Overview of Deep Learning

Click on a question number to see how your answers were marked and, where available, full solutions.

Question Number	Score
Question 1	1 / 1
Question 2	1 / 1
Question 3	1 / 1
Question 4	1 / 1
Question 5	1 / 1
Question 6	1 / 1
Question 7	1 / 1
Question 8	1 / 1
Total	**8 / 8 (100%)**

The pass rate for the questions in the tutorial is 50%, if you score less than this you might want to revisit the questions you had difficulty with and read some of the resources pertaining to that topic.

Thank you for using this tool, in order to improve the system please complete the questionnare linked below:

https://forms.ncl.ac.uk/view.php?id=6719176 (https://forms.ncl.ac.uk/view.php?id=6719176)

Performance Summary

Exam Name:	Overview of Deep Learning
Session ID:	16838052913
Exam Start:	Thu Dec 10 2020 11:53:47
Exam Stop:	Thu Dec 10 2020 11:54:28
Time Spent:	0:00:41

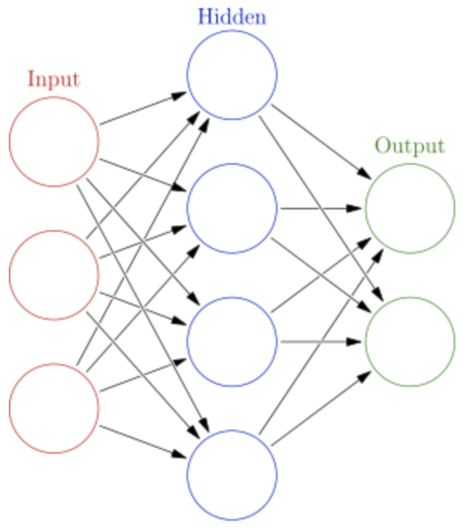
Question 1

Artificial Neural Networks (ANN)

What is an ANN?

As the 'neural' element of the name suggests, artificial neural networks are algorithms influenced by the brain and are loosely based on how humans learn. They are one of the main tools used in deep learning and are excellent at finding patterns within data.

In an ANN we have a unit for calculation called a *neuron*, these neurons are connected by *weighted values.* We have sets of neurons called *layers.* In an ANN we have an *input layer*, multiple *hidden layers* and an *output layer.*



What is needed for an ANN?

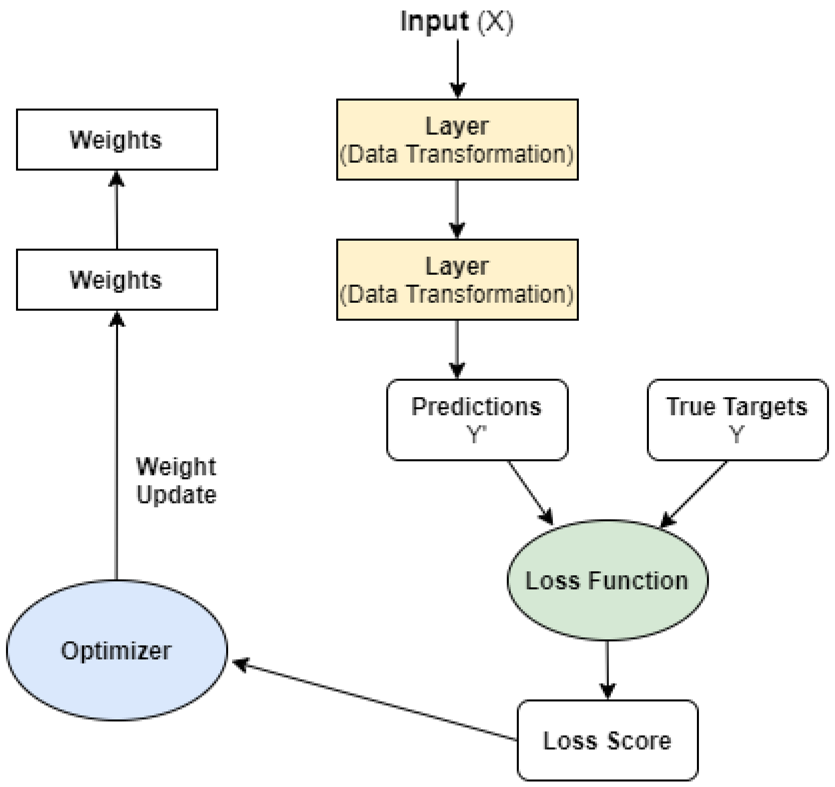
An ANN requires the following:

- **Layers** - these are the main building blocks of a neural network model. A layer works as a sort of filer for the data, they are data processors which extract useful representations.
- **Input data and comparable targets**
- **Activation function -** an activation function is linked to each neuron and determines whether the neuron should fire or not.
- **Loss function** - a function which defines the type of feedback signal used for learning, it is how the model will be able to measure its performance on the training data.
- **Optimizer** - this influences how learning occurs. The optimizer is how the model updates itself based upon the data and the loss function.

