Outfit Recommender application implementing Machine Learning Techniques

By Joseph Mckeown

# Declaration

I, Joseph Mckeown, of the Department of Computer Science, University of Reading, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person’s work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly .I give consent to a copy of my report being shared with future students as an exemplar. I give consent for my work to be made available more widely to members of UoR and public with interest in teaching, learning and research.

Joseph Mckeown October 27, 2020

Contents

[Declaration 2](#_Toc70257699)

[Contents 3](#_Toc70257700)

[Abstract 4](#_Toc70257701)

[Introduction 5](#_Toc70257702)

[Background 5](#_Toc70257703)

[Problem statement 5](#_Toc70257704)

[Aims and Objectives 5](#_Toc70257705)

[Solution Approach 6](#_Toc70257706)

[Literature Review 7](#_Toc70257707)

[Background on Neural Network 7](#_Toc70257708)

[Recommender systems 8](#_Toc70257709)

[Recommender systems implementing Artificial Neural Networks 9](#_Toc70257710)

[Applied to the project 10](#_Toc70257711)

[Critique of existing works/main findings and evaluation 10](#_Toc70257712)

[Summary 11](#_Toc70257713)

[Methodology 12](#_Toc70257714)

[Screens of the Application 12](#_Toc70257715)

[Screen1 - randomiser 12](#_Toc70257716)

[Screen2 – item editing 13](#_Toc70257717)

[Screen3 – item adding 14](#_Toc70257718)

[Clothing and Wardrobe System 16](#_Toc70257719)

[The neural network 19](#_Toc70257720)

[Implementation 24](#_Toc70257721)

[NeuralNet 26](#_Toc70257722)

[Network 26](#_Toc70257723)

[Node 27](#_Toc70257724)

[Clothes/Outfit 28](#_Toc70257725)

[Main Activity 29](#_Toc70257726)

[Item Adding 33](#_Toc70257727)

[Wardrobe Activity 35](#_Toc70257728)

[Testing and Discussion 38](#_Toc70257729)

[Conclusion 46](#_Toc70257730)

[Conclusion and Future Works 48](#_Toc70257731)

[References 49](#_Toc70257732)

# Abstract

# Introduction

## Background

This project is to develop and create an app for smart devices which give the users recommendation on outfit they can wear from clothes in their wardrobe. It is also an exploration into how implementing a neural network into the recommending process can improve the user’s satisfaction though predictions. The theory is that a neural network will be able to learn the user’s tastes clothing from engaging with the application over a period of time. This is to see if implementing it will leave the user being more satisfied in the predictions made which should lead to greater user retention. This project will be developed in android studio using java with the code being created from the ground up to provide an application with the goal of having it run on most modern android devices.

## Problem statement

In modern times, people have access to more choices than ever before. This is due to how many more options have been presented to us due to the expansion of the internet and technology. However due to this it has become more common to experience decision fatigue. Decision fatigue is the phenomenon where the quality of decisions become worse the more decision which must be made throughout the day [1]. This project suggests that this can be circumvented to an extent by using technology to automate one a task everyone is faced with everyday being choosing what to wear. This project plans to implement a smart system to present outfit to the user in a similar way to how social media sites like YouTube and online Storefront like amazon recommend their content to it is user. This is by learning what the user likes and filtering out the content which the user would find irrelevant. This would help to alleviate the sense of decision fatigue as the number of decisions would be reduced through the system, learning more of what the user would want. As well as this, the project implements techniques from neural net to provide more suggestions the users like to improve their satisfaction with the application.

## Aims and Objectives

The aim of this project is to create an android application which will recommender outfits to the user which they will like. The aim is that by being learning the tastes from the user the app will be able to accurately predict good outfit making the user experience more enjoyable. It aims to include features to improve the user experience beyond the recommender alone such as customisations option which can allow the user to tailor the experience to their liking. Another aim is for the application to be accessible and for to be easy to use without having to explicitly explain how to use the app. The project also aims to present then recommendation to the user in an appealing way which takes advantage of the visual aspects of the UI.

The Objects to complete this task are:

* Develop an application which can run on android devices.
* Create a neural network which is able to learn the tastes of user over a period of time.
* An application with multiple screens providing different kinds of functionality.
* Easily accessibly UI design with clear understanding on what each interactable in the app does
* Create a visual interpretation of the outfit which is being recommended to the user though using picture of the clothes that they have added to the application.

## Solution Approach

To create the application itself, android studio will be used to develop and android application which will be compatible any device with android 5.0 or higher. The app will be built by using the tools provides by the IDE to create the UI with all the functionalities being written independently by using Java Class scripts. Creating a Neural network will be done through creating multiple interlinked Java class script which are able to perform all the necessary functions to run a neural network. It will also have its structure based of the models of deep learning neural network. The app will have multiple screens which with be created as individual android activities. On each of these screens a specific function will be assigned to it such as customisation, adding and recommendation. Each outfit in the application will be represented by a JPEG image which will be taken by the user. This will be stored for each item of clothing in the app and will be displayed when appropriate.

## Organisation of the report

The report is split into 5 sections. The 1st section is the literature review where the research and analysis of the relevant fields to this project are discussed. The 2nd section is the methodology where the ideas and plans used to build the application are discussed, outlined and displayed. The 3rd section is the implementation where the ideas from the methodology are implemented into the app with the reasoning and explanations of the functionality. The 4th section is the testing and discussion where the application is tested to see how well it performs. Finally the 5th section concluding the report along with a discussion into the future works for this project.

# Literature Review

## Background on Neural Network

Development of neural network has had theory written about if for many decades. This has stemmed from the idea of try to create an artificial way to represent the human brain and how human process thought and learn. One of the earliest discoveries was in the 1943 with the development of the McCulloch-Pits (MCP)Neuron model.[2] This model was the basic design of a biological model. In the biological model of the brain the Dendrite receives signal from another neuron which is processed by the Soma. If the signal received is strong enough from the Dendrite it will send a signal through the Axon which is excitatory or inhibitory. [3] This was then translated to create the MCP neuron where numeric inputs would be received by the computer and if the sum of the inputs exceed the threshold for the neuron it would send a representation of a excretory or inhibitory signal being a 1 or 0. This development allowed for basic model of Neuron to be able to learn certain Boolean problem such as, And, or and Not.

This model would be further expanded with development such as the perceptron in 1958 from Frank Rosenblatt [4]. This helped expand the fundamental ideas about neural network by introducing weighting into the model. This is where each input into a neuron would have a weight associated with it which would affect how strong the input is. Though changing the value of the weight based on their error, it would facilitate a way for the neuron to be able to learn problems. This allowed for linearly solvable classification to be solved as the weight could be adjusted relating to the error of the output of the neuron to the desired target.

However, issues arose as even though a perceptron could learn certain problem it would struggle with other which were not linearly separable such as the XOR problem. This would then be solved by the later development the Multi-Layer Perceptron(MLP). To help with training these network multiple method were invented. One of the most popular and prevalent ones is back propagation which was first proposed by Paul Werbos.

In recent years, as the process power of computers has increased, artificial neural networks have become far more prevalent. This is brought by the deep learning. This is the idea of an unsupervised model being fed huge amounts of data and being able to distinguish trends for it to learn the problem. Unsupervised networks are one’s which are given information to learn without any labels to train against. This means there is no human intervention in learning the data. The deep neural network discovers the trends in the data and finds which features are important for each classification [5] Due to its success, it has been implemented far more commercially, such as the success of deep learning voice recognition system being used in more application such as Siri and Amazon Alexa[6]. It is also constantly being improved with implementation of deep learning network in areas such as image recognition and manipulation. This is seen with applications such as Facesawp being one of the leading open-source version of deepfake software [7].

## Recommender systems

This is a system which is a subclass of information filtering which predicts the preference of the user. These are usually things the user wants, and the goal is to be able to predict these from the users’ habits. “RS are used primarily for individuals who lack sufficient personal experience or competence to evaluate all the choices.” [8]

A recommender system is defined by the users’ preferences and constraints. For knowing these recommendations can be made which conform to each users preference. There are multiple way data is collected to fuel recommender systems. This can be explicit like their reviews or they can be interpreted from a user’s action or though associated likes. Some of these techniques are used in modern example such as Facebook with their collaborative filtering system which creates ratings for recommendations based of people with similar interests[9]. With using metrics based on a user’s action it could be the about of retention a user has on a certain site, video or article. The goal for recommender system is to filter all the options provided and find the best option based on the user. This is done as it predicts the rank a predicted item would have with each user which is dictated by the type of technique used in the system.

An effective method for recommender systems is called collaborative-filtering. This is uses comparisons of other users who have similar tastes to the desired user to find items which could be recommended to them. The idea is that if one user agree on liking the same things there is a high chance that another thing one of them like will be enjoyed by the other.[10] The similarity with other users would increase the rank of an item making it more likely to be shown. This has been shown to be an effective method as many companies like Amazon use method similar to this in how they recommend products to its users.

Recommendation become more and more important in modern life as due to increased ease of access to information and items though the internet, we are faced with more and more decisions each day. It has also become much easier to make more accurate recommendations. This is due to the amount of data which is able to be harvested allowing for the recommenders to makes more and more accurate suggestions.

This makes recommendation system more important in trying make it not only benefit the user but also the owner. For the user, it provides the ability to have a better selection of choices while the owner benefits as it often mean greater profit. This can be due to the user finding something they want to buy, or the customer experience being improved leading to longer use of the application/site and establishing consumer loyalty.

