AUTOMATIC EBAY LISTER

Aaron Moorey Candidate Number: 9462, Center number:64395, Godalming College

Contents

Research

The aim of this project is to create a system that allows eBay sellers to list items much faster than by using eBay's website. I will aim to make a program which takes an image of a football shirt inputted by the user, works out the club of the shirt, and lists it on eBay, the user will also provide the year and size of the shirt and their PayPal email. I will use a neural network to work out what club the shirt image is, and then once the program has found out what the club is, it will search on eBay using the eBay API to find similar sold items and list it on to eBay in a similar way, with a similar price and title to others that have sold. There will be minimum input from the user which I believe makes it a useful program to someone that wants to list items on eBay because of the time saved.

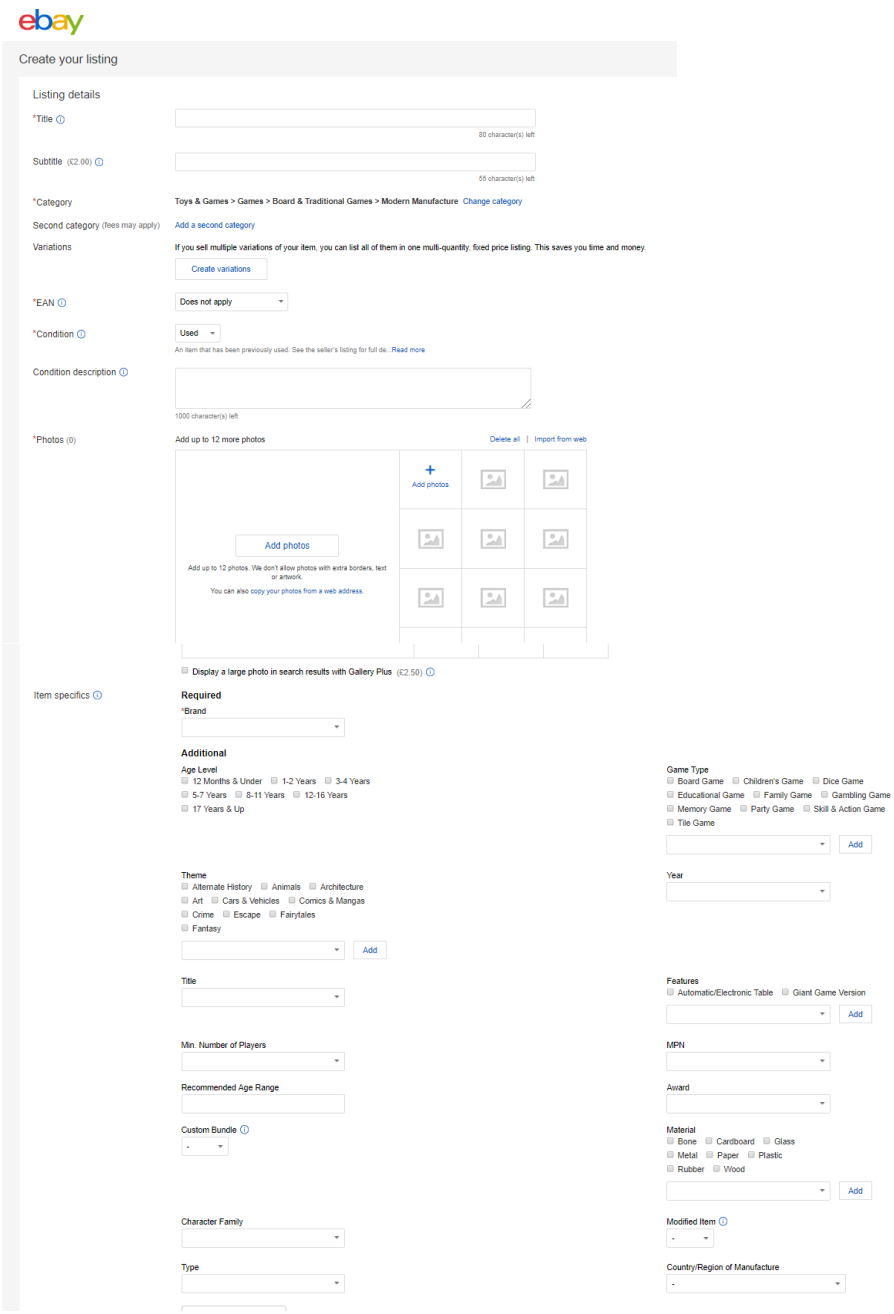
From this research I will conclude whether someone that lists on eBay would find this program useful, what a neural network is, what an eBay item is and consists of, and what Python Tkinter and the eBay API are.

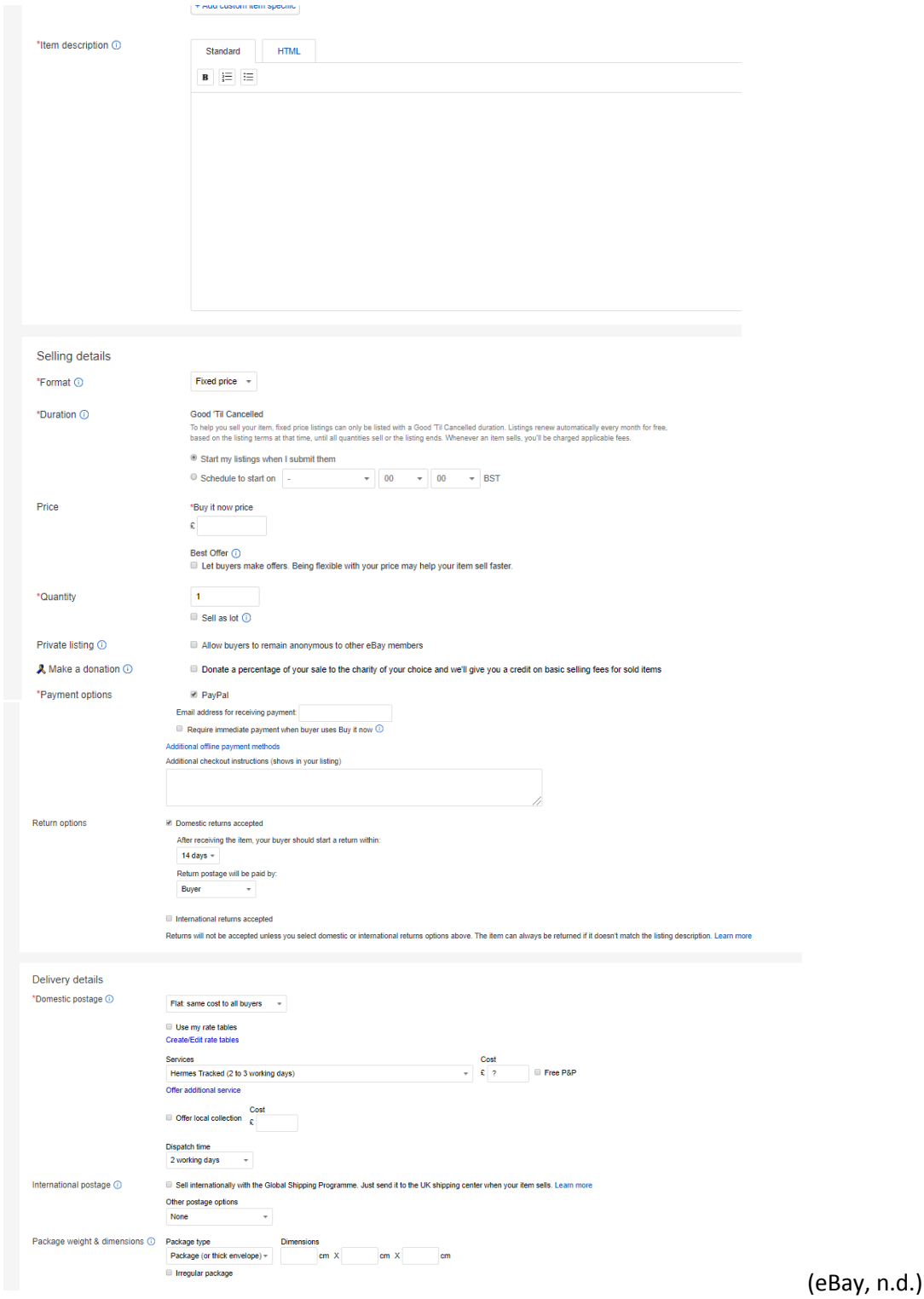
What is eBay and what is an eBay item?

eBay is a worldwide online shopping site. Anyone can post items to sell on eBay, either on auction (where buyers can bid and the item ends on a certain date), or on buy it now (BIN, where the item has a set price which the buyer must pay unless another price between the seller and buyer is agreed). Selling on eBay is used by the general public and by businesses so is an ideal platform for anyone.

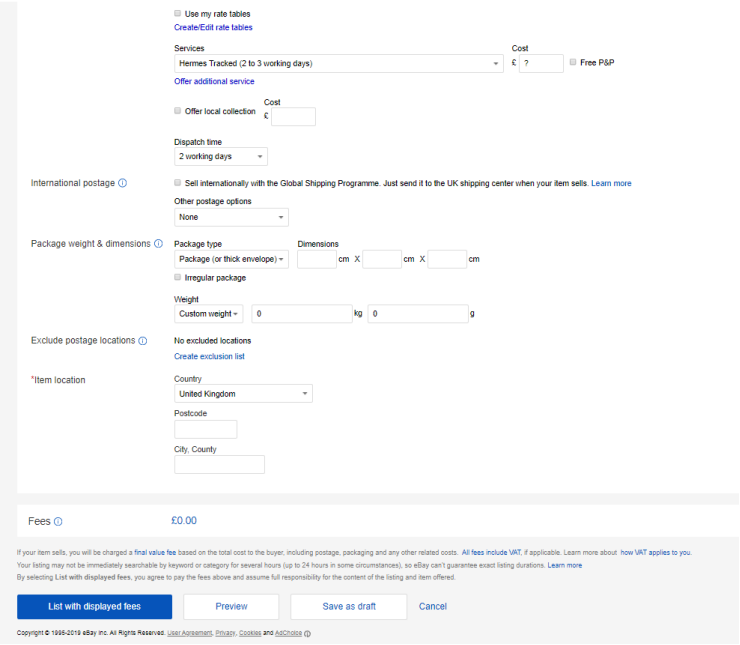
From my personal experience of using eBay, I have found that listing items on eBay can be time consuming, because you have to search through any sold items to check how best to title your item and what the best price is. My program will save a substantial amount of time for sellers on eBay because they will not have to find out the price and title themselves, the program will do this for them. However, they will have the option to tweak any specifics about the item if they would like.

Below shows what listing an eBay item manually looks like:




*Item description
Standard
HTML
Selling details
*Format
Fixed price
*Duration
Good 'Til Cancelled
To help you sell your item, fixed price listings can only be listed with a Good 'Til Cancelled duration. Listings renew automatically every month for free, based on the listing terms at that time, until all quantities sell or the listing ends. Whenever an item sells, you'll be charged applicable fees.
Start my listings when I submit them
Schedule to start on
BST
Price
*Buy it now price
Best Offer
Let buyers make offers. Being flexible with your price may help your item sell faster.
*Quantity
Sell as lot
Private listing
Allow buyers to remain anonymous to other eBay members
Make a donation
Donate a percentage of your sale to the charity of your choice and we'll give you a credit on basic selling fees for sold items
*Payment options
PayPal
Email address for receiving payment:
Require immediate payment when buyer uses Buy it now
Additional offline payment methods
Additional checkout instructions (shows in your listing)
Return options
Domestic returns accepted
After receiving the item, your buyer should start a return within:
14 days
Return postage will be paid by:
Buyer
International returns accepted
Returns will not be accepted unless you select domestic or international returns options above. The item can always be returned if it doesn't match the listing description. Learn more
Delivery details
*Domestic postage
Flat: same cost to all buyers
Use my rate tables
Create/Edit rate tables
Services
Cost
Hermes Tracked (2 to 3 working days)
Free P&P
Offer additional service
Offer local collection
Cost
Dispatch time
2 working days
International postage
Sell internationally with the Global Shipping Programme. Just send it to the UK shipping center when your item sells. Learn more
Other postage options
None
Package weight & dimensions
Package type
Dimensions
Package (or thick envelope)
Irregular package

(eBay, n.d.)



ebay.co.uk 1

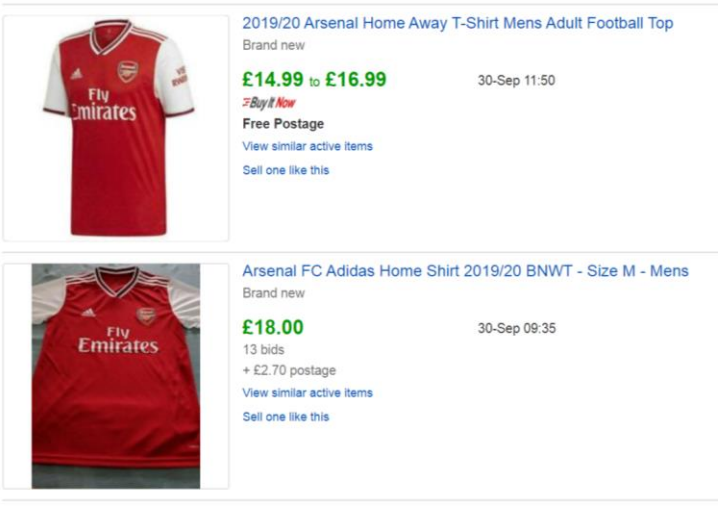
(eBay, n.d.)

Football Shirts

I have decided that I will aim to get my program to work for telling the difference between certain football clubs' shirts. This is because the way that photos are taken of them is relatively similar for all football shirts on eBay.

I will obtain the pictures which I will use for my training data (explained below) from previously sold eBay items. To do this I will write a separate program to get images of certain football clubs' shirts and put them into a file and then resize them all to 50 x 50 pixel JPEG images.

Here is an example of what listings of football shirts looks like on eBay:



ebay.co.uk 2

(eBay, n.d.)

What is a neural network?

A neural network is a computing system based around biological neural networks. The system can learn to solve problems without being told what to do specifically. A neural network has a large number of processors that operate in parallel with each other and are arranged as tiers. The first tier receives raw input, each tier after that then receives input from the tier before it and then passes on its output to the tier in after it. The last tier gives a final output. Each tier is made up by nodes, which are connected with the nodes in the tiers before and after it. These nodes perform certain operation on the input they receive and pass on the result as output to the next tier.

Types of neural networks:

Feedforward – this is where the inputted data passes through the nodes until it reaches the output node. The sum of the inputs and their weights are calculated and are then transferred to the output.

Multilayer Perceptron – this has at least three layers. Every node in one layer is connected to each node in the next layer, so is fully connected. It also uses an activation function, which defines the output of a node.

Convolutional – this uses more than one multilayer perceptron's. It performs a convolutional operation on the input before passing it in, this means the network can be much deeper but with fewer parameters. It contains one or more convolutional layers which can be interconnected or pooled.

Recurrent neural network – this is where the output of a layer is saved and fed back to the input which helps predict the outcome of the layer. If the prediction is wrong, then the system learns and uses backpropagation to make the right prediction.

(A.Mehta, 2015)

For my neural network, I will use pictures of sold items of football shirts for my training data (the data which will be put into the neural network to make the neural network learn). Each input will consist of an image of a football shirt. Each input will have a target array matched up with it. For example a Chelsea shirt input might have a target array output of [1,0,0,0,0]. The network will train with my training data and by the end of the training it should be able to work out which football shirt an inputted image is.

The user will be asked to input an image of a shirt and this will be passed through the neural network, the program will then display what club it thinks the shirt belongs to. It will ask the user to confirm that the program has got the correct club name. If it is not the same then the user will have to input the correct club name the network will train again with the image inputted. When the user confirms it's the same, they will have to input certain details about the shirt, like the size, year and also their PayPal email.

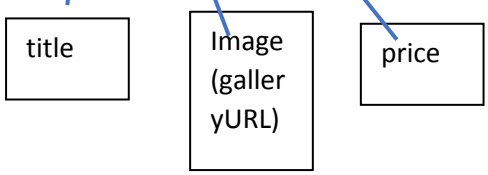
It will then use these details to perform a search using the eBay API to find others of the same items to it that have previously sold. Using the information of the previously sold

items, it will make a listing for the item the user input, by working out what price it should have, and the title.

Finally, the user will be offered the chance to make any changes to the listing and then confirm that it is fine to be listed on to eBay.

Here is an example of item information I can pull out using the eBay API by searching by a specific category, you can also search by other factors such as by keywords in the title:

{"findItemsByCategoryResponse":[{"ack":["Success"],"version":["1.13.0"],"timestamp":["2019-09-18T13:57:06.438Z"],"searchResult":[{"@count":"2","item":[{"itemId":
["312772447156"],"title":["Pearl Flute Quantz 665 Series Professional PF-665RBE PF-665"],"globalId":["EBAY-US"],"primaryCategory":[{"categoryId":["10183"],"categoryName":
["Flutes"]}],"galleryURL":["https:\/\/thumbs1.ebaystatic.com\/m\/mmIJIVLEGBc_bXPVzJNuwtA\/140.jpg"],"viewItemURL":["https:\/\/www.ebay.com\/itm\/Pearl-Flute-Quantz-665-Series-
Professional-PF-665RBE-PF-665-\/312772447156"],"paymentMethod":["CashOnPickup","PayPal","VisaMC","AmEx","Discover"],"autoPay":["false"],"postalCode":["93534"],"location":
["Lancaster,CA,USA"],"country":["US"],"shippingInfo":[{"shippingServiceCost":[{"@currencyId":"USD","__value__":"0.0"}],"shippingType":["Free"],"shipToLocations":
["Worldwide"],"expeditedShipping":["false"],"oneDayShippingAvailable":["false"],"handlingTime":["1"]}],"sellingStatus":[{"currentPrice":
[{"@currencyId":"USD","__value__":"5.5"}],"convertedCurrentPrice":[{"@currencyId":"USD","__value__":"5.5"}],"bidCount":["8"],"sellingState":["Active"],"timeLeft":
["P4DT10H32M57S"]}],"listingInfo":[{"bestOfferEnabled":["false"],"buyItNowAvailable":["false"],"startTime":["2019-09-18T00:30:03.000Z"],"endTime":["2019-09-
23T00:30:03.000Z"],"listingType":["Auction"],"gift":["false"],"watchCount":["13"]}],"returnsAccepted":["true"],"condition":[{"conditionId":["3000"],"conditionDisplayName":
["Used"]}],"isMultiVariationListing":["false"],"topRatedListing":["true"]},{"itemId":["392431842590"],"title":["Yamaha 221 Flute"],"globalId":["EBAY-US"],"primaryCategory":
[{"categoryId":["10183"],"categoryName":["Flutes"]}],"galleryURL":["https:\/\/thumbs3.ebaystatic.com\/m\/mGDRd8aV5sS-mxAz3I5KCww\/140.jpg"],"viewItemURL":
["https:\/\/www.ebay.com\/itm\/Yamaha-221-Flute-\/392431842590"],"paymentMethod":["PayPal"],"autoPay":["false"],"postalCode":["97501"],"location":["Medford,OR,USA"],"country":
["US"],"shippingInfo":[{"shippingServiceCost":[{"@currencyId":"USD","__value__":"18.0"}],"shippingType":["Flat"],"shipToLocations":["Worldwide"],"expeditedShipping":
["false"],"oneDayShippingAvailable":["false"],"handlingTime":["3"]}],"sellingStatus":[{"currentPrice":[{"@currencyId":"USD","__value__":"51.0"}],"convertedCurrentPrice":
[{"@currencyId":"USD","__value__":"51.0"}],"bidCount":["7"],"sellingState":["Active"],"timeLeft":["P6DT10H32M3S"]}],"listingInfo":[{"bestOfferEnabled":
["false"],"buyItNowAvailable":["false"],"startTime":["2019-09-18T00:29:09.000Z"],"endTime":["2019-09-25T00:29:09.000Z"],"listingType":["Auction"],"gift":["false"],"watchCount":
["6"]}],"returnsAccepted":["true"],"condition":[{"conditionId":["3000"],"conditionDisplayName":["Used"]}],"isMultiVariationListing":["false"],"topRatedListing":
["false"]}]}],"paginationOutput":[{"pageNumber":["1"],"entriesPerPage":["2"],"totalPages":["75840"],"totalEntries":["151680"]}],"itemSearchURL":
["https:\/\/www.ebay.com\/sch\/10181\/i.html?_ddo=1&_ipg=2&_pgn=1"]}]}



To find out how I will make my neural network and use the eBay API I have read many online resources and I have also watched YouTube videos. These have helped me understand what is needed for my program to function correctly. I have looked at what is needed in a neural network and also the logic and maths behind it to help me better understand it; in addition I have also looked at ways to make it more effective (with more or less layers, or more or less nodes in each layer).

Interview with two eBay sellers

Interviewee number 1 – he has been selling on eBay for over a year so has good experience about the current eBay listing system.

Interviewee number 2 – W. Horsley, he has also been selling on eBay for over a year so has good experience of the current listing process too.

1. What do you like about eBay?
   It's easy to use
   I like how it is easy to set parameters on searches etc, particularly on my phone
2. What don't you like about eBay?

eBay almost always sides with the buyer in any dispute
I don't like how long it takes to extract pictures from my phone onto my laptop and then onto eBay and then resizing and rotating them

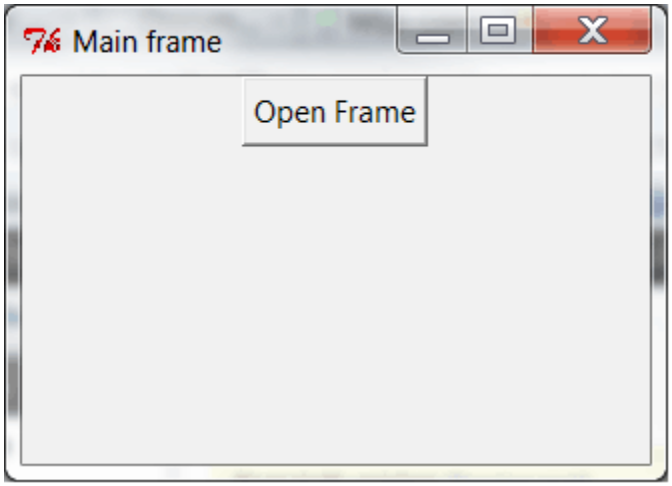
3. About how long does it take you to list an item?
   Around 5 minutes, depending on the item
   9-10 minutes
4. Do you feel it takes longer than necessary?
   No not really, only when relisting
   Absolutely
5. Do you spend lots of time looking up the price of other items?
   Yes
   Much longer than necessary
6. Do you find it interesting seeing the price of other items like yours?
   Yes
   I find it interesting to find out the value of what I am selling
7. Do you spend lots of time looking up what to include in your title for an item?
   No
   I spend longer than I want because I have to check a number of sold items as well as listed items on the product I am listing
8. Do you find it interesting finding out what to include in your title?
   No
   No I find it time consuming having to find lots of relevant words from listed and sold items to help maximise the searches I will get on the listing
9. What don't you like about the current listing method on eBay?
   Having to rotate and crop the images
   The number of separate stages there are to listing. Finding a price, finding a title and adding the pictures
10. Would you find something that lists items for you automatically from you just inputting an image useful?
    Yes
    Yes
11. Why/ why not?
    It would be less time consuming and less commitment for me
    Because it would mean I could list more items in a given period and it would reduce the time researching things like price, title etc

In conclusion, from these interviews I have found that both interviewees would find my program useful because it would save time. Additionally, both find it interesting to see similar sold items, therefore I will have to include this in my program somewhere. I have also found that both find that cropping and rotating images takes a long time.

eBay API and Tkinter

Tkinter provides an easy to use simple graphical user interface for python. I have had to learn how to use Tkinter, this will make my project look much better and will make it much easier for the user to use. By using Tkinter this will give my program a user friendly interface which will make it possible for anyone who wants to list a football shirt on eBay, be able to list the shirt simply and quickly. Here there will also be validation on user inputs, such as the user typing in the year of the shirt. This will also help with the ease of using the program by making sure there are no errors while it is running.

Here's an example of python Tkinter:



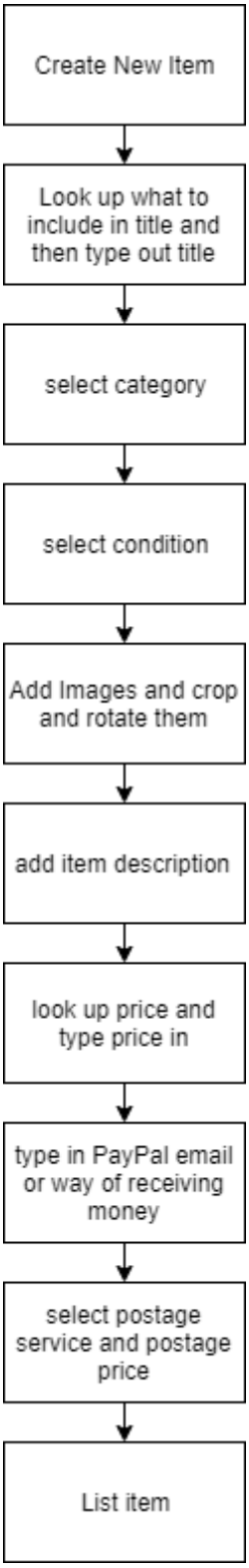
The eBay API (application programming interface) is used for interacting with the eBay database. The communication occurs over the internet. There are buy, sell, commerce and developer APIs. These allow you to search for items, list items, buy items and many other possibilities through code and is crucial for my program. For example, I will be using the trading API to list items and the finding API to search for recently sold similar items. For this I will need a file to store my API personal user keys to allow me to have access to eBay, however I will not show these in the documentation because they are private to my account. These will be stored in the file '`ebay.yaml`'.

(developer.ebay.com, 2020)

Overall, I will be making a program that takes an image of a football shirt that the user has inputted and lists the item on eBay by using a neural network and the eBay API to work out what football club the shirt inputted by the user belongs to and therefore work out the price, title and other attributes the listing should have. This should make the listing process for eBay sellers more efficient and save them time.

Analysis

Existing system flowchart

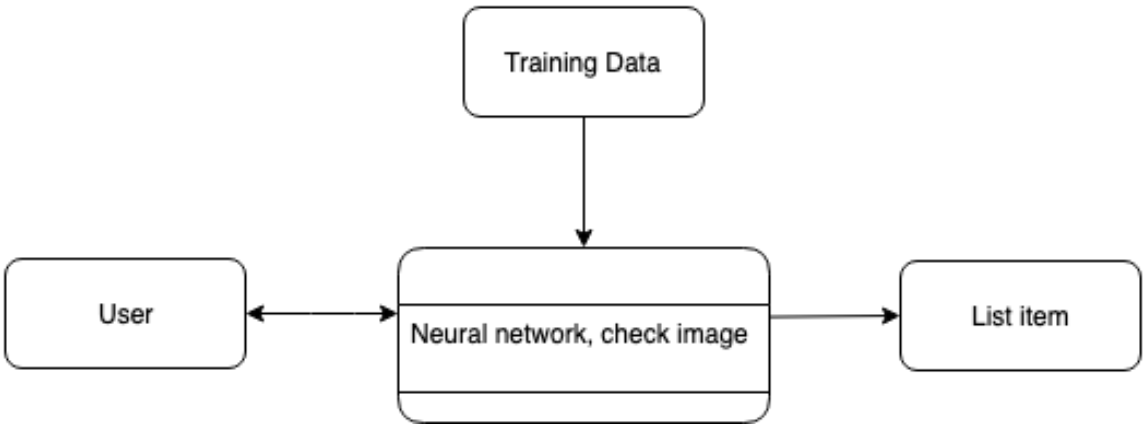


As you can see from the flow chart, listing an item requires many steps and is very repetitive. My program should stop it from taking so long to list items and make there less steps for the user, hopefully making it a less dull process.