How does an ANN work?

The following explanation and image by F.Chollet (2018) nicely illustrates how a ANN works:

The network (model), composed of layers that are chained together, maps the input data to predictions. The loss function then compares these predictions to the targets, producing a loss value (a measure of how well the network's predictions match what was expected). The optimzer uses this loss value to update the network's weights.



Different types of neural network

Over the course of the Deep Learning tutorial we will cover a number of types of ANN including *feedforward neural networks*, *recurrent neural networks (RNN)* and *convolutional neural networks (CNN).*

References and Resources:

F. Chollet (2018) Deep Learning with Python. New York: Manning

S. Raschka , V. Mirjalili (2017) Python Machine Learning. Birmingham: Packt Publishing

https://medium.com/technology-invention-and-more/everything-you-need-to-know-about-artificial-neural-networks-57fac18245a1 (https://medium.com/technology-invention-and-more/everything-you-need-to-know-about-artificial-neural-networks-57fac18245a1)

NN Architecture Image from: https://upload.wikimedia.org/wikipedia/commons/thumb/4/46/Colored_neural_network.svg/300px-Colored_neural_network.svg.png (https://upload.wikimedia.org/wikipedia/commons/thumb/4/46/Colored_neural_network.svg/300px-Colored_neural_network.svg.png)

https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/ (https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/)

In an ANN, which of the components extracts representations from the data?

Loss Function [✓ Layers] Optimizer

Expected answer:
Loss Function [✓ Layers] Optimizer

✓

✓ You chose a correct answer. You were awarded **1** mark.

You scored **1** mark for this part.

Score: 1/1 ✓

Question 2

How Humans Learn

Neural networks are a pivotal topic to understand within the field of AI, the aim of this tutorial is to aid your comprehension of this model through explanation of the origins of this domain-human learning.

What is a neuron?

The brain contains billions of information processing units called neurons. Neurons are connected via synapses which pass on the information between neurons. The neurons form complex networks (neural networks) which conjoin and disssociate as we learn and unlearn- these changes between connections and pathways is referred to as plasticity.

Each neuron may be connected to up to thousands of other neurons and information flows in a number of directions due to the amount of feedback connections in the brain.

What is the relation between artificial neural networks and how humans learn?

Neurons are represented in an ANN by the different nodes in a layer, the synapses are represented by the weighted values which connect the different layers via the individual nodes. The connections between the neurons have differing strengths, therefore the activity in one neuron can affect the activity in another. It is through these fluctuating strengths that the model learns to map input patterns to output patterns.

Regarding the feedback connections in the brain, the backpropagation algorithm (we will cover this in a later tutorial) can be seen as a form of feedback as it feeds an error from the output units back to the input units.

J.Hawkins (2004) outlines one of the major differences beteen ANNs and human learning in that "artificial auto-associative memorries fail to recognise patterns if they are moved, rotated, rescaled, or transformed in any of a thousand other ways, whereas our brains handle these variations with ease." Clearly AI research has a far way to go before we realistically can model human learning.

References and Resources:

Collins (2019, 2nd Ed) "Neuroscience for Learning and Development" London: Kogan Page Limited

J. Hawkins (2004) On Intelligence. New York: St. Martin's Press.

In an ANN what is it that represents the synapse?

Nodes [✓ Weighted Values] Plasticity

Expected answer:

Nodes [✓ Weighted Values] Plasticity

✓

✓ You chose a correct answer. You were awarded **1** mark.

You scored **1** mark for this part.

Score: 1/1 ✓

Question 3

Gradient Descent

Optimization Review

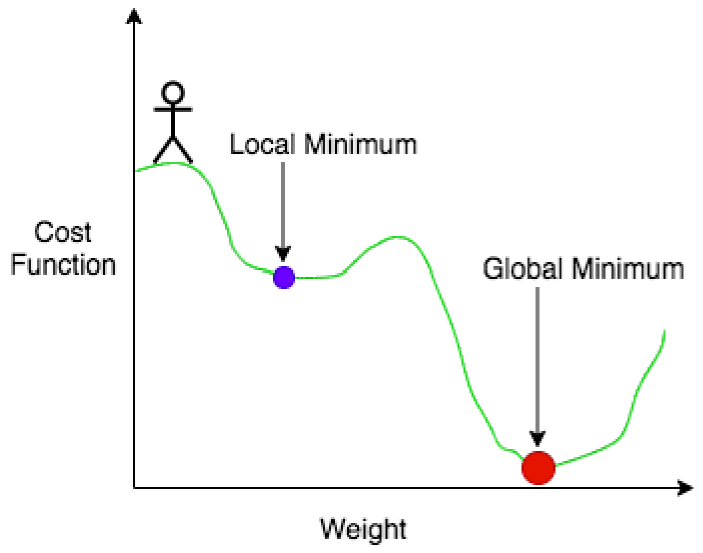
Optimization refers to the process of altering a model to get the best possible performance in the training data. An *optimizer* updates the model in response to the output of the loss function which you may also see referred to as the cost function or error function. However the *loss function* calculates the error for a single training example and the *cost function* is the average of all of the loss functions for the entire training set.