Recommendation systems are important as they can a fair amount of influence on everyday life and routines and choices. It streamlines the decision process and reduces the amount of choice having to be made. This can help by reducing decision fatigue. This being caused by the overwhelming about of choice which is facilitated.

## Recommender systems implementing Artificial Neural Networks

Recently these two techniques have been crossed over more often. This is due how deep learning can process large amounts of data and be able to recognise trends in the data while unsupervised. This benefits recommender systems as it provides a way to produce far more accurate predictions for recommendation. The style of network use to for this is called Neural collaborative filtering(NCF). This uses multi-layer perceptron network to learn the interaction with the users. Though using networks like the performance of a recommender system is improved significantly. It is also able to help alleviate shortcoming which are experienced with methods such as matrix factorisation. NCF has benefits over traditional methods as due to it’s nature of a deep neural network based model, it is able to learn problems which contain more non linearly solvable relationships which other method would struggle or miss the connection.[11]

This system has been adopted into a lot of modern companies recommender systems. One such example is YouTube which use data obtained from their Tensor flow to train their deep neural network. [12] As google have a wealth of data from it large userbase, this means that their recommender models trained through NCF will have process more accurate results that more traditional method, like collaborative filtering would have.

## Applied to the project

The project’s goal is to be able to create a outfit randomiser application which learns the user’s tastes. To achieve this goal, the application will need to take advantage of Neural Networks to be able to learn the tastes and to implement recommender system techniques in deciding on the prediction. As the app would want to learn the taste over time, a multiple layer network would be able to accomplish this well. This is because implementing this allows for more accurate prediction. As well as this, by implementing a MLP network it will allow for greater expansion in the project to implement deep learning. This aspirational goal would greatly benefit the performance of the app as, a deep neural net will be able to learn non linear trends in the data. This would be beneficial as it would be able to learn the aspect of an outfit which the user likes and priorities them. To be able to take advantage of this, each item of clothing and outfit would need to contain enough fielded of information for these connections to be formed.

In terms of recommender system, they could be implemented into the application as a way of deciding the outfits. The is through the recommender system assigning values based on a potential an item must be chosen by a user based on the user’s information. This would prove beneficial as it would provide.

## Critique of existing works/main findings and evaluation

In the field of outfit application there is a majority of application which aspire to be virtual wardrobe.

Most of these application have the functionally to create a digital version on someone wardrobe and not much more than that. The focus seems to be more on the organisational aspects of the concept as well as the social media aspect.

One of the biggest examples of this is the Combyne. With 149,848 user reviews and a average review score of 4.5, it is one of the most popular fashion and outfit related apps on the google play store. This app creates an environment to add clothes in the form of picture and to demo them virtual mannequins. The focus is the ability to see different clothes together without having to wear the outfit itself. As well as this it boasts integration with online outlets as well as social features. This allows the users to be able to share their outfits with their friends and be connected to a retailer when an item seen is desired to be bought.

An app which is very similar to the project is the app Shuffle Outfits. This app, unlike Combyne is much smaller functionality wise as it has a very simple function. It gives the user to ability to create a virtual wardrobe and then use that wardrobe to create random outfits. These outfits are completely randomly chosen but follow the format of having an outfit comprise of shoes, something for the lower half of the body such as trousers and something for the top half such as a hoodie.

With these examples in mind, it shows that there is a gap in the market for an application which is more personalised in how it presents it’s outfits to it’s users. Some aspects of the these app which are well done are how Combyne has a smart system for cropping clothes to a model and it’s clean design of it’s UI. With Shuffle Outfits, the way it categories an outfit through having simple yet defined definition of 3 items of different type could be used to make the classification of an outfit easier.

## Summary

In summary, it has been found that the future of neural network is going toward deep learning-based implementation. This will need to be adapted and try and be implemented to work as a solution to learning the taste in clothes. In the design of the application, it will take inspiration form the networks used in social media site such as YouTube and their integration of recommender system and Neural nets to provide the best results. This is also with consider to the vast difference in available data to train the network with.

# Methodology

The design of the app would need to be able to facilitate the main function of the app. These functions being:

* Neural network recommendation
* Wardrobe management
* Clothes additions / deletions
* Outfit storage

To accommodate this, the app can have separate screens to focus on each task. With this idea, 3 screen would need to be created. It would also need a central screen which would start the app and would be where the user would spend most their time. For this the recommendation makes the most sense as the main screen as this it the key functionality of this application.

## Screens of the Application

### Screen1 - randomiser

For this main screen the user would need to be able to request a recommendation from the current items of clothing available and to present it on screen. Along with this it also need the functionality for the user to decide if they like or dislike the recommended outfit. This can be done through a having button either side of the recommend button. These would provide each option respectively. As well, due to it being the main screen of the application it would need to have suitable menu functionality. This can be done by having buttons at each corner at the top of the screen. These can lead to both the clothing item addition screen and the wardrobe management screen. This would follow standard designs where the options and extra feature are accessible in other screens.

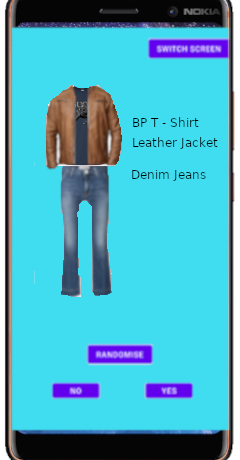


Figure Early Demo of main Screen

In the main screen the main process will need to be the recommendations. When the user is using this screen, they should follow a general cycle of interactions. This would be them obtaining random outfits and deciding whether they like the outfits or not with the occasionally adding cloths to the application. From this the design for the usage of the main screen can made. In this it also shows the basic flow of the recommender function as well. It’s structure is that once a request is made to get a recommendation it will find the outfit with the best fitness at that point and the present it for the user to judge.

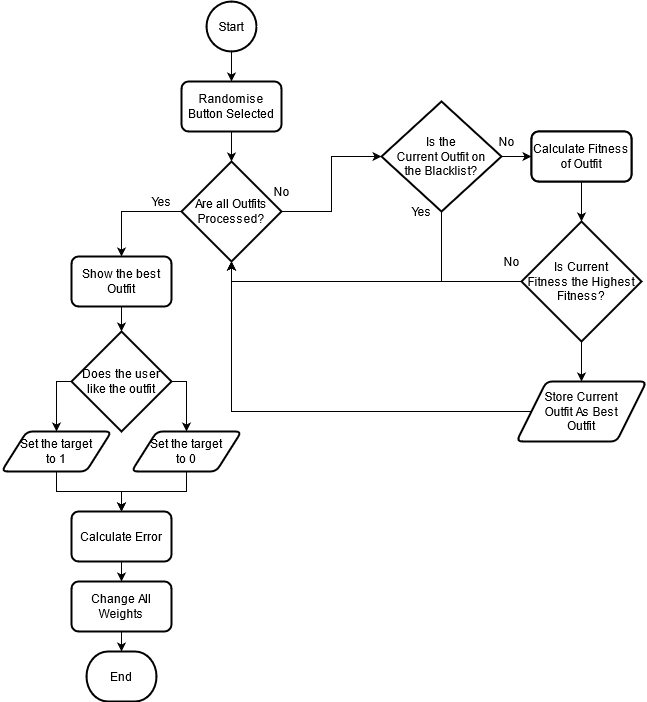


Figure Flowchart of the main Screens main process

### Screen2 – item editing

The next screen of importance is the wardrobe screen. in this the user will be able to see the entirety of the clothes which they have entered into the application. It will allow the user to browse through each item and highlight them to gain more information about the item of clothing. This can be information like a score based on it’s current fitness. These kinds of feature can help the users better understand how the application is functioning and can also allow help them with highlighting items of clothing which they prefer. This will help towards the goal of a better user experience as it can help them better understand their tastes through this app along with potential highlighting patterns of decision which the user was not aware of previously.

Another necessary feature in this screen is to edit the properties of the item which are in the wardrobe. The user should be able to change the majority of the key information about an object like the name, image representing it , delating the item and even what category it belongs in. As this I changing key information about the item’s the NN is based off some consideration have to be made. For both the Name and the Images they should be able to be delt with within a Clothes class. This is because they are the properties to define the clothes which are not intertwined with the NN. However If the user would want to delete or to change which sub category the item belonged in accommodation would need to be made to ensure the NN says accurate. This would be ensuring the weights are removed accordingly with the deletion of an item along with removing any outfit which may have contained the item. For the changing which type it belongs to, it would need to have the have all the outfit weights containing that item removed as those outfits would not longer be possible. Along with that it would need to create whole new outfit weights for all the new outfits this type change would have created.

### Screen3 – item adding

Another screen of the application is the Item addition screen. Here would be the main hub for the users to be able to add the items of their wardrobe into the application virtual wardrobe to be created. This screen would need to have the option for the user to input all the relevant information about each item of clothing. This would will probably be the name of the item, a picture of the item and the sub-type which the item fits under. With all of this information it should be ready to add a new item to the virtual wardrobe and the NN.

To add new item to the NN it would need to consider how it would create the new weights for it. individually, all it would need to set up the creation of a new weight. When it come to outfits, all the new outfit created due to the inclusion of this new item would need to be indexed. Along with this the weights would need to be created for the respective outfits which are created. This may take a long time as the with lots new outfits being added it means that there’s a larger wardrobe so may take some time to calculate all the outfits as the number of outfits added per item is relevant to the amount of other item in the other categories. This could be resolved by either implementing multi-threading when it is possible on a device or to have an item limit where only a certain about of items can be added where the outfit limit would not be exceeded.