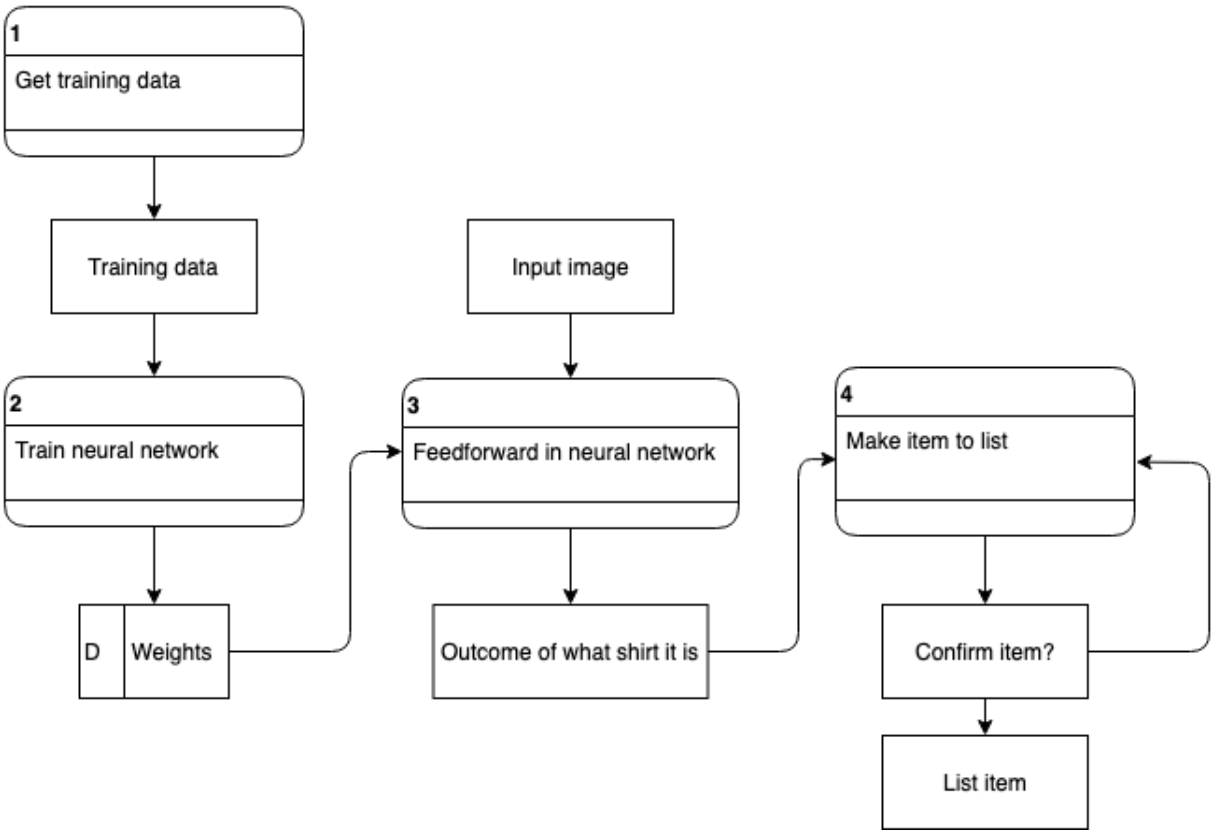
Data Flow Diagrams

Here are two data flow diagrams, one level 0 and two level 1s showing simply how the program I will make is expected to work.

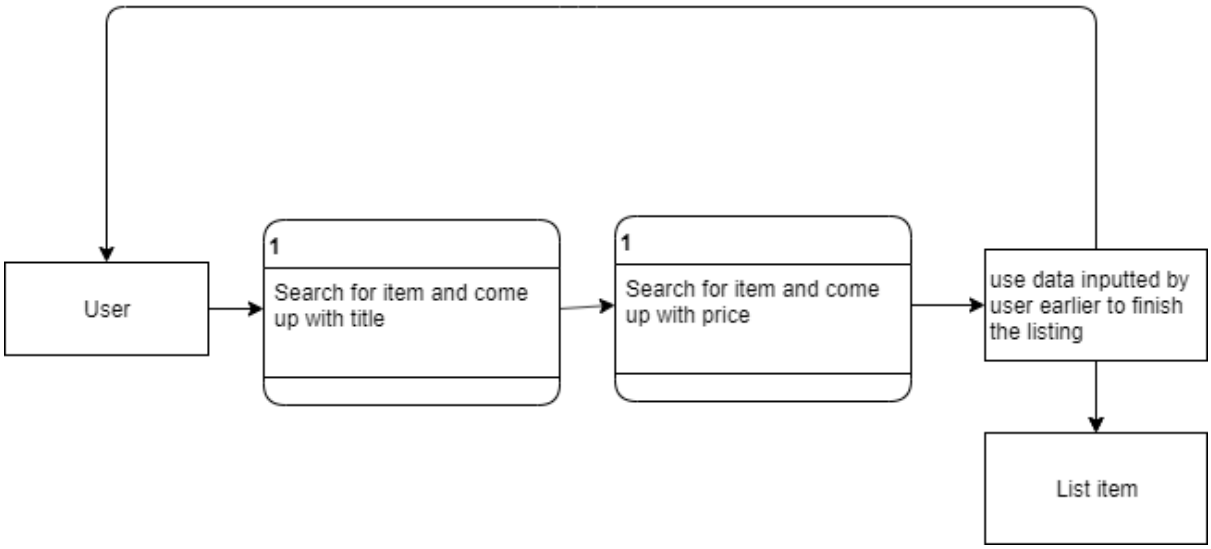
Level 0:



Level 1:



This diagram shows how the listing part of the program will work



Images

I will require many photos of each football club shirt for my training data. This will be collected by searching through current listings on eBay and taking the photo of each listing. A search for each club shirt will be carried out and then saved to a different folder for each club. There should be around 100 images for each club.

I will then manually look through each folder to make sure the images are of the correct shirts and delete any that are incorrect. The pictures will then be renamed by a separate program so that they are all named in numerical order.

I will use the python module Pillow to read the images. Each image will be resized to 50 x 50 pixels and saved as a jpg. I will then use Pillow to convert each image to an array. Each pixel is made up of 3 numbers for the RGB (red green blue) values for the colour. This means that an array will contain 7500 (50x50x3) numbers for each image.

Matrices

A matrix is a rectangular array of numbers in rows and columns.

These are examples of matrices:

$$A = \begin{bmatrix} 1 & 3 & 5 \\ -2 & 6 & 8 \\ -1 & -5 & 3 \end{bmatrix} \quad B = \begin{bmatrix} 3 & 5 & -2 \\ 0 & 6 & 0 \\ -3 & 2 & -3 \end{bmatrix}$$

$$C = \begin{bmatrix} 4 & 8 & 3 \\ -2 & 12 & 8 \\ -4 & -3 & 0 \end{bmatrix} \quad D = \begin{bmatrix} -2 & -2 & 7 \\ -2 & 0 & 8 \\ 2 & -7 & 6 \end{bmatrix}$$

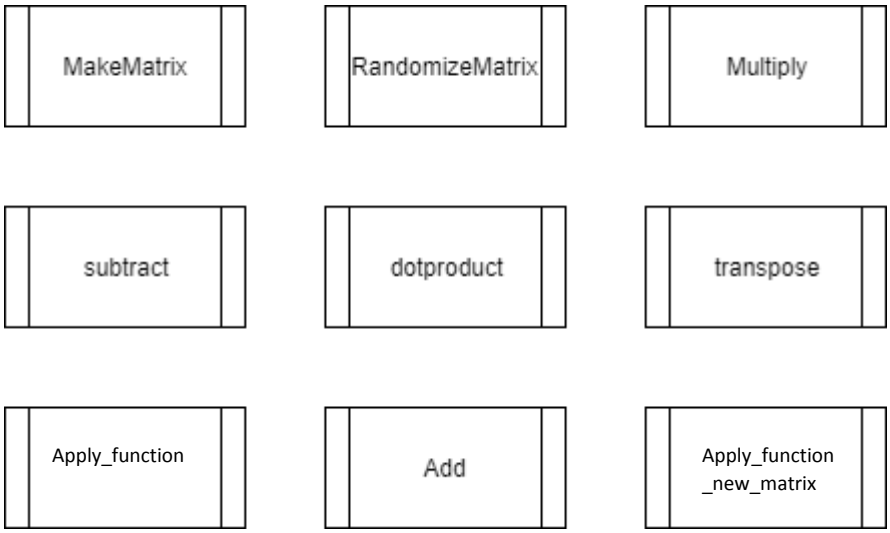
inquiry maths 1

(maths, n.d.)

Using matrices to store the weights and data in a neural networks is very useful, which I will go into more detail later showing how they are used.

I will require a separate class to do any matrix calculations required for the neural network, this is described below. The matrix class will take rows and columns as parameters in the initialisation function.

These are the sub procedures that will be in the class:



MakeMatrix – this sub creates an empty matrix with the dimensions specified by rows and columns.

RandomizeMatrix – this sub assigns a random number between -1 and 1 to each position in the matrix.

Multiply – this sub multiplies each element in the matrix by a number given as a parameter or multiplies two matrices together.

Subtract – this is where one matrix is subtracted from another and a new matrix is returned. Each element in the two matrices are matched up and subtracted from one another:

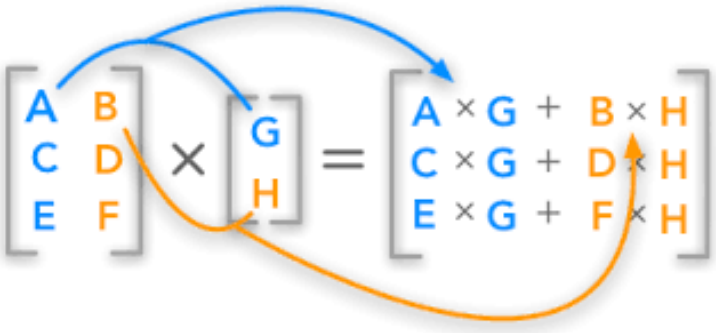
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} - \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1-9 & 2-8 & 3-7 \\ 4-6 & 5-5 & 6-4 \\ 7-3 & 8-2 & 9-1 \end{bmatrix}$$

$$= \begin{bmatrix} -8 & -6 & -4 \\ -2 & 0 & 2 \\ 4 & 6 & 8 \end{bmatrix}$$

codeforwin.org, 2015 1

(Codeforwin, codeforwin, 2015)

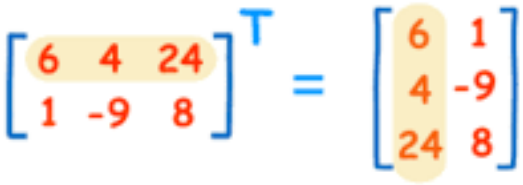
Dotproduct – the diagram below demonstrates the dotproduct multiplication, the rows and columns of two different matrices must match for this multiplication to work:



hadrienj - github, 2018 1

(github, 2018)

Transpose – this is where the matrix's rows are turned into a new matrix's columns, the diagram below shows how this works:



java67.com, 2016 1

(Java67.com, 2016)

Apply_function – this is where a specified function is applied to each element in the matrix.

Add - this is where one matrix is added to another and a new matrix is returned. Each element in the two matrices are matched up and added to one another:

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}+\begin{bmatrix}9&8&7\\6&5&4\\3&2&1\end{bmatrix}=\begin{bmatrix}1+9&2+8&3+7\\4+6&5+5&6+4\\7+3&8+2&9+1\end{bmatrix}$$

$$=\begin{bmatrix}10&10&10\\10&10&10\\10&10&10\end{bmatrix}$$

codeforwin.org, 2017 1

(Codeforwin, codeforwin, 2017)

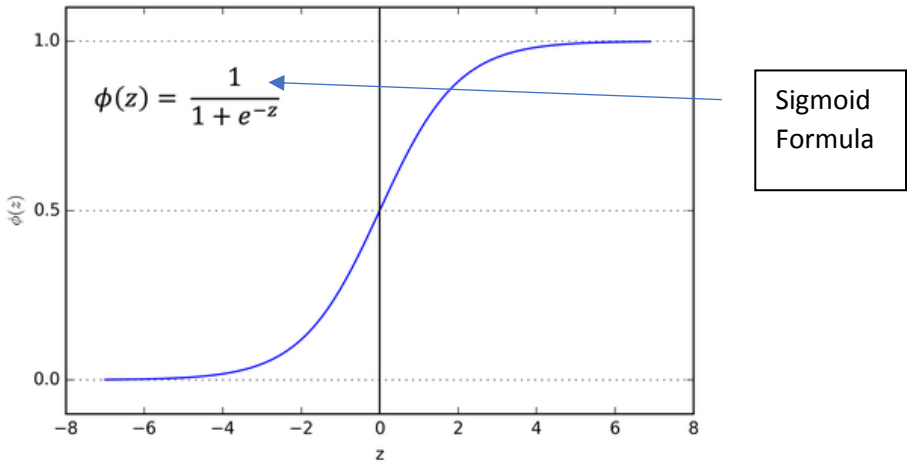
Apply_function_new_matrix – this is where a specified function is applied to each element in the matrix and a new matrix is created with these values in.

Activation functions

Activation functions take away the linear properties of the neural network and give it non-linear properties. They convert the input node to an output node which is then fed into the next layer as input. Giving the neural network non linear properties which means they can learn almost anything as linear functions are easy to solve and have limited complexity.

Possible activation functions:

The sigmoid function:



Sigmoid Formula

towardsdatascience.com, 2017 1

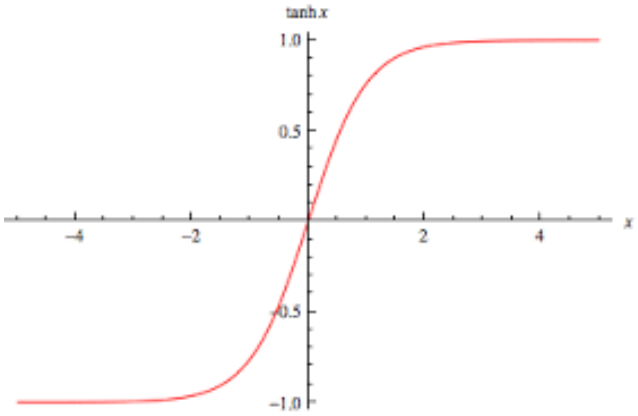
(Towardsdatascience, Towards data science, 2017)

This function returns a value in the range of 0 and 1 which is useful for normalising the weights for the neural network. However, this function makes the gradient changes go too

far in different directions and has a vanishing gradient problem as shown in the diagram on the left where the gradient turns to 0 at either end.

Hyperbolic tangent function:

This produces values between -1 and 1, which makes it 0 centred, therefore optimization is easier than with sigmoid. However, this still has the vanishing gradient like sigmoid.



mathworld.wolfram.com 1

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

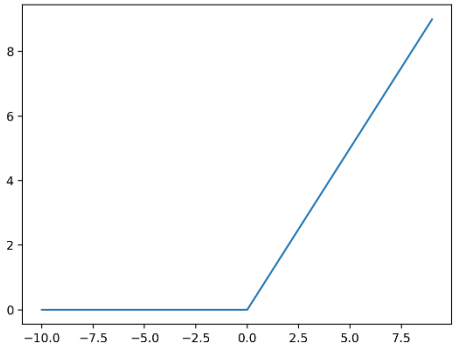
oreilly.com 1

(Mathworld, n.d.)

Rectified linear units:

This is where if x < 0 then R(x) = 0 and if x >= 0 then R(x) = x

This is very simple and also loses the vanishing gradient problem found in both sigmoid and the hyperbolic tangent function. However, this should only be used for the hidden layers.

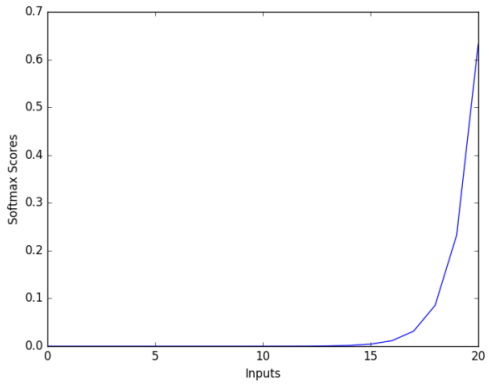


machinelearningmastery.com, 2019 1

(machinelearningmastery, 2019)

Therefore, a softmax function should be used for the output layer, this returns a value between 0 and 1 and all the outputs add up to 1. The high value of the outputs will have the highest probability.

The softmax graph looks like this:



dataaspirant.com, 2017 1

(Dataaspirant, 2017)

Softmax formula:

$$F(X_i) = \frac{Exp(X_i) \quad i = 0, 1, 2, \ldots k}{\sum_{j=0}^{k} Exp(X_j)}$$

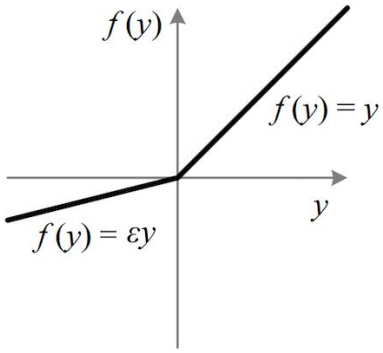
dataaspirant.com, 2017 2

(Dataaspirant, 2017)

Leaky ReLu:

Some gradients with ReLu can be fragile during training and die, this could result in dead neurons.

To fix this leaky ReLu is used which has a small slope before 0 to stop neurons dying.



ayearofai.com, 2016 1

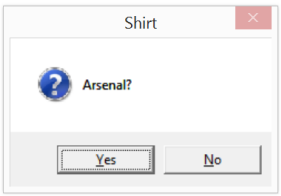
(ayearofai, 2016)

I will be using sigmoid and softmax for my program as the softmax gives probabilities which works well for me working out which football shirt an image is.

(Sharma, 2017) – activation functions

After the user has inputted the image

Once the user has inputted the image, it will then be converted to an array of numbers and fed through the neural network. An output will be displayed showing which club the program thinks the shirt belongs to:



Here the user can say that the program got the club correct or incorrect by selecting yes or no. If the user says the program got the incorrect club then they should input the actual club name and the neural network will train again with the image the user inputted and the actual club name they just inputted.

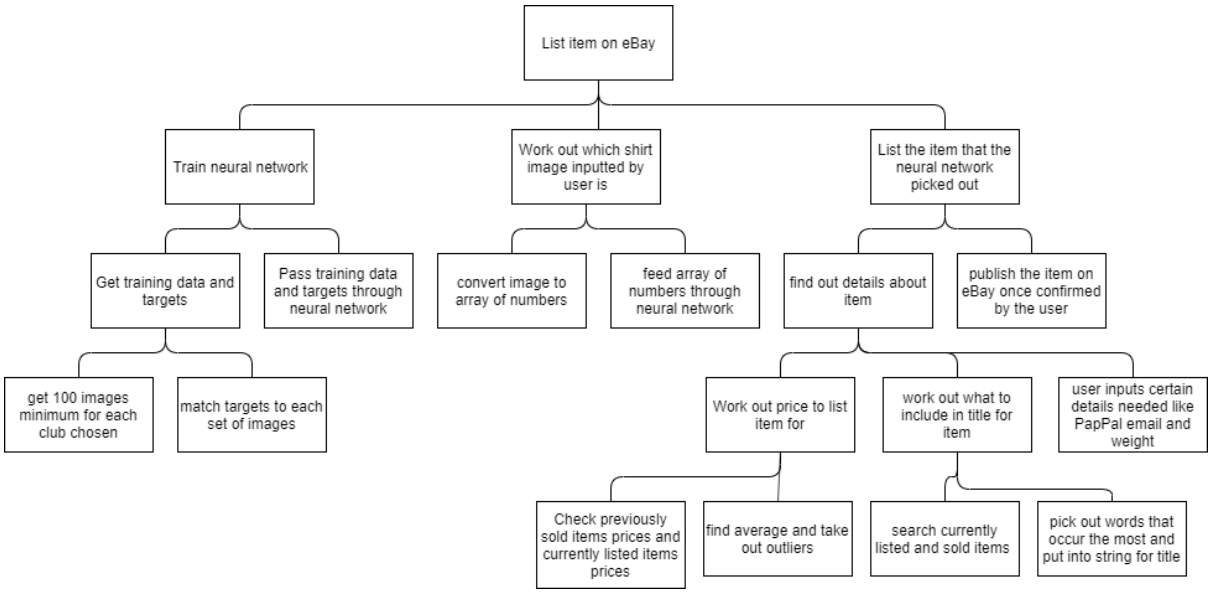
If the user says the program got the correct club, the user will then have to enter details about the shirt, such as the year it was from, the size and the weight, and also their PayPal email which is needed for listing the item on eBay.

Once they have inputted this information which will be validated (email by using regex, year of shirt by using boundaries so only certain years you can input, etc.), the program will then use the information inputted to search through previously sold similar items on eBay using the eBay API. It will create a suggested title, description, postage price and price (made by taking out outlier prices and then finding an average). All of these suggested features will be made possible to be changed by the user however, these are meant to be the ideal features for the eBay item. E.g. the title will feature the most common words used by other eBay sellers for similarly sold items which should help the item sell.

Then the user should be allowed to make any final changes before listing the item on to eBay. Once they have listed the item on eBay they will then be able to choose to exit the program or list another item.

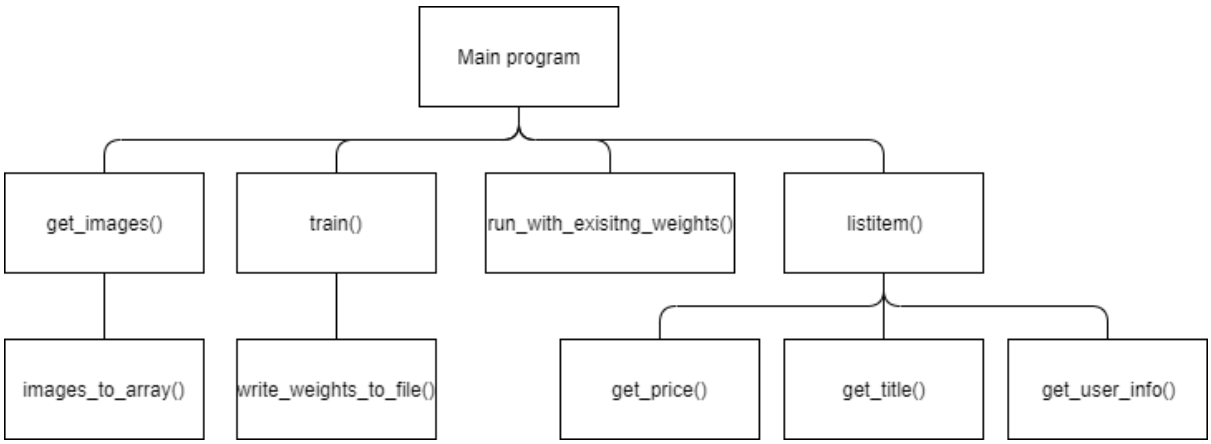
Top down diagram

This is a top down diagram to show the basic processes involved in the planned program.



Organizational chart

This organizational chart shows the basic processes involved in my program including planned sub procedures



Document Specification Sheet (online listing on eBay)

Volumetrics				
Document description	System	Document	Name	Sheet
eBay online listing	Listing an item online	1	Aaron Moorey	1
Stationery ref.	Size	Number of parts	Method of preparation	
		1	typed	
Filing sequence	Medium	Prepared by		
	Computer	Person listing an item on eBay		
Frequency of preparation	Retention period	Location of file		
Every time an item is needed to be listed on eBay		online		

	Minimum	Maximum	Av/Abs	Growth rate/fluctuations
Volume				

Users/receipts	Purpose	Frequency of use
People listing on eBay	To list an item on eBay	Whenever an item is needed to be listed on eBay

Data Dictionary

Ref	Name	Data Type	Regex	Occurrence	Source of data / description
1	title	string		Once per listing	User input
2	Category	string		Once per listing	User input
3	Condition	string		Once per listing	User input
4	Photos	images		Max 12 times per listing,	User input

				otherwise have to pay extra	
5	Description	string		Once per listing	User input
6	Format	string		Once per listing	User input
7	price	number		Once per listing	User input

Proposed General Solutions

Here I have two possible solutions which I could use to make my program.

Solution 1:

In this system it would require the program to iterate through current and sold eBay items. From these items it would pull out the image of the item. It would have to work out which items are the same and which aren't from the titles of the listings. Then the neural network would train by selecting two random images and outputting a 1 if the images were the same and a 0 if the images were different. The weights would then be stored in a notepad file once the training is finished.

The user would then input their image and using the eBay API, the program would search through the items in the football shirt category on eBay and use the neural network to compare the image inputted by the user and the image of the item currently being looked at in the football shirt category. If the computer thinks they are different, the program will move onto the next listing and compare the image of that listing.

If a number near 1 is outputted, then the program will take the title from the item. This will then be used to search for other items with similar titles and the program will then be able to work out what price the item should be listed at and what title the item should have.

The user would input their PayPal email, the weight of the item and then add anything to the description if they would like. The item will then be showed to the user and they will be able to make any changes that they would like to make, and they can then confirm listing the item.

Solution 2:

In this system the training data would be collected from current eBay listings' photos. At least 100 images of each football shirt (for at least five teams in the English Premier League) will be collected using a separate program and then stored in folders with the team assigned to the correct folder. The neural network will then train using the images collected from the eBay search, selecting random images from the folders. The weights will then be stored in a text file.

The user will then have to input an image. This image will be resized to 50x50 pixels, then converted to an array of numbers, 3 RGB numbers representing each pixel in the image. This array will then be fed forward through the neural network using the weights stored in the text file. An output will be given of which football club it thinks the shirt belongs too. The user will then have to confirm whether this is correct.

Once the football club and shirt have been confirmed to be correct by the user, they would have to input information like shirt size and year. The program can then use this information to search on eBay for similar items and decide on a title and price for the shirt. The user will

then have to input their PayPal email, the weight of the item and can choose to edit the description. After this the user will have the option to make any changes to the item and then they can confirm that they are happy for the item to be listed.

Solution I have chosen:

Solution 1 requires two images to be passed through the neural network. This would take twice as long to train and feed forward as solution two would making it less efficient. Solution 2 would require me to check every image in the training data to make sure it is the right shirt which would take a lot of time but is not too much of a problem. Solution 1 would also require a lot more training data as it is checking for if images are the same or different whereas solution 2 is just matching an image with an output. Also, solution 1 may cause some problems with the training data when having to try and find similar images. Overall, I have decided that I will use solution 2 as it is more efficient and matches all the criteria needed for the user.

Requirements

1. To be easy to use
   1.1 The program must be able to be used by anyone that wants to list a football shirt on eBay
   1.2 Must have a good user-friendly interface
   1.3 There must be validation to avoid any errors while the program runs
2. Training data must be gathered
   2.1 get around 100 images for each club, some clubs may be harder to find lots of images
   2.2 images must be checked manually to make sure they are the correct shirt for the club
      2.2.1 inappropriate images should be deleted from the folders
      2.2.2 images should then be renamed in numerical order
3. The neural network should then train
   3.1 randomly selected images from the training data will be selected along with an output value, which will then be used to train the neural network
   3.2 the weights from the outcome of the training will be stored in a txt file which will be used later in the program
4. User must be able to input an image
   4.1 the image must be resized to a 50x50 pixel image
   4.2 the image must then be converted to an array of numbers, 3 numbers representing each pixel in the image
   4.3 this image must then be able to be passed through the neural network. An output should be displayed of what football club the computer thinks the shirt belongs to using the weights stored in the text file
   4.4 The user must then be able to confirm whether the computer got the correct club for the picture of the shirt they inputted
5. If the computer got the wrong club then the following should happen
   5.1 The user should input the actual club for the shirt
   5.2 The neural network should train again using the image inputted by the user, and therefore updating its weights
   5.3 The program should then continue as it would from point 6
6. List item
   6.1 The user must input some information:
      6.1.1 the year the shirt was from
      6.1.2 their PayPal email
      6.1.3 the size of the shirt
      6.1.4 the weight of the shirt
   6.2 The program should then use the club, size and the year to search for the item using the eBay API
   6.3 Get title for item
      6.3.1 The program should look at currently listed items and sold items to decide on a title for the item

6.3.2 Should pull out key words that occur regularly and format correctly

6.4 Get price for item

6.4.1 The program should look at recently sold items to work out a price for the item

6.4.2 Any outlier prices should not be used

6.4.3 The user should also be able to see recently sold items of their particular shirt as requested in the interview in my research

6.5 The user will then be shown the listing and given the opportunity to make any changes and confirm they are happy for the item to be listed

7. User then has option to exit program or list another shirt

Design

Ipso charts

Get training data

Get images of shirts for different clubs which will be used for training data in the neural network

Input • List of clubs	Processes • Search through eBay and download images of club shirts
Storage • Save images in folder of name of club	Outputs • Images saved in folder

Train neural network

Training data is passed into the neural network and adjusts weights in the network to get closer to match the actual output with the expected output

Input	Processes
• Randomly selected shirts from training data folders	• Image turned into array of numbers representing pixels of image • Array passed through training part of neural network
Storage	**Outputs**
• Weights saved to text file when training finished	• Conformation that training is complete

Work out which shirt

User inputs a shirt and is sent through the neural network to work out which club the shirt inputted is

Input	Processes
• User image of shirt	• Convert image to array of numbers representing the pixels of the image • Feed the array of numbers through the neural network
Storage	**Outputs**
• Save which shirt the neural network thinks it is in the program	• Display which club shirt the neural network thinks the user input

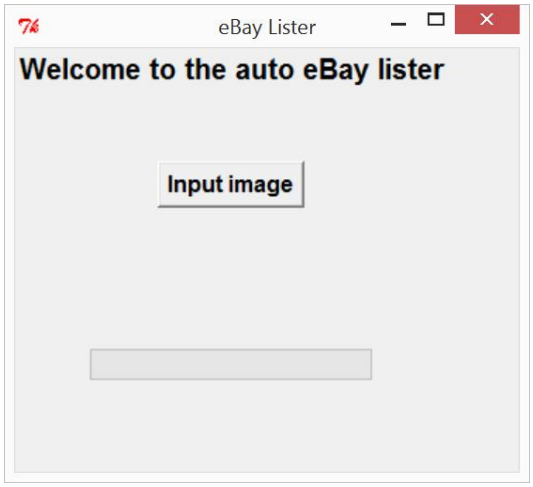
List item

Search on eBay to work out price and title for the listing of the shirt and then list it on eBay

Input	Processes
• Year of shirt • Size of shirt • PayPal email • Weight of shirt	• Work out price to list shirt at • Work out title to use for listing • List the shirt using eBay API
Storage	**Outputs**
• Use shirt discovered earlier to work out price and title • ebay.yaml config file for eBay API token	• List the item on eBay

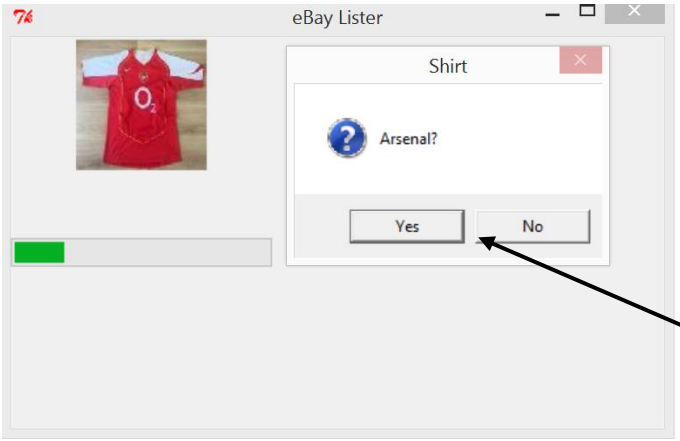
Interface

These are some simple designs for how the user interface will look, they will have the simple inputs and outputs the program will require. This is by using the module Tkinter.