We can use gradient descent, which is an optimization algorithm to discover the weights which minimise our cost function.

Gradient Descent

The gradient is the derivative of a tensor operation (for further information on derivatives refer to the Maths and Stats tutorial-linked below), the derivative is good for mimimizing the function because it can tell us how to improve y by telling us how we should change x.

An easy way to think of gradient descent is as a person climbing down a hill, we want to climb down until we reach some minimum cost.



Gradient descent is an iterative algorithm in which we take a step in the opposite direction of the gradient to reduce the loss.

We must decide the size of the step to take, this is called the *learning rate.* If we have a *high learning rate* we run the risk of going too far past the lowest point, however we do cover more ground quickly with a higher learning rate. If we have a *lower learning rate* we can be more precise, however the process will be more time consuming.

Local Minimum - this is where the loss function is at its mimimum but only for that local region.

Global Mimimum - the lowest loss we can a acheive accross the full domain.

Once we converge into a local minima we can get stuck and it can be very hard to get out of this mimimum and to the global minimum.

Stochastic Gradient Descent

One of the methods to help gradient descent get out of the local minima is to use stochastic gradient descent. The method of gradient descent described above is called *batch gradient descent* as it processes all of the samples in a single batch.

Stochastic gradient descent differs as we only compute the gradient of the loss of *one* randomly sampled example.

Due to the process of calculating the losses step by step, stochastic gradient descent can be computationally expensive, an alternative method called *mini-batch gradient descent* can solve this issue.

Mini-Batch Gradient Descent

In this form of gradient descent we use a fixed number of samples to form a mini-batch.

Choosing the size of the mini-batch is important as we want to maintain enough stochasticity to still prevent the local minima.

Recap:

Batch Gradient Descent - uses the full dataset

Stochastic Gradient Descent - uses a single random sample

Mini-Batch Gradient Descent - uses a fixed number of samples

References and Resources:

Maths and Stats Tutorial: https://numbas.mathcentre.ac.uk/exam/10572/maths-and-stats-for-machine-learning/embed/?token=d633f234-aaa4-434c-b549-0c0eebfb93a5

https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html (https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html)

https://machinelearningmastery.com/gradient-descent-for-machine-learning/ (https://machinelearningmastery.com/gradient-descent-for-machine-learning/)

https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent (https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent)

https://www.deeplearningbook.org/contents/numerical.html (https://www.deeplearningbook.org/contents/numerical.html)

F. Chollet (2018) Deep Learning with Python. New York: Manning Publications Co.

S. Raschka , V. Mirjalili (2017) Python Machine Learning. Birmingham: Packt Publishing

https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/ (https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/)

What is the learning rate?

[✓ The step size] The gradient The direction

Expected answer:

[✓ The step size] The gradient The direction

✓

✓ You chose a correct answer. You were awarded **1** mark.

You scored **1** mark for this part.

Score: 1/1 ✓

Question 4

Backpropagation

Backpropagation is a very important algorithm within multi-layer neural networks. It allows us to understand how a change in the weights and biases can change overall network behaviour.

Backpropagation occurs after each forward pass through the network, backpropagation completes a backward pass and adjusts the model parameters (the weights and biases).

The overall aim of backpropagation is to minimize the cost function through adjusting the weights and biases.

How Does it Work?

Backpropagation computes the partial derivatives of the cost function- it allows us to see how quickly the cost changes when we change the weights or biases.

We need to discover the loss at every node in the neural net as we need to ascertain which node is responsible for the majority of loss in every layer.

Once we've discovered which node is responsible for the most loss we can penalize this node by assigning it a smaller weight, therefore lessening the total loss of the model.

This tutorial aims to give a broad overview of the backpropagation algorithm, for greater detail into this algorithm have a look at the additional resources below.

References and Resources:

http://neuralnetworksanddeeplearning.com/chap2.html (http://neuralnetworksanddeeplearning.com/chap2.html)

https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd (https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd)

S. Raschka , V. Mirjalili (2017) Python Machine Learning. Birmingham: Packt Publishing

https://www.deeplearningbook.org/contents/mlp.html (https://www.deeplearningbook.org/contents/mlp.html)

https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7 (https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7)

https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/ (https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/)

What does the backpropagation algorithm aim to minimize?

[✓ Cost function] ○ Number of layers ○ Number of hidden units

✓ Expected answer:

[✓ Cost function] ○ Number of layers ○ Number of hidden units

✓ You chose a correct answer. You were awarded **1** mark.

You scored **1** mark for this part.

Score: 1/1 ✓

Question 5

Convolutional Neural Networks (CNN)

Convolutional Neural Networks are a collection of models which are inspired by the process in which human brains recognise objects. They are especially useful for processing data with a grid like topology such as time series data and images. They are commonly used for image proccessing and classification.

A CNN works by applying filters to capture both spatial and temporal dependencies.

