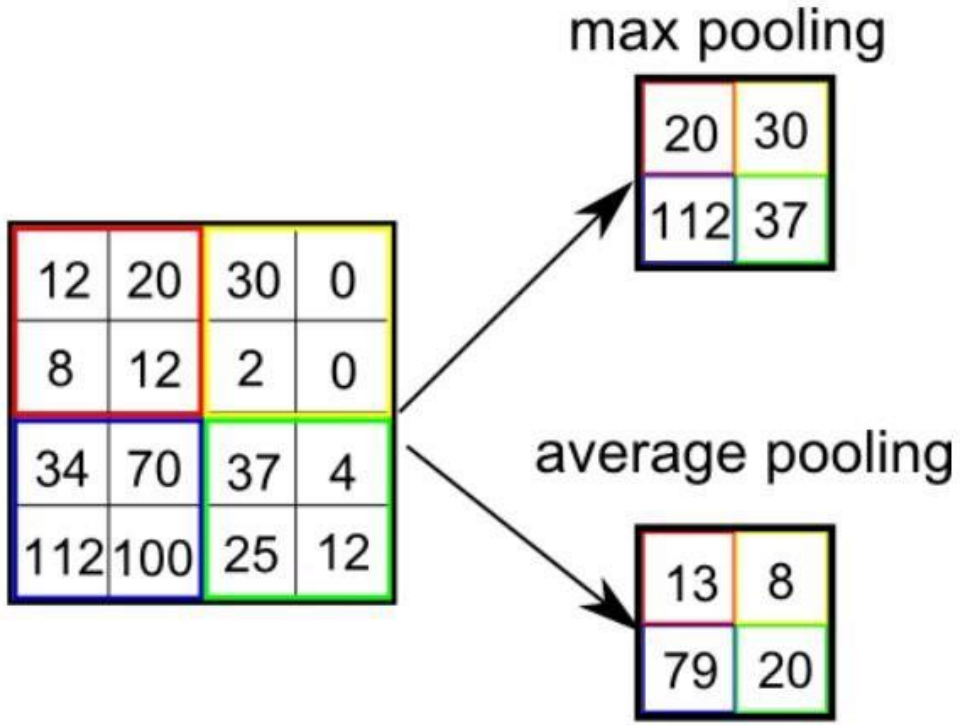
- This matrix is multiplied by the values in our window to produce our smaller convolved feature matrix
- This is a good practical way to reduce the data size whilst still maintaining the important high level features



Credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

MAX POOLING

- Following our convolutional layer, we often still have a very large output space.
- In order to reduce this further we use a technique known as max pooling
- The idea is similar; we have a small sliding window that looks at a subset of the convolved features. It takes the maximum value of each of those pixels as it's new value
- This turns out to be an effective way to reduce dimensionality whilst retaining the important features



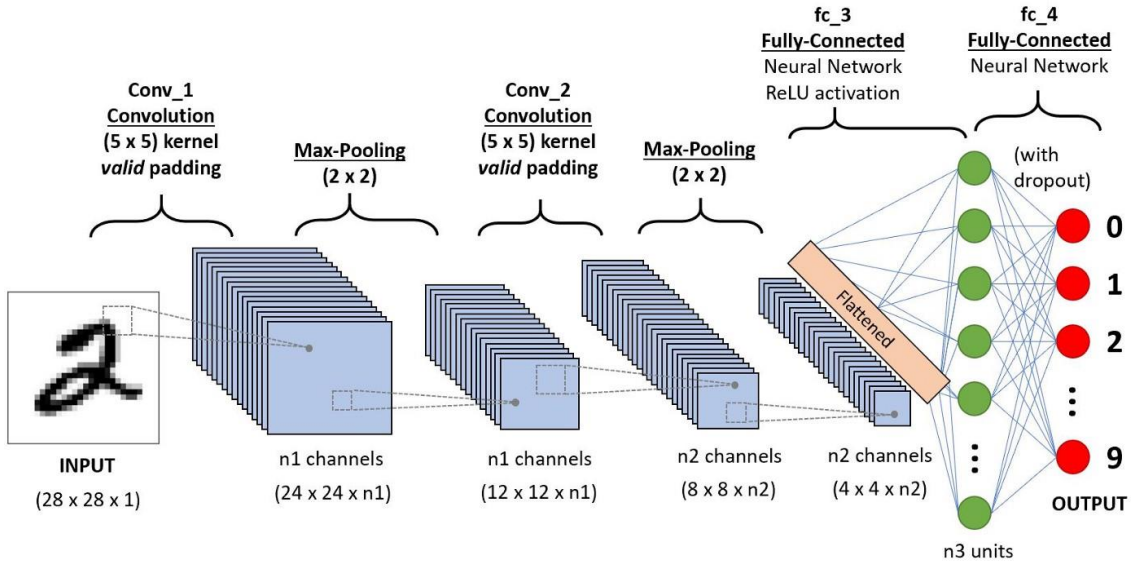
3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