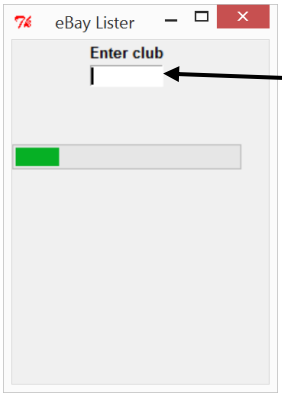


Button for user to press to be able to select an image to be sent through the neural network. There is a progress bar at the bottom to show how far the user is through the listing process.

When button is pressed, opens up a place for the user to browse and select the image they want to send to the neural network, **only will be allowed to input a JPEG image to stop errors occurring for validation**.



It will then display the image the user inputted and ask the user whether the neural network has got the club of the shirt correct, which the user will then have to select 'yes' or 'no'. Giving the user the only options of yes and no means that whichever they select, an error will not occur, so is more robust than asking them to type it manually.



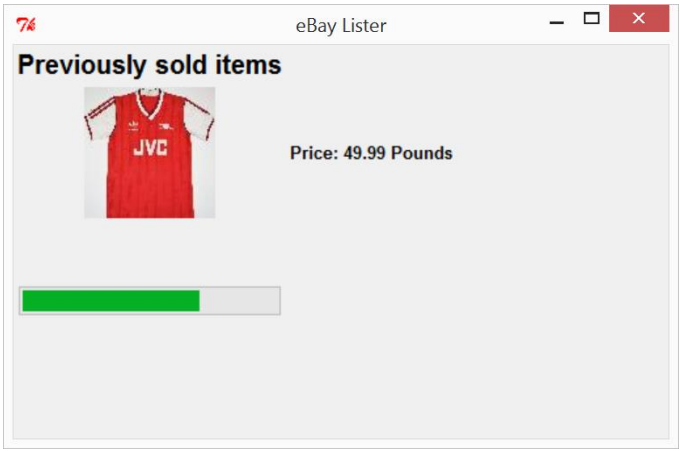
Will be validated to ensure that the user only can enter one of the allowed clubs

If the user answers 'no', then the user will have to enter the actual club of the shirt. Then the image inputted by the user will be trained by the neural network with the actual club name just inputted by the user. The program will then continue as it would for if the user had selected 'yes'.

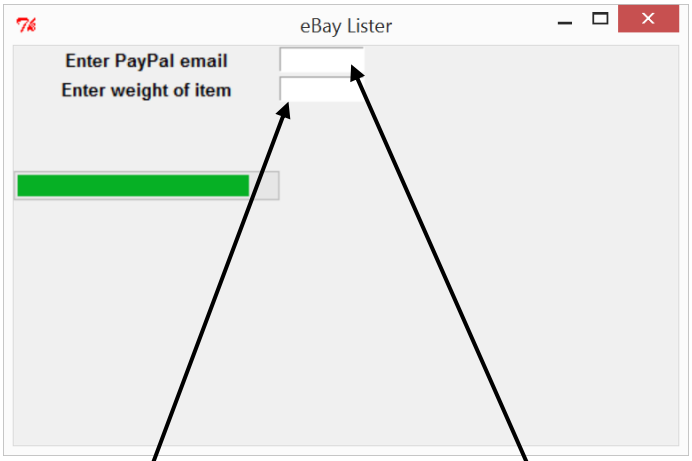
User then needs to input the size of the shirt and the year it was from. This will then be used to search on eBay for similarly sold items, for the program to be able to obtain a title and price for the shirt for the users listing.

Year of the shirt will be validated by user only being able to input years between and including 1900 and 2020.

Size of shirt will be a drop-down box where there are only certain sizes the user can select (e.g. XS, S, M, L, XL, etc.)



Then the interface should display previously sold items with their prices so that the user can see what other shirts similar have sold at. This was suggested in my interview with two eBay sellers in my research and will help the user know whether they should stick with the program's recommendation for price or change it.



The user should then enter their PayPal email and the weight of their item. The PayPal email is needed for listing the item on eBay, so that when the item is bought, the money paid by the buyer has somewhere to go.

The weight is needed so that the postage cost for the item can be calculated. Different weights have different postage costs.

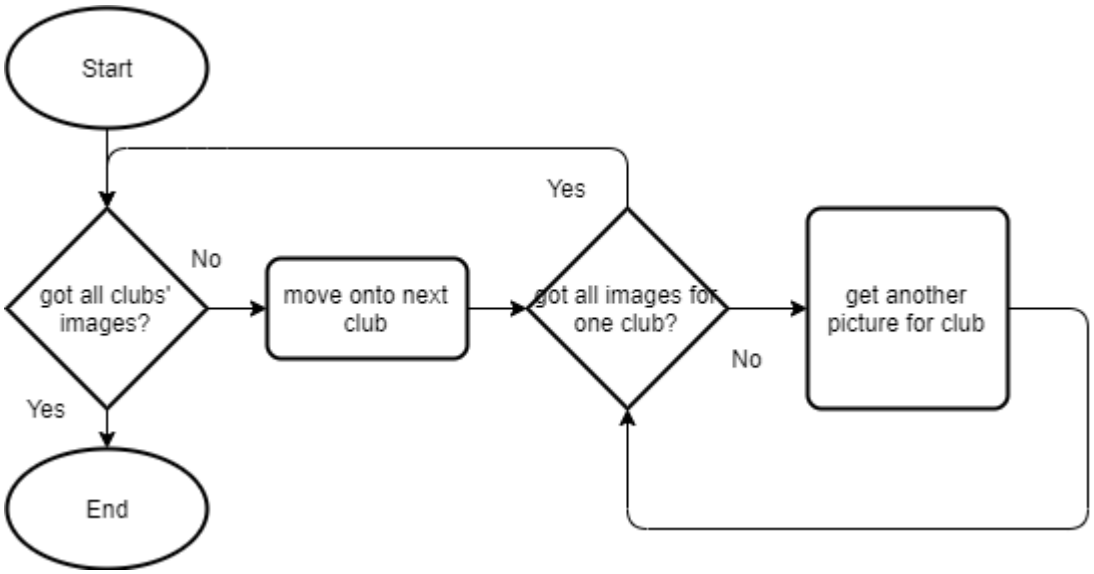
The email will be validated using regex, to make sure the user inputs a valid email.

The weight will be a drop-down box, giving the user only certain options for the weight. This means I can avoid having to validate raw user text input and is also quicker for the user.

The user will then be shown what their new listing will look like, here they will have the opportunity to make changes to the title, price, description and postage price. Once they are happy with the listing, they will then be able to confirm that they are happy for the shirt to be listed on eBay. Once it has been listed on eBay, the user will then have the option to list another shirt or exit the program.

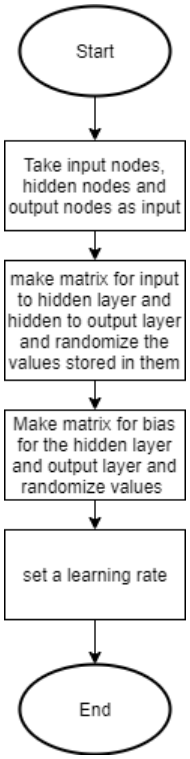
Flowcharts

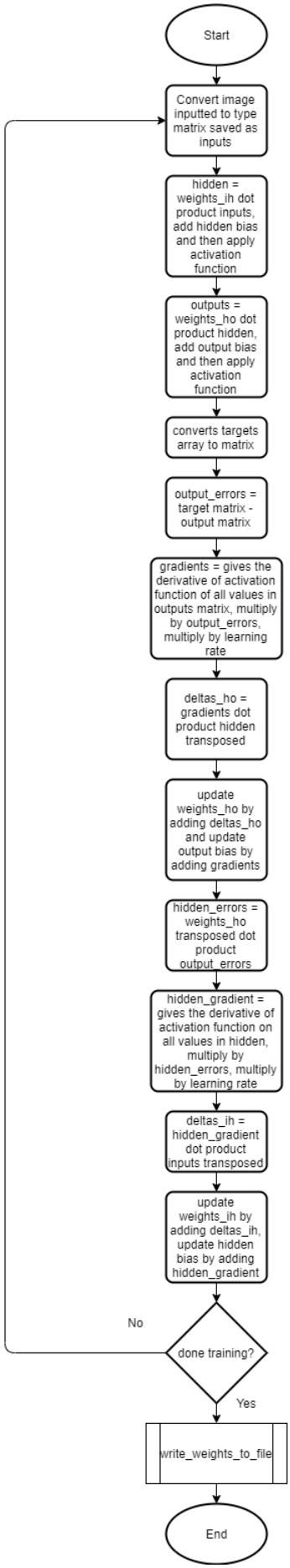
This flowchart shows how the training data will be gathered



Here are some flow charts showing how I plan to develop the neural network, they only include one hidden layer however the final solution will include two hidden layers.

This flowchart shows the basic setup of the neural network when the class is initialised



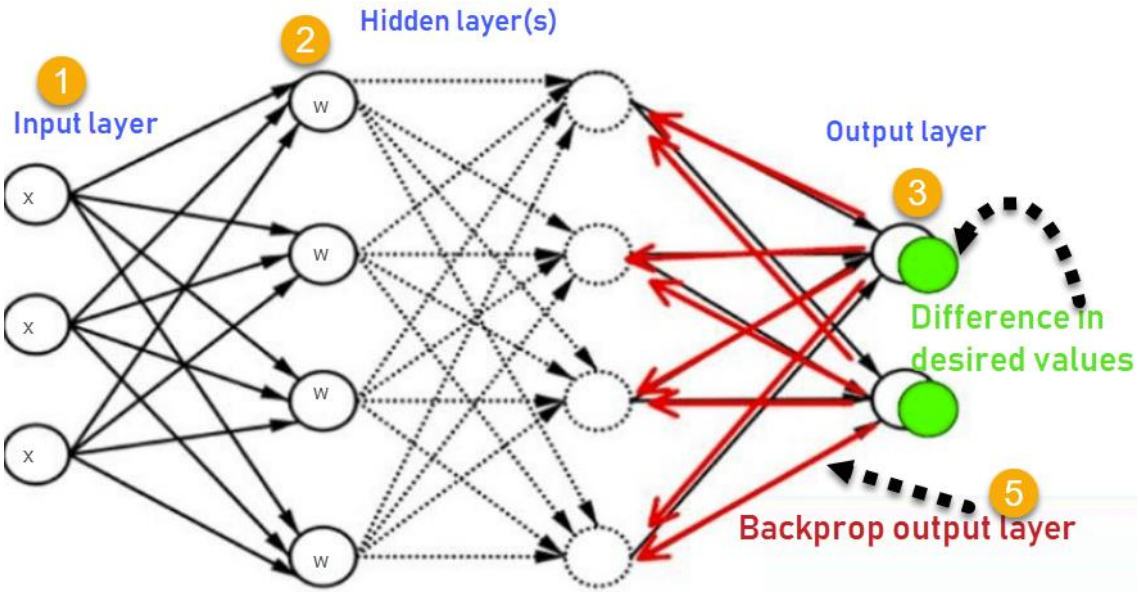


This flowchart shows a simplified version of how the training section works for the neural network

Weights_ih stands for the weights in between the input and hidden layer.

Weights_ho stands for the weights in between the hidden and output layer

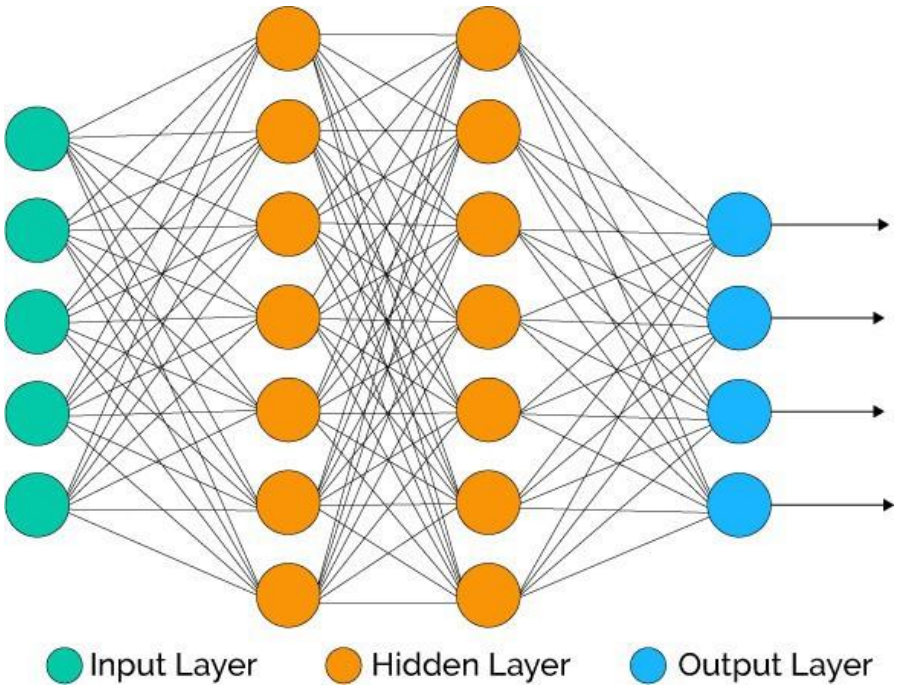
These diagrams show a similar training process used by my neural network in the flowchart above:



guru99.com 1

(guru99, n.d.)

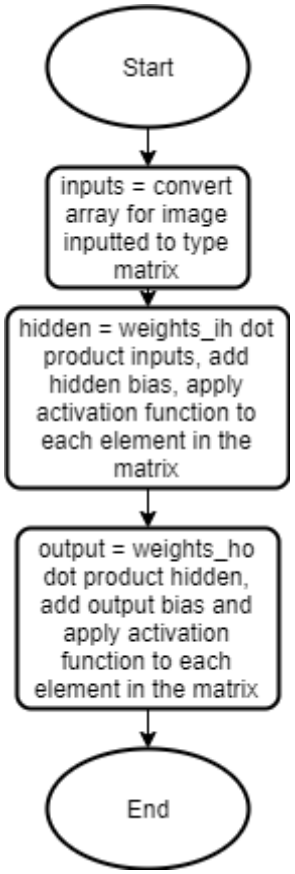
Once the actual output has been produced, the difference between the actual output and expected output is calculated. This difference is then used to for the back-propagation process where the weights are updated.



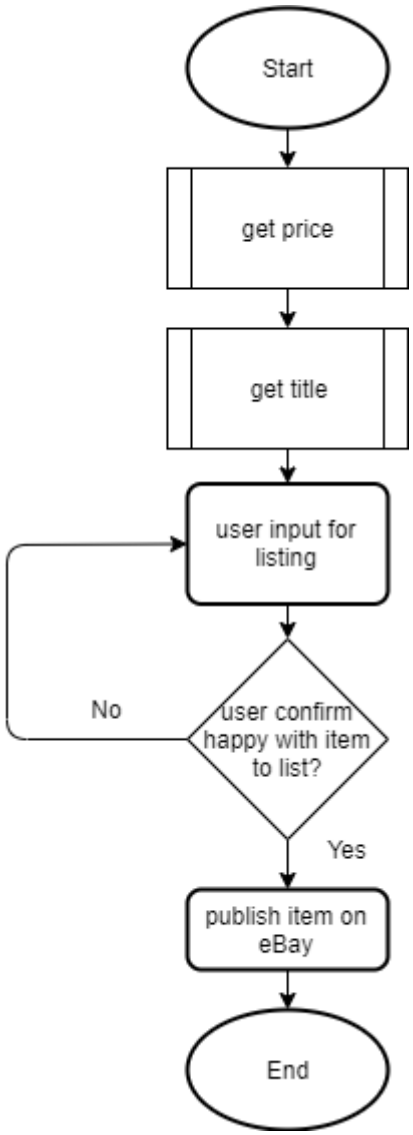
towardsdatascience.com, 2017 2

(Towardsdatascience, Towardsdatascience, 2017)

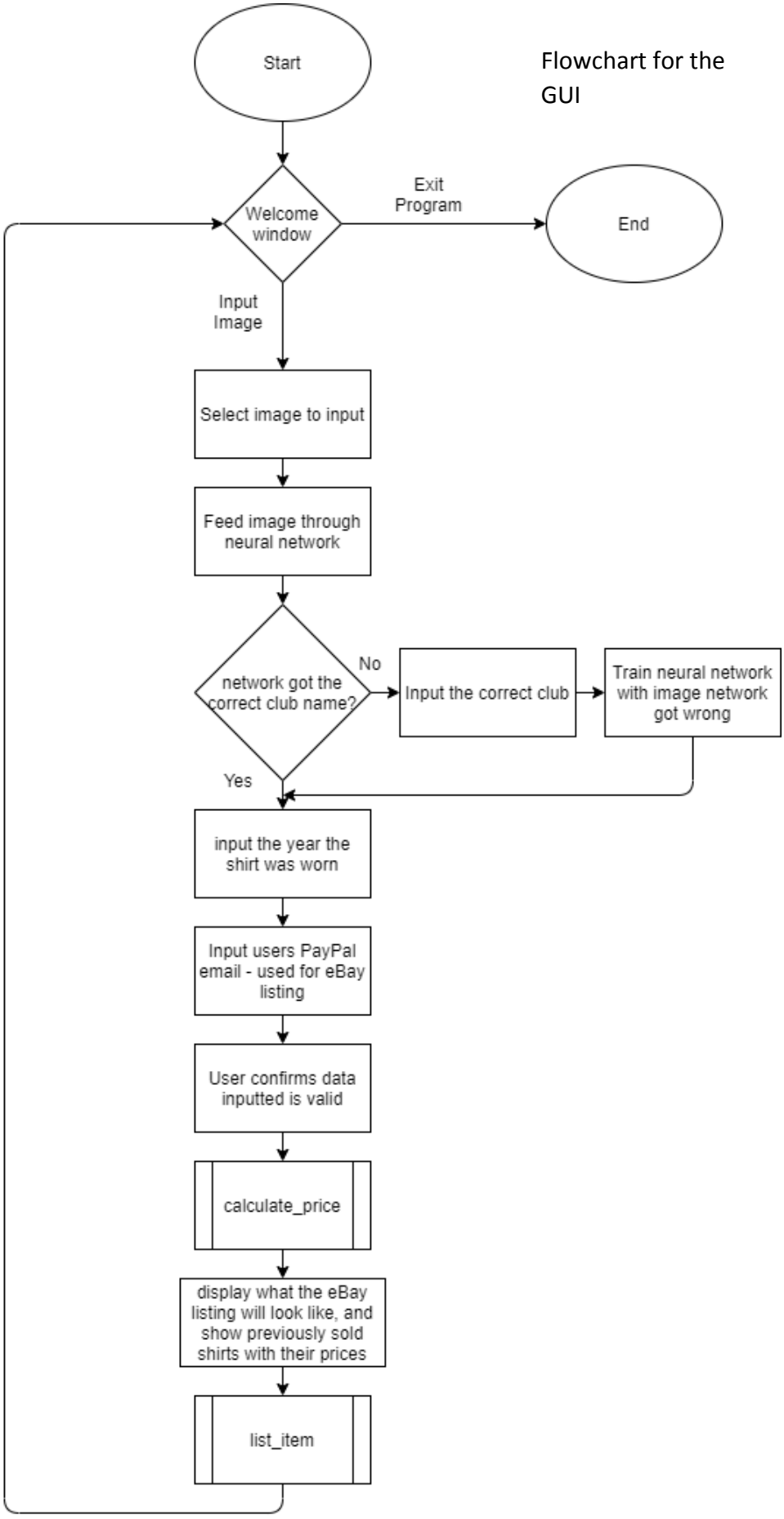
Flowchart showing the feed forward process of the neural network:



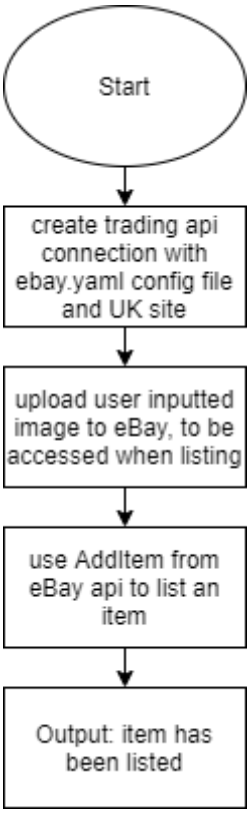
Flow chart showing the basic process of listing the item on eBay



Flowchart for the GUI

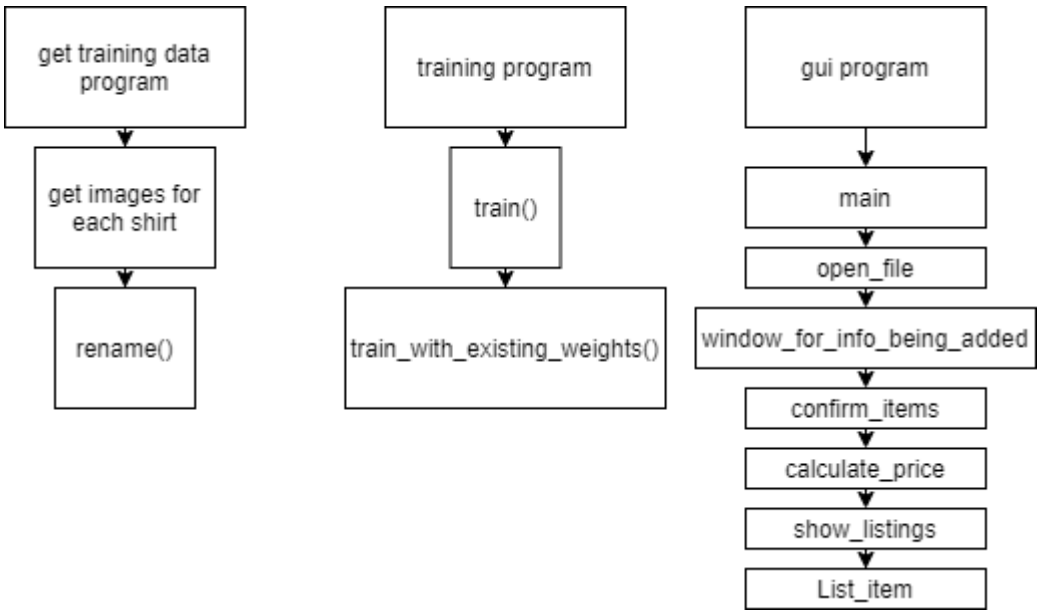


List_item sub procedure



Organizational chart changes

New organizational chart, updated from analysis:



Pseudocode

Matrix class

These are the sub routines in the matrix class:

MakeMatrix

- Creates a matrix with width self.cols and height self.rows, each element in the matrix will be equal to 0

```
FUNCTION MakeMatrix(self):
    FOR I in range(0,self.rows):
      self.matrix.append([])
      FOR j in range(0,self.cols):
        self.matrix[i].append(j)
        self.matrix[i][j] ← 0
      END FOR
    END FOR
    RETURN self.matrix
END FUNCTION
```

Randomize Matrix

- Puts random values in each element of a matrix, each value will be between -1 and 1

```
FUNCTION RandomizeMatrix(self):
    FOR I in range(0,self.rows):
      FOR j in range(0,self.cols):
```

```
            self.matrix[i][j] ← random.uniFORm(-1,1)
        END FOR
    END FOR
    RETURN self.matrix
END FUNCTION
```

Multiply

- Multiplies each element in matrix by n or multiplies two matrices together

```
FUNCTION multiply(self, n):
    IF isinstance(n, Matrix):
        FOR I in range(0,self.rows):
            FOR j in range(0,self.cols):
                self.matrix[i][j] *← n.matrix[i][j]
            END FOR
        END FOR
        RETURN self.matrix
    ELSE:
        FOR I in range(0,self.rows):
            FOR j in range(0,self.cols):
                self.matrix[i][j] *← n
            END FOR
        END FOR
        RETURN self.matrix
    END IF
END FUNCTION
```

Add

- Adds n to each element in matrix or adds two matrices together

```
FUNCTION add(self, n):
    IF isinstance(n, Matrix):
      FOR I in range(0,self.rows):
        FOR j in range(0,self.cols):
          self.matrix[i][j] +← n.matrix[i][j]
        END FOR
      END FOR
      RETURN self.matrix
    ELSE:
      FOR I in range(0,self.rows):
        FOR j in range(0,self.cols):
          self.matrix[i][j] +← n
        END FOR
      END FOR
      RETURN self.matrix
    END IF
END FUNCTION
```

Apply_function

- Applies a function, fun, to each element in the matrix, for example it may apply a function called double which doubles each element in the matrix

```
FUNCTION apply_function(self, fun):
    FOR I in range(0, self.rows):
      FOR j in range(0, self.cols):
        val ← self.matrix[i][j]
        self.matrix[i][j] ← fun(val)
      END FOR
    END FOR
```

```
    RETURN self.matrix
END FUNCTION
```

Subtract

- Subtracts one matrix from another

```
FUNCTION subtract(self, n):
    result ← Matrix(self.rows, self.cols)
    result.MakeMatrix()
    FOR I in range(0,result.rows):
      FOR j in range(0,result.cols):
        result.matrix[i][j] ← self.matrix[i][j] – n.matrix[i][j]
      END FOR
    END FOR
    RETURN result
END FUNCTION
```

Transpose

- Shown in my analysis section what the transpose procedure is, page 17

```
FUNCTION transpose(self):
    result ← Matrix(self.cols, self.rows)
    result.MakeMatrix()
    FOR I in range(0, self.rows):
      FOR j in range(0, self.cols):
        result.matrix[j][i] ← self.matrix[i][j]
      END FOR
    END FOR
    RETURN result
END FUNCTION
```

Dot product

- Applies dot product between 2 matrices, shown in my analysis, page 17

```
FUNCTION dotproduct(self, n):
    IF isinstance(n, Matrix):
        IF self.cols !← n.rows:
            OUTPUT “Not equal cols and rows”
        ELSE:
            result ← Matrix(self.rows, n.cols)
            result.MakeMatrix()
            FOR I in range(0, result.rows):
                FOR j in range(0,result.cols):
                    total ← 0
                    FOR k in range(0, self.cols):
                        total +← self.matrix[i][k] * n.matrix[k][j]
                    END FOR
                    result.matrix[i][j] ← total
                END FOR
            END FOR
            RETURN result
        END IF
    ELSE:
        OUTPUT “Not matrix”
    END IF
END FUNCTION
```

apply_function_new_matrix

- Applies a function to each element in matrix and returns a new matrix, same as apply_function but a new matrix is made with the new values in it

```
FUNCTION apply_function_new_matrix(self, fun):
    result ← Matrix(self.rows, self.cols)
    result.MakeMatrix()
    FOR I in range(0, result.rows):
      FOR j in range(0, result.cols):
        val ← self.matrix[i][j]
        result.matrix[i][j] ← fun(val)
      END FOR
    END FOR
    RETURN result
END FUNCTION
```

Images class

This class manages all the images inputted by the user or used for the training data. It uses the module PIL which is imported at the start of the program. The two procedures are resize and toarray.