CNNs differ from other neural networks in that they use convolution in at least one of their layers as opposed to general matrix multiplication. The convolution operation extracts high level features (e.g. edges) from the input image.

CNN Architecture

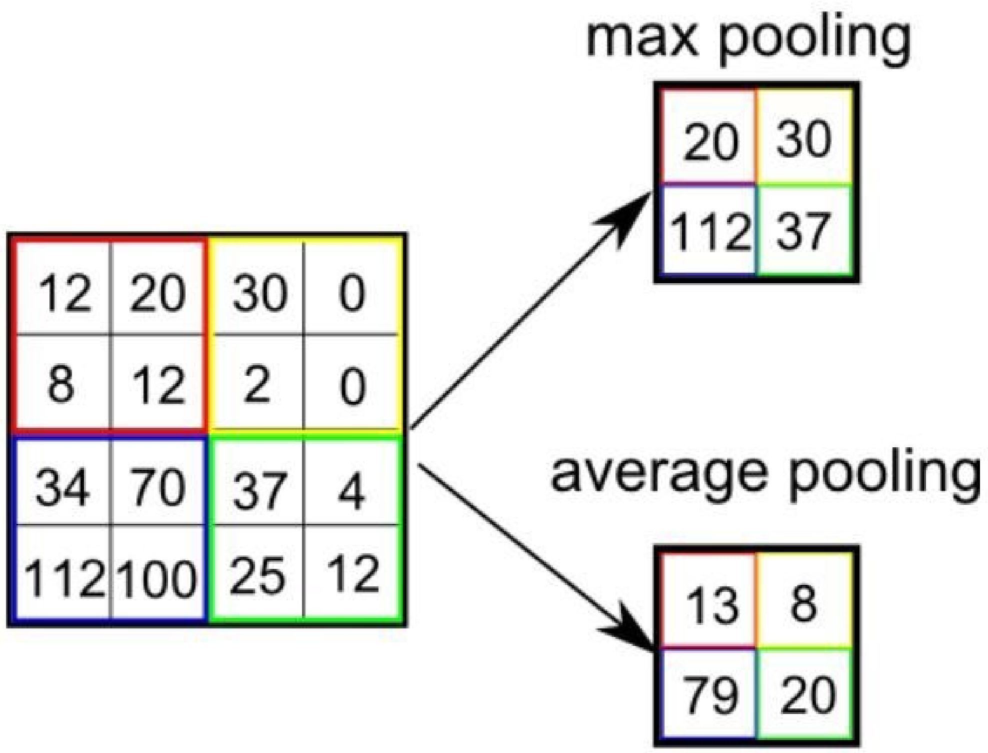
Convolution Operation - convolutions perform over 3D tensors which we refer to as *feature maps*, these have a height, weight and depth axis. The convolution operation obtains segments from the input feature map and applies some form of transformation (undertaken by the kernel), it does this for all of the segments and produces an output feature map. One of the parameters of the convolution operation we need to define is size of the window which will slide over all locations and extract the features. The size usually used is either 3x3 or 5x5. Each segment is then transformed into a 1D vector, these vectors are then reassembled into a 3D output map, maintaining the spatial locations. The main objective of the convolution operation is to extract high level features, e.g. edges from an image.

Pooling Layer - pooling layers have a number of uses including reducing the size of the feature map from the convolution operation (this reduces computation strain due to dimensionality reduction), it can also be used to extract prominent features. The pooling layer does not contain weights or biases.

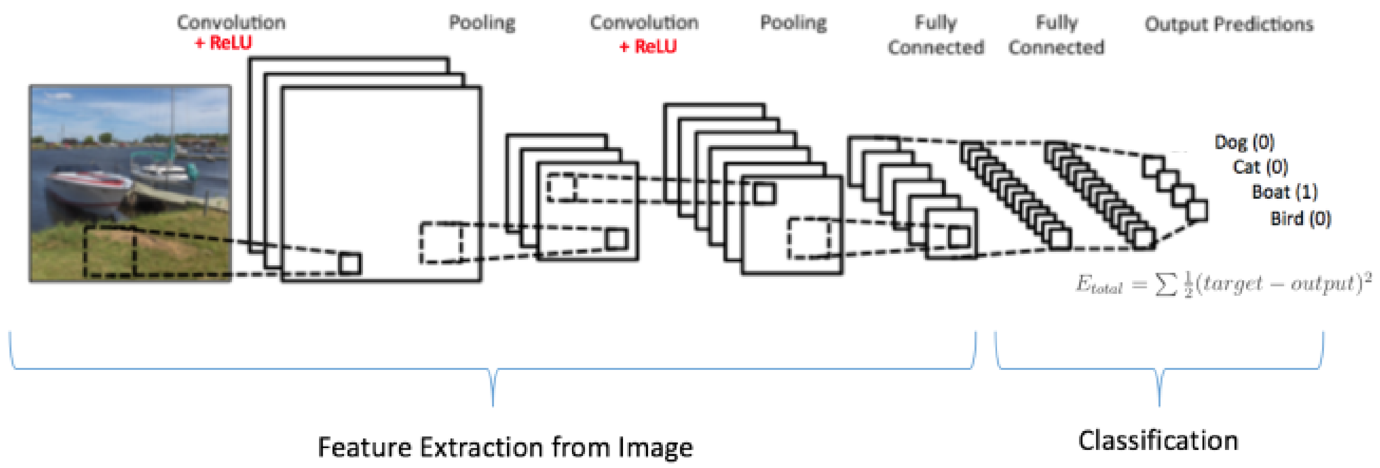
There are two options for pooling- max pooling or average pooling.