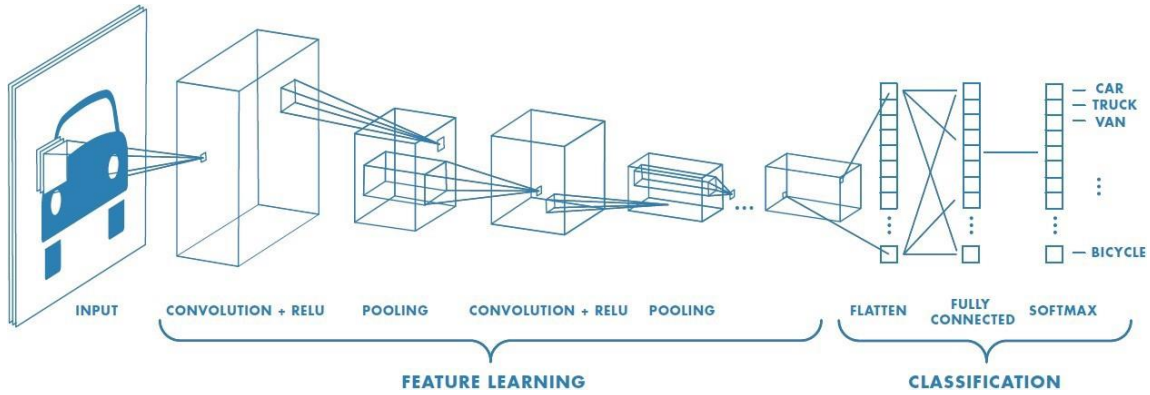
3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

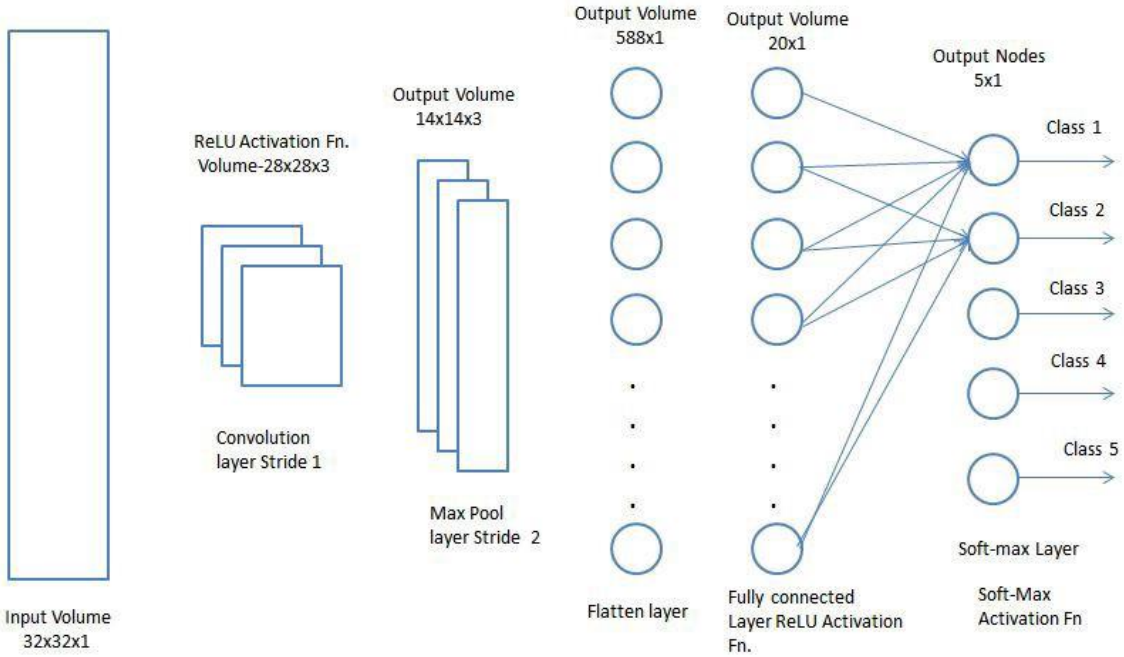
Credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