Resize

- This resizes an image to a new 50 x 50 pixel image

```
FUNCTION resize(self, filename2):
    img← Image.open(self.filename)
    new_img ← img.resize((50,50))
    TRY:
      new_img.save(settings_array[4] + filename2 + ".jpg")
    EXCEPT:
      OUTPUT "File path does not exist in the settings file, please update and then restart"
      sys.exit()
END FUNCTION
```

Toarray

- this converts the image to an array of integers, 3 integers for each pixel because of the RGB colours.

```
FUNCTION toarray(self):
    img ← Image.open(self.filename, 'r')
    w, h ← img.size
    pix ← list(img.getdata())
    x ← [pix[n:n+w] FOR n in range(0, w*h, w)]
    arr ← []
    FOR I in range(0, len(x)):
      FOR j in range(0, len(x[i])):
        FOR k in range(0, len(x[i][j])):
          arr.append(round(3*((x[i][j][k]) / float(1000)),3))
        END FOR
```

```
        END FOR
    END FOR
    RETURN arr
END FUNCTION
```

Training data program

Separate program which gets the training data

- This program searches through eBay using the eBay API and downloads images of football shirts and saves them to folders named by club. This uses imports of urllib2, json, requests, Images, PIL and os.

```
FROM urllib2 IMPORT urlopen

IMPORT json

IMPORT requests

FROM images IMPORT *

FROM PIL IMPORT Image

IMPORT os


search_term ← “club_name%20shirt%20home”

url ← (‘https://svcs.ebay.com/services/search/FindingService/v1\

?OPERATION-NAME←findItemsByKeywords&paginationInput.pageNumber←1&SERVICE-VERSION←1.0.0\

&SECURITY-APPNAME←ebay_key&\

RESPONSE-DATA-FORMAT←JSON&REST-PAYLOAD&keywords←’ + search_term)

apiresult ← requests.get(url)

api_return ← apiresult.json()

index ← 0

FOR item in (api_return[“findItemsByKeywordsResponse”][0][“searchResult”][0][“item”]):

   pic ← item[“galleryURL”][0]

   Images(urlopen(pic), “pic” + str(index)).resize()

   index+←1

END FOR
```

Rename

- This procedure renames all the files in the folders after I have gone through manually and deleted any photos which should not be in the files. It renames them in numerical order which makes the images much easier for me to use for the neural network training.

```
FUNCTION rename():
  I ← 0
  FOR filename in os.listdir("location_of_where_images_are_stored"):
    IF filename ← "Thumbs.db":
      OUTPUT "not an image"
    ELSE:
      new_name ←"pic" + str(i) + ".jpg"
      current_name ← location_of_where_images_are_stored + filename
      new_name ← location_of_where_images_are_stored + dst
      os.rename(current_name, new_name)
      I +← 1
    END IF
  END FOR
END FUNCTION
```

Neural network class

- The program with this class in imports the matrix and images modules above as well as math and sys.

These are the sections for the neural network class:

Initialization procedure

- this takes in the amounts of nodes in the neural network for each layer and makes a matrix for each layer for the weights and biases and randomizes the contents to give random weights for the neural network

```
FUNCTION __init__(self, inputnodes, hiddennodes1, hiddennodes2, outputnodes):

    self.inodes ← inputnodes
    self.hnodes1 ← hiddennodes1
    self.hnodes2 ← hiddennodes2
    self.onodes ← outputnodes

    self.weights_ih ← Matrix(self.hnodes1, self.inodes)
    self.weights_h1h2 ← Matrix(self.hnodes2, self.hnodes1)
    self.weights_ho ← Matrix(self.onodes, self.hnodes2)

    self.weights_ih.MakeMatrix()
    self.weights_h1h2.MakeMatrix()
    self.weights_ho.MakeMatrix()
```

```
    self.weights_ih.RandomizeMatrix()
    self.weights_h1h2.RandomizeMatrix()
    self.weights_ho.RandomizeMatrix()

    self.bias_h ← Matrix(self.hnodes1, 1)
    self.bias_h2 ← Matrix(self.hnodes2, 1)
    self.bias_o ← Matrix(self.onodes, 1)

    self.bias_h.MakeMatrix()
    self.bias_h2.MakeMatrix()
    self.bias_o.MakeMatrix()

    self.bias_h.RandomizeMatrix()
    self.bias_h2.RandomizeMatrix()
    self.bias_o.RandomizeMatrix()

    self.learningrate ← 0.01
END FUNCTION
```

Feedforward

- Takes image array, feeds it through the neural network which applies matrix calculations and activations functions to the array and returns an output matrix

```
FUNCTION feedforward(self, input_array):

    inputs ← Matrix(len(input_array), 1)
    inputs.MakeMatrix()
    FOR I in range(0, len(input_array)):
       inputs.matrix[i][0] ← input_array[i]
    END FOR
    hidden ← self.weights_ih.dotproduct(inputs)
```

```
        hidden.add(self.bias_h)
        hidden.func(activation_function)

        hidden2 ← self.weights_h1h2.dotproduct(hidden)
        hidden2.add(self.bias_h2)
        hidden2.func(activation_function)

        output ← self.weights_ho.dotproduct(hidden2)
        output.add(self.bias_o)
        output.func(activation_function)

        output = softmax(output)
        RETURN output
END FUNCTION
```

Softmax

- Applies softmax function to a matrix, explained in analysis 19

```
FUNCTION softmax(outputs):
 denominator ← 0
 FOR I in range(0,outputs.rows):
  FOR j in range(0,outputs.cols):
   denominator +← math.exp(outputs.matrix[i][j])
  END FOR
 END FOR
 FOR I in range(0,outputs.rows):
  FOR j in range(0,outputs.cols):
   OUTPUT (math.exp(outputs.matrix[i][j]))/denominator
  END FOR
 END FOR
END FUNCTION
```

Activation functions

The activation function are explained in analysis, on page 18

Sigmoid:

- Applies sigmoid function

```
FUNCTION sigmoid(x):
 IF x < 0:
  RETURN 1- 1 / (1 + math.exp(x))
 RETURN 1 / (1 + math.exp(-x))
 END IF
END FUNCTION
```

Derivative of sigmoid (used for backpropogation):

- Applies derivative of sigmoid

```
FUNCTION dsigmoid(x):
 RETURN x * (1 – x)
END FUNCTION
```

Relu:

- Applies relu function

```
FUNCTION relu(x):
  IF x < 0:
    RETURN x * 0.01
  ELSE:
    RETURN x
  END IF
END FUNCTION
```

Derivative of relu (used for backpropogation):

- Applies derivative of relu

```
FUNCTION drelu(x):
 IF x < 0:
  RETURN 0.01
 ELSE:
  RETURN 1
 END IF
END FUNCTION
```

- Tanh activation function can be used with the import module math

Train

- This function is used to train the neural network, it takes an image, inputs_array, a target_array and nn, which is the neural network which has been instantiated earlier. The image is converted to an array and then a matrix, which is fed through the network. The error between the actual output and the expected output is calculated, then the backpropogation occurs where the weights are adjusted.

```
 FUNCTION train(self, inputs_array, targets_array, nn):

    inputs ← Matrix(len(inputs_array), 1)
    inputs.MakeMatrix()
    FOR I in range(0, len(inputs_array)):
      inputs.matrix[i][0] ← inputs_array[i]
    END FOR
    #generating hidden output
    hidden ← self.weights_ih.dotproduct(inputs)
    hidden.add(self.bias_h)
    #activation function
    hidden.func(activation_function)
```

```
hidden2 ← self.weights_h1h2.dotproduct(hidden)
hidden2.add(self.bias_h2)
hidden2.func(activation_function)

#generate output
outputs ← self.weights_ho.dotproduct(hidden2)
outputs.add(self.bias_o)
#activation function
outputs.func(activation_function)

#put targets array into matrix
targets ← Matrix(len(targets_array), 1)
targets.MakeMatrix()
FOR I in range(0, len(targets_array)):
   targets.matrix[i][0] ← targets_array[i]
END FOR

#calculate output errors
output_errors ← targets.subtract(outputs)

#calculate gradients
gradients ← outputs.stfunc(derivative_of_activation_function)
gradients.multiply(output_errors)
gradients.multiply(self.learningrate)

#calculate deltas
hidden2_transposed ← hidden2.transpose()
weight_ho_deltas ← gradients.dotproduct(hidden2_transposed)
```

```
#adjust ho weights by deltas
self.weights_ho.add(weight_ho_deltas)
#adjust bias by deltas
self.bias_o.add(gradients)

#calculate hidden layer errors
weights_ho_transposed ← self.weights_ho.transpose()
hidden2_errors ← weights_ho_transposed.dotproduct(output_errors)

hidden2_gradient ← hidden2.stfunc(derivative_of_activation_function)
hidden2_gradient.multiply(hidden2_errors)
hidden2_gradient.multiply(self.learningrate)

hidden1_transposed ← hidden.transpose()
weight_h1h2_deltas ← hidden2_gradient.dotproduct(hidden1_transposed)

self.weights_h1h2.add(weight_h1h2_deltas)
self.bias_h2.add(hidden2_gradient)

weights_h1h2_transposed ← self.weights_h1h2.transpose()
hidden_errors ← weights_h1h2_transposed.dotproduct(hidden2_errors)

#calculate hidden gradients
hidden_gradient ← hidden.stfunc(derivative_of_activation_function)
hidden_gradient.multiply(hidden_errors)
hidden_gradient.multiply(self.learningrate)

#calculate input to hidden deltas
inputs_transposed ← inputs.transpose()
```

```
    weight_ih_deltas ← hidden_gradient.dotproduct(inputs_transposed)


    #adjust ih weights
    self.weights_ih.add(weight_ih_deltas)
    #adjust hidden bias by deltas
    self.bias_h.add(hidden_gradient)


    nn.write_weights_to_file(self.weights_ih.matrix, self.bias_h.matrix,
self.weights_h1h2.matrix, self.bias_h2.matrix, self.weights_ho.matrix, self.bias_o.matrix)
END FUNCTION
```

Write_weights_to_file

- This writes all the weights of the neural network to a text file, it writes a comma between each weight, and a new line between each layer

```
FUNCTION write_weights_to_file(self, wih, bh1, wh1h2, bh2, wh2h3, bh3, wh3o, bo):
    f ← open(settings_array[5], "w+")
    FOR I in wih:
      FOR j in i:
        f.write(str(j)+",")
      END FOR
    END FOR
    f.write("\n")
    FOR I in bh1:
      FOR j in i:
        f.write(str(j)+",")
      END FOR
    END FOR
    f.write("\n")
    FOR I in wh1h2:
```

```
        FOR j in i:
            f.write(str(j)+",")
        END FOR
    END FOR
    f.write("\n")
    FOR I in bh2:
        FOR j in i:
            f.write(str(j)+",")
        END FOR
    END FOR
    f.write("\n")
    FOR I in wh2o:
        FOR j in i:
            f.write(str(j)+",")
        END FOR
    END FOR
    f.write("\n")
    FOR I in bo:
        FOR j in i:
            f.write(str(j)+",")
        END FOR
    END FOR
    f.write("\n")
    f.close()
END FUNCTION
```

Run_with_existing_weights

- This procedure uses the weights in the text file and then feedforwards an image passed in which produces an output. The weights from the files are converted to type matrix. Then using these matrices, the image inputted is converted to a matrix and fed through the network, where an output is then produced.

```
FUNCTION run_with_existing_weights(self, input_array):
    inputs ← Matrix(len(input_array), 1)
    inputs.MakeMatrix()
    FOR I in range(0, len(input_array)):
      inputs.matrix[i][0] ← input_array[i]
    TRY
      f ← open(settings_array[5], "r")
    EXCEPT
      OUTPUT "weights file in settings file does not exist or file path is incorrect, please update and restart"
      sys.exit()
    TRY
      ih1 ← f.readline()
      ih1split ← ih1.split(",")
      ih1_weights ← []
      FOR I in range(0,len(ih1split)-1):
        ih1_weights.append(ih1split[i])
      END FOR
      weights_ih1 ← Matrix(self.hnodes1,self.inodes)
      weights_ih1.MakeMatrix()
      count_ih1 ← 0
      FOR I in range(0,self.hnodes1):
        FOR j in range(0,self.inodes):
          weights_ih1.matrix[i][j] ← float(ih1_weights[count_ih1])
          count_ih1 +← 1
```

```
    END FOR
  END FOR
bh1 ← f.readline()
bh1split ← bh1.split(“,”)
bh1_weights ← []
FOR I in range(0,len(bh1split)-1):
    bh1_weights.append(bh1split[i])
END FOR
weights_bh1 ← Matrix(self.hnodes1,1)
weights_bh1.MakeMatrix()
count_bh1 ← 0
FOR I in range(0,self.hnodes1):
    FOR j in range(0,1):
        weights_bh1.matrix[i][j] ← float(bh1_weights[count_bh1])
        count_bh1 +← 1
    END FOR
END FOR
h1h2 ← f.readline()
h1h2split ← h1h2.split(“,”)
h1h2_weights ← []
FOR I in range(0, len(h1h2split)-1):
    h1h2_weights.append(h1h2split[i])
END FOR
weights_h1h2 ← Matrix(self.hnodes2,self.hnodes1)
weights_h1h2.MakeMatrix()
count_h1h2 ← 0
FOR I in range(0,self.hnodes2):
    FOR j in range(0,self.hnodes1):
        weights_h1h2.matrix[i][j] ← float(h1h2_weights[count_h1h2])
```

```
        count_h1h2 +← 1
    END FOR
END FOR
bh2 ← f.readline()
bh2split ← bh2.split(“,”)
bh2_weights ← []
FOR I in range(0,len(bh2split)-1):
    bh2_weights.append(bh2split[i])
END FOR
weights_bh2 ← Matrix(self.hnodes2,1)
weights_bh2.MakeMatrix()
count_bh2 ← 0
FOR I in range(0,self.hnodes2):
    FOR j in range(0,1):
        weights_bh2.matrix[i][j] ← float(bh2_weights[count_bh2])
        count_bh2 +← 1
    END FOR
END FOR
h3o ← f.readline()
h3osplit ← h3o.split(“,”)
h3o_weights ← []
FOR I in range(0, len(h3osplit)-1):
    h3o_weights.append(h3osplit[i])
END FOR
weights_h3o ← Matrix(self.onodes,self.hnodes2)
weights_h3o.MakeMatrix()
count_h3o ← 0
FOR I in range(0,self.onodes):
    FOR j in range(0,self.hnodes2):
```

```
            weights_h3o.matrix[i][j] ← float(h3o_weights[count_h3o])
            count_h3o +← 1
        END FOR
    END FOR
    bo ← f.readline()
    bosplit ← bo.split(“,”)
    bo_weights ← []
    FOR I in range(0,len(bosplit)-1):
        bo_weights.append(bosplit[i])
    END FOR
    weights_bo ← Matrix(self.onodes,1)
    weights_bo.MakeMatrix()
    count_bo ← 0
    FOR I in range(0,self.onodes):
        FOR j in range(0,1):
            weights_bo.matrix[i][j] ← float(bo_weights[count_bo])
            count_bo +← 1
        END FOR
    END FOR
    f.close()
EXCEPT
    OUTPUT “invalid weights file”
    sys.exit()

hidden1 ← weights_ih1.dotproduct(inputs)
hidden1.add(weights_bh1)
hidden1.func(activation_function)

hidden2 ← weights_h1h2.dotproduct(hidden1)
```

```
    hidden2.add(weights_bh2)
    hidden2.func(activation_function)

    output ← weights_h3o.dotproduct(hidden2)
    output.add(weights_bo)
    output.func(activation_function)

    RETURN output
END FUNCTION
```

Train_with_existing_weights

- This procedure trains the neural network with the weights stored in the text file. The weights in the text file are converted to matrices. The image inputted is then converted to a matrix where it is fed through the neural network using the new matrices from the text file. The output error between the expected output and the actual output is calculated and the backpropagation occurs. The new weights are then stored in the text file.

```
  FUNCTION train_with_existing_weights(self, inputs_array, targets_array, nn):

    inputs ← Matrix(len(inputs_array), 1)
    inputs.MakeMatrix()
    FOR I in range(0, len(inputs_array)):
      inputs.matrix[i][0] ← inputs_array[i]
    END FOR
    TRY
      f ← open("weights.txt", "r")
    EXCEPT
      OUTPUT "weights file in settings file does not exist or file path is incorrect, please update and restart"
      sys.exit()
    TRY
```

```
ih1 ← f.readline()
 ih1split ← ih1.split(“,”)
ih1_weights ← []
FOR I in range(0,len(ih1split)-1):
        ih1_weights.append(ih1split[i])
 END FOR
weights_ih1 ← Matrix(self.hnodes1,self.inodes)
weights_ih1.MakeMatrix()
count_ih1 ← 0
FOR I in range(0,self.hnodes1):
   FOR j in range(0,self.inodes):
        weights_ih1.matrix[i][j] ← float(ih1_weights[count_ih1])
        count_ih1 +← 1
  END FOR
END FOR

bh1 ← f.readline()
bh1split ← bh1.split(“,”)
 bh1_weights ← []
FOR I in range(0,len(bh1split)-1):
   bh1_weights.append(bh1split[i])
END FOR
weights_bh1 ← Matrix(self.hnodes1,1)
weights_bh1.MakeMatrix()
count_bh1 ← 0
FOR I in range(0,self.hnodes1):
    FOR j in range(0,1):
      weights_bh1.matrix[i][j] ← float(bh1_weights[count_bh1])
      count_bh1 +← 1
```

```
    END FOR
 END FOR
h1h2 ← f.readline()
h1h2split ← h1h2.split(“,”)
h1h2_weights ← []
FOR I in range(0, len(h1h2split)-1):
    h1h2_weights.append(h1h2split[i])
 END FOR
weights_h1h2 ← Matrix(self.hnodes2,self.hnodes1)
weights_h1h2.MakeMatrix()
count_h1h2 ← 0
FOR I in range(0,self.hnodes2):
   FOR j in range(0,self.hnodes1):
       weights_h1h2.matrix[i][j] ← float(h1h2_weights[count_h1h2])
       count_h1h2 +← 1
    END FOR
 END FOR
bh2 ← f.readline()
bh2split ← bh2.split(“,”)
bh2_weights ← []
FOR I in range(0,len(bh2split)-1):
            bh2_weights.append(bh2split[i])
 END FOR
weights_bh2 ← Matrix(self.hnodes2,1)
weights_bh2.MakeMatrix()
count_bh2 ← 0
FOR I in range(0,self.hnodes2):
            FOR j in range(0,1):
       weights_bh2.matrix[i][j] ← float(bh2_weights[count_bh2])
```

```
        count_bh2 +← 1
    END FOR
END FOR

h3o ← f.readline()
h3osplit ← h3o.split(“,”)
h3o_weights ← []
FOR I in range(0, len(h3osplit)-1):
    h3o_weights.append(h3osplit[i])
END FOR
weights_h3o ← Matrix(self.onodes,self.hnodes2)
weights_h3o.MakeMatrix()
count_h3o ← 0
FOR I in range(0,self.onodes):
    FOR j in range(0,self.hnodes2):
        weights_h3o.matrix[i][j] ← float(h3o_weights[count_h3o])
        count_h3o +← 1
    END FOR
END FOR

bo ← f.readline()
bosplit ← bo.split(“,”)
bo_weights ← []
FOR I in range(0,len(bosplit)-1):
    bo_weights.append(bosplit[i])
END FOR
weights_bo ← Matrix(self.onodes,1)
weights_bo.MakeMatrix()
count_bo ← 0
```

```
    FOR I in range(0,self.onodes):
        FOR j in range(0,1):
            weights_bo.matrix[i][j] ← float(bo_weights[count_bo])
            count_bo +← 1
        END FOR
    END FOR
    f.close()
EXCEPT
    OUTPUT “invalid weights file”
    sys.exit()

inputs ← Matrix(len(inputs_array), 1)
inputs.MakeMatrix()
FOR I in range(0, len(inputs_array)):
    inputs.matrix[i][0] ← inputs_array[i]
END FOR

hidden1 ← weights_ih1.dotproduct(inputs)
hidden1.add(weights_bh1)
hidden1.func(activation_function)

hidden2 ← weights_h1h2.dotproduct(hidden1)
hidden2.add(weights_bh2)
hidden2.func(activation_function)

outputs ← weights_h3o.dotproduct(hidden2)
outputs.add(weights_bo)
outputs.func(activation_function)
```

```
targets ← Matrix(len(targets_array), 1)
targets.MakeMatrix()
FOR I in range(0, len(targets_array)):
   targets.matrix[i][0] ← targets_array[i]
END FOR

#calculate output errors
output_errors ← targets.subtract(outputs)

#calculate gradients
gradients ← outputs.stfunc(derivative_of_activation_function)
gradients.multiply(output_errors)
gradients.multiply(self.learningrate)

#calculate deltas
hidden2_transposed ← hidden2.transpose()
weight_ho_deltas ← gradients.dotproduct(hidden2_transposed)

#adjust ho weights by deltas
weights_h3o.add(weight_ho_deltas)
#adjust bias by deltas
weights_bo.add(gradients)

#calculate hidden layer errors
weights_ho_transposed ← weights_h3o.transpose()
hidden2_errors ← weights_ho_transposed.dotproduct(output_errors)

hidden2_gradient ← hidden2.stfunc(derivative_of_activation_function)
hidden2_gradient.multiply(hidden2_errors)
```

```
    hidden2_gradient.multiply(self.learningrate)

    hidden1_transposed ← hidden1.transpose()
    weight_h1h2_deltas ← hidden2_gradient.dotproduct(hidden1_transposed)

    weights_h1h2.add(weight_h1h2_deltas)
    weights_bh2.add(hidden2_gradient)

    weights_h1h2_transposed ← weights_h1h2.transpose()
    hidden_errors ← weights_h1h2_transposed.dotproduct(hidden2_errors)

    #calculate hidden gradients
    hidden_gradient ← hidden1.stfunc(derivative_of_activation_function)
    hidden_gradient.multiply(hidden_errors)
    hidden_gradient.multiply(self.learningrate)

    #calculate input to hidden deltas
    inputs_transposed ← inputs.transpose()
    weight_ih_deltas ← hidden_gradient.dotproduct(inputs_transposed)

    #adjust ih weights
    weights_ih1.add(weight_ih_deltas)
    #adjust hidden bias by deltas
    weights_bh1.add(hidden_gradient)

    nn.write_weights_to_file(weights_ih1.matrix, weights_bh1.matrix,
weights_h1h2.matrix, weights_bh2.matrix, weights_h3o.matrix, weights_bo.matrix)
END FUNCTION
```

Gui program

Contents of the gui program sections:

Imports

- Start of program, imports, makes two arrays containing contents of the settings and clubs files. The settings and clubs files are validated, making sure they exist and match up correctly.

```
IMPORT Tkinter
IMPORT tkMessageBox
IMPORT ttk
IMPORT tkFileDialog
IMPORT tkSimpleDialog as simpledialog
FROM PIL IMPORT ImageTk, Image

IMPORT re
FROM urllib2 IMPORT urlopen
```

```
IMPORT json
IMPORT requests
IMPORT math
IMPORT io
IMPORT sys

FROM ebaysdk.trading IMPORT Connection as Trading

FROM neuralnetwork IMPORT *
FROM images IMPORT *
continue_listing → True
#settings file
settings_array → []
TRY:
   settings_file → open("settings.txt", "r")
EXCEPT:
   OUTPUT "the settings file doesn't exist, the program will end now"
   sys.exit()
FOR line in settings_file:
   settings_array.append(line.strip('\n'))
END FOR
#clubs file
clubs → []
TRY:
   clubs_file → open("clubs.txt", "r")
EXCEPT:
   OUTPUT "the clubs file does not exist, the program will end now"
   sys.exit()
FOR club in clubs_file:
```

```
    clubs.append(club.strip('\n'))
END FOR
main()
```

main

- Welcome screen with button to exit program or input an image

```
FUNCTION main():
WHILE continue_listing == True:
        main_menu → Tkinter.Tk()
        welcome_label → Tkinter.Label(input_image, text = "Welcome To The Auto eBay Lister")
        input_image_button → Tkinter.Button(input_image, text="Input image", command=open_file)
        progress_bar → ttk.Progressbar(input_image, length=200)
        button_exit → Tkinter.Button(input_image, text="Exit program", command=end)
        main_menu.mainloop()
END FUNCTION
```

End

- Ends the program

```
FUNCTION end():
    input_image.destroy()
    sys.exit()
END FUNCTION
```

open_file

- This sub routine opens a window for the user to select their image of a shirt (will be validated to only allow JPEG images inputted) and then that gets passed through the neural network. User is asked if shirt network got is correct, if it is then it proceeds to the subroutine window_for_info_being_added(). If the shirt guessed by the network was incorrect, the user will type the correct club, then the neural network will train with the image inputted by the user and then will go to the window_for_info_being_added() sub routine.

```
FUNCTION open_file(main_menu):
  IF len(clubs) !→ int(settings_array[6]):
    tkMessageBox.showinfo('Problem with clubs or settings file', 'The number of clubs and number of output nodes do not match in the settings and club files')
    end(main_menu)
  END IF
  valid_file → False
  WHILE valid_file → False:
    file_path → tkFileDialog.askopenfilename(filetypes → [('Jpeg Files', '*.jpg')])
    IF file_path !→ '':
      img → Image.open(file_path)
      width, height → img.size
      IF width >→ 500 and height >→ 500:
        valid_file → True
      ELSE:
        tkMessageBox.showinfo('Invalid image', 'Your image is too small, please input another')
        resize → tkMessageBox.askquestion('Resize?', 'Would you like your image to be resized?\nIt is recommended you get a better quality image')
        IF resize → 'yes':
          img → Image.open(file_path)
          width, height → img.size
          new_img → img.resize((500, 500))
```

```
            new_img.save(file_path)
            valid_file → True
          END IF
        END IF
      END IF
    END WHILE

    Images(file_path).resize(settings_array[1])
    nn → neuralNetwork(7500,200,20,int(settings_array[6]))
    TRY:
      image → nn.run_with_existing_weights(Images(settings_array[2]).toarray()).matrix
    EXCEPT:
      tkMessageBox.showinfo('Problem with settings file', 'The number of output nodes in the settings file is incorrect or the file name is incorrect, please correct this and restart the program')
      end(main_menu)
    highest_index → 0
    highest → 0
    club_name → ""
    image_softmax → softmaxtrain(image)
    FOR I in range(0,len(image_softmax)):
     IF image_softmax[i] > highest:
      highest → image_softmax[i]
      highest_index→ i
     END IF
    END FOR
    IF highest > 0.18:
      club_name → clubs[highest_index]
      main_menu.destroy()
      check_shirt_window → Tkinter.Tk()
```

```
    TRY:
      image_shirt → ImageTk.PhotoImage(Image.open(settings_array[2]))
    EXCEPT:
      tkMessageBox.showinfo('Problem with settings file', 'The file path in the settings file does not exist, please update the settings file and then restart the program')
      check_shirt_window.destroy()
      sys.exit()
    panel_shirt → Tkinter.Label(check_shirt_window, image → image_shirt)
    panel_shirt.image → image_shirt
    progress_bar → ttk.Progressbar(check_shirt_window, length→200)
    progress_bar['value']→20
    correct_shirt → tkMessageBox.askquestion('Shirt', club_name + '?')
    IF correct_shirt → 'yes':
      check_shirt_window.destroy()
      window_for_info_being_added(file_path, club_name)
    ELIF correct_shirt → 'no':
      valid_title → False
      while valid_title → False:
        TRY:
          club_name → simpledialog.askstring('Input actual club','Please input the actual club name of the shirt', parent→check_shirt_window)
          IF club_name !→ '':
            IF club_name in clubs:
              valid_title → True
            ELSE:
              tkMessageBox.showinfo('Invalid club', 'Please input a valid club')
            END IF
          ELSE:
            tkMessageBox.showinfo('Invalid club', 'Please input a valid club')
          END IF
```

```
            EXCEPT:
                tkMessageBox.showinfo('Invalid club', 'Please input a valid club')
            TRY:
                inputs → Images(settings_array[3]).toarray()
            EXCEPT:
                tkMessageBox.showinfo('Problem with settings file', 'The file path in the settings file does not exist, please update the settings file and then restart the program')
                check_shirt_window.destroy()
                sys.exit()
            target → []
            FOR club in clubs:
                IF club → club_name:
                    club_index → clubs.index(club)
                END IF
            END FOR
            FOR I in range(0,int(settings_array[6])):
                target.append(0)
            END FOR
            target[club_index] → 1
            nn.train_with_existing_weights(inputs,target,nn)
            check_shirt_window.destroy()
            window_for_info_being_added(file_path, club_name)
        END IF
    ELSE:
        tkMessageBox.showinfo('Unknown shirt', 'Unsure what club this is, please try another photo')
END FUNCTION
```

window_for_info_being_added

- User inputs the year, size, weight of shirt and their PayPal email, which is all validated and displayed to the user. When confirm button is pressed, go to confirm_items sub routine.

```
FUNCTION window_for_info_being_added(file_path, club_name):
    input_shirt_details_window → Tkinter.Tk()
    valid_year → False
    valid_email → False
    label_year → Tkinter.Label(input_shirt_details_window, text→"Year of shirt")
    label_size → Tkinter.Label(input_shirt_details_window, text→"Select the size of the shirt")
    label_email → Tkinter.Label(input_shirt_details_window, text→"PayPal email")
    label_weight → Tkinter.Label(input_shirt_details_window, text→"Select the weight of the shirt")
    label_year_number → Tkinter.Label(input_shirt_details_window,text→"")
    combo_size → ttk.Combobox(input_shirt_details_window)
    combo_size['values']→ ("XXS", "XS", "S", "M", "L", "XL", "XXL", "XXXL")
    combo_size.current(0)
    label_email_display → Tkinter.Label(input_shirt_details_window,text→"")
    combo_weight → ttk.Combobox(input_shirt_details_window)
    combo_weight['values']→ ("w<1","1<w<2","2<w<10","10<w<15")
    combo_weight.current(0)
    progress_bar → ttk.Progressbar(input_shirt_details_window, length→200)
    progress_bar['value']→30
    button_confirm_text → Tkinter.Button(input_shirt_details_window, text→"Confirm"),
command→lambda: confirm_items(label_year_number.cget("text"),
combo_size.get(),label_email_display.cget("text"),combo_weight.get(),
input_shirt_details_window, file_path, club_name))
    WHILE valid_year → False:
        TRY:
```

```
        year → int(simpledialog.askstring('Year','Please input the year for the shirt',
parent→input_shirt_details_window))
        IF year < 1900 or year > 2020:
          tkMessageBox.showinfo('Invalid year', 'Please input a valid year')
        ELSE:
          valid_year → True
        END IF
      EXCEPT:
        tkMessageBox.showinfo('Invalid year', 'Please input a valid year')
   END WHILE
   label_year_number.configure(text→str(year))

   WHILE valid_email → False:
      TRY:
        email → str(simpledialog.askstring('Email','Please input your PayPal email',
parent→input_shirt_details_window))
        regex → re.search("[a-z0-9!#$%&'*+/→?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/→?^_`{|}~-
]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?", email)
        IF regex → None:
          tkMessageBox.showinfo('Invalid email', 'Please input a valid email')
        ELSE:
          valid_email → True
         END IF
      EXCEPT:
        tkMessageBox.showinfo('Invalid email', 'Please input a valid email')
   END WHILE
   label_email_display.configure(text→str(email))
END FUNCTION
```