Max pooling - outputs the maximum value from the segment of the image enclosed by the kernel.

Average pooling - outputs the average value from the segment of the image enclosed by the kernel.



After undertaking these operations, the model now understands the features, we can now flatten this output and we feed it into a regular neural network for classification. The image below shows an example CNN architecture.



References and Resources:

S. Raschka , V. Mirjalili (2017) Python Machine Learning. Birmingham: Packt Publishing

CNN Architecture Image from: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ (https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/)

https://www.deeplearningbook.org/contents/convnets.html (https://www.deeplearningbook.org/contents/convnets.html)

Pooling Image from:https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53)

Which type of pooling layer extracts the maximum value?

Average pooling [✓ Max pooling]

Expected answer:

Average pooling [✓ Max pooling]

✓

✓ You chose a correct answer. You were awarded **1** mark.

You scored **1** mark for this part.

Score: 1/1 ✓

Question 6

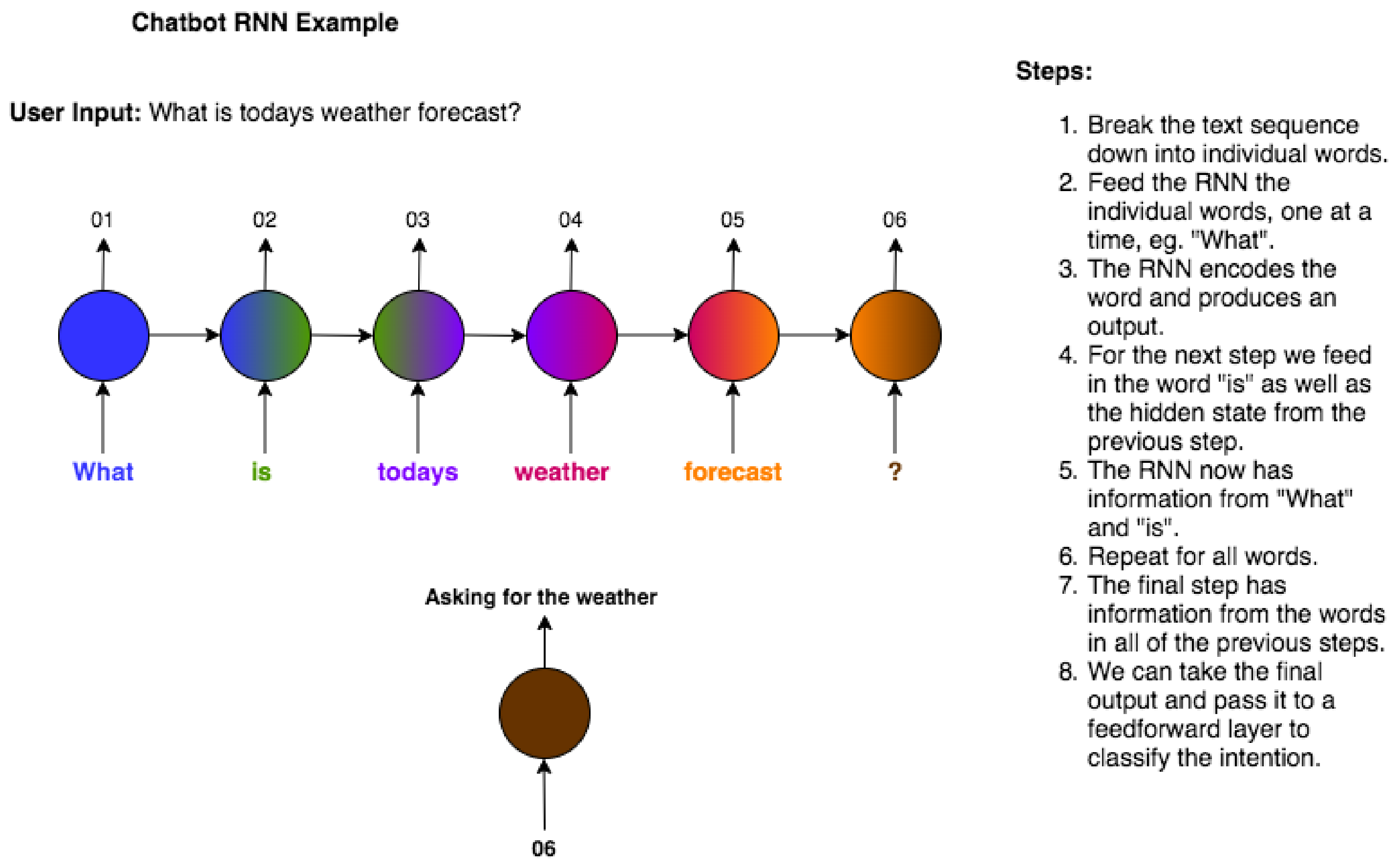
Recurrent Neural Networks (RNN)

RNNs are a very popular deep learning model, they have been used for a number of tasks such as speech recognition and image captioning, they are particularly useful for modelling sequential data.