This screen would be implemented through creating a separate activity within the app. Within this activity, it contains the main features mention, being the ability to choose our options for a new item of clothing.

On the screen, the user is provided multiple options:

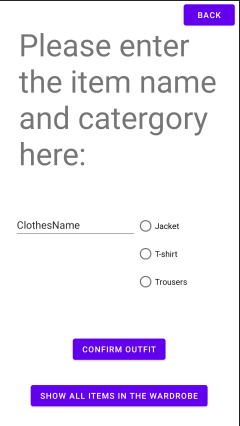


Figure The UI of the item adding activity

* 3 buttons which align with the 3 main categories for the clothes in the clothing classifier.
* A button to show all the items which are currently within the wardrobe
* Textbox for the user to input the name of the item of clothing
* Confirmation button which creates a new item of clothing in the wardrobe

With these option the user is able to engage with adding item to the app. Some consideration need to be made when implementing as to ensure that an invalid item is not entered into the wardrobe, the option are checked on confirmation of the Item. This require that the would be clothing item to be entered is validated. This means it has to be checked to ensure it has the right attributes being a name and a type. Once validated it will be added to wardrobe. If the requirements for an item clothing in the app was expanded the validation would equally need to be extended. To prevent the app from crashing a try catch statement was used. This would TRY adding the item of clothing into the wardrobe and if any part of the process would error it would stop the process and display a message on the main screen telling the user to choose valid inputs. Doing this prevents the app from crashing from invalid input while also making the user aware of the issue.

Another consideration is how to send the data on the wardrobe and clothes between the activities. This is because when you switch between screens the app closes the previous screen meaning the key information needs to be passed around each activity. This can be done by sending intents across. Intents are used to communicate between multiple components in an application. By using intents, we can send the wardrobe information between each activity. This means that each screen needs to accommodate from packing the data into intents to send over and also to receive and unpack the intents. This can be done through be sending the data as multiple variable arrays or taking the object and making it a passable/bundle.

## Clothing and Wardrobe System

The main objects within the application are the clothes. There needs to be a way to represent the clothes of a user within the app. To do this it would need multiple parameters to define what the item of clothing was. This could be done be categorising each item into a general theme of similar items. This would make it easier to be able to create outfits as there could be easily defined parameters to create an outfit. To try and solve this problem, the idea of a virtual wardrobe is created. this is where the all the representations of the items of clothing from the user are stored in a particular order and method in a virtual wardrobe object. This would help aid in the classification of the item for creating an outfit as it an outfit would need to pick the required items of clothing from each section of the virtual wardrobe.

To achieve this each item of clothing in the application would need to be classified as a type. The goal of which is to prevent item which would be incompatible in real life E.G. not wear 3 pairs of jeans. A way in which this can be executed is by having each of the clothes to be sub-dived into 3 main categories being a Top, Under Top, Bottom. These would represent the basic components of what would make a usual outfit. This can be expanded so that the minimum which an outfit could comprise of is a under top and a bottom clothing item as generally this is the least which is considered acceptable. By using this clothing classification, it makes it simple and effective to classify clothes and prevent conflicts when creating outfits. It can also allow for further customisation with the user as you could change the minimum and maximum requirements needed for an outfit to be considered. Another way in which it can be expand in is to include more sub-categories. This could be very useful as it gives more specific option for the users to classify their item which will help track their preference in certain subtype of clothes. This can help provide more accurate results as it the NN can have the influence of preference in certain subtype play a heavier role it which it selects as the most appropriate outfit. This is because the more specific the categories the easier it become to implement category based filtering/learning. This because the more specific the categories are, the less generalisation have to be made if implementing it into the learning process. For example If the user likes jeans but they are classified under long bottoms(E.g. jeans, trousers, leggings) it will recommend others within that sub category even if the user has no affinity for clothe of that type.

A benefit for more defined sub-types it can allow for more implementation of more traditional recommender system methods to be introduced like a collaboration filter. If the application was used by multiple users and the data was taken and stored, then a collaboration filter could be created. This could compare other users on the application who have similar tastes or trends in their clothes and outfits to try and recommend other thing which those similar users like. This makes it much easier to present recommendation when there is little data on a user as it can pull from users with perceived similar tastes. However, this would need a large data pool size to effective. This is both because of taste in clothing very subjective there would be cases where is the pool is not large enough more niches taste within the system would benefit less from the collaborative filtering and could lead to potentially longer times to learn the users tastes.

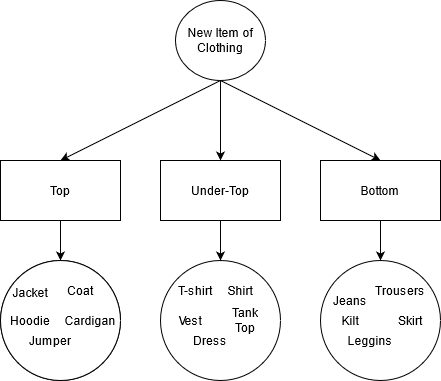


Figure The Design of the Clothes Classification System

The implementation of this was to create a class for the items of clothing. Within this class, each object would contain the name, id, type of item which it is. This would allow for every item of clothing in the app to have a standardised format of information which makes it easier to create and access information about the item of clothing. These objects would be created when ever a new item was created within the app and then would be stored within a virtual wardrobe.

To represent the virtual wardrobe, it would need to be able to stores each individual item of clothing added to the app along with being able to be changed through addition, deletion and edit whenever the user so chooses. application. The way this can be implemented is through creating a multidimensional ArrayList. This would allow for clothing items to be moved on and off the list with ease. The multiple dimensions of the ArrayList are to accommodate for the number of types which are used in the clothing classification system. This is because each dimension represents an unique array list containing all of the item of clothing within that type so for 3 types there would be 3 dimensions respectively.

This is done to create to help to ease the process of creating an outfit. This is because the criteria can be set to one of each item from each dimension of the ArrayList to create a new outfit. This format helps to properly define what is considered an outfit in a wardrobe. By defining it, all the possible outfit from the wardrobe can be easily created through by creating outfits from 1 item from a category and all other items from the other 2 categories. Along with this it would affect any sub-types which may be introduced as they would still be categorised under these main three categories.

## The neural network

This is a key piece of functionality for the application. The goal for a Neural Network(NN) in this application is to be able to learn the users taste in clothing over time. This means that the inputs for the network need to be the outfits. Along with this the app should be able to create a distinction for outfits as a whole and items individually. The idea of which is because an individual item of clothing may not be popular with the user on it’s own but when paired with other specific items the outfit it more favour. Another factor when designing the NN is that the user will be able to choose if they like the outfit or not. This means suitable method needs to be introduced to facilitate that. It also means that the problem of taste is not linearly separable. This means that this problem cannot be solved by a single layer perceptron network[13].

For the create the inputs, it will take them from the current item of clothing/outfit being rated. This works by having the current item/s having their associated input value being set to 1 while any other item that is not in the current outfit will have an input of 0. This will ensure that only the values of the relevant items are used to produce a result along with ensuring the right values are changed when the network is learning. This referred to as the activation function, where 1 represents an on node and 0 represent off.[14]

As well, each input has its own weight attached to it. This affect how strong the signal from the input be within the network. These weights affect each input individually and are changed throughout usage, either positively or negatively, depending on the user. This allows for more favoured outfits to be discovered as the items weight will be higher allowing for a greater input in the network, producing a higher output overall. The design of which is see below:

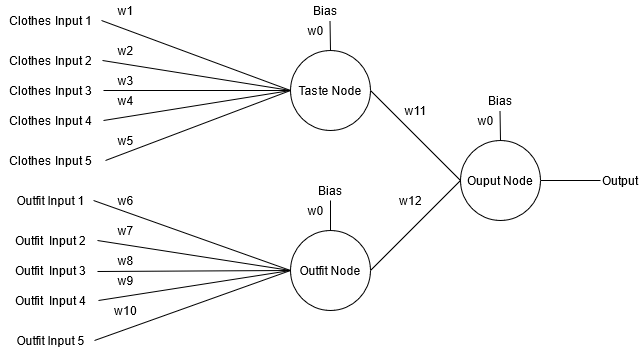


Figure Inputs from the wardrobe into the neural network

Through this model, the neural net is able to receive the input of both the preference in each item of an outfit but the outfit together.

As the problem can not be linearly separated, a multiple layer perceptron model would have to be used. This kind of network contain an input layer, an output layer and potentially multiple layers of perceptrons between the input output. These layers of perceptron are called the hidden layer. The hidden layer and the network nonlinear activation(look up) allow this network to be bale to solve the problem which not linearly separable.

To facilitate the learning of the network, a goal has to be set for the outputs. In this network it would be done by the users decision on an outfit. If they like it an choose yes, the goal will be 1 while the would be 0 if the decision is no. This will be used to change the weight within the network accordingly. Due to the network being a MLP network, the weights are changed through back propagation. Back propagation “calculates the gradient of the error function with respect to the neural network's weights”[15]. This calculates the error of each layer based of the layer before it starting at the final output layer and working it’s way back. The error of the a layer in backpropagation is calculated by multiplying the previous layers error by the connected weight between the two layers This would be described as the chain rule[5]. The goal of this is employ gradient decent to each layer within the network. Gradient decent is when we want the weight in the network to produce the smallest amount of error, as if the overall error is decreasing, we want to increase the weight.