Confirm_items

- Changes the abbreviation of size to actual word for size (e.g. “S” to “Small”) which is used for the actual listing and works out postage price based on weight of shirt. Searches through previously sold eBay items using eBay API and picks out the images and prices and titles. The title is decided by finding out the most used words, the price is calculated by using the calculate_price subroutine. Then the images and prices are used in the scrollbox to show the user what similarly sold items sold at.

```
FUNCTION confirm_items(year, size, email, weight, window, file_path, club_name):
    size_actual →’’
    IF size → “S”:
        size_actual → “Small”
    ELIF size → “M”:
        size_actual → “Medium”
    ELIF size → “L”:
        size_actual → “Large”
    END IF
    IF size_actual → ‘’:
        size_actual → size
    END IF
    postage_price → 0
    IF weight → “w<1”:
        postage_price → 2.89
    ELIF weight → “1<w<2”:
        postage_price → 4.05
    ELIF weight → “2<w<10”:
        postage_price → 6.49
    ELSE:
        postage_price → 8.99
    END IF
```

```
url → (settings_array[0] + club_name + "+shirt+home+" + str(year)+"+"+size_actual)
internet_connection → False
WHILE internet_connection → False:
    TRY:
        apiresult → requests.get(url)
        internet_connection → True
    EXCEPT:
        tkMessageBox.showinfo('No Internet Connection or invalid URL', 'Please connect to the internet or I the URL in the settings file and restart the program')
END WHILE
TRY:
    json_format_of_listings → apiresult.json()
    array_for_prices →[]
    array_for_images →[]
    array_for_titles →[]
    FOR item in (json_format_of_listings["findCompletedItemsResponse"][0]["searchResult"][0]["item"]):
        picture_of_listing → item["galleryURL"][0]
        ebay_title → item["title"][0]
        ebay_title_split → ebay_title.split()
        array_for_titles.append(ebay_title)
        array_for_images.append(picture_of_listing)
        price_of_shirt → item['sellingStatus'][0]["convertedCurrentPrice"][0]['__value__']
        array_for_prices.append(price_of_shirt)
    END FOR
    window.destroy()
    price → calculate_price(array_for_prices, year)
    words → []
    word_array → []
    word_count_array → []
```

```
IF len(array_for_titles) > 0:
    FOR title in range(0,len(array_for_titles)):
        split_title → array_for_titles[title].split()
        FOR word in split_title:
            word → word.lower()
            words.append(word)
        END FOR
    END FOR
    FOR word in words:
        IF len(word_array) → 0:
            word_array.append(word)
            word_count_array.append(1)
        ELSE:
            IF word in word_array:
                index → word_array.index(word)
                word_count_array[index] → word_count_array[index] + 1
            ELSE:
                word_array.append(word)
                word_count_array.append(1)
            END IF
        END IF
    END FOR
    title_words → []
    index → 0
    FOR num in word_count_array:
        IF num >= len(array_for_titles)/2.75:
            title_words.append(word_array[index])
        END IF
        index +→1
```

```
        END FOR
        FOR title_word in range(0,len(title_words)):
            title_words[title_word] → title_words[title_word].capitalize()
        END FOR
        title →  " ".join(title_words)
    ELSE:
        title → club_name + " Football Soccer Home Shirt Year " + year + " Size UK " +
size_actual + " Good Condition"
  END IF
    show_listing_window → Tkinter.Tk()
    show_listings(title, price, postage_price, show_listing_window, email, file_path)
  EXCEPT:
    title → club_name + " Football Soccer Home Shirt Year " + year + " Size UK " +
size_actual + " Good Condition"
    valid_price → False
    price → 0
    WHILE valid_price → False:
      TRY:
        price → float(simpledialog.askstring('Price','No listings found, please input a price',
parent→window))
        IF price < 0.99:
          tkMessageBox.showinfo('Invalid price', 'Please input a valid price')
        ELSE:
          valid_price → True
        END IF
      EXCEPT:
        tkMessageBox.showinfo('Invalid price', 'Please input a valid price')
    END WHILE
    window.destroy()
    show_listing_window → Tkinter.Tk()
```

```
        show_listings(title, price, postage_price, show_listing_window, email, file_path)
    IF len(array_for_images)>0:

scroll_window→Tkinter.Frame(show_listing_window,relief→Tkinter.GROOVE,width→500,h
eight→500,bd→1)
        global canvas
        canvas→Tkinter.Canvas(scroll_window)
        scroll_frame→Tkinter.Frame(canvas)

myscrollbar→Tkinter.Scrollbar(scroll_window,orient→"vertical",command→canvas.yview)
        canvas.configure(yscrollcommand→myscrollbar.set)
        myscrollbar.pack(side→"right",fill→"y")
        canvas.pack(side→"left")
        canvas.create_window((0,0),window→scroll_frame,anchor→'nw')
        scroll_frame.bind("<Configure>",scroll_box)
        FOR I in range(0,len(array_for_prices)):
            img → ImageTk.PhotoImage(Image.open(urlopen(array_for_images[i])))
            panel → Tkinter.Label(scroll_frame, image → img)
            panel.image → img
            panel.grid(column→0,row→i)
            Tkinter.Label(scroll_frame,text→str(array_for_prices[i])).grid(row→I,column→2)
        END FOR
    END IF
END FUNCTION
```

Scroll_box

- Used in confirm_items, to make the scroll box for showing images of previously sold shirts

```
FUNCTION scroll_box(event):
    canvas.configure(scrollregion=canvas.bbox("all"),width=500,height=450)
END FUNCTION
```

Calculate_price

- Used in confirm items to work out the price to list the item at. The list of prices passed in is sorted into numerical order using the quicksort algorithm. Here the lower and upper quartiles can be found out, and therefore the interquartile range. This allows outlier bounds to be calculated and therefore outliers can be removed. An average of the prices left is then calculated. The average price is then rounded up to the nearest integer with 0.01 removed to make the price look more appealing to buyers on eBay (the price will be x.99).

```
FUNCTION calculate_price(array_for_prices, year):
    total_price → 0
    copy_of_array_for_prices → []
    FOR I in range(0,len(array_for_prices)):
        copy_of_array_for_prices.append(array_for_prices[i])
    END FOR
    sorted_array → quicksort(copy_of_array_for_prices, 0, len(copy_of_array_for_prices)-1)
    length_of_array → len(array_for_prices)
    lower_quartile_position → int(math.ceil(length_of_array/4))
    upper_quartile_position → int(math.ceil((length_of_array/4)*3))
    inter_quartile_range → float(sorted_array[int(upper_quartile_position)])-
float(sorted_array[int(lower_quartile_position)])
    IF int(year) < 1980:
        difference → 0.05 * inter_quartile_range
```

```
    ELIF int(year) < 2000:
        difference → 0.1* inter_quartile_range
    ELIF int(year) < 2010:
        difference → 0.2*inter_quartile_range
    ELIF int(year) < 2015:
        difference → inter_quartile_range
    ELSE:
        difference → 1.5*inter_quartile_range
    lower_bound → float(sorted_array[int(lower_quartile_position)])-difference
    upper_bound → float(sorted_array[int(upper_quartile_position)])+difference
    FOR I in range(0,len(sorted_array)):
        IF float(sorted_array[i]) < lower_bound or float(sorted_array[i])>upper_bound:
            sorted_array[i] → 0
        END IF
    END FOR
    FOR I in range(0,len(sorted_array)):
        total_price +→ float(sorted_array[i])
    END FOR
    price → (math.ceil(total_price / length_of_array))-0.01
    RETURN price
END FUNCTION
```

Quicksort

- used in calculate_price, sorts the array of prices in order, uses recursion to make this an efficient algorithm

```
FUNCTION quicksort(array, start, end):
    low → start
```

```
    high → end
    pivot → array[int((low+high)/2)]
    WHILE low<=high:
        WHILE array[low] < pivot:
            low +→1
        END WHILE
        WHILE pivot < array[high]:
            high-→1
        END WHILE
        IF low <= high:
            temp → array[low]
            array[low] → array[high]
            array[high] → temp
            low+→1
            high-→1
        END IF
    END WHILE
    IF start<high:
        quicksort(array, start, high)
    END IF
    IF end > low:
        quicksort(array,low, end)
    END IF
    RETURN array
END FUNCTION
```

Show_listings

- displays what the listing will look like to the user (shows title, image, price, description and postage price) and gives them the opportunity to make any changes (any changes they make will be validated, like making sure the price is not below 0.99). Also shows the recently sold similar items (images and prices) and has a button which if you press, lists the item on eBay.

```
FUNCTION show_listings(title, price, postage_price, show_listing_window, email, file_path):
    label_title = Tkinter.Label(show_listing_window, text=title)
    image_of_shirt = ImageTk.PhotoImage(Image.open(settings_array[2]))
    panel = Tkinter.Label(show_listing_window, image = image_of_shirt)
    panel.image = image_of_shirt
    label_description = Tkinter.Label(show_listing_window, text=title)
    label_price = Tkinter.Label(show_listing_window, text=str(price))
    label_postage_price = Tkinter.Label(show_listing_window, text=str(postage_price))
    label_sub_heading = Tkinter.Label(show_listing_window, text="Recently sold shirts")
    button_change_title = Tkinter.Button(show_listing_window, text="Change title",
command=lambda: change_title(show_listing_window, label_title))
    button_change_description = Tkinter.Button(show_listing_window, text="Change
description", command=lambda: change_description(show_listing_window,
label_description))
    button_change_price = Tkinter.Button(show_listing_window, text="Change price",
command=lambda: change_price(show_listing_window, label_price))
    button_change_postage_price = Tkinter.Button(show_listing_window, text="Change
postage price", command=lambda: change_postage_price(show_listing_window,
label_postage_price))
    button_confirm_listing = Tkinter.Button(show_listing_window, text="List", bg="green",
command=lambda:
list_item(label_title.cget('text'),label_description.cget('text'),label_price.cget('text'),
label_postage_price.cget('text'), show_listing_window, email, file_path))
    progress_bar = ttk.Progressbar(show_listing_window, length=200)
    progress_bar['value']=75
END FUNCTION
```

Change_price

- used in the show_listings sub routine, used for the user wanting to make a change to the price, validates the input to make sure it is an integer or float and that it is greater than 0.99.

```
FUNCTION change_price(window, existing_label):
    valid_price → False
    WHILE valid_price → False:
        TRY:
            price → float(simpledialog.askstring('Change price','Please input a price', parent→window))
            IF price < 0.99:
                tkMessageBox.showinfo('Invalid price', 'Please input a valid price')
            ELSE:
                valid_price → True
            END IF
        EXCPET:
            tkMessageBox.showinfo('Invalid price', 'Please input a valid price')
    END WHILE
    existing_label.configure(text=str(price))
END FUNCTION
```

Change_title

- used in the show_listings sub routine, used for the user wanting to make a change to the title, validates to make sure the user inputs something.

```
FUNCTION change_title(window, existing_label):
    valid_title → False
    WHILE valid_title → False:
```

```
    TRY:
      title → simpledialog.askstring('Change title','Please input a title', parent→window)
      IF title !→ '':
        existing_label.configure(text=title)
        valid_title→True
      END IF
    EXCEPT:
      tkMessageBox.showinfo('Invalid title', 'Please input a valid title')
  END WHILE
END FUNCTION
```

Change_description

- used in the show_listings sub routine, used for the user wanting to make a change to the description, validates to make sure the user inputs something.

```
FUNCTION change_description(window, existing_label):
  valid_description → False
  WHILE valid_description → False:
    TRY:
      description → simpledialog.askstring('Change description','Please input a description', parent→window)
      IF description !→ '':
        existing_label.configure(text→description)
        valid_description→True
      END IF
    EXCEPT:
      tkMessageBox.showinfo('Invalid description', 'Please input a valid description')
  END WHILE
END FUNCTION
```

Change_postage_price

- used in the show_listings sub routine, used for the user wanting to make a change to the title, validates to make sure the user inputs an integer or float and that it is greater than 0.01

```
FUNCTION change_postage_price(window, existing_label):
    valid_price → False
    WHILE valid_price → False:
        TRY:
            price → float(simpledialog.askstring('Change postage price','Please input postage price', parent→window))
            IF price < 0.01:
                tkMessageBox.showinfo('Invalid postage price', 'Please input a valid postage price')
            ELSE:
                valid_price → True
            END IF
        EXCEPT:
            tkMessageBox.showinfo('Invalid postage price', 'Please input a valid postage price')
    END WHILE
    existing_label.configure(text→str(price))
END FUNCTION
```

List_item

- Lists the item on eBay using eBay api, called in the show_listings sub routine when 'list' button is pressed. Uploads the image to eBay to be able to access it and use it for the listing. Uses the details for title, image, description, postage price and PayPal email for the listing. Validates to make sure you are connected to the internet.

```
FUNCTION list_item(title, description, price, postage_price,window, email, file_path):
    api → Trading(config_file→"ebay.yaml", siteid→3)
    WITH Image.open(file_path) AS user_image:
        user_image.thumbnail((1600,1600))
        WITH io.BytesIO() AS image:
            user_image.save(image, "JPEG")

            files → {'file': ('EbayImage', image.getvalue())}
            pictureData → {
                "WarningLevel": "High",
                "PictureSet":'Supersize',
                "PictureName": "Test"
            }
            internet → False
            WHILE internet → False:
                TRY:
                    response → api.execute('UploadSiteHostedPictures', pictureData, files→files)
                    picture → (response.reply.SiteHostedPictureDetails.FullURL)
                    internet → True
                EXCEPT:
                    tkMessageBox.showinfo('No internet connection or invalid eBay token', 'Please connect to the internet or get new eBay token')
            END WHILE
```

File used to store my personal user eBay API keys

Site ID for the UK

```
api_request → {
  "Item": {
    "Title": title,
    "Country": "GB",
    "Location": "GB",
    "Site": "UK",
    "ConditionID": "3000",
    "PaymentMethods": "PayPal",
    "PayPalEmailAddress": email,
    "PictureDetails": {"PictureURL": [picture]},
    "PrimaryCategory": {"CategoryID": "123490"},
    "Description": description,
    "ListingType": "FixedPriceItem",
    "ListingDuration": "GTC",
    "StartPrice": price,
    "Currency": "GBP",
    "ReturnPolicy": {
      "ReturnsAcceptedOption": "ReturnsAccepted",
      "RefundOption": "MoneyBack",
      "ReturnsWithinOption": "Days_30",
      "ShippingCostPaidByOption": "Buyer"
    },
    "ShippingDetails": {
      "ShippingServiceOptions": {
        "FreeShipping": "False",
        "ShippingService": "UK_myHermesDoorToDoorService",
        "ShippingServiceCost": postage_price
      }
    },
```

```
            "DispatchTimeMax": "2"
        }
    }
    TRY:
        api.execute("AddItem", api_request)
        tkMessageBox.showinfo('Listing complete', 'Your item has been published on eBay')
    EXCEPT:
        tkMessageBox.showinfo('Invalid listing', 'This listing already exists')
    window.destroy()
END FUNCTION
```

Training neural network

- this is a separate program which uses the neural network class to train

```
FROM testsagain IMPORT *
FROM images IMPORT *
IMPORT sys

#settings file
settings_array → []
TRY:
   settings_file → open("settings.txt", "r")
EXCEPT:
   OUTPUT "the settings file doesn't exist, the program will end now"
   sys.exit()
FOR line in settings_file:
   settings_array.append(line.strip('\n'))
END FOR

FUNCTION softmaxtrain(outputs):
 arr → []
 denominator → 0
 FOR I in range(0,len(outputs)):
  FOR j in range(0,1):
   denominator +→ math.exp(outputs[i][j])
  END FOR
 END FOR
 FOR I in range(0,len(outputs)):
  FOR j in range(0,1):
   arr.append((math.exp(outputs[i][j]))/denominator)
```

Softmax function which is explained in analysis, page 19

```
    END FOR
  END FOR
  RETURN arr
END FUNCTION
```

Add the training inputs to an array (adds all the pictures in the training data folders and converts them to arrays of floats)

```
training_inputs → []
FOR I in range(0, 113):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/99ewcast/pic" + str(i) + ".jpg").toarray())
END FOR
FOR I in range(0,108):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/arsenal/pic" + str(i) + ".jpg").toarray())
END FOR
FOR I in range(0,59):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/Norwich/pic" + str(i) + ".jpg").toarray())
END FOR
FOR I in range(0,106):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/mancity/pic" + str(i) + ".jpg").toarray())
END FOR
FOR I in range(0,87):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/99ewcastle/pic" + str(i) + ".jpg").toarray())
END FOR
FOR I in range(0,123):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/99ewcastle/pic" + str(i) + ".jpg").toarray())
END FOR
```

Add the training targets to an array (each shirt has a specific output, so outputs are added in the same quantity and in the same order as the training data)

```
training_targets → []
FOR I in range(0,113):

training_targets.append([1,0,0,0,0,0])#chelsea
END FOR
FOR I in range(0,108):
 training_targets.append([0,1,0,0,0,0])#arsenal
END FOR
FOR I in range(0,59):
 training_targets.append([0,0,1,0,0,0])#norwich
END FOR
FOR I in range(0,106):
 training_targets.append([0,0,0,1,0,0])#man city
END FOR
FOR I in range(0,87):
 training_targets.append([0,0,0,0,1,0])#tottenham
END FOR
FOR I in range(0,123):
 training_targets.append([0,0,0,0,0,1])#newcastle
END FOR

nn → neuralNetwork(7500,200,20,6)
```

Here a randomly selected image is chosen and then the correct output is assigned. This trains with the randomly generated weights and writes them to the text file (seen in the train subroutine)

```
FOR I in range(0,1):
 inputs → random.choice(training_inputs)
 index → training_inputs.index(inputs)
 target → training_targets[index]
 nn.train(inputs,target)
```

```
 OUTPUT “trained”
END FOR


ind → 0
FOR I in range(0,1000):
 ind +→1
 inputs → random.choice(training_inputs)
 index → training_inputs.index(inputs)
 target → training_targets[index]
 nn.train_with_existing_weights(inputs,target,nn)
 OUTPUT “trained” + str(ind)
 IF ind → 100 or ind → 200 or ind → 300 or ind → 400 or ind → 500 or ind → 600 or ind → 700 or ind → 800 or ind → 900 or ind → 1000:
    OUTPUT “101ewcast → [1,0,0,0,0,0]”
    FOR I in range(1,11):
       c → nn.run_with_existing_weights(Images(“chelseatest” + str(i) + “.jpg”).toarray()).matrix
       OUTPUT softmaxtrainI
    END FOR
    OUTPUT “arsenal → [0,1,0,0,0,0]”
    FOR I in range(1,11):
       a → nn.run_with_existing_weights(Images(“arsenaltest” + str(i) + “.jpg”).toarray()).matrix
       print softmaxtrain(a)
    END FOR
    OUTPUT “101ewcast → [0,0,1,0,0,0]”
    FOR I in range(1,11):
       nor → nn.run_with_existing_weights(Images(“norwichtest” + str(i) + “.jpg”).toarray()).matrix
       print softmaxtrain(nor)
    END FOR
```

The way you train the neural network can be customized a lot (e.g. you can change the training data, number of iterations and the learning rate). This is just one example of training the network for 1000 iterations, printing out the results of some images every 100 iterations. Every 100 iterations here each club has 10 testing shirts and these testing shirts are fed through the neural network, using the run_with_existing_weights sub routine. The outputs are then displayed so you are able to see how the network is performing

```
    OUPUT “man city → [0,0,0,1,0,0]”
    FOR I in range(1,11):
        man → nn.run_with_existing_weights(Images(“mancitytest” + str(i) +
“.jpg”).toarray()).matrix
        OUTPUT softmaxtrain(man)
    END FOR
    OUTPUT “102ewcastle → [0,0,0,0,1,0]”
    FOR I in range(1,11):
        w → nn.run_with_existing_weights(Images(“102ewcastle” + str(i) +
“.jpg”).toarray()).matrix
        OUTPUT softmaxtrain(w)
    END FOR
    OUTPUT “102ewcastle → [0,0,0,0,0,1]”
    FOR I in range(1,11):
        new → nn.run_with_existing_weights(Images(“102ewcastle” + str(i) +
“.jpg”).toarray()).matrix
        OUTPUT softmaxtrain(new)
    END FOR
  END IF
 END FOR
```

Validation

Images

The image inputted by the user will be validated to ensure that only JPEG images can be inputted.

Year of shirt

The year of the shirt will be validated to ensure that an integer between 1900 and 2020 is inputted.

Size and weight of shirt

The size and weight of the shirts will be validated by using drop down boxes. Here only certain values will be allowed to be selected ensuring any value selected is valid.

PayPal email

the PayPal email will be validated using Regex to confirm that a valid email has been inputted.

The regex I will use is:

[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?

(Regexr.com, n.d.)

Connecting to the API

I will also validate that when the eBay API is used, like searching through previously sold items, uploading an image to eBay and listing the item on eBay, that the user has internet connection to avoid any errors when trying to connect to the API.

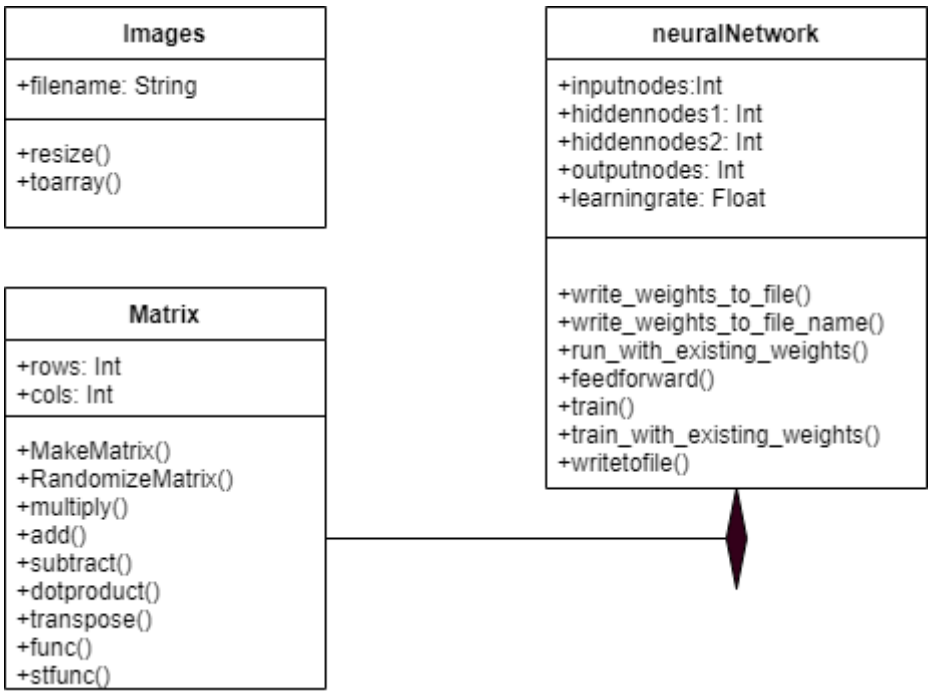
Validation on the user making changes to the listing

There will be validation for if the user wants to change the price – it will be ensured that they input an integer or float and that it is greater than 0.99.

There will be validation for if the user wants to change the description or title – it will be ensured that they input at least one character.

There will be validation for if the user wants to change the postage price – it will be ensured that they input an integer or float and that it is greater than 0.01.

UML class diagram



Data Storage

Each image for the training data will be resized to a 50 x 50 pixel image. This will normalize my data and make all images the same size. Then each image will be stored in an array with 3 elements per pixel (for the RGB colour values), so an array of length 7500 (50*50*3).

Each image will also have a matching target array for outputs. For example:

Liverpool [0,0,1]

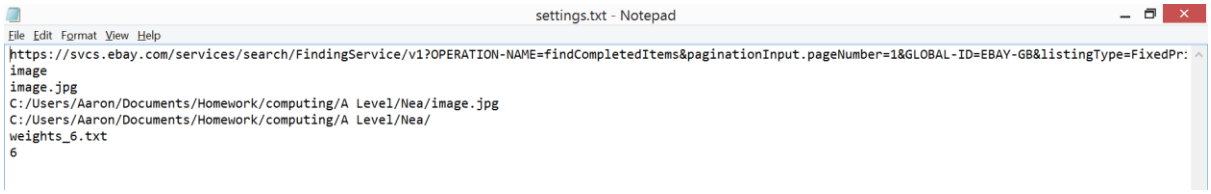
Arsenal [0,1,0]

Chelsea [1,0,0]

The weights that will be created by the neural network will be stored in a notepad file, they will be able to be updated and used throughout the program.

Here is an example of how I will store the weights for my neural network. Each weight will be separated by a comma and each layer will be separated by a new line.

Settings.txt file:



The settings file has any file paths or web URLs, which means you only have to change the settings file not the actual code if making changes. E.g. if changing where you want to save images to or changing the weights file.