A basic definition of the RNN is that it is a feedforward neural network with a looping mechanism, this enables the RNN to remember past information which can be used to influence decisions.

The RNN generates an output which is an outcome of a function of the input and hidden state, the *hidden state* works as a representation of the previous inputs.

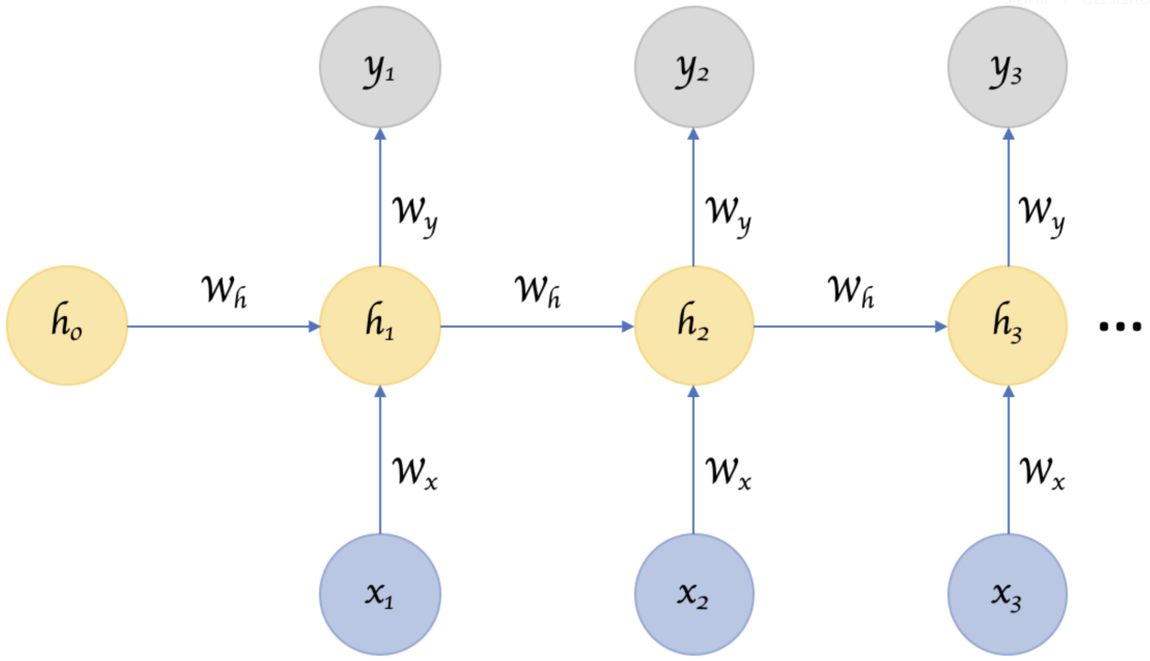
Below is an example of an RNN used to generate and operate a chatbot.



Steps:

1. Break the text sequence down into individual words.
2. Feed the RNN the individual words, one at a time, eg. "What".
3. The RNN encodes the word and produces an output.
4. For the next step we feed in the word "is" as well as the hidden state from the previous step.
5. The RNN now has information from "What" and "is".
6. Repeat for all words.
7. The final step has information from the words in all of the previous steps.
8. We can take the final output and pass it to a feedforward layer to classify the intention.

Below is a more technical diagram detailing the input, hidden state and output. RNN inputs can consist of a number of input vectors and can produce one or multiple output vectors. The outputs are determined not only by the weights (exactly like a regular neural network) but also by the hidden state which represents important information from previous inputs in the series. Along the way we also update the hidden state based upon the current input and then use this in processing the next input.



One of the issues we can encounter with what we term "vanilla" recurrent neural networks is that they can have trouble retaining information from previous steps- they have a short-term memory. If we refer back to the chatbot RNN example, the words "What" and "is" would be barely present at the final step.

Two specialisations of the RNN are availble to counteract this issue - LSTMs and GRUs. Both of these models essentially function like a normal RNN except they over comes the short term memory issue by using gates.

Gates essentially learn what information is the most relevant and should be added or removed to the hidden state.

LSTM (Long Short Term Memory)

The main idea behind an LSTM is that we have three gates which oversee the flow of information. The three gates are called the *forget gate*, the *input gate* and the *output gate*.

Forget gate - this gate determines which information is the most relevant and should be kept.

Input gate - this gate determines which information is the most relevant to add to the current step.