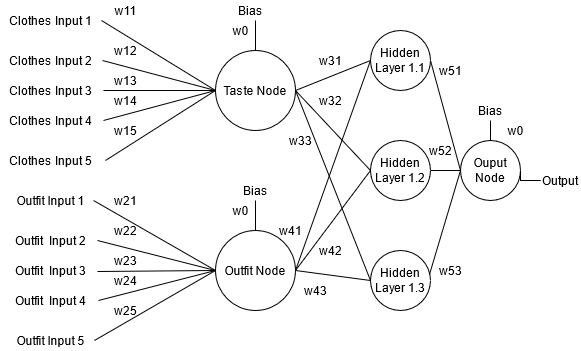


Figure MLP design of the Neural Network with a single hidden layer

As this type of NN is not using a data set as training for the application. In neural network based in classification they would be trained on sample data to learn train the weight of the network. However this algorithm will not know which kinds of outfits will be accepted or rejected by the user. This means that the network will have to be able to learn the taste quickly. The need for the network to constantly learn is due to the nature of clothing and how it is constantly changing in the tastes of the user. Therefore, the algorithm needs to be able to adapt quickly to new item being added to the system. A way in which this can be achieved is by having the network itself set to a higher learning rate. This is because the learning rate of a network dictates how large a change is made to the weights in a network. Generally smaller change is better as it allows for more nuances to be learned. It also helped prevent outliers in data drastically changing the weight and output. This is important as is the network changes too much due to outlier it could lead to it learning the wrong information about the data or having it unlearn something.

As this type of problem is a prediction, the output of the network would be used to predict what outfit is most likely to be preferred by the user. This means that there need to be selection occurring in the network. To do this it needs all the possible results of the outfits in the network need to be found. From this the output would need to be compared to each other to decided which outfit shall be the predicted outfit.

#### Evolutionary Algorithm implementation

To accomplish this techniques from Evolutionary Algorithms(EA) can be used. EAs are inspired by biological evolution and are often used to find the answer to a problem from the multiple generations of their populations. Two techniques which they use are Evaluation and Selection. In evaluation the population if given a score, called fitness, based on how close a chromosome in the population was to solving a problem. This fitness score is then used within evaluation, where the parents of the next generation are selected based of their fitness and the algorithm’s selection technique. Some selection techniques include rank based, where the chromosomes with the best fitness are chosen to reproduce with other high fitness parent, and roulette, where the each chromosome is given a proportional percentage chance to be chosen based on the chromosome’s fitness. These techniques can be implemented into the neural network as the chromosomes would be represented by each outfit, with their fitness being the result they produce in the neural network. From this an appropriate selection algorithm can be used to decided which outfit should be chosen as a prediction.

A rank-based selection method could work with this kind of network as it would always provide the best possible out to the user. However, an issue arises as this method would keep presenting the same outfit, if the user like the outfit. This would mean that there would be low to no variety of outfits being shown. Even though this would always produce the best result, it is also important that the users are presented with variety of outfits.

A method which was experimented with to try alleviating this issue, was to blacklist. This blacklist would exclude a chosen outfit from participating in the rank selection whilst on the blacklist. This would have worked in a similar manner as dating sites such as tinder where once a decision has been made on a person they are taken out of the queue. This idea would be adapted so that an outfit would be eligible to re-join the potential outfit after spending enough time there or when the amount of clothes blacklisted exceed a threshold of the overall wardrobe size. Any outfits which would eligible would have a chance to be added back each randomisation. This however would not be a good enough solution as it this could lead to outfit which are well liked being removed from the potential wardrobe for a long time. This could result in lower user satisfaction, as they may not understand why an outfit isn’t appearing anymore.

In the end, the network would use a roulette-based selection method. This is because, it would provide outfit which are greater preference to be chosen if they have performed well previously and visa versa if seen negatively. This would be achieved while still being able to have another outfit be chosen. This makes it a more preferable option than a rank-based selection as it will still make preferred outfit have the better chance of being chosen while also allowing for less liked outfit, but still equally viable outfits to be chosen.

#### Equations of complexity for the wardrobe

Each run of the randomiser will have for (o), with 0 being the amount of outfit. O = i\*j\*k; i=number of tops, j=number of underTops, k = number of bottoms. This means when ever a new item is added to the wardrobe the complexity is increased by i\*j for a k, j\*k for a I and i\*k for a j. This means the number of outfits increases at a bigger rate the more item of clothing are already in the wardrobe.

As the user decided how the neural network learn, this would make it an supervised network but a more unconventional one. This is because in a supervised neural network, once a result was computed, it would be compared to the label being the target result. This is done in the neural network, but the user decided what the label be after each run. This means it is still trying to reach a label defined goal but that label in not defined until the user makes the decision. Along with, the users decision is not permanent due to tastes changing. This means a definitive label can not be put on any outfit. This is why the EA roulette selection technique works well as it will allocates a chance an outfit will be chosen based on its ability to produce a positive or negative response. Unlike this, an unsupervised network is one which can “discover hidden patterns or data groupings without the need for human intervention”[16].

# Implementation

The platform which was decided to create the application was android. This is because the android OS is has the majority market share in smartphone demographic. Because of this, it would allow for the application to be run on more devices, making it easier to distribute. To develop the app itself, android studio, the official IDE for android, was used. This is because the IDE provides a lot of feature such as comprehensive layout editors, virtual device emulators and useful debugging tool, all of which made it appealing to use. Android studio uses both Java and Kotlin as the primary languages. For the application, Java was used. This is because, the developer has more experience using Java, along with it being an object oriented language which is a necessity for the creation of the application.

Android application are comprised of multiple activities. In each of these the UI can be created from the layout editor which generates a XML file with the UI design. Along with this each activity has an associated java file. This contains any of the functions and process needed to run the activity.

In the application, it is broken down into 3 activities:

* MainActivity
* itemAdding
* WardrobeActivity

These activities each contain one of the main processes of the screens discussed earlier.

Along with the activities, the application also contain multiple separate java files. These classes help to manage all the process for its respective task. These classes are used within the activity classes. The java classes which are in the application are:

* NeuralNet
* Network
* Node
* Clothes
* Outfit

For each of the 3 activities there some universal functions which they all share. These are mostly relevant to the transversal of the application. In each screen there is are 1 or 2 traversal button in the corners of the screen. As the main Activity is the core of the application the screen go between the main activity and the other activities. Below shows how each screen is connected to each other.

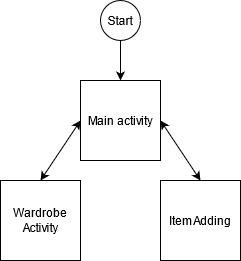


Figure application screen flow

To leave an activity for another, certain information needs to be passed through. This is information which is necessary for the functioning of the app. This means that the Neural Network object needs to be passed from one activity to another. This can be done by adding extras to an intent object. Intent objects are what allow other activities to be run. Along with this each intent can store information in the form of extras. To pass the wardrobe object information form one screen to another, its important information is stored as extras. Once the new activity has started it will extract the intent information sent over and use another constructor to rebuild the NeuralNet object in the new activity. This constructor takes the passed extras and inputs them back into a NeuralNet object. For each of these traversal functions, it implements a try can catch when adding the extra information. This is so the user can traverse though the screens even if there is no information to pass from the neuralNet. Along with that it prevent the application from crashing if there are any issues with the data.

All of these function are implemented into button in each respective screen. Each screen sets the button’s onClickListener on it’s initialisation. It is slightly different for the mainActivity as there are 2 traversal buttons. These button onClickListeners are set so the call the same function but pass a different class object. This allows for the same function to be used for traversing to both screens. 

Figure Main Activity Screen Swap button onClickListener

## NeuralNet

The NeuralNet class is used to contain and process all the information for a neural net to run in the application. This class links with multiple other classes as it contains multiple object from other classes.

The wardrobe of clothes is stored in this class. The wardrobe is implemented as an ArrayList of Clothes objects. Along with this all the possible outfits are stored in another ArrayList only made of Outfit objects. Both of these array lists dynamically change in its size and contents throughout using the application. This is from new Clothes objects being added to the application and old objects being removed.

The net itself is built from creating a Network object.

## Network

This class contains all the information to be able to create a neural network. It does this by storing an ArrayList of an array of node objects called nodeLayers. By having the node objects being stored in an array format, it can represent multiple layers of the network. For example to represent an output layer of 2 output nodes, an array of 2 nodes would be added to the list. This makes it easier to apply changes for an entire layer. As well it helps with connecting nodes from one layer to another. Along with that this makes it very easy to implement hidden layers as node layers can be created of a set amount of nodes to a set amount of layers. Also having the layers in this format help greatly with implementing back propagation. This is because it can be implemented into a for loop for the number of layers, working until it reaches the input layer.

The main function of this class is to calculate most of the network related information like setting up the network layers, applying the changes in the network, calculating the output of the network and the relevant getters and setters for the network.

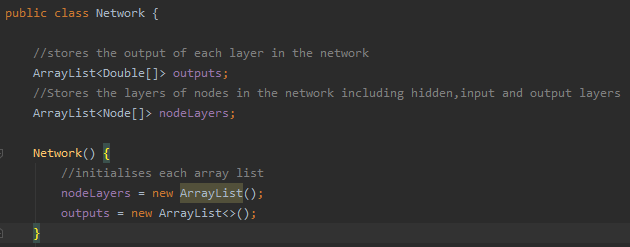


Figure Network constructor

## Node

A neural network is comprised of lots of nodes, which are formed to create a network. To represent this the node class is used. The function of this class is to be able to calculate all the relevant information for an induvial node. The includes calculating the nodes output, changing the weight attached to the node along with providing the relevant getters and setters. Each node object has the shared characteristics of containing weights, change in weights, delta and a bias. For the weights, they are stored as a 2 dimensional double ArrayList. It is organised like this as makes processing the inputs from the individual items of clothing easier to manage. This is because the wardrobe object in the neuralnet class is also an 2 dimensional arraylist. This structure is mirrored so the weight of the clothes input node would correspond with the exact item of clothing within the wardrobe list. This also does not affect how any other nodes function as they will only have list of 1 dimensions.