This is the structure of the file:

Line 1 – search URL for eBay finding API

Line 2 – name of file to save resized image inputted by user

Line 3 – to use the resized image inputted by the user

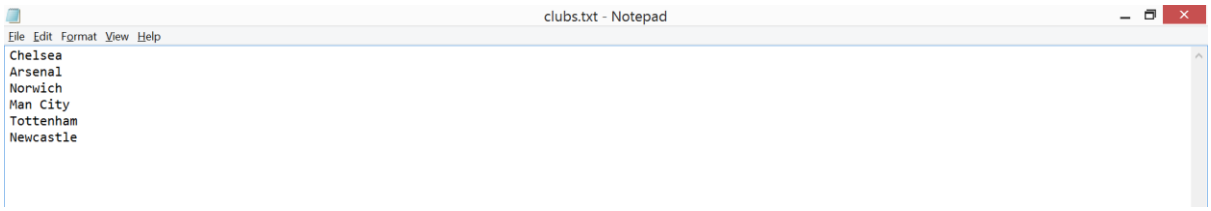
Line 4 – exact location of resized image inputted by user

Line 5 – used for the images class, for where images resized are saved

Line 6 – the name of the weights file for the neural network

Line 7 – number of output nodes for network (number of clubs network is working for)

Clubs.txt file



This file stores the name of all the clubs that the program works for and will be used for training the network if the program gets the wrong club.

Testing

Test Table

Test number	Description	Data type	Expected result	Pass/Fail	Cross reference
1	User presses input image button and comes up with window to select an image from	Typical	Comes up with a window to select an image	Pass	Validation part 1 – 9 secs
2	Image user inputs must be a jpeg	Typical	Only allow user to input jpeg images	Pass	Validation part 1 – 20 secs
3	User inputs no image	Erroneous	Ask user to input image again	Pass	Validation part 1 – 54 secs
4	When user has option to say whether shirt neural network guesses is correct, the user selects 'no' button	Typical	Comes up with a text input to type in the correct club of the shirt	Pass	Validation part 1 – 1 min 13 secs
5	When user has to input actual shirt if neural network guesses wrong, test whether allowed shirt names work	Typical	Trains the neural network with the image inputted and the actual club name inputted by the user	Pass	Validation part 1 – 1 min 45 secs
6	When user has to input actual shirt if neural network guesses wrong, test with any string that isn't one of the allowed	Erroneous	User asked to input club again	Pass	Validation part 1 – 3 mins

	club names is inputted				
7	When user has option to say whether shirt neural network guesses is correct, the user selects 'yes' button	Typical	Window for inputting year of shirt comes up	Pass	Validation part 1 – 3 mins 20 seconds
8	User inputting year of shirt, input a number between 1900 and 2020	Typical	Window for inputting PayPal email	Pass	Validation part 1 – 3 mins 30 seconds
9	User inputting year of shirt, input a number below 1900 and above 2020	Erroneous	Ask user to input year again	Pass	Validation part 1 – 3 mins 47 seconds
10	User inputting year of shirt, input boundaries of 1900 and 2020	Extreme	Window for inputting PayPal email	Pass	Validation part 1 – 4 mins 4 seconds
11	User inputting year of shirt, input a string	Erroneous	Ask user to input year again	Pass	Validation part 1 – 4 mins 25 seconds
12	User inputting year of shirt, input a float	Erroneous	Ask user to input year again	Pass	Validation part 1 – 4 mins 33 seconds
13	User inputting email, input a valid email	Typical	Goes to the confirm item window	Pass	Validation part 1 – 4 mins 44 seconds
14	User inputting email, input an invalid email	Erroneous	Asks user to input email again	Pass	Validation part 1 – 5 mins 2 seconds
15	User presses 'confirm' button	Typical	Displays a window with previously sold items and their prices and the user's listing with options to change title, price,	Pass	Validation part 1 – 5 mins 14 seconds, I say 16 but mean 15

			description and postage price, and the option to list the item		
16	If can't find any previously sold items, then asks user to input their own price, input an integer	Typical	Accepts the input and displays the user's listing	Pass	Validation part 1 – 5 mins 54 seconds
17	If can't find any previously sold items, then asks user to input their own price, input a float	Typical	Accepts the input and displays the user's listing	Pass	Validation part 1 – 6 mins 25 seconds
18	If can't find any previously sold items, then asks user to input their own price, input a string	Erroneous	Asks user to input a price again	Pass	Validation part 1 – 6 mins 37 seconds
19	If can't find any previously sold items, then asks user to input their own price, input nothing	Erroneous	Asks the user to input a price again	Pass	Validation part 1 – 6 mins 43 seconds
20	User presses 'change title' button, inputs a valid string	Typical	Changes the title and displays it on the window	Pass	Validation part 1 – 6 mins 54 seconds
21	User presses 'change title' button, inputs nothing	Erroneous	Asks the user to input title again	Pass	Validation part 1 – 7 mins 8 seconds
22	User presses 'change description' button, inputs a valid string	Typical	Changes the description and displays it on the window	Pass	Validation part 1 – 7 mins 20 seconds

23	User presses 'change description' button, inputs nothing	Erroneous	Asks the user to input the description again	Pass	Validation part 1 – 7 mins 30 seconds
24	User presses 'change price' button, inputs an integer	Typical	Accepts the input and displays the new price on window	Pass	Validation part 1 – 7 mins 40 seconds
25	User presses 'change price' button, inputs a float	Typical	Accepts the input and displays the new price on the window	Pass	Validation part 1 – 7 mins 45 seconds
26	User presses the 'change price' button, inputs a string	Erroneous	Asks the user to input the price again	Pass	Validation part 1 – 7 mins 55 seconds
27	User presses the 'change price' button, inputs nothing	Erroneous	Asks the user to input the price again	Pass	Validation part 1 – 8 mins 2 seconds
28	User presses the 'change postage price' button, inputs an integer	Typical	Accepts the input and displays the new postage price on the window	Pass	Validation part 1 – 8 mins 10 seconds
29	User presses the 'change postage price' button, inputs a float	Typical	Accepts the input and displays the new postage price on the window	Pass	Validation part 1 – 8 mins 20 seconds
30	User presses the 'change postage price' button, inputs a string	Erroneous	Asks the user to input the postage price again	Pass	Validation part 1 – 8 mins 30 seconds
31	User presses the 'change postage price' button, inputs nothing	Erroneous	Asks the user to input the postage price again	Pass	Validation part 1 – 8 mins 35 seconds
32	User presses 'change price' button, input a price below 0.99	Erroneous	Asks the user to input the price again	Pass	Validation part 1 – 8 mins 40 seconds

33	User presses 'change price' button, input a price of 0.99	Extreme	Accepts the input and displays the new price on the window	Pass	Validation part 1 – 9 mins 5 seconds
34	User presses the 'change postage price' button, input a price below 0.01	Erroneous	Asks the user to input the postage price again	Pass	Validation part 1 – 9 mins 12 seconds
35	User presses the 'change postage price' button, input a price of 0.01	Extreme	Accepts the input and displays the new price on the window	Pass	Validation part 1 – 9 mins 25 seconds
36	User presses the 'list' button	Typical	Lists the item on eBay	Pass	Validation part 2 – 8 seconds
37	User presses the 'list' button, with no internet connection	Erroneous	Asks the user to connect to the internet first before being able to list the item	Pass	Vid 2 – 30 seconds
38	User presses the confirm button with no internet connection	Erroneous	Asks the user to connect to the internet first before being able to search through previously sold items	Pass	Validation part 2 – 1 minute 10 seconds
39	Put a file name that doesn't exist in line 3 of settings file	Erroneous	End the program with a message	Pass	Validation part 2 – 1 minute 45 seconds
40	Put an incorrect number of output nodes in the setting file at line7	Erroneous	End the program with a message	Pass	Validation part 2 – 2 mins 35 seconds
41	Put a file path which doesn't exist in settings file on line 4	Erroneous	Ends the program with a message	Pass	Validation part 2 – 3 mins 10 seconds

42	Put a URL that is incorrect in the settings file on line 1	Erroneous	Tells user to update settings file and restart the program	Pass	Validation part 2 – 3 mins 50 seconds
43	Put a file path that doesn't exist in settings file on line 5	Erroneous	Ends the program with a message	Pass	Validation part 2 – 4 mins 50 seconds
44	Input a file name that doesn't exist for the weights in the settings file at line 6	Erroneous	Ends the program with a message	Pass	Validation part 2 – 5 mins 30 seconds, I say test 46 but mean 44
45	Make the number of clubs in the club txt file different to the number of output nodes in the settings file	Erroneous	Ends the program with a message	Pass	Validation part 2 – 6 mins
46	Start the program with the clubs txt file not existing	Erroneous	Ends the program with a message	Pass	Validation part 2 – 6 mins 30 seconds
47	Start the program with the settings txt file not existing	Erroneous	Ends the program with a message	Pass	Validation part 2 – 6 mins 50 seconds

Videos for testing

Video running through code – https://www.youtube.com/watch?v=puf1-pwU_8A

Video of validation part 1 – https://www.youtube.com/watch?v=JwkNnsa6iq0

Video of validation part 2 – https://www.youtube.com/watch?v=fy9s2GheT0Y

Video of training the neural network – https://www.youtube.com/watch?v=j5UU94mfsq8

The matrix calculations

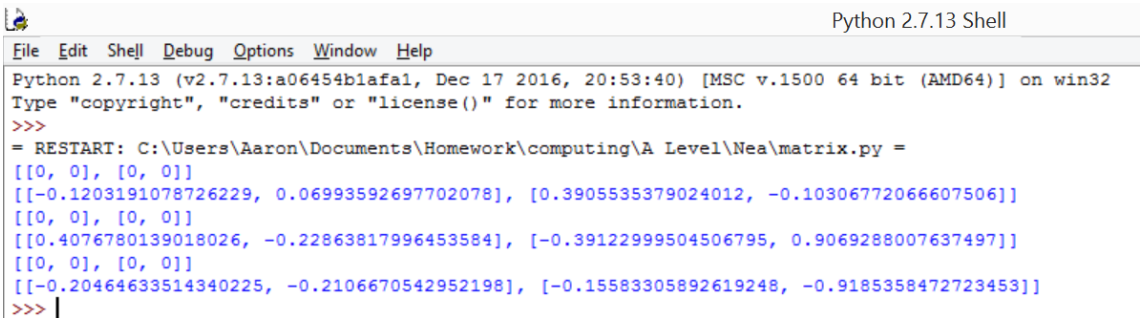
Here I will compare handwritten matrix calculations by myself against the outputs my matrix class produces, to show that the matrix class is performing as it should.

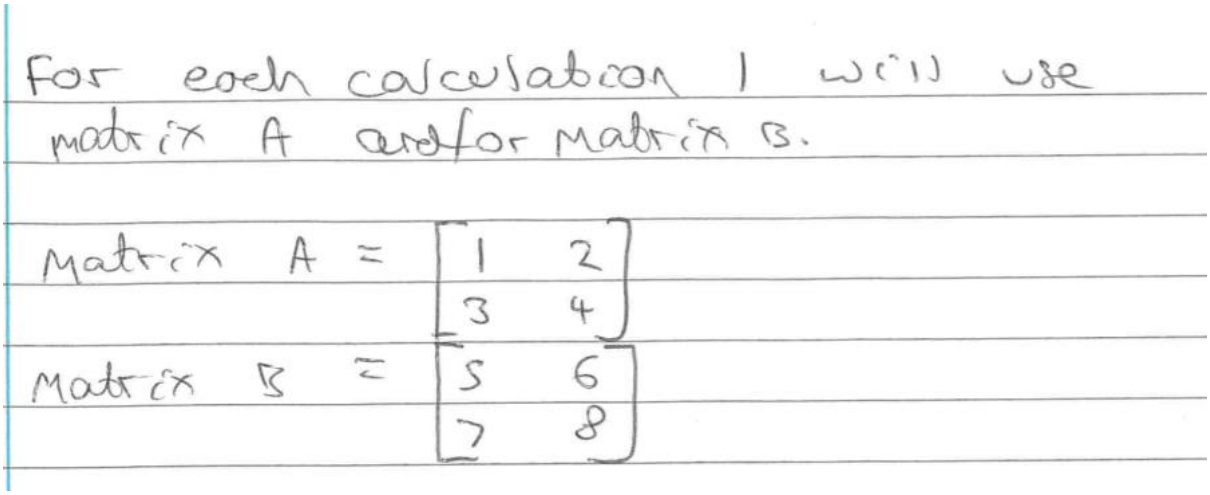
Showing it produces random matrices:

This is the code I used, I am generating 3 matrices and printing them to show they are random (between -1 and 1):

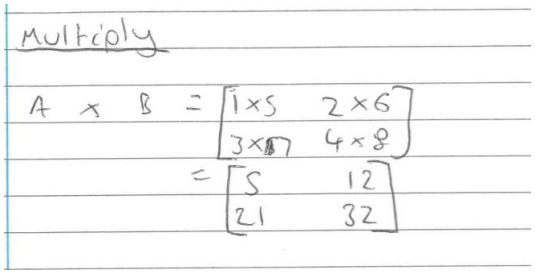
```
1.  for I in range(0,3):
2.      m = Matrix(2,2)
3.      m.MakeMatrix()
4.      print m.matrix
5.      m.RandomizeMatrix()
6.      print m.matrix
```

And the output:



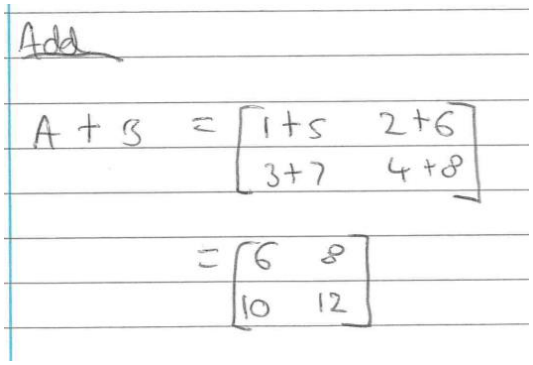


Multiply:



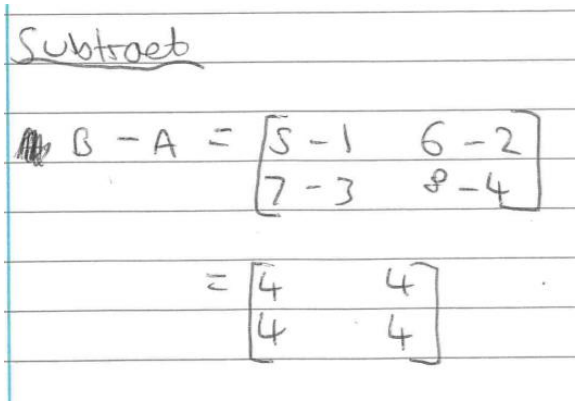

```
Matrix A
[[1, 2], [3, 4]]
Matrix B
[[5, 6], [7, 8]]
Matrix A x Matrix B
[[5, 12], [21, 32]]
>>>
```

Add:



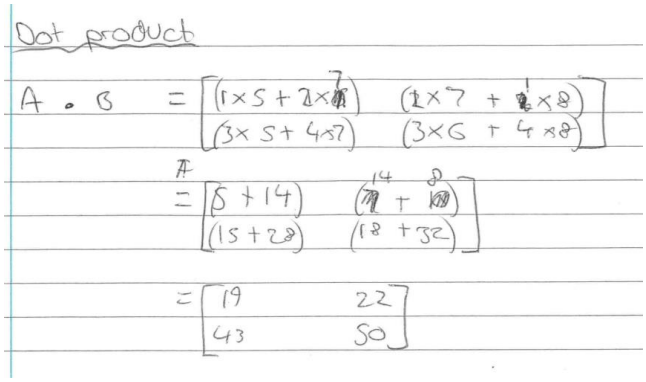

```
Matrix A
[[1, 2], [3, 4]]
Matrix B
[[5, 6], [7, 8]]
Matrix A + Matrix B
[[6, 8], [10, 12]]
>>>
```

Subtract:



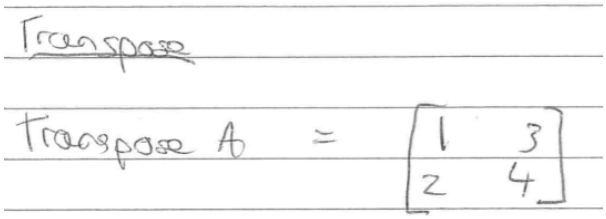

```
Matrix A
[[1, 2], [3, 4]]
Matrix B
[[5, 6], [7, 8]]
Matrix B - Matrix A
[[4, 4], [4, 4]]
>>>
```

Dot product:



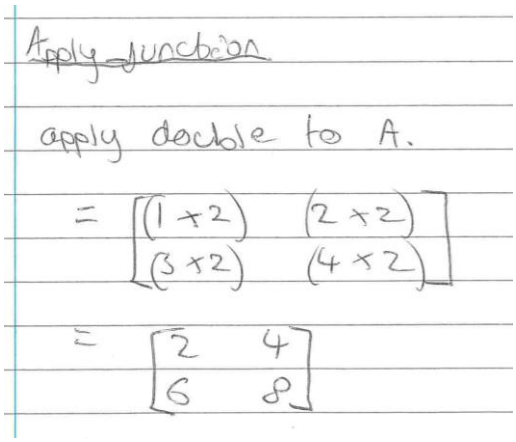

```
Matrix A
[[1, 2], [3, 4]]
Matrix B
[[5, 6], [7, 8]]
Matrix A . Matrix B
[[19, 22], [43, 50]]
```

Transpose:




```
Matrix A
[[1, 2], [3, 4]]
Matrix A transposed
[[1, 3], [2, 4]]
>>>
```

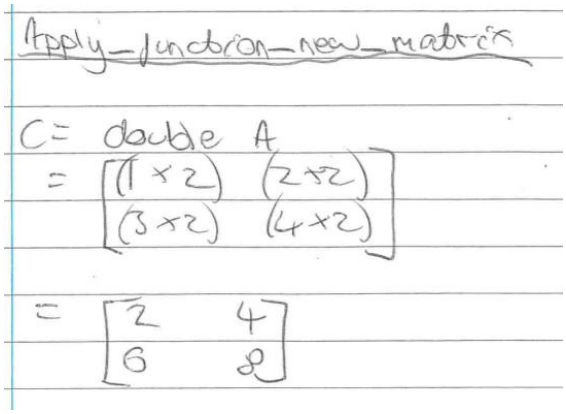
Apply_function:




```
1.  def double(x):
2.      return x *2
3.
4.  print "Matrix A doubled"
5.  print A.apply_function(double)
```

```
Matrix A
[[1, 2], [3, 4]]
Matrix A doubled
[[2, 4], [6, 8]]
>>>
```

Apply_function_new_matrix:



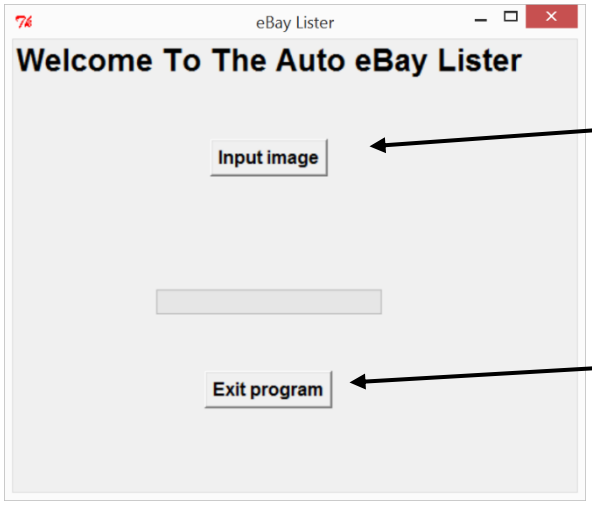

```
1.  def double(x):
2.      return x *2
3.
4.  print "Matrix A doubled new matrix"
5.  new = A.apply_function_new_matrix(double)
6.  print new.matrix
```

```
Matrix A
[[1, 2], [3, 4]]
Matrix A doubled new matrix
[[2, 4], [6, 8]]
```

GUI

Here I am showing and explaining the user interface of my program:

1.

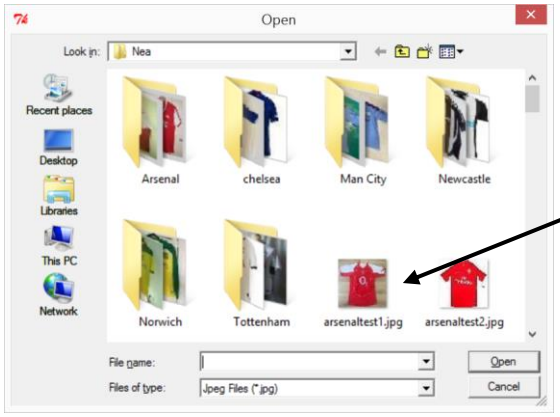


Option to input an image

Requirement 4

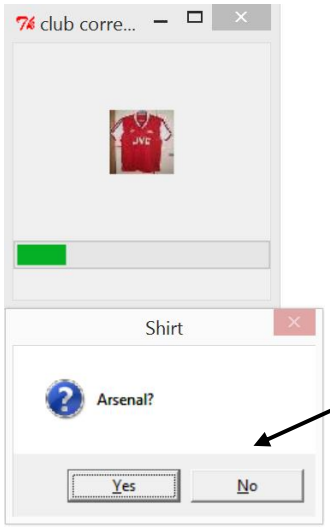
Option to end the program

2.



User can then select an image to input, but can only be a JPEG

3.



After selecting an image, the program then uses the neural network to work out what club the shirt inputted was, and then asks the user to confirm if the neural network got the club correct

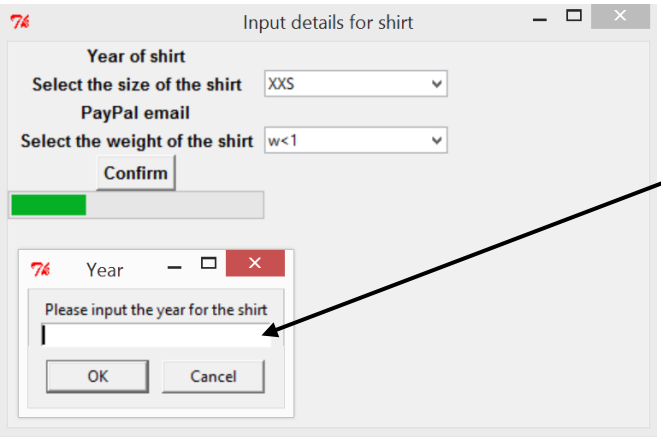
Requirements 4.3 and 4.4

4.

If the user says no, then they are asked to input the actual club, and then the neural network will train with the image inputted and the actual club name and then will go to stage 5.

Requirements 5.1, 5.2 and 5.3

5.

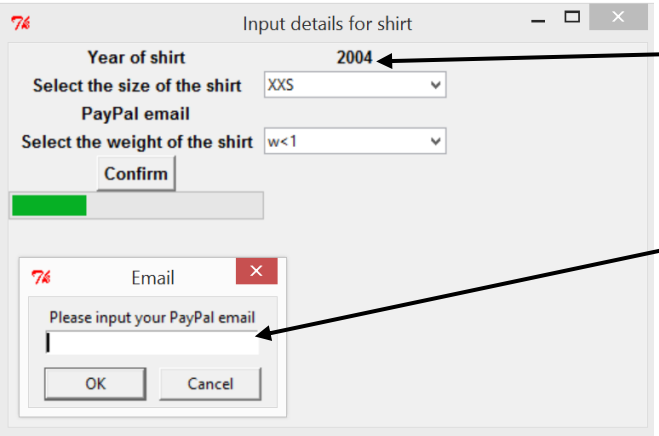


This is the window the user will be shown straight after pressing 'yes' or after pressing 'no' once the neural network has trained.

Here the user is asked to input the year of the shirt

Requirement 6.1.1

6.

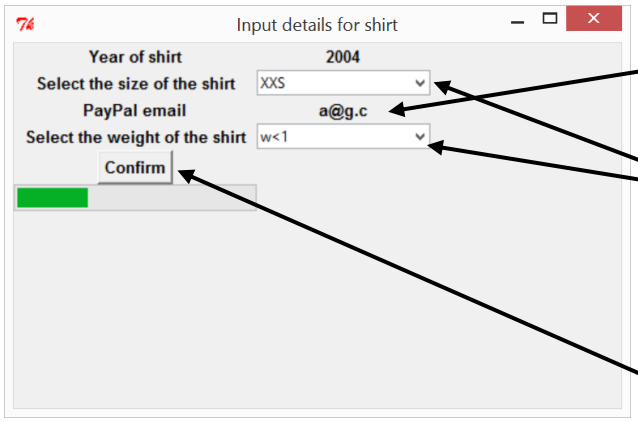


The year inputted is then shown on the window

And now the user is asked to input their PayPal email, which is used for listing the item on eBay

Requirement 6.1.2

7.



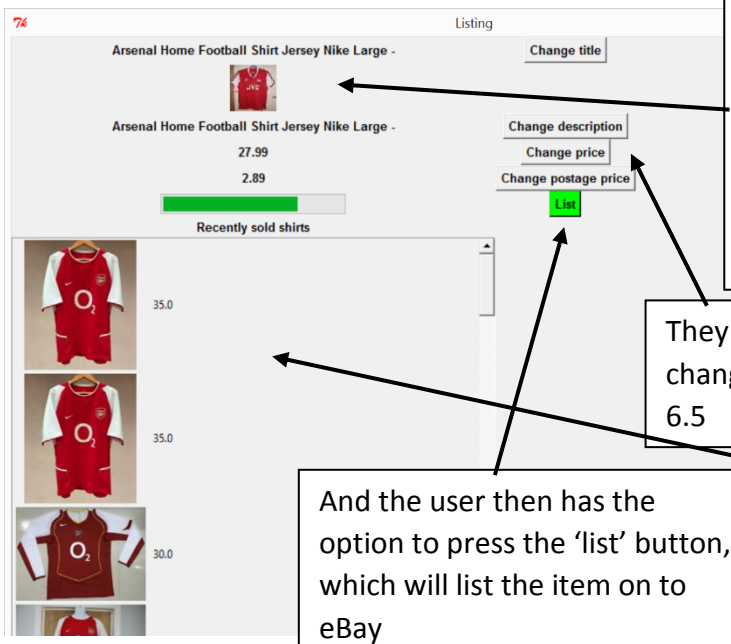
The PayPal email inputted is then displayed on the window

The user can then change the size of the shirt and the weight of the shirt

Requirements 6.1.3 and 6.1.4

When they are happy with the details they have inputted they can press the confirm button

8.



Once they have pressed confirm, the user is shown what their current listing will be, the title, image, description, price and postage price – Requirements 6.2, 6.3.1, 6.3.2, 6.4.1, 6.4.2

They also have the option to make changes to their listing – Requirement 6.5

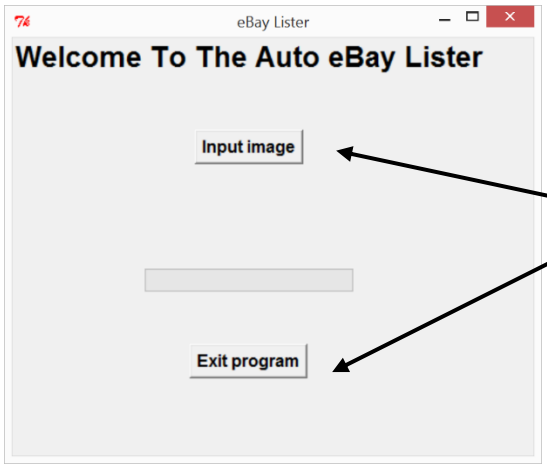
And the user then has the option to press the 'list' button, which will list the item on to eBay

Requirement 6.5

It also shows previously sold items on eBay along with their prices as requested in my interview

Requirement 6.4.3

9.



The user then has the option to list another shirt or exit the program

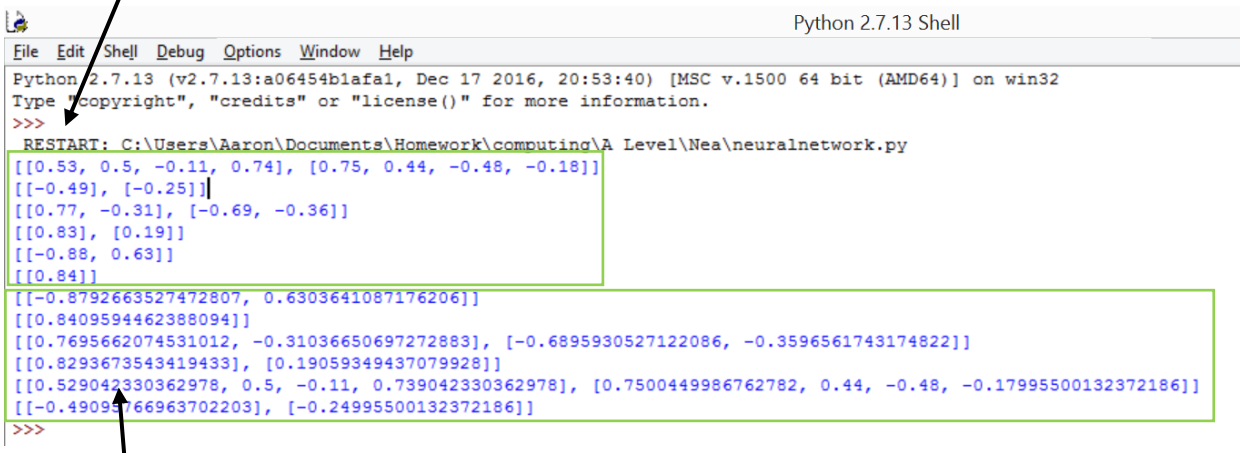
Requirement 7

Training the neural network

Worked example of the back propagation for the neural network

Here is a handwritten example of me going through the calculations the neural network should be doing and coming up with an output. Then I will compare my output against the output of the neural network. This will show one iteration of training. I will be using a neural network with 4 input nodes, 2 hidden layer 1 nodes, 2 hidden layer 2 nodes and 1 output node (I can't use the same number of nodes for my actual network because there's too may of them, however it is the same structure).