CNN

- Typically CNN's will use Convolutional layers and pooling layers a number of different times. The final output of the system will be attached to a fully connected neural network to learn from the extracted features







Credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

CNN

- It turns out CNN's are the most versatile neural network architecture choice and perform very well on a range of problems
- CNN's are especially powerful in Computer vision domains (e.g. image recognition)
- There are a whole range of other architecture types. However, CNN and ANN's are by far the two most common ones.
- They can be applied to countless problems.

RNN

- I will also briefly mention that current research often focus on recurrent neural networks (RNN).
- These use long short term memory networks (LSTM).
- The maths is very complicated.
- CNN and ANN take a single datapoint and produces a result
- LSTM's can process a series of datapoints, allowing them to be used in temporal tasks more effectively
- However, any more detail is far beyond the scope of this course.
- You should simply be aware that they exist and are good for processing series of data (often along the time axis). For example, meteorological readings may be used by an LSTM to produce a weather forecast.

LIBRARIES

- In order to train neural networks, seriously optimised and low level code is needed to train them in a reasonable time window
- Many companies have private ML libraries that have been highly tested and optimised
- However, the vast majority of programmers use a small range of publicly available and free to use libraries.
- These libraries are open source and can be used for all purposes (including commercial) free of charge

LIBRARIES

- Most machine learning projects tend to use “TensorFlow” or “PyTorch”. These are libraries which can be interacted with through the python programming language.
- Model creation and training can be achieved with just a few lines of code
- These libraries have been written by a large team of programmers. This means all functionality is very well tested and that the implementation is efficient
- Additionally, there is detailed documentation available in order to easily use it
- As these libraries are so widely used, there’s a large community that can potentially offer support should any issues be encountered

TENSORFLOW

- TensorFlow is the most used ML framework in the world
- Python is used to implement the API. However, the actual implementation is extremely fast C code
- TensorFlow can be installed here: https://www.tensorflow.org/install/pip
- Note that a working installation of Python (amongst other things) may be required depending on your system.

TENSORFLOW

- To demonstrate TensorFlow, I will use it to solve the MNIST problem we solved earlier. I'll first do this using an ANN.
- The script is loosely derived from this: https://github.com/tensorflow/docs/blob/master/site/en/tutorials/quickstart/beginner.ipynb
- The script can be found here (in "annMnist.py"): https://github.com/philipmortimer/AI-Course/tree/main/Programs/Part%205/TFMnist
- Recommended IDE: "PyCharm"

TENSORFLOW

- Let's look at code!

CNN

- Next, let's make a CNN for the MNIST task.
- Code can be found here (at "cnnMnist.py"): https://github.com/philipmortimer/AI-Course/tree/main/Programs/Part%205/TFMnist
- This model is much bigger, but achieves a remarkable performance of ~99.3% after 5 epochs
- Lets look at the code!

SUMMARY

- We reinforced our understanding of training of artificial neural networks by demonstrating how we can train one to diagnose cancer.
- We discussed the adam optimiser
- Softmax activation
- Categorical cross entropy loss
- We discussed CNN as a common ML model that uses feature extraction to improve on the ANN model.
- We looked at popular deep learning library TensorFlow and made our own models using a CNN and ANN to recognise handwritten digits.

QUESTIONS

FURTHER READING

- CNN - https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- TensorFlow tutorials - https://www.tensorflow.org/tutorials