The nodes includes change in weights which is based of the change in weights for that node’s weight in the previous cycle. This is stored so that momentum can be included within the changes in weight. This is done as it can have the improve the speed and accuracy of the network along with having the affect to “dampen the oscillation, making the resulting model more accurate”[17]. This is because having values go between positive and negative change, momentum with lessen the effects an push the change of weight more on the general trend the weight had been heading.

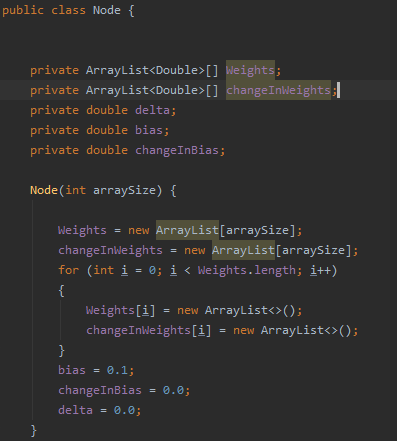


Figure Node Constructor

## Clothes/Outfit

This class is the basis for every item of clothing within the system. It achieves this by storing all the necessary information about an item being the name, type, id and image path. Form this, a clothes object is defined. The expanded version of this class is Outfit with is an object comprised of 3 clothes objects which is this application definition of an outfit. Each of these are used to store info about the clothes and so are mostly used for getting and setting information about the items.

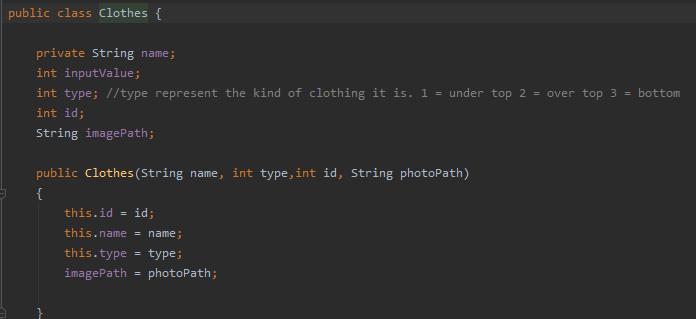


Figure Clothes Constructor

## A screenshot of a computer Description automatically generated with medium confidenceMain Activity

Figure Main Screen with a recommender outfit

The main activity’s main function in to run the Outfit Recommender. This is run through the 3 buttons in the main screen. This allows for the loop the process loop of the user get’s a recommendation, they decided if the like the suggested outfit or not, and the neural network if adjusted accordingly. To visualise the process, a space in the activity is dedicated to displaying the outfits. This is displayed with a purple outlined box in the centre of the screen. This contains a horizontal linear layout which contains 3 other vertical linear layout constraints. Within each 3 vertical linear layout constraints are a Textview object and a ImageView object. On resolution of the randomise button OnClick function it will set these object to the relevant clothes as seen to the right.

For the randomise button, it’s onClickListener triggers the function randomOutfit. This function tries to run the public function of a NeuralNet object called running and on resolution change set the textviews and imageViews in the screen to the chosen outfits. The NerualNet function running is what does the bulk of the processing.

The running function’s purpose is to do a full run of every possible outfit combination within the wardrobe stored within the NeuralNet object. Once it obtains every possible results it then uses it’s selection function to decide which object get’s shown. For calculating the outputs of the each outfit the function outfitProb is called. It is able to get every result by looping for each item in the wardrobe in each type. For this is creates an Outfit Object from the current cycles clothes objects, whose results are calculated and then stored in the function. This is so it can be later used for the selection process.

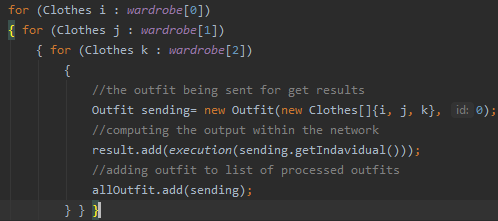


Figure Calculating all the outfit probabilities.

To calculate the output of a select outfit the execution function is used. This passes the outfit which is currently needing to be processed and running it through the neuralNet. It calculates the inputs which are needed for the neural network by sending the passe outfit into the calcInputs functions. These functions return back an ArrayList of 1’s and 0’s. These are the inputs for the neural network with the 1’s representing the clothing item or outfit. This allows for only the potential outfit’s item of clothing and the outfit itself can be considered when calculating the output.

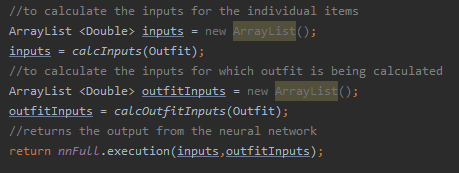


Figure NeuralNet Execution function

Once the inputs needed for the neural network have been calculated, they are passed into the Network object’s function execution. In this function, it loops through every layer within the network. I initially calculates the outputs for the first layer based on the inputs of the clothes and the outfit. Form there is will loop for the amount of hidden layers which have been set up. The more layer which are implemented the better the performance in learning trends should be. Once the final layer of the network has been reached, the result from the output node will be sent as the return value.

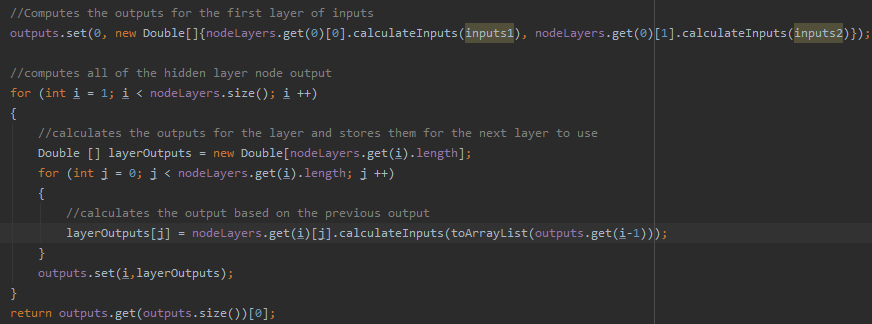


Figure Processing all the outputs for each layer in the neural network

The output from each node in a layer is calculated from the weight attached to the node and the input which is passed into the function. It loops for every weight stored in the node along with the basis. The of this loop is return and stored for the next layer to use.

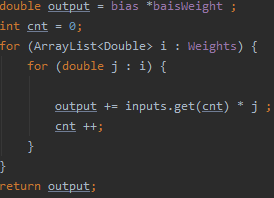


Figure Calculating node Output.

This process occurs for every outfit in the Object. Once all the outfits output have been calculated the selection algorithm is then used to choose which outfit is chosen. As the project is using a roulette selection method for the selecting the outfit, sum of all the outputs is found. This is used to choose a random double value between the sum and 0.0. The value produced is the value which will be selected as the outfit to be presented to the user. To obtain the index of the outfit, it loop though all the outfits, adding their respective results to the variable sum, which keep track of the sum of the outputs up to that point. It then checks to see if the new sum would be greater than the random position value and if so it will return that outfit. This returned outfit will be stored as the network’s current best outfit. This done so that it can be displayed and also changed later when the network is learning.

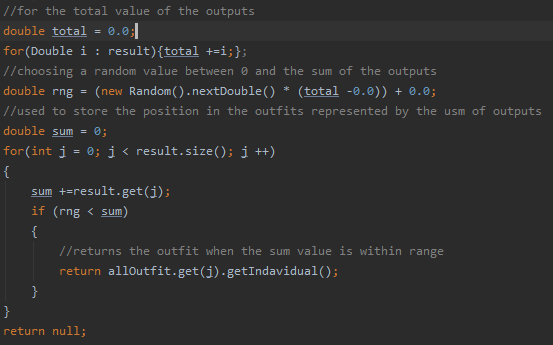


Figure Roulette selection method

Once the result is shown the user can either like or dislike the outfit which they are presented with. Choosing either option will run the NeuralNet process outcomeChange. This function has the decision made by the user being representing liking the outfit being a 1 or a 0 respectively. From there the changeAllWeigths function is called while passing the decision and current best outfit. The reason why 1 or 0 is pass is that this allows for the calculation of the error to occur. This is because if the result is positive the error is calculated as the 1 – the output. this is because the best possible value and outfit will get is 1. This work the same way for if the decision is 0. In this function, the learning rate and the momentum for the network is set up. This will affect the rate in which the weight will be changed. From this the network is set up to the current best outfit. This restores all the output calculated in each layer of nodes. Once set up, the network will adjust the weights of the network based on the inputs and the initial layer calculated.