Weights before training



Weights after training (should be the same as what I get for the results of my calculations)

Backpropagation worked example.

Inputs = $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

Target = [1]

Network layout = 4, 2, 2, 1

Learning rate = 0.01

weights_ih = [[0.53, 0.5, -0.11, 0.74],
[0.75, 0.44, -0.48, -0.36]]

bias_h = [[-0.49], [-0.25]]

weights_h1h2 = [[0.77, -0.31], [-0.69, -0.36]]

bias_h2 = [[0.83], [0.19]]

weights_ho = [[-0.88, 0.63]]

bias_o = [[0.84]]

All the weights above have come from a random number generator and have been hardcoded for the start of the neural network for this example.

Sigmoid = $x < 0$:
$1 - 1/(1 + e^{x})$
$x \geq 0$:
$1/(1 + e^{-x})$

dsigmoid = $x \times (1 - x)$

Calculations.

hidden = self.weights_ih.dot(inputs)

$$\text{hidden} = \begin{bmatrix} 0.53 & 0.5 & -0.11 & 0.74 \\ 0.75 & 0.44 & -0.48 & -0.36 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\text{hidden} = \begin{bmatrix} (0.53 \times 1) + (0.5 \times 0) + (-0.11 \times 0) + (0.74 \times 1) \\ (0.75 \times 1) + (0.44 \times 0) + (-0.48 \times 0) + (-0.36 \times 1) \end{bmatrix}$$

$$= \begin{bmatrix} 0.53 + 0 + 0 + 0.74 \\ 0.75 + 0 + 0 + -0.36 \end{bmatrix}$$

$$= \begin{bmatrix} 1.27 \\ 0.39 \end{bmatrix}$$

$$\text{hidden} = \text{hidden} + \text{bias\_h} = \begin{bmatrix} 1.27 \\ 0.39 \end{bmatrix} + \begin{bmatrix} -0.49 \\ -0.25 \end{bmatrix}$$

$$= \begin{bmatrix} 0.78 \\ 0.14 \end{bmatrix}$$

hidden with sigmoid applied.

$$= \begin{bmatrix} (1 / (1 + e^{-0.78})) \\ (1 / (1 + e^{-0.14})) \end{bmatrix}$$

$$= \begin{bmatrix} 0.685680113\,9 \\ 0.534942945\,2 \end{bmatrix}$$

hidden2 = weights_h1h2 . dotproduct (hidden)

$$= \begin{bmatrix} 0.72 & -0.31 \\ -0.69 & -0.36 \end{bmatrix} \cdot \begin{bmatrix} 0.685689011139 \\ 0.5349429452 \end{bmatrix}$$

$$= \begin{bmatrix} (0.72 \times 0.685689011139) + (-0.31 \times 0.5349429452) \\ (-0.69 \times 0.685689011139) + (-0.36 \times 0.5349429452) \end{bmatrix}$$

$$= \begin{bmatrix} 0.493689682 + -0.165832313 \\ -0.473119278 6 + -0.192579460 3 \end{bmatrix}$$

$$= \begin{bmatrix} 0.327857369 \\ -0.665698738 9 \end{bmatrix}$$

$$\text{hidden2} = \text{hidden2} + \text{bias\_h2} = \begin{bmatrix} 0.327857369 \\ -0.665698738 9 \end{bmatrix} + \begin{bmatrix} 0.83 \\ 0.19 \end{bmatrix}$$

$$= \begin{bmatrix} 1.157857369 \\ -0.475698738 9 \end{bmatrix}$$

hidden 2 with sigmoid applied.

$$= \begin{bmatrix} (1 / (1 + e^{-1.157857369})) \\ (1 - 1/(1 + e^{-0.475698738 9})) \end{bmatrix}$$

$$= \begin{bmatrix} 0.760943169 6 \\ 0.383268318 5 \end{bmatrix}$$

outputs = weights_ho . dotproduct (hidden2)

$$= \begin{bmatrix} -0.88 \\ 0.63 \end{bmatrix} \cdot \begin{bmatrix} 0.7609431696 \\ 0.3832683185 \end{bmatrix}$$

$$= \left[(-0.88 \times 0.7609431696) + (0.63 \times 0.3832683185)\right]$$

$$= \left[-0.6696249892 + 0.2414590407\right]$$

$$= \left[-0.4281709485\right]$$

outputs = outputs + bias_o $= \left[-0.4281709485\right] + \left[0.84\right]$

$$= 0.4118290515$$

outputs' with sigmoid applied.

$$= \left[\left(1 \;/\; \left(1 + e^{-0.4118290515}\right)\right)\right]$$

$$= 0.6015263699$$

output_errors = targets - outputs

$$= \left[1\right] - \left[0.6015263699\right]$$

$$= \left[0.3984736301\right]$$

gradients = outputs with dsigmoid applied.

$= 0.3984736301 \times (1 - 0.3984736301)$

$= 0.2396923962$

gradients = gradients × output errors.

$= 0.2396923962 \times 0.3984736311$

$= 0.09551109944$

gradients = gradients × learning rate

$= 0.09551109944 \times 0.01$

$= 0.0009551109944$

hidden_2 transposed $= [0.7609431696, 0.3832683185]$

weight_ho deltas = gradients.dotproduct(hidden2_transposed)

$= [0.0009551109944] \cdot [0.7609431696, 0.3832683185]$

$= [(0.0009551109944 \times 0.7609431696), (0.000955... \times 0.38326...)]$

$= [0.0007267851874, 0.0003660637848]$

Update weights_ho and bias_o

weights_ho = weights_ho + weight_ho_deltas

$= [-0.88, 0.63] + [0.0007267851874, 0.0003660637848]$

$= [-0.8792732148, 0.6303660638]$

Update bias_o

bias_o + gradients = [0.84] + [0.000955110944]

= 0.84095511

weights_ho transposed = $\begin{bmatrix} -0.879273214 \\ 0.630366063 \end{bmatrix}$

hidden 2 errors = weights_ho transposed . dotproduct(output errors)

= $\begin{bmatrix} -0.879273214 \\ 0.630366063 \end{bmatrix} \cdot [0.398473631]$

= $\begin{bmatrix} (-0.879273214 \times 0.398473631) \\ (0.630366063 \times 0.398473631) \end{bmatrix}$

= $\begin{bmatrix} -0.350367189 \\ 0.251184253 \end{bmatrix}$

hidden 2 gradient = hidden 2 with dsigmoid applied.

= $\begin{bmatrix} 0.760943169 \times (1 - 0.760943169) \\ 0.383268318 \times (1 - 0.383268318) \end{bmatrix}$

= $\begin{bmatrix} 0.181908662 \\ 0.236373714 \end{bmatrix}$

hidden2 gradient × hidden2 errors

$$= \begin{bmatrix} 0.1819086622 \\ 0.2363737145 \end{bmatrix} \times \begin{bmatrix} -0.3503671898 \\ 0.2511842537 \end{bmatrix}$$

$$= \begin{bmatrix} -0.0637348267\overline{8} \\ 0.0593733550\overline{7} \end{bmatrix}$$

hidden2 gradient × learning rate

$$= \begin{bmatrix} -0.0637348267\overline{8} \\ 0.0593733550\overline{7} \end{bmatrix} \times 0.01$$

$$= \begin{bmatrix} -0.0006373482678 \\ 0.0005937335507 \end{bmatrix}$$

hidden1 transposed = $[0.6856801139, 0.5349429452]$

weight_h1h2_deltas = hidden2_gradient.dotproduct(hidden1_transposed)

$$= \begin{bmatrix} -0.0006373482678 \\ 0.0005937335507 \end{bmatrix} \bullet [0.6856801139, 0.5349429452]$$

$$= \begin{bmatrix} 0.000637\ldots \times 0.6856\ldots, & -0.000637\ldots \times 0.53494\ldots \\ 0.000593\ldots \times 0.6856\ldots, & 0.000593\ldots \times 0.53494\ldots \end{bmatrix}$$

$$= \begin{bmatrix} -0.0004370170329, & -0.0003409449595 \\ 0.0004071112887, & 0.0003176135743 \end{bmatrix}$$

update weights_hlh2

$= weights\_hlh2 + weight\_hlh2\_deltas$

$= \begin{bmatrix} 0.72 & -0.31 \\ -0.69 & -0.36 \end{bmatrix} + \begin{bmatrix} -0.000437\ldots & -0.0003409\ldots \\ 0.000407\ldots & 0.0003176\ldots \end{bmatrix}$

$= \begin{bmatrix} 0.719562983 & -0.3103409\,45 \\ -0.689592\,8887 & -0.3596823864 \end{bmatrix}$

update bias_h2

$= bias\_h2 + hidden2\_gradient$

$= \begin{bmatrix} 0.83 \\ 0.19 \end{bmatrix} + \begin{bmatrix} -0.000637348\,2678 \\ 0.000593733\,5507 \end{bmatrix}$

$= \begin{bmatrix} 0.829362651\,7 \\ 0.190593733\,6 \end{bmatrix}$

$weights\_hlh2\ transposed = \begin{bmatrix} 0.719562983 & -0.689592\,8887 \\ -0.3103409\,45 & -0.3596823864 \end{bmatrix}$

$hiddenerrors = weights\_hlh2\_transposed.dotproduct(hidden2\_errors)$

$= \begin{bmatrix} 0.719562983 & -0.689592\,8887 \\ -0.3103409\,45 & -0.3596823864 \end{bmatrix} \cdot \begin{bmatrix} -0.3503671898 \\ 0.2511842537 \end{bmatrix}$

$= \begin{bmatrix} (0.719\ldots \times -0.3503\ldots) + (-0.6895\ldots \times 0.2511\ldots) \\ (-0.3103\ldots \times -0.3503\ldots) + (-0.3596\ldots \times 0.2511\ldots) \end{bmatrix}$

$= \begin{bmatrix} -0.252111\,2602 + -0.173214\,8751 \\ 0.108733\,2848 \quad -0.090346\,5518 \end{bmatrix}$

$= \begin{bmatrix} -0.425326\,1353 \\ 0.018386\,733 \end{bmatrix}$

hidden_gradient = hidden with dsigmoid applied

$$= \begin{bmatrix} 0.685680\,1139 \times (1 - 0.685680\,1139) \\ 0.534942\,9452 \times (1 - 0.534942\,9452) \end{bmatrix}$$

$$= \begin{bmatrix} 0.2155228953 \\ 0.2487789906 \end{bmatrix}$$

hidden_gradient × hidden_errors

$$= \begin{bmatrix} 0.2155228953 \\ 0.2487789906 \end{bmatrix} \times \begin{bmatrix} -0.4253261353 \\ 0.01830\,6733 \end{bmatrix}$$

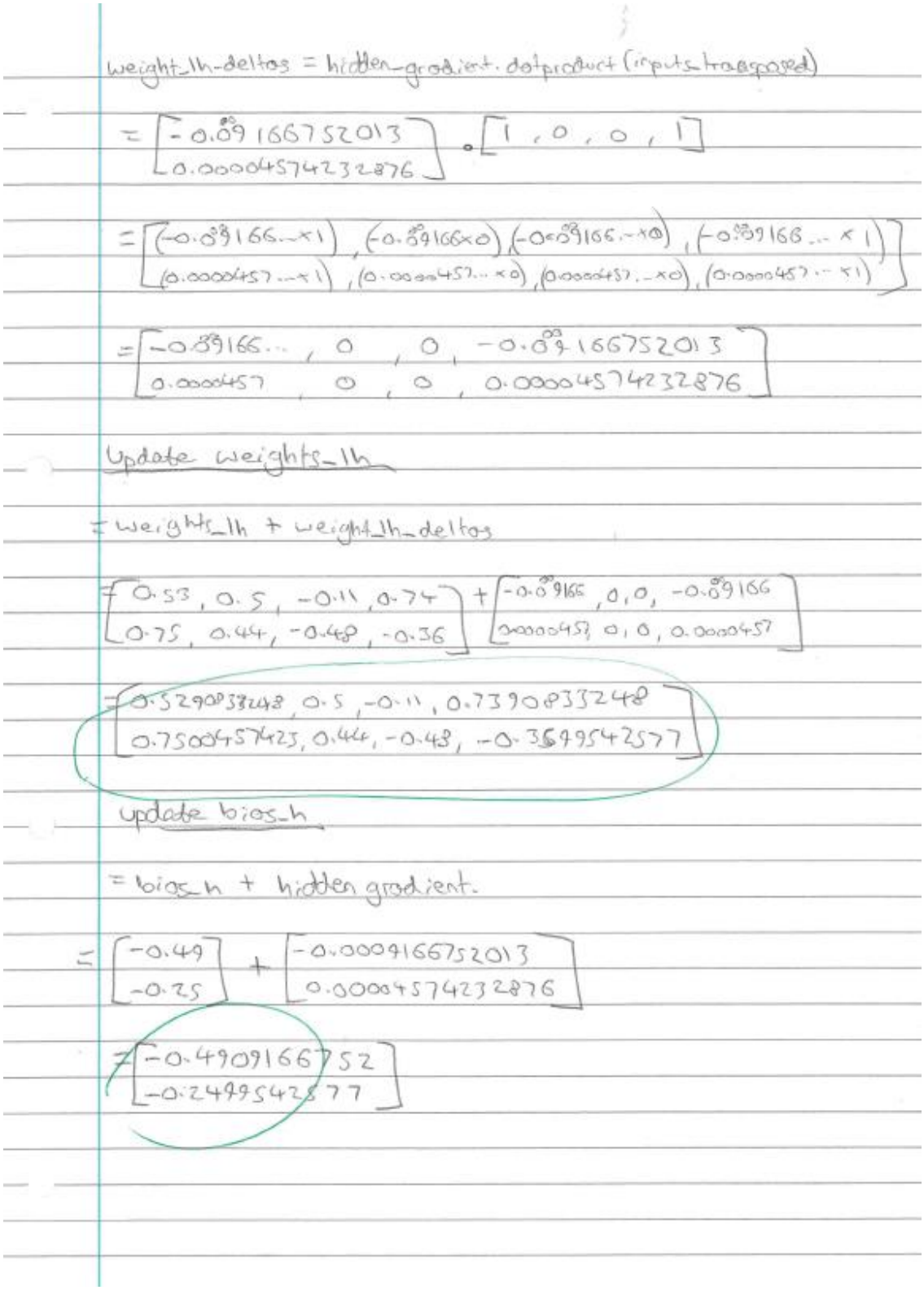
$$= \begin{bmatrix} -0.0916675201\,3 \\ 0.0045742328\,76 \end{bmatrix}$$

hidden_gradient × 0.01

$$= \begin{bmatrix} -0.0916675201\,3 \\ 0.0045742328\,76 \end{bmatrix} \times 0.01)$$

$$= \begin{bmatrix} -0.0009166752013 \\ 0.0000457423276 \end{bmatrix}$$

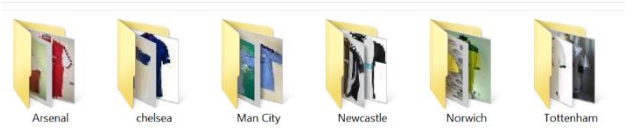
Inputs transposed = $[1, 0, 0, 1]$



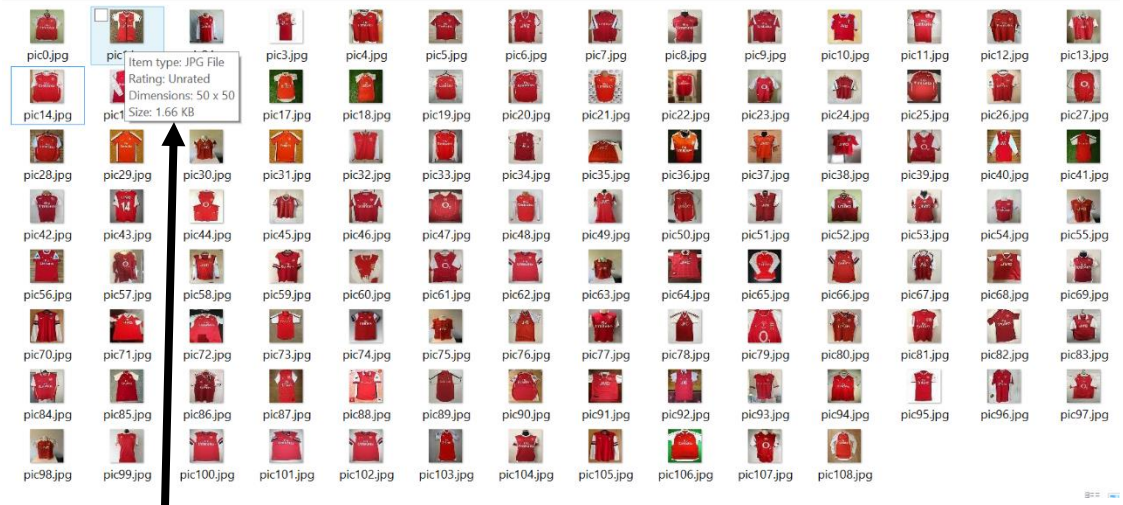
Here we can see that the weights produced after training the program were the same as the weights after training that I calculated (circled in green). Hence, showing that the back-propagation algorithm is working as it should.

Images for training data:

Here I am running the neuralNetwork class to train the network to work for 6 different shirts: Chelsea, Arsenal, Norwich, Man City, Tottenham and Newcastle.



For each club I have a folder, called the club name, with around 100 images of the shirt in it



This is the folder of the Arsenal images. Also as you can see each image has been resized to 50 x 50 pixels.

The code I used for the training:

Here is how I made the training_inputs array. Each picture in the folder for each shirt is converted to an array of 7500 numbers, and that array is added to the training_inputs array.

All of what I mention here will be covered in more detail in the video at the end

```
1.  training_inputs = []
2.  for I in range(0, 113):
3.    training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Leve
    l/Nea/133ewcast/pic" + str(i) + ".jpg").toarray())
4.  for I in range(0,108):
5.    training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Leve
    l/Nea/arsenal/pic" + str(i) + ".jpg").toarray())
6.  for I in range(0,59):
7.    training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Leve
    l/Nea/Norwich/pic" + str(i) + ".jpg").toarray())
8.  for I in range(0,106):
9.    training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Leve
    l/Nea/mancity/pic" + str(i) + ".jpg").toarray())
10. for I in range(0,87):
11.   training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Leve
    l/Nea/133ewcastle/pic" + str(i) + ".jpg").toarray())
12. for I in range(0,123):
13.   training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Leve
    l/Nea/133ewcastle/pic" + str(i) + ".jpg").toarray())
14.
15. training_targets = []
16. for I in range(0,113):
17.   training_targets.append([1,0,0,0,0,0])#chelsea
18. for I in range(0,108):
19.   training_targets.append([0,1,0,0,0,0])#arsenal
20. for I in range(0,59):
21.   training_targets.append([0,0,1,0,0,0])#norwich
22. for I in range(0,106):
23.   training_targets.append([0,0,0,1,0,0])#man city
24. for I in range(0,87):
25.   training_targets.append([0,0,0,0,1,0])#tottenham
26. for I in range(0,123):
27.   training_targets.append([0,0,0,0,0,1])#newcastle
28.
29. nn = neuralNetwork(7500,200,20,6)
```

Here is how I made the training_targets. There are 6 possible outputs (1 for each shirt), so ive assigned an output for each shirt. The number of outputs for each shirt is the same as the number of shirts for the training data. The targets for the shirts are added in the same order as the inputs so that they match up.

Instantiate neuralNetwork class with 7500 input nodes, 200 hidden layer 1 nodes, 20 hidden layer 2 nodes and 6 output nodes

For each club I have 10 test images I got from google, these are used to test the neural network while it is training and the results are outputted to show me how it is doing.

```
1.  for I in range(0,1):
2.    inputs = random.choice(training_inputs)
3.    index = training_inputs.index(inputs)
4.    target = training_targets[index]
5.    nn.train(inputs,target)
6.    print “trained”
```

This is the code I used first for training, using the train sub routine which then writes the weights to the file

I then ran the code below twice, so 2000 iterations of training, printing the outputs of the test images every 100 iterations

```
1.  ind = 0
2.  for I in range(0,1000):
3.    ind +=1
4.    inputs = random.choice(training_inputs)
5.    index = training_inputs.index(inputs)
6.    target = training_targets[index]
7.    nn.train_with_existing_weights(inputs,target,nn)
8.    print “trained” + str(ind)
```

```
    if ind == 100 or ind == 200 or ind == 300 or ind == 400 or ind == 500 or ind == 6
00 or ind == 700 or ind == 800 or ind == 900 or ind == 1000:
        print "135ewcast = [1,0,0,0,0,0]"
        for I in range(1,11):
            c = nn.run_with_existing_weights(Images("chelseatest" + str(i) + ".jpg").
toarray()).matrix
            print softmaxtrainI
        print "arsenal = [0,1,0,0,0,0]"
        for I in range(1,11):
            a = nn.run_with_existing_weights(Images("arsenaltest" + str(i) + ".jpg").
toarray()).matrix
            print softmaxtrain(a)
        print "135ewcast = [0,0,1,0,0,0]"
        for I in range(1,11):
            nor = nn.run_with_existing_weights(Images("norwichtest" + str(i) + ".jpg"
).toarray()).matrix
            print softmaxtrain(nor)
        print "man city = [0,0,0,1,0,0]"
        for I in range(1,11):
            man = nn.run_with_existing_weights(Images("mancitytest" + str(i) + ".jpg"
).toarray()).matrix
            print softmaxtrain(man)
        print "135ewcastle = [0,0,0,0,1,0]"
        for I in range(1,11):
            w = nn.run_with_existing_weights(Images("135ewcastle" + str(i) + ".jpg").
toarray()).matrix
            print softmaxtrain(w)
        print "135ewcastle = [0,0,0,0,0,1]"
        for I in range(1,11):
            new = nn.run_with_existing_weights(Images("135ewcastle" + str(i) + ".jpg"
).toarray()).matrix
            print softmaxtrain(new)
```

I then ran the below code for 10 iterations, which works out how many of the test images gave the correct output. If all 6 clubs had 70% or more correct test images then the program would end and that file would be stored

```
ind = 0

for I in range(0,10):
  ind +=1
  inputs = random.choice(training_inputs)
  index = training_inputs.index(inputs)
  target = training_targets[index]
  nn.train_with_existing_weights(inputs,target)
  print "trained" + str(ind)

  cc = 0
  print "135ewcast = [1,0,0,0,0,0]"
  for I in range(1,11):
    135ewcast = nn.run_with_existing_weights(Images("chelseatest" + str(i) + ".jpg"
).toarray()).matrix
    cindex = 0
    chigh = 0
    chelseaout = softmaxtrain(135ewcast)
    print chelseaout
    for I in range(0,len(chelseaout)):
      if chelseaout[i] > chigh:
```

```
        chigh = chelseaout[i]
        cindex= I
    if cindex == 0:
      cc+=1
  print cc

  ac = 0
  print "arsenal = [0,1,0,0,0,0]"
  for I in range(1,11):
    arsenal = nn.run_with_existing_weights(Images("arsenaltest" + str(i) + ".jpg").
toarray()).matrix
    aindex = 0
    ahigh = 0
    arsenalout = softmaxtrain(arsenal)
    print arsenalout
    for I in range(0,len(arsenalout)):
      if arsenalout[i] > ahigh:
        ahigh = arsenalout[i]
        aindex= I
    if aindex == 1:
      ac+=1
  print ac

  nc = 0
  print "136ewcast = [0,0,1,0,0,0]"
  for I in range(1,11):
    136ewcast = nn.run_with_existing_weights(Images("norwichtest" + str(i) + ".jpg"
).toarray()).matrix
    nindex = 0
    nhigh = 0
    norwichout = softmaxtrain(136ewcast)
    print norwichout
    for I in range(0,len(norwichout)):
      if norwichout[i] > nhigh:
        nhigh = norwichout[i]
        nindex= I
    if nindex == 2:
      nc+=1
  print nc


  mc = 0
  print "man city = [0,0,0,1,0,0]"
  for I in range(1,11):
    mancity = nn.run_with_existing_weights(Images("mancitytest" + str(i) + ".jpg").
toarray()).matrix
    mcindex = 0
    mchigh = 0
    mcout = softmaxtrain(mancity)
    print mcout
    for I in range(0,len(mcout)):
      if mcout[i] > mchigh:
        mchigh = mcout[i]
        mcindex= I
    if mcindex == 3:
      mc+=1
  print mc

  tc = 0
  print "136ewcastle = [0,0,0,0,1,0]"
  for I in range(1,11):
    136ewcastle = nn.run_with_existing_weights(Images("136ewcastle" + str(i) + ".jp
g").toarray()).matrix
    tcindex = 0
    tchigh = 0
    tcout = softmaxtrain(136ewcastle)
```

```
    print tcout
    for I in range(0,len(tcout)):
      if tcout[i] > tchigh:
        tchigh = tcout[i]
        tcindex= I
    if tcindex == 4:
      tc+=1
  print tc

  nec = 0
  print "137ewcastle = [0,0,0,0,0,1]"
  for I in range(1,11):
    137ewcastle = nn.run_with_existing_weights(Images("137ewcastle" + str(i) + ".jpg").toarray()).matrix
    neindex = 0
    nehigh = 0
    neout = softmaxtrain(137ewcastle)
    print neout
           for I in range(0,len(neout)):
             if neout[i] > nehigh:
               nehigh = neout[i]
               neindex= I
           if neindex == 5:
             nec+=1
         print nec

         if cc > 6 and ac > 6 and nc > 6 and mc > 6 and tc > 6 and nec > 6:
             print "70% correct for each"
             sys.exit()
```

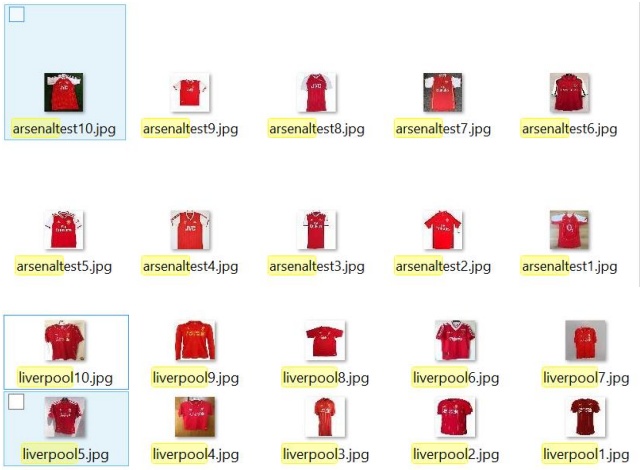
Video for neural network training

I now have a video showing what happened when I trained the neural network, here I go into more detail about the code I used and show how the outputs of the test images changed throughout the training, plus a part at the end on showing the neural network working for same colour shirts (below):

https://www.youtube.com/watch?v=j5UU94mfsq8

Showing the neural network works for the same colour shirts

For my eBay listing program, it could be thought that the neural network only works for different colour shirts. However, here I disprove this and show that the neural network can work out the difference between two red shirts (Arsenal and Liverpool), this is also shown in the video of training the neural network just above.




```
arsenal = [1,0]
[0.5011227572754343, 0.4988772427245657]
[0.456565183725492, 0.543434816274508]
[0.5553731967980587, 0.44462680320194115]
[0.5230678077749793, 0.4769321922250207]
[0.5232899843353452, 0.4767100156646547]
[0.4766352767077701, 0.5233647232922298]
[0.5037920839133349, 0.49620791608666515]
[0.5850219264224625, 0.4149780735775375]
[0.5266341263736695, 0.4733658736263306]
[0.40873939244985386, 0.5912606075501461]
liverpool = [0,1]
[0.5207571056369625, 0.4792428943630375]
[0.3887033733149898, 0.6112966266850103]
[0.5847299006644576, 0.4152700993355424]
[0.39594049782025303, 0.604059502179747]
[0.4985461599806872, 0.5014538400193128]
[0.42287245505300924, 0.5771275449469907]
[0.534813339143362, 0.465186660856638]
[0.4369831659228794, 0.5630168340771206]
[0.4552053919498304, 0.5447946080501695]
[0.40090614432456184, 0.5990938556754382]
```

Test images

Here this shows that 70% of the test images have correct outputs so it is working for telling the difference between Arsenal and Liverpool shirts

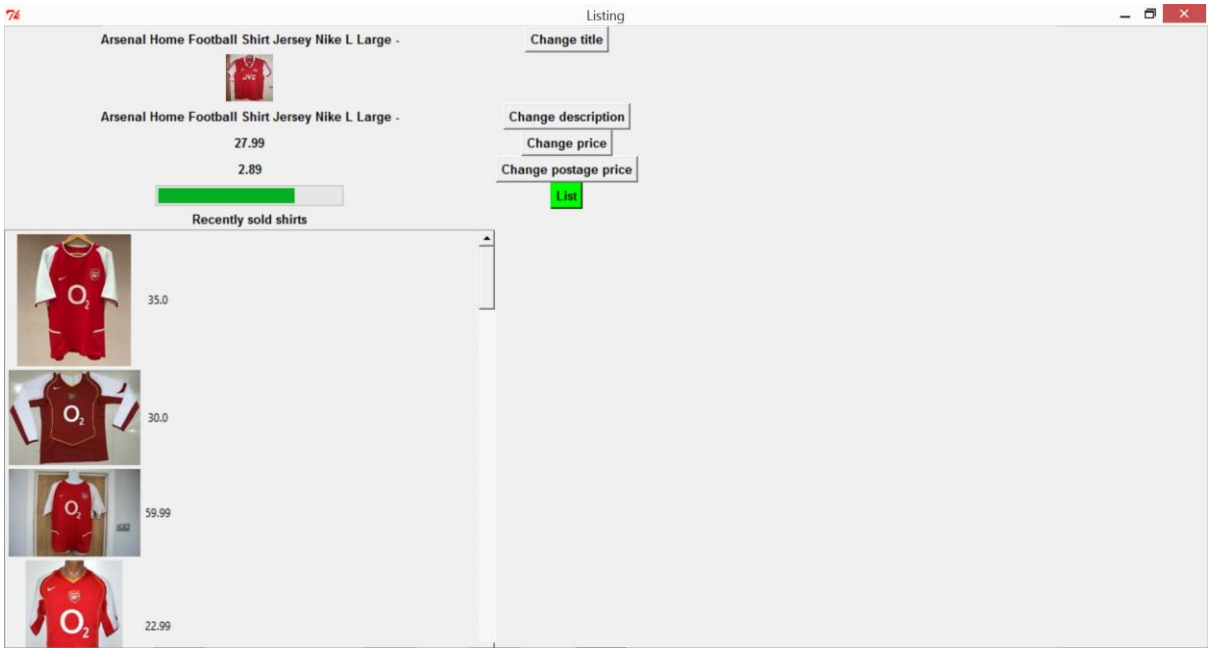
Output

Showing the eBay API connection works correctly

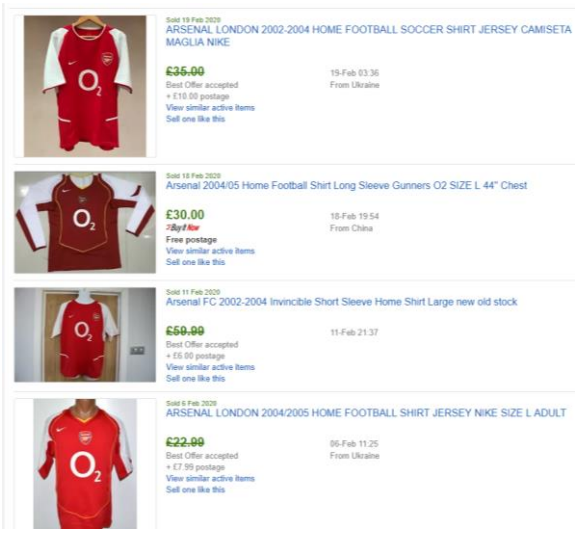
Images

Firstly, this shows the images and prices my program finds compared with what I find when searching manually on the eBay website.