Output gate - this gate determines what the next hidden state should be.

GRU (Gated Recurrent Units)

A GRU only has two gates as opposed to the three in the LSTM. These two gates are the *update gate* and the *reset gate.*

Update gate - this gate performs a similar task to the forget gate and the input gate in the LSTM. The update gate determines which of the information is to be thrown away and what new information to add.

Reset gate - the main purpose of the reset gate is to determine how much of the previous information to forget.

Which One To Use?

Normal RNNs are fine to use when you are not modelling long sequences. However if you are modelling sequences which are longer it is better to look at using either a GRU or LSTM.

GRUs require fewer tensor operations so are usually quicker to train than LSTMs, however it is usually best to try both an LSTM and a GRU to see which works best for your task.

References and Resources:

Technical diagram from: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce (https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce)

http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Chatbot example altered from: https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9 (https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9)

https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 (https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21)

Which variation of RNN has three gates?

GRU [✓ LSTM] Regular RNN

Expected answer:

GRU [✓ LSTM] Regular RNN

✓

✓ You chose a correct answer. You were awarded **1** mark.

You scored **1** mark for this part.

Score: 1/1 ✓

Question 7

Generative Adversarial Network (GAN)

A generative adversarial network comprises two neural networks a *generator* and a *discriminator*, these two networks work in competition against each other (hence *adversarial*).

The *generator* neural network creates new instances of data.

The *discriminator* decides whether each sample of data is authentic or not- does it belong to the ground truth dataset or has it been created by the generator.

GANs are mainly used on images for translation such as text- to- image translation and for image creation.

What is generative modelling?

To gain a better understanding of the purpose of GANs, we are briefly going to review the difference between generative and discriminative models.

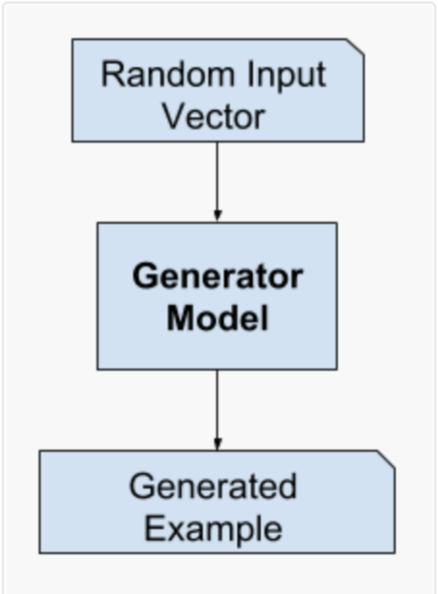
A predictive modeling task such as classification can be defined as *discriminative* modeling. Discriminative modeling/algorithms attempt to classify data, e.g. map features to labels.

Generative models do the opposite to discriminative models, they try to predict features given a certain label, this may then be used to generate new examples.

GANs can be categorised as a deep learning generative model.

The Generator

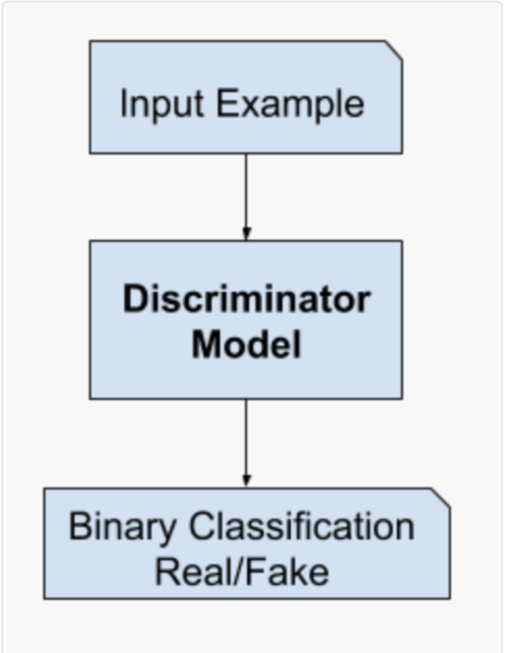
The generator is responsible for creating new instances of data, it takes an input of a random vector and generates a sample within the domain.



Example of the GAN Generator Model

The Discriminator

The discriminator takes in a sample from the domain as input (this sample can be real or fake) and outputs a probability which represents whether it predicts the sample as being real or fake as created by the generator.