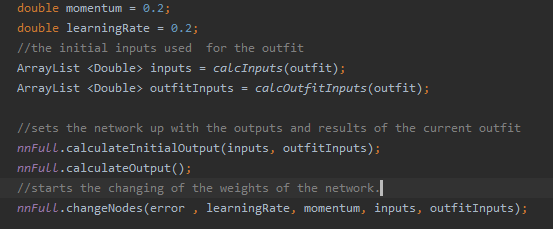


Figure Configuring the weight learning system

Very similar to how the output was calculated, the weights are changed in a very similar manner. For all the nodes in the network their errors are set. This uses back propagation so it work from the output layer and set the errors for each layer until the input layer. The error for each layer’s node is the sum of all of the next Layer’s node error multiplied by the connecting weight from the current layer’s node and the next layer’s node. This becomes each nodes delta and is used to change the weight in that node. Once the error has be set for every node in the layer, the weight for each respective node is then changed based on the set delta values and the output from the previous layers. To change all the weight in a node, it loop for every weight value in the node. For there each weight has it’s individual weight changed based on the learning rate, momentum , delta and the input for that weight. The change in the weights is also always rounded to 5 decimal places.

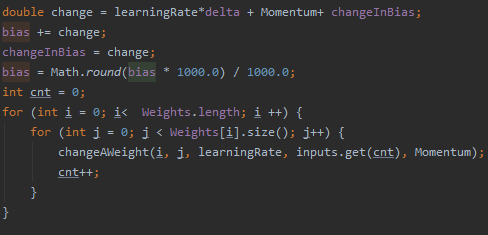


Figure changing all the weights in a node.

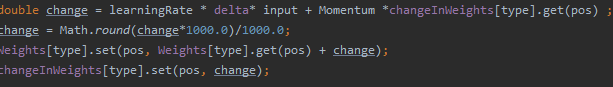


Figure Changing a singular weight.

## Item Adding

Graphical user interface, application

Description automatically generatedThe main function of this screen is giving the user the ability to add items into the wardrobe. To do this the activity has been designed to clearly show what properties are needed to add an item to the wardrobe. This can been seen with the editable textView, the Radio buttons defining the categories and the button to take a picture of the item to be added. When the user want to add an item by pressing the confirm item, it will try to add all the information from the activity and add them to a new Clothes Object. It will attempt all of these, but if any component is missing it will change the textview message at the top of the screen to tell the user what they need to fill in. This will prevent items being added which do not enough information while also informing the user on how to fix this.

With getting the type of clothing and the name of the item is done through pulling the set value through and passing them though. However to get a picture taken another activity needs to be run. This intent is using a MediaStore class activity to start up image capturing. To be able to store the picture which is taken, a File object needs to be set up for the picture. The function will try to create a new File object and is it succeeds it will create a URI object. This will sets up the location to where the File object will get saved to. On the resolution of the picture capturing activity the image is stored to where the URI is set up to. Along with this, the picture which was just taken would then be displayed on a imageView in the Item Adding activity. This is done by taking the file path where the image is stored and using the bitmap factory object to decode the file so that I can be displayed. Most of this implementation was inspired by the Android studio guide to implementing image capturing into applications. [https://developer.android.com/training/camera/photobasics]

With all of the necessary information filled in for a item, the neural network will be able to execute the function, additem. This function uses the passed information to create a new clothes object which is then added to the wardrobe ArrayList at the respective type. As whenever an item is added to the wardobe this creates new possible outfits. To add these new outfit, the new item is added with every other possible combination of the other 2 types. All of these outfit would then be added to the Outfits list. For the neural network, new weights need to be introduce. For the clothes inputs, a new weight is created for the respective type that the item is. For the outfit inputs, it will add a new weight to them for every new outfit which is created during this process.

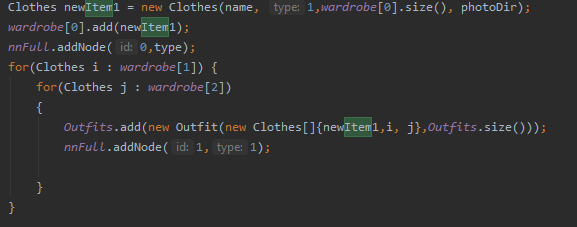


Figure An example of a type 1 item being added to the wardrobe

## Wardrobe Activity

This activity lets the user see all the item which are in the wardrobe and gives them the option to edit the items. When the user enters the screen, all the item are displayed in the purple wardrobe box. On this initial screen the user is met by a drop drown bar. This bar contains all 3 types which the clothes can be. The user can select a type from the spinner which on selecting the type which they want will only display items from that type. This is obtained from a Neural Net function called getAllNamesOfType. From the passed value, sent on the selection of the spinner type, it will loop for the length of that wardrobe type, adding each item’s name into a string. This string is comma separated so that once the function has completed, the returned string then be separated and used for the TextView text.

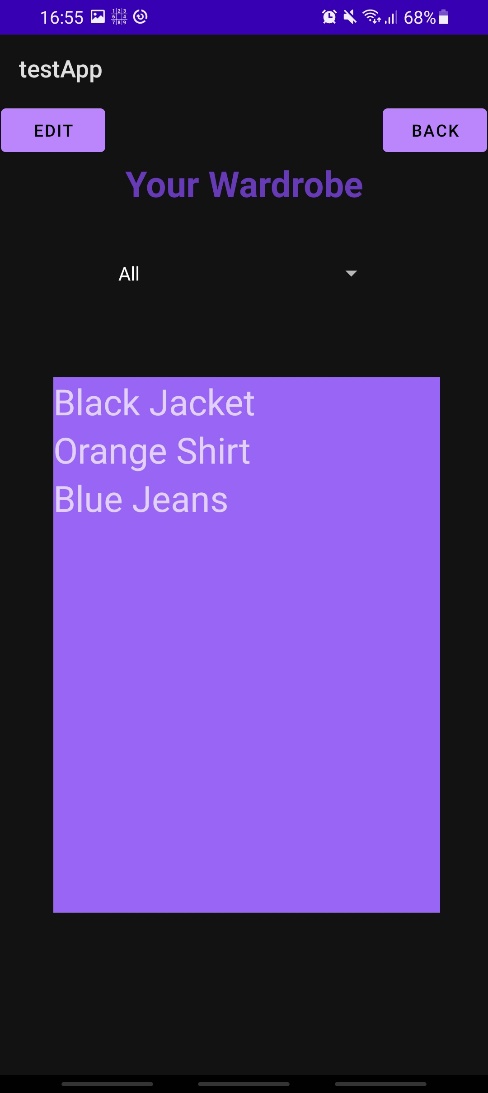


Figure Displaying the wardrobe

In the wardrobe box, there a multiple textview object all stored in a linear layout constrictor. Whenever parts of the wardrobe are shown, the layout is looped for the amount of children it has, being the textviews, and changing the text to the respective item of clothing.

When items are in the wardrobe box they can be selected. This is when a red box is draw around the item which is selected. This allows for the user to see which item they have have selected and tells the activity which item to edit. This is done through having each Textview object have an onClickListener. This is initialised once the Activity has been started and when the wardrobe is being read. This means when ever a TextView is pressed it will run the highlight function. This function works by having each TextView pass their ID within the linear layer constraint. This is then stored within the activity as it will be used for other functions. The function works where is no other object in the item has already been selected it will highlight the object which was passed. If there was already an item which was selected, it would turn that item’s background to transparent before changing the currently pressed object to the red highlighted colour. This works the same way if the an already highlighted item has been pressed again. Along with that it would also set the currently highlighted value to -1 to show nothing was selected anymore. This important as to make the editing screen work as intended, it needs to prevent the user form trying to edit no items due to no objects being selected. Along with that -1 is also a value which could not be produced by an index.

With an item selected the user would be able to select the edit button. This changes the wardrobe box to have all the information about the selected item. This is done by making the wardrobe layout constraints visibility set to gone and turning the editing layout constraints visibility to visible. In the Editing layout constraint it shows the item’s name, Type, and weight in the neural network. Along with this they are presented with multiple options to change elements about the item they have selected. First is changing the name of the item, which reads the text int the editTextView and passes that to a function within the NeuralNet which uses a Clothes object function to set the new name.

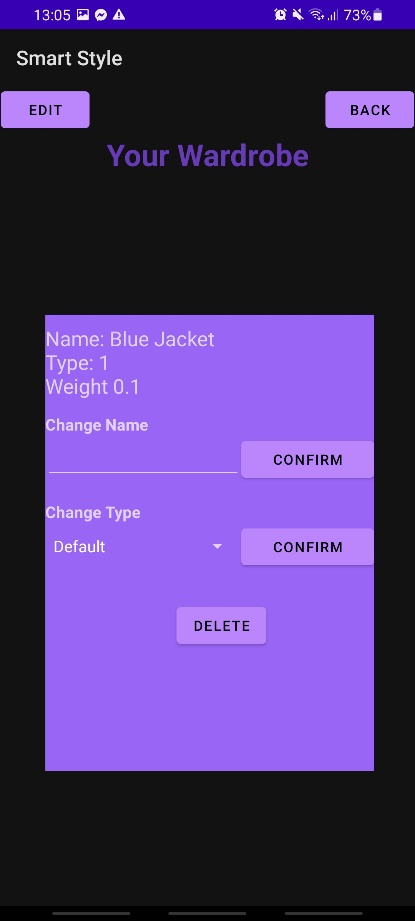


Figure Editing screen for an item

When the user wants to get rid of an item them can use the delete button which is provided. Certain considerations needed to be made as the item needs to be removed without changing the Neural network irreversibly. To do this the NeuralNet function removeAnItem is used. This function gets the desired Clothes object to be removed and stores it in a temporary clothes object. It first removes the item from the wardrobe itself using the old item’s id and type to remove the item at the correct location. After that the weight associated in the individual items inputs in the first layer of the network. Finally it has to remove all the possible outfits it was a part off. To do this is loops for the entire Oufits list and adds the Outfits IDs when the item is found comprising the outfit. Once all the Outfits which need to be removed have been found, it will loop through the list from back the highest id to the last removing the Outfits objects and the associated Weights in the input layer one by one. It is done this was as when an item in the list is removed, any item items after it on the list are moved down the list meaning that their index would change. Therefore it is needed to go from the newest outfit with the item in to the oldest. Once completed the selected object would have been removed from the Network and the user would be taken back to the original wardrobe screen.



Figure Process to delete an item

Another button is the changing type button. This changes the type of the item to any of the other types. This is done by taking the value selected by the Spinner object, and pass it through to the NeuralNet. From there the setType function takes the all the information about the object and stores it. From there it delete the current object form the wardrobe. This takes advantage of the preestablished removeAnItem function. Once removed from the wardrobe, the a new item is introduced with all the old information from the item but with the new type. This will lead to the item being successful being transferred from one type to another without leaving the original behind. This ensures that the neural network doesn’t try and recommended the old object as it would not longer exist in the same capacity.

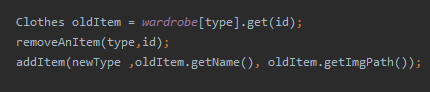


Figure Transferring process

# Testing and Discussion

The main aspect of the application which needs to be testes, is its performance in being able to learn the users taste in clothes. To test the network, the accuracy of the prediction after a set amount of train at multiple intervals would be used. This would test the network after training from 0 – 200 epochs going in iteration of 10 testing epochs to 100 where the next testing values would be 150 epoch and 200 epochs. For testing, a wardrobe and tastes for that wardrobe needed to be created. This was done by using a testProfile. The test profile would simulate a user using the app. With each profile it would contain all the items of clothing they have and the relationship ships between each item. This would act as their tastes. To decided whether or not the simulated user will like or dislike the outfit, a score it created from the outfit and the outfit score needs to be able to exceed the threshold. This threshold is a score of 1 or more compatible items of clothing. For example profile A might have Top X which they like with Bottom Y but dislike with UnderTop Z. This will mean that the user will not choose the item as this outfit would produce a score of 0 as the relationship with Top X and Bottom Y would be +1 while the relationship between Top X and UnderTop Z would be -1. This would lead to an overall score of 0 which would not be accepted for the threshold.

This testing method does have limitation however. This is due to not have having historical data of real people outfits choices. This means that we have to rely on a simulation of it which has the limitation of not be able to properly predict the how the user will react due to potential spontaneity of results. As well as this, as users taste change over time it means that the relationships between item would not be static and would be prone to changing. As well of this there are external factor when address what outfit to wear as the user may like the outfit but not be suitable for the current weather conditions leading to a favour outfit being rejected. Another limitation of testing the network is that there are elements of randomness, even though it is skewed. This means that multiple answer will be obtain from the same settings. To alleviate this for each test case, an average of 5 separate testing runs at the same setting will be used as the value for the respective epoch. This is done in the effort to get a more accurate results and being less susceptible to outliers in results. Along with this, it will be tested while have a single hidden layer of neurons.

The goal of the testing is to prove that the neural network can learn tastes over multiple recommendations. It should show an upward trend with the % of good recommendation increasing at a reasonable rate. It should also show the ideal settings of the neural network to be set at. This would be found by a combination of speed of learning, accuracy, and variance in results. The setting which will be changed are the learning rate, momentum, and the profiles. The hypothesis is that by increasing the learning rate the accuracy will increase at a faster rate, but may be prone to large variance due to potentially unlearning key bits of information, while lowering the rate should produces a slower but smoother learning curve. With the momentum, it’s expected that it will help the learning when increased as it should be able to emphasise the direction the weights were learning in. This should help lessen the effects of a popular item being used in clashing outfit.

Relationship tables

Profile 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Hoodie | Black Jacket | Band Shirt | Black Shirt | Grey Shirt | Black Jeans | Grey Jeans |
| Hoodie | N/A | N/A | +1 | -1 | 0 | 0 | +1 |
| Black Jacket | N/A | N/A | 0 | +1 | 0 | +1 | 0 |
| Band Shirt | +1 | 0 | N/A | N/A | N/A | -1 | 0 |
| Black Shirt | -1 | +1 | N/A | N/A | N/A | -1 | -1 |
| Grey Shirt | 0 | 0 | N/A | N/A | N/A | +1 | 0 |
| Black Jeans | 0 | +1 | -1 | -1 | +1 | N/A | N/A |
| Grey Jeans | +1 | 0 | 0 | -1 | 0 | N/A | N/A |

Profile 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Suit Jacket | Hoodie | Coat | Jumper | T-shirt | | Long Sleave | Vest | Polo | Band Shirt | Smart Trousers | Trousers | Shorts |
| Suit Jacket | N/A | N/A | N/A | N/A | | -1 | 0 | -1 | 1 | -1 | +1 | 0 | -1 |
| Hoodie | N/A | N/A | N/A | N/A | | +1 | 0 | 0 | -1 | +1 | 0 | 0 | 0 |
| Coat | N/A | N/A | N/A | N/A | | 0 | 0 | -1 | 0 | 0 | +1 | +1 | -1 |
| Jumper | N/A | N/A | N/A | N/A | | 0 | 0 | 0 | 0 | 0 | 0 | +1 | -1 |
| T-shirt | -1 | +1 | 0 | 0 | | N/A | N/A | N/A | N/A | N/A | 0 | 0 | 0 |
| Long sleave | 0 | 0 | 0 | 0 | | N/A | N/A | N/A | N/A | N/A | 0 | +1 | 0 |
| Vest | -1 | 0 | -1 | 0 | | N/A | N/A | N/A | N/A | N/A | -1 | 0 | +1 |
| Polo | +1 | -1 | 0 | 0 | | N/A | N/A | N/A | N/A |  | +1 | 0 | 0 |
| Band Shirt | -1 | +1 | 0 | 0 | | N/A | N/A | N/A | N/A | N/A | -1 | 0 | 0 |
| Smart Trousers | +1 | 0 | +1 | 0 | | 0 | 0 | -1 | +1 | -1 | N/A | N/A | N/A |
| Trousers | 0 | 0 | +1 | +1 | | 0 | +1 | 0 | 0 | 0 | N/A | N/A | N/A |
| Shorts | -1 | 0 | -1 | -1 | | 0 | 0 | +1 | 0 | 0 | N/A | N/A | N/A |

These tables show all the relationship between all the items in the profile’s wardrobe. Any item with the relationship value N/A means that either it’s a duplicate item or that it is an item which can not be in an outfit together and therefore can have no relationship.

**Test 1 –** Profile 1, Learning rate = 0.2, Momentum = 0.2

This test was to see if the NeuralNet would be able to learn a small sample size with the default settings. From the results in the graph, it shows that the NueralNet is capable to learn the a users taste. This is displaced as there the trend in the result is generally positive. This shows that the more time the network is trained the great the accuracy normally is. Along with this it shows that the learning slows down the closer the network gets to getting 100% accuracy. This is because the increments in learning become much smaller the closer the accuracy gets.

**Test 2 –** Profile 1, Learning rate = 0.5, Momentum = 0.2

As expected by increasing the learning rate to over double, it has cause the consistency of the prediction to decrease significantly. This is seen as the graphs has severe jumps from both high and low accuracies. However, it does still follow a general upward trend showing that it is generally learning the problem. The peaks and valleys shows may be due to the fact that with such a significantly higher learning will cause the weights in the network to go through more substantial change per epoch. This could be leading to the network unlearning to much information on certain epochs causing a larger influence in the direction of the learning. This shows that plainly increasing the learning rate is not a good solution to learning the tastes faster as the loss of consistency is unacceptable.

**Test 3 –** Profile 1, Learning rate = 0.1, Momentum = 0.2

In this test it has shown a very smooth curve for learning. The curvature of the learning is fairly shallow and is close to being more a line. This is indicative of the more consistent nature of the network learning the information. As expected, this is in anthesis of increase the learning rate which produced more inconsistency. However, a side effect of lowering the learning rate is that the time to learn more of the taste has increased as the peak in this is fairly lower than pervious examples which were able to get near perfect at their peak. This shows that even though a lower learning rate does improve consistency, there needs to be a middle ground where it can learn faster without hurt the consistency. This is because the average user would be unlikely to reach usage in the hundreds meaning it is vital to be able to provide a good experience without it taking too long to learn.

**Test 4 –** Profile 1, Learning rate = 0.2, Momentum = 0.5

When compared to the initial settings used for testing, the increase in momentum follows a very similar upward trend in learning creating a mostly smooth curve with as few outliers. Unlike when the learning rate was increased, the learning is seen to be much consistent. This is while being at a faster rate of learning than the initial settings. This is possibly due to the how momentum only emphasises the trend the weight had previously been on rather than in the learning rate’s case where it is control of the size of the adjustments made. From these results it shows that most likely, a higher momentum will be a greater benefit to improving the speed of learning without sacrificing the consistency.

**Test 5 –** Profile 1, Learning rate = 0.35, Momentum = 0.35

This graph shows the dramatic effect increasing both the momentum and the learning rate in tandem can have with the performance of the neural net. It shows how initially it learns the problem quickly as it take around 30 epochs to starting producing accuracy of >80%. This is overall an improvement as compared to the initial settings which took around 70 epochs of training to reach a similar value. This also produces a similar result as with only increasing the momentum as that reached this milestone at 50 epoch. This demonstrates that with the combination of both momentum and learning rate being increased the learning is improved considerably and more than each of them individually being altered. However this also emphasises prior issues as there is an area of inconsistency introduced as from 70 to 90 epochs of training the overall performance dropped. This was most likely due to the influence of learning rate indicating that to obtain better results for the task as the learning rate being increased at a lower rate than the momentum may be beneficial.