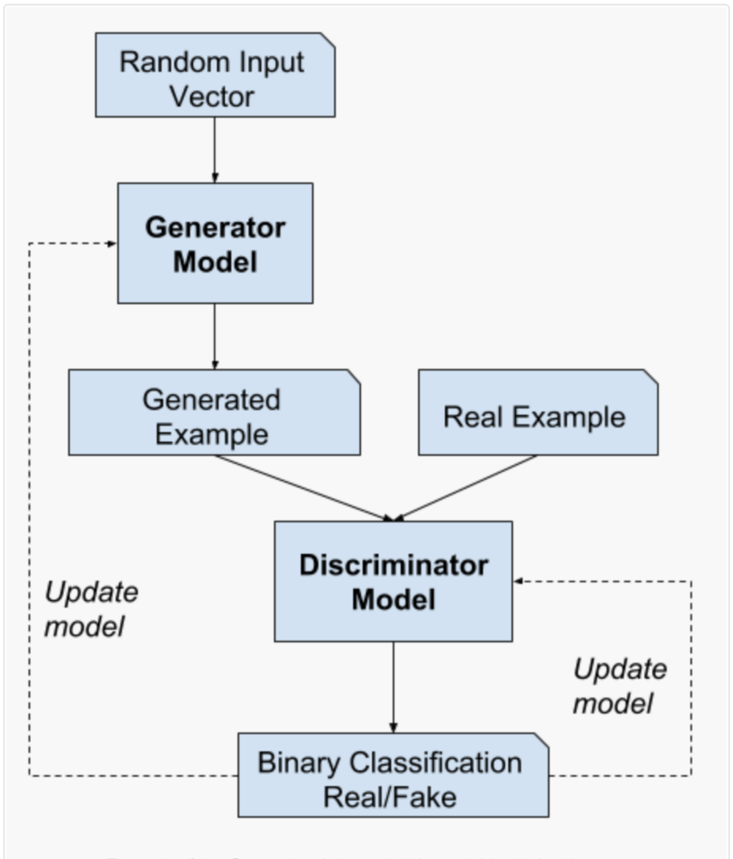


Example of the GAN Discriminator Model

How Do GANs Work?

The generator and descriminator are trained together, the generator creates a batch of samples which are fed into the discriminator alongside samples from the actual domain, the discriminator then classifies these samples as either real of fake.

The discriminator and the generator are then updated. The descriminator is updated to try and improve its discrimination accuracy regarding real and fake samples for the next round and the generator is updated depending on how well the generated samples decived the discriminator.



Example of the Generative Adversarial Network Model Architecture

References and Resources:

Images from- https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/ (https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/)

https://pathmind.com/wiki/generative-adversarial-network-gan (https://pathmind.com/wiki/generative-adversarial-network-gan)

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29 (https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29)

Which part of the GAN creates new instances of data?

[✓ Generator] Descriminator

Expected answer:

[✓ Generator] Descriminator

✓

✓ You chose a correct answer. You were awarded **1** mark.

You scored **1** mark for this part.

Score: 1/1 ✓

Question 8

Autoencoders

The main idea behind autoencoders is that they take the input, create a compressed version and then use this to try and reconstruct the original input. Autoencoders can be used for dimensionality reduction as well as learning important data features depending on the type of autoencoder used. They are commonly used for image reconstruction or for denoising.

The Essentials Needed for an Autoencoder:

1. Input data - the data to be fed into the encoder
2. Encoding function - this takes the input data and encodes it
3. Decoding function- this takes the encoded data as input and decodes it
4. Loss function - the autoencoder is considered to be working well when the output from the decoder is similar to the original input data - this would be a small loss.

The architecture of the autoencoder is important to ensure that the model creates a compressed representation of the original input and does not just memorise the input. We want our autoencoder to only retain the variations in the data which are important for the input reconstruction and to remove redundancies.

Different Types of Autoencoders

Undercomplete autoencoders - in this model we limit the number of nodes in the hidden layer to force the autoencoder to learn the best attributes of the input data by having having a smaller dimension.

Denoising autoencoders - in this model we can add some noise/corrupt input data and try to get the autoencoder to learn to remove it.

Sparse autoencoders- we use our loss function to penalise the activations within a layer to try and get our network to only activate a small number of neurons while encoding and decoding.

Variational autoencoders - variational autoencoders are a generative model which are most commonly used for image generation (refer to https://numbas.mathcentre.ac.uk/question/55839/generative-adversarial-network-gans/embed/?token=52d09dd6-8f83-4cbd-ac79-0922bcd8bdee for further information on generative models).

References and Resources:

https://towardsdatascience.com/autoencoder-neural-networks-what-and-how-354cba12bf86 (https://towardsdatascience.com/autoencoder-neural-networks-what-and-how-354cba12bf86)

https://www.jeremyjordan.me/autoencoders/ (https://www.jeremyjordan.me/autoencoders/)

https://www.deeplearningbook.org/contents/autoencoders.html (https://www.deeplearningbook.org/contents/autoencoders.html)

https://www.kaggle.com/shivamb/how-autoencoders-work-intro-and-usecases (https://www.kaggle.com/shivamb/how-autoencoders-work-intro-and-usecases)

https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f (https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f)

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73 (https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73)

Which of the different types of autoencoder is a generative model?

Denoising autoencoder Sparse autoencoder [✓ Variational autoencoder]

✓

Expected answer:

Denoising autoencoder Sparse autoencoder

[✓ Variational autoencoder]

✓ You chose a correct answer. You were awarded **1** mark.

You scored **1** mark for this part.

Score: 1/1 ✓