**Test 6 –** Profile 2, Learning rate = 0.2, Momentum = 0.2

This graph displays how even when the problem is scaled up it is still able to solve the problem and produce results with high accuracy. Unlike the comparatively smaller wardrobe size of the first profile, the rate at which the accuracy improves has flattened out much more than the curve seen in the original. This highlights a potential issue with scalability as the larger the wardrobe the slower the learning may become if the settings are changed to alleviate this. Though the network work is still able to produce good accuracy despite this, just over a longer period of testing. This is due to the fact that the wardrobe size is 5 times larger as profile 1 has a total of 12 outfit whiles profile 2 has a total of 60 outfits making it theoretically 5 times harder to learn the tastes. To improve the performance with a larger wardrobe needs to be considered.

**Test 6 –** Profile 2, Learning rate = 0.25, Momentum = 0.35

As tested previously, the setting of a higher momentum and a slightly high learning rate have proved to be successful in improving the performance of the network when given a larger amount of clothes to work with. This curve, while having dips, is generally smooth showing that progression in learning the problem is fair standard by having larger jumps in performance early and the difference in performance decreasing the longer more it’s learned. One thing which is highlighted is that even with the performance being better, it never reaches the same peaks as with a smaller wardrobe. However, this is to be expected due to the increase in possibilities and relationships to learn. Though the peak performance seen here at over 90% would still be very satisfactory for the goal of the network.

## Conclusion

Overall the testing has displayed that the neural is very much capable of learning the relationships between item of clothing in the wardrobe and is able to learn to pick better outfits over times. It has also been discovered that in long term usage of the application there may arise some scalability as the network may take a lot longer to learn tastes with greater scale of outfits. This could be solved in the future by implementing changing learning rate/Momentum depending of the size of the wardrobe. As well it’s has be discovered that momentum is the more consistent tool to speed up the learning of the network. This is due to how is keeps the weight on it’s original trend meaning bad results will affect it less if it was previously highly favoured. Along with this learning rate was found to be good at increasing the peaks at which the network would learn but would also be very volatile as the larger the jumps in every time the network learns the bigger the impact of the future of the network is which can lead to the wardrobe tastes being unlearned, hurting the performance and the consistency.

# Conclusion

In conclusion, the project has been able to produce an android application for purpose of outfit recommendation. It has been able to implement a neural network into the app which is designed to learn the tastes in each user’s clothes. The neural net was able to implement Deep learning-esque model implemented a multi-layer perceptron network to better learn the information providers and learn the trend of information. The network has also been proven that through usage over time, the accuracy of the it increases to a point where most of the predictions are ones which would be received positively. This shows that it can learn the tastes within a reasonable time improving the user experience through more accurate recommendation. Along with this by having the network be proven to give improved prediction over time, it will be able to reduce the amount of decision which have to be made by the user, which will have fulfilled the goal of reducing the amount decision fatigue over time.

The functionality of the app has also been successfully implemented as it has multiple screens developed for multiple specific function. It has been able to give the user the ability to receive recommendations on outfits, add new items of clothing into the app, and to view, edit, and delete any item within the app. It was also able to produce a visual representation of each outfit being recommend from the picture taken of each object being displayed on the recommender screen activity. The editor screen has also given the user all the functionality to change items in the wardrobe on a whim, fulfilling the need aim of creating customisability in the application.

Overall the project has been able to satisfy all the basic aim which it set out to achieve with a functional android application being the end result of the development.

# Future Works

The project has been able to succeed in creating the foundation for the desired application. However what has been created is still very surface level. Even though it was able to implemented a Neural Network, this network runs on a few layers of hidden layers. This leaves room for this to be expanded for greater learning. Along with this the way that each item of clothing is ranked is simplistic as it only considers the item as an individual item and as part of an outfit. This is compounded by the limited but broad classification system for the items of clothing. This leading to limiting the design of outfits. As well even though the feature available with this app are satisfactory more could be implemented to expand to greatly improve the level customisability.

In the future if the project was to be expanded, I would have expanded the depth to the functionality of the application. This would be seen mostly in the amount of information each item of clothing had. One way this could be expanded in the future would be having temperature values associated with each item. This would have allowed features such as temperature of the outfits being a consideration in the selection process. This would have been done by taking the current weather at the user’s location and using that to filter out outfits which would exceed the range of outfits which would be comfortable in that weather. Along with using weather as another factor in recommendation, the clothes classification system would be expanded to the have more sub categories. This would not only allow for more specific descriptions of each item of clothing, it would allows for each sub-type of clothing to having their own influence in the recommendation process. This would mean that each item would be judged by each item’s individual score, the outfit score but also the score produced by the sub-type of clothes. This would provide more data to for the network classify the items with which would potentially lead to even better accuracy.

To build of what has already been created, implementation of social media aspect could be a good way to expand the current system. Implementing this could increase the user’s satisfaction due to being able to connect and show of their outfit they got recommended to them with other users along with being able to get inspiration from other users. As well as this by creating implementing more online aspects to the app, it could use more use collaborative filtering technique to speed up the initial learning of the users taste by basing them of similar taste of other users in the application.

# References

1. (Lamothe, 2019)

Lamothe, C. (2019). *Decision Fatigue: What It Is and How to Avoid It*. Healthline. Retrieved 21 April 2021, from <https://www.healthline.com/health/decision-fatigue>.

1. ("Nervous system - The nerve cell"

*Nervous system - The nerve cell*. Encyclopedia Britannica. Retrieved 22 April 2021, from <https://www.britannica.com/science/nervous-system/The-nerve-cell>

1. (Chandra, 2017)

Chandra, A. (2017). *McCulloch-Pitts Neuron — Mankind’s First Mathematical Model Of A Biological Neuron*. Medium. Retrieved 22 April 2021, from <https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1>.

1. ROSENBLATT, 1958)

ROSENBLATT, F. (1958). *THE PERCEPTRON: A PROBABILISTIC MODEL FORINFORMATION STORAGE AND ORGANIZATIONIN THE BRAIN* [Ebook]. Retrieved 21 April 2021, from <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.335.3398&rep=rep1&type=pdf>.

1. (Team, 2017)

Team, S. (2017). *Hey Siri: An On-device DNN-powered Voice Trigger for Apple’s Personal Assistant*. Apple Machine Learning Research. Retrieved 22 April 2021, from <https://machinelearning.apple.com/research/hey-siri>.

1. ("Welcome - Faceswap", 2021)

*]Welcome - Faceswap*. Faceswap. (2021). Retrieved 22 April 2021, from https://faceswap.dev/#content.

1. ("What is Deep Learning?", 2020)

*What is Deep Learning?*. Ibm.com. (2020). Retrieved 21 April 2021, from

<https://www.ibm.com/cloud/learn/deep-learning>.

1. (Ricci, 2011)

Ricci, F. (2011). *Introduction to Recommender Systems*. Retrieved 21 April 2021, from.

1. (Ilic & Kabiljo, 2015)

Ilic, A., & Kabiljo, M. (2015). *Recommending items to more than a billion people - Facebook Engineering*. Facebook Engineering. Retrieved 22 April 2021, from <https://engineering.fb.com/2015/06/02/core-data/recommending-items-to-more-than-a-billion-people/>.

1. (Kuraa, 2019)

Kurama, V. (2019). *A Simple Introduction to Collaborative Filtering*. Built In. Retrieved 22 April 2021, from <https://builtin.com/data-science/collaborative-filtering-recommender-system>.

1. (Sharma, 2019)

Sharma, A. (2019). *Neural Collaborative Filtering*. Medium. Retrieved 22 April 2021, from <https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401>.

1. Covington, P. *Deep Neural Networks for YouTube Recommendations*. Storage.googleapis.com. Retrieved 22 April 2021, from https://storage.googleapis.com/pub-tools-public-publication-data/pdf/45530.pdf.
2. (Werbos, 1974)

Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and analysis in the behavioral sciences*.

1. *A multilayer perceptron,Neural Networks lecture notes*. Faadooengineers.com. (2021). Retrieved 21 April 2021, from http://www.faadooengineers.com/online-study/post/eee/neural-networks-and-fuzzy-logic/1478/a-multilayer-perceptron.

("A multilayer perceptron,Neural Networks lecture notes", 2021)

1. *What are Activation Functions in Neural Networks?*. GreatLearning Blog: Free Resources what Matters to shape your Career!. (2020). Retrieved 21 April 2021, from <https://www.mygreatlearning.com/blog/activation-functions/>.

("What are Activation Functions in Neural Networks?", 2020)

1. McGonagle, J. (2021). *Backpropagation | Brilliant Math & Science Wiki*. Brilliant.org. Retrieved 21 April 2021, from https://brilliant.org/wiki/backpropagation/.

(McGonagle, 2021)

1. Divakar, V. (2019). *Understanding The Chain Rule*. QuantInsti. Retrieved 21 April 2021, from <https://blog.quantinsti.com/understanding-chain-rule/>.

(Divakar, 2019)

1. *What is Unsupervised Learning?*. Ibm.com. (2020). Retrieved 21 April 2021, from <https://www.ibm.com/cloud/learn/unsupervised-learning>.

("What is Unsupervised Learning?", 2020)