My program:

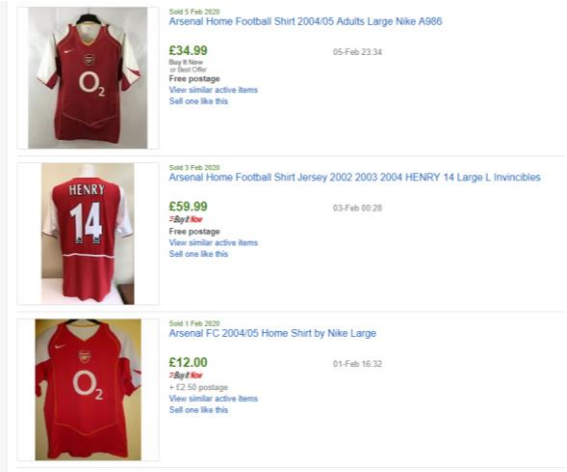


Actual eBay:

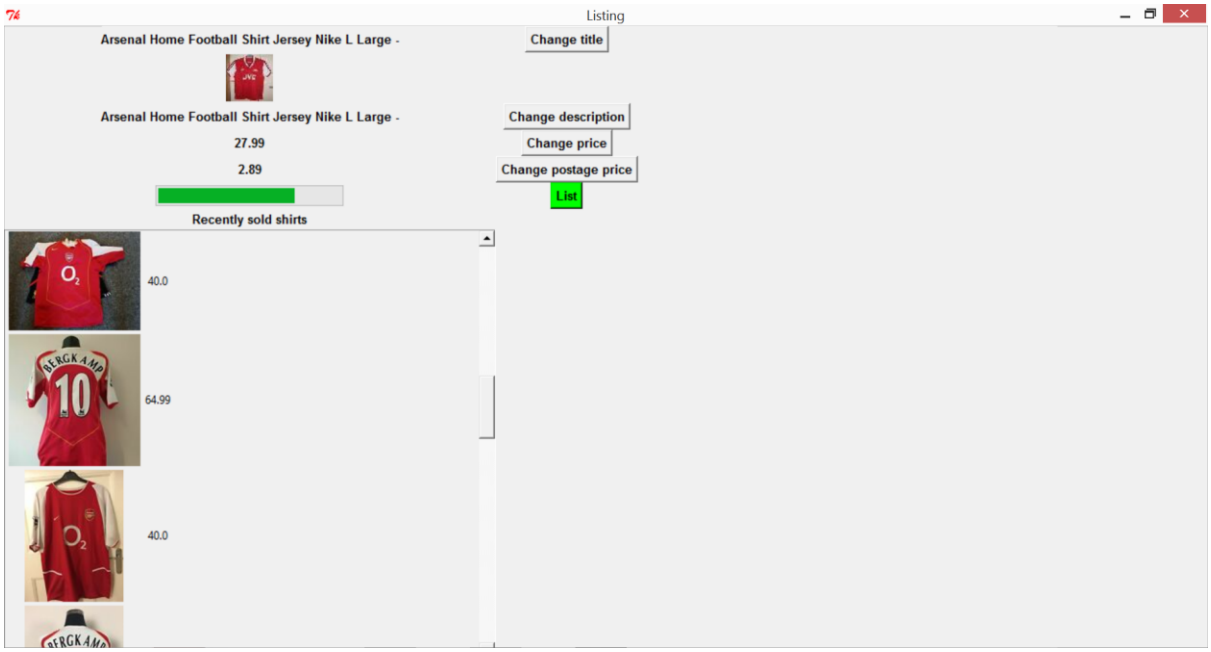


My program:

Actual eBay:

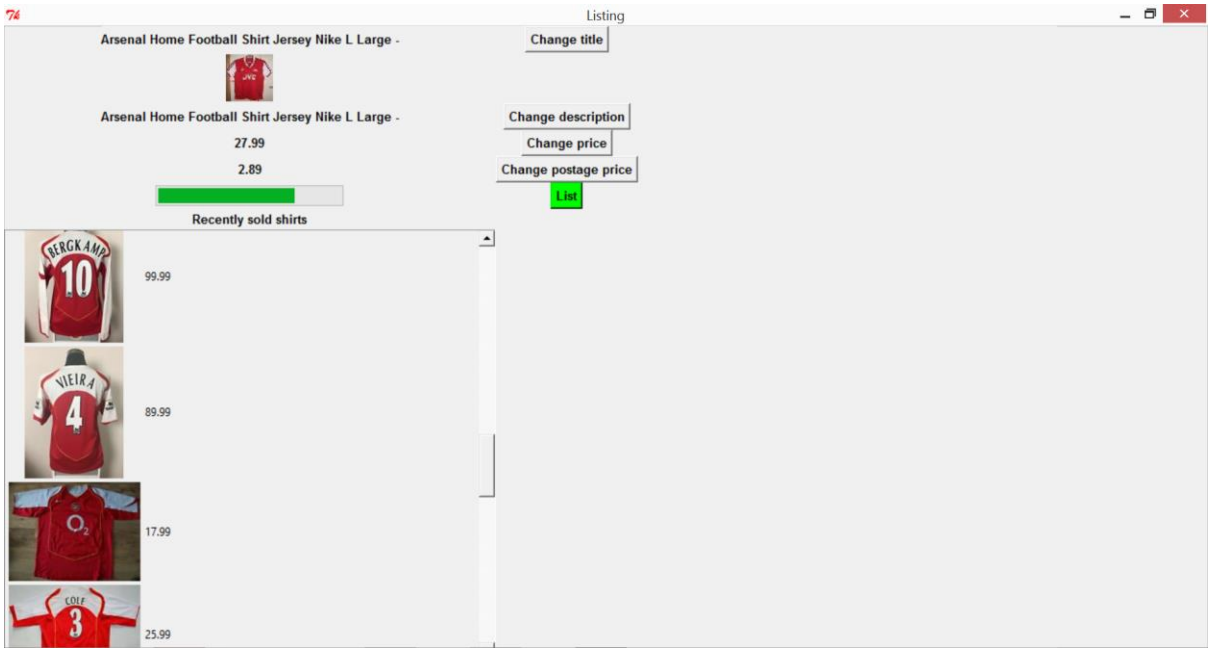


My program:

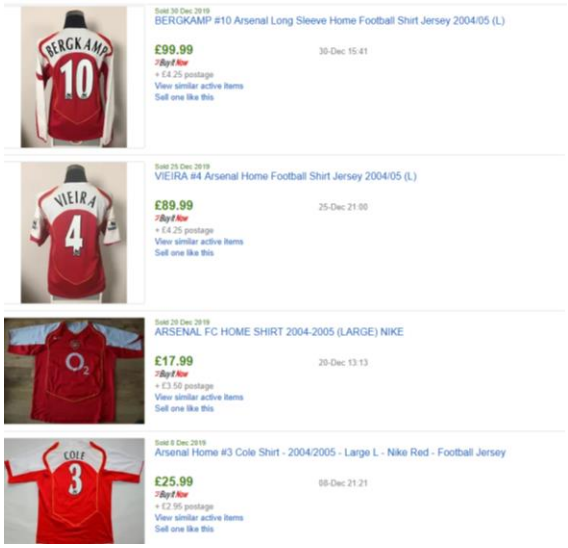


Actual eBay:

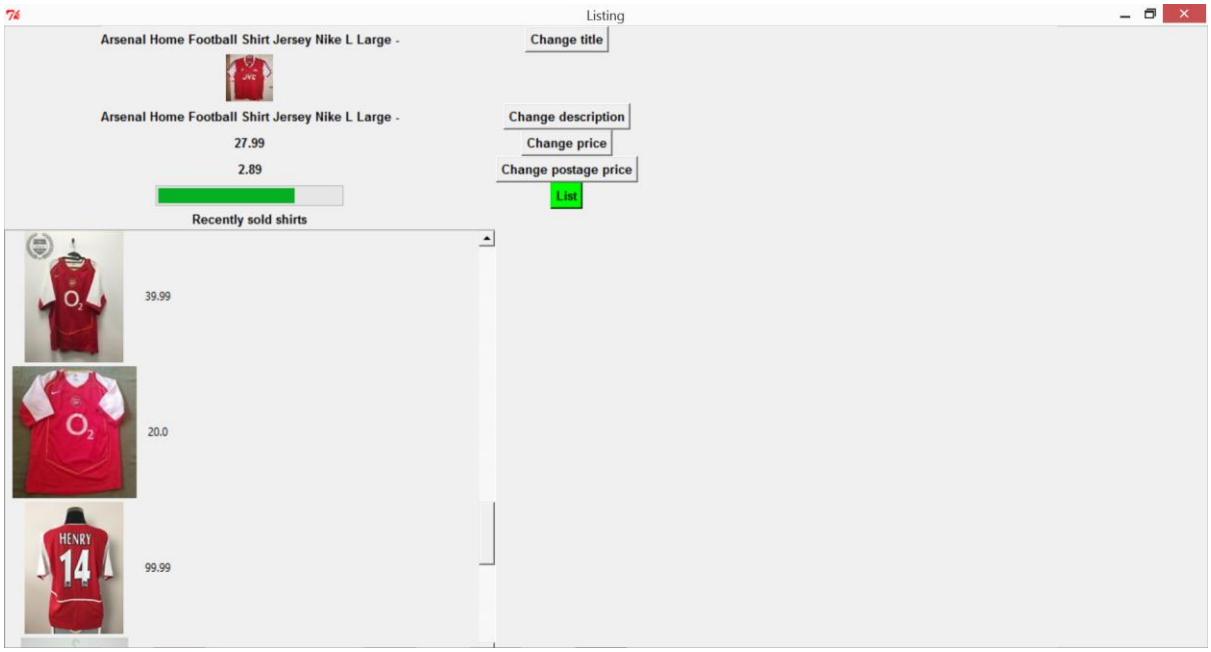
My program:



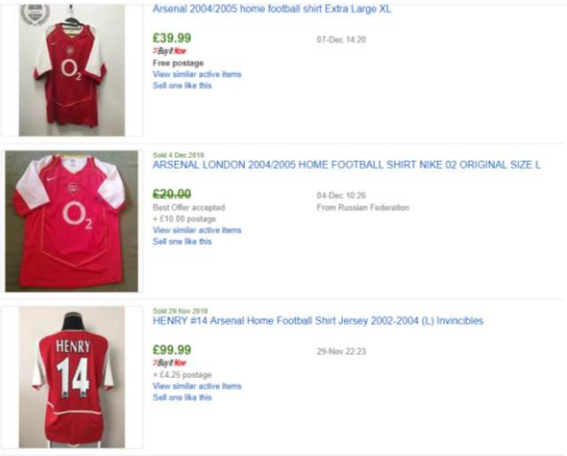
Actual eBay:



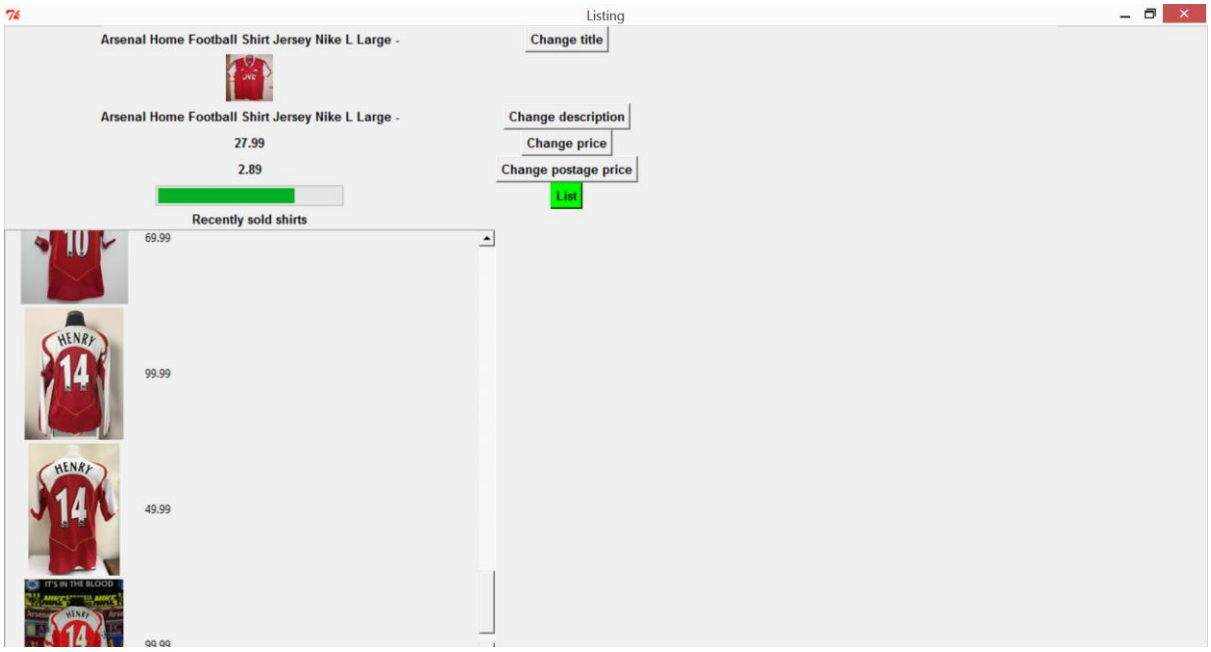
My program:



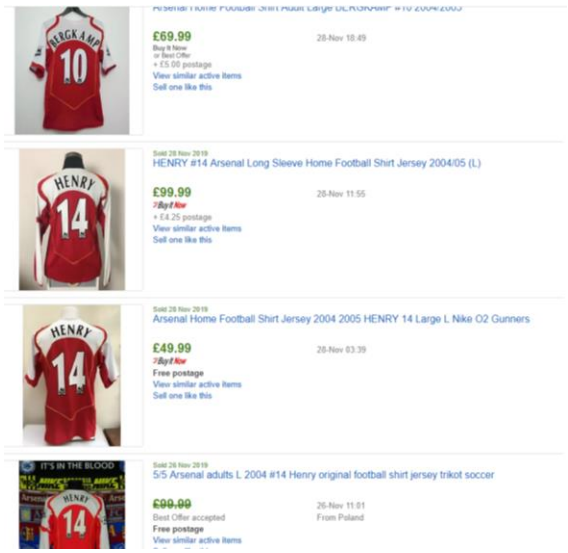
Actual eBay:



My program:



Actual eBay:



eBay.co.uk 3

(eBay, n.d.)

This shows that the images and prices my program pulls out match up with the actual images and prices on eBay, therefore the API is working correctly.

Checking the title

Here I go through the titles from the actual eBay site and make a tally for each separate word, picking out the most common words, making my own title and comparing it with the title the program produces.

The code I use to create a suggested title:

```
try:
        json_format_of_listings = apiresult.json()
        array_for_prices =[]
        array_for_images =[]
        array_for_titles =[]
        for item in (json_format_of_listings["findCompletedItemsResponse"][0]["searchResult"][0]["item"]):
            picture_of_listing = item["galleryURL"][0]
            ebay_title = item["title"][0]
            ebay_title_split = ebay_title.split()
            array_for_titles.append(ebay_title)
            array_for_images.append(picture_of_listing)
            price_of_shirt = item['sellingStatus'][0]["convertedCurrentPrice"][0]['__value__']
            array_for_prices.append(price_of_shirt)
        window.destroy()
        price = calculate_price(array_for_prices, year)
        words = []
        word_array = []
        word_count_array = []
        if len(array_for_titles) > 0:
            for title in range(0,len(array_for_titles)):
                split_title = array_for_titles[title].split()
                for word in split_title:
                    word = word.lower()
                    words.append(word)
            for word in words:
                if len(word_array) == 0:
                    word_array.append(word)
                    word_count_array.append(1)
                else:
                    if word in word_array:
                        index = word_array.index(word)
                        word_count_array[index] = word_count_array[index] + 1
                    else:
                        word_array.append(word)
                        word_count_array.append(1)
            title_words = []
            index = 0
            for num in word_count_array:
                if num >= len(array_for_titles)/2.75:
                    title_words.append(word_array[index])
                index +=1
            for title_word in range(0,len(title_words)):
                title_words[title_word] = title_words[title_word].capitalize()
            title =  " ".join(title_words)
        else:
            title = club_name + " Football Soccer Home Shirt Year " + year + " Size UK " + size_actual + " Good Condition"
```

1

2

3

4

Default title

These text boxes match up with the numbers above

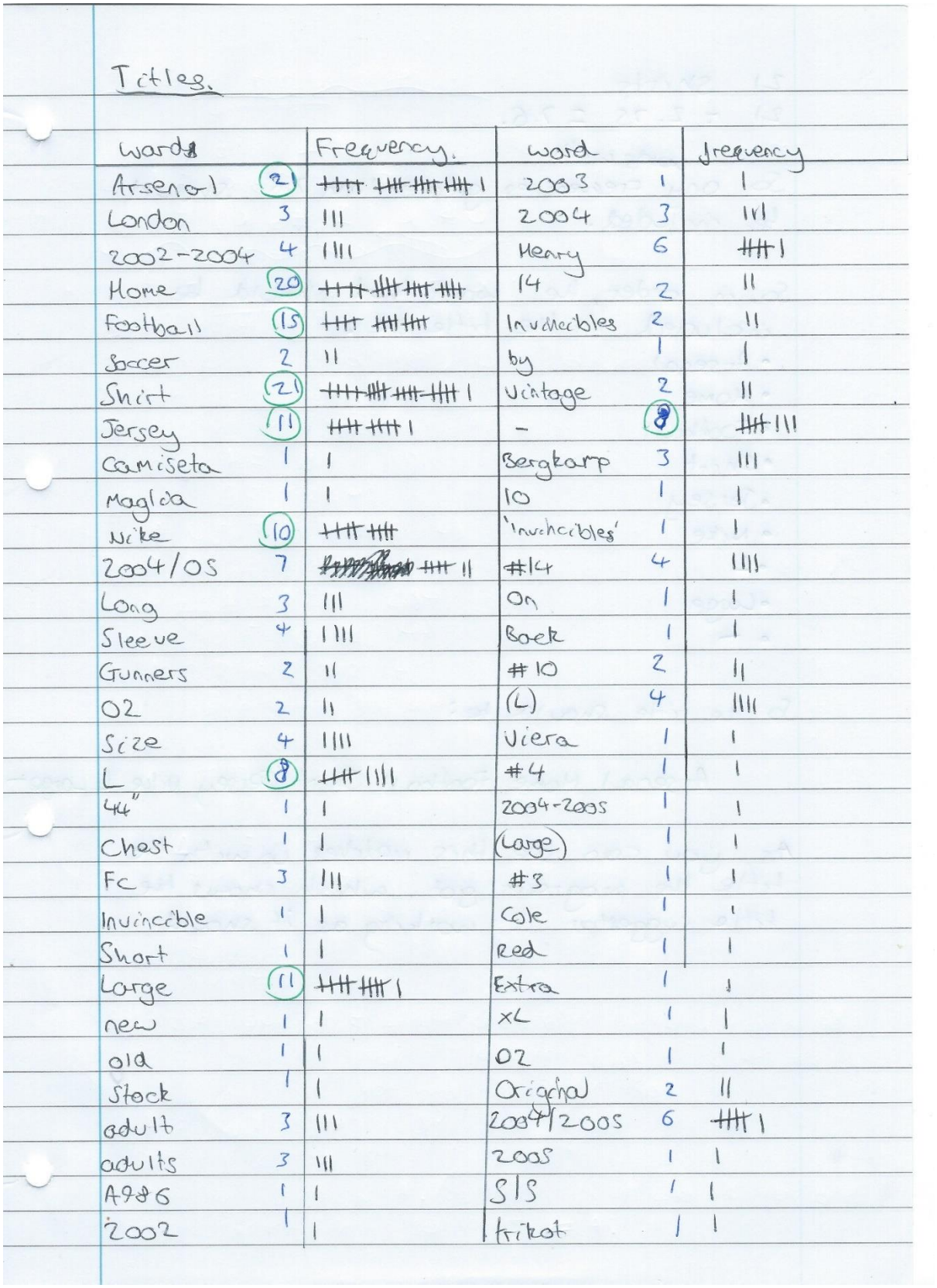
1. For each item on eBay, add the title to the array_for_titles

2. Split all titles into singular words and add all those words to the words array

3. Have an array for words called word_array and a separate array for the number of occurrences of each word called word_count _array. Go through each word in the words array and either add it to the words_array or increase the correct index of the words_count_array by 1.

4. Pick out the most common words and add them to title_words array, then go through the title_words array and combine all the words into one string and that is the title

How I worked out my title:



Titles.

Words		Frequency.	Word		Frequency
Arsenal	(21)	𝍸 𝍸 𝍸 𝍸 I	2003	1	I
London	3	III	2004	3	III
2002-2004	4	IIII	Henry	6	𝍸 I
Home	(20)	𝍸 𝍸 𝍸 𝍸	14	2	II
Football	(15)	𝍸 𝍸	Invincibles	2	II
Soccer	2	II	by	1	I
Shirt	(21)	𝍸 𝍸 𝍸 𝍸 I	Vintage	2	II
Jersey	(11)	𝍸 𝍸 I	-	(8)	𝍸 III
Camiseta	1	I	Bergkamp	3	III
Maglia	1	I	10	1	I
Nike	(10)	𝍸 𝍸	'Invincibles'	1	I
2004/05	7	~~𝍸 𝍸~~ 𝍸 II	#14	4	IIII
Long	3	III	On	1	I
Sleeve	4	IIII	Back	1	I
Gunners	2	II	#10	2	II
02	2	II	(L)	4	IIII
Size	4	IIII	Viera	1	I
L	(8)	𝍸 IIII	#4	1	I
44"	1	I	2004-2005	1	I
Chest	1	I	(Large)	1	I
Fc	3	III	#3	1	I
Invincible	1	I	Cole	1	I
Short	1	I	Red	1	I
Large	(11)	𝍸 𝍸 I	Extra	1	I
new	1	I	XL	1	I
old	1	I	02	1	I
Stock	1	I	Original	2	II
adult	3	III	2004/2005	6	𝍸 I
adults	3	III	2005	1	I
A986	1	I	SIS	1	I
2002	1	I	Trikot	1	I

21 shirts
21 ÷ 2.75 = 7.6.
words with
So only frequencies greater than 7.6 should be included.

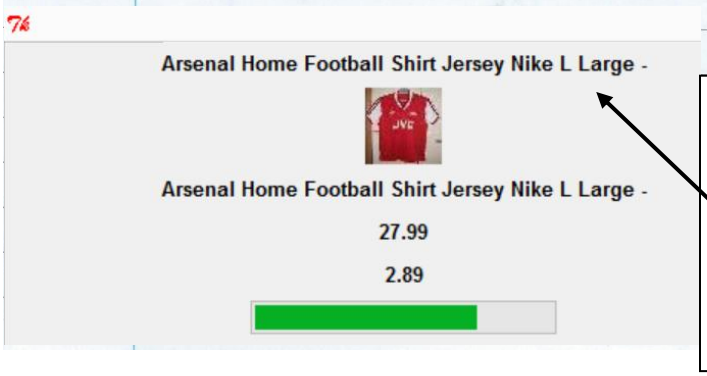
So, in order, the words that should be included in the title are:

- Arsenal
- Home
- Football
- Shirt
- Jersey
- Nike
- L
- Large
- —

So the title should be:

Arsenal Home Football Shirt Jersey Nike L Large –

As you can see this matches up with the title the program got, which shows the title suggestor is working as it should.



So here we can see that the title my program produces is the same as the title I got, which shows that the title suggesting section of code is working as it should.

Checking the price

Here I go through the prices from the actual eBay site from the listing pulled out by my program shown in the Images section just above. I take out the outliers and work out the price.

Price.

1 Prices:

35, 30, 59.99, 22.99, 34.99, 59.99, 12,
40, 64.99, 40, 99.99, 89.99, 17.99, 25.99, 39.99,
20, 99.99, 69.99, 99.99, 49.99, 99.99.

In order:

12, 17.99, 20, 22.99, 25.99, 30, 34.99, 35,
39.99, 40, 40, 49.99, 59.99, 59.99, 64.99,
69.99, 89.99, 99.99, 99.99, 99.99, 99.99.

LQ = 21/4 = 5.25 ↑ = 6.
So LQ = 30

UQ position = (21/4) × 3 = 15.75 ↑ = 16
So UQ = 69.99

IQR = 69.99 − 30
= 39.99

Year < 2010 so difference = 0.2 × IQR
So difference = 7.998

Lower bound for outliers = 30 − 7.998 =
= 22.002

Upper bound for outliers = 69.99 + 7.998
= 77.988

So any prices below 22.002:
12, 17.99, 20.

Any prices above 77.988:
89.99, 99.99, 99.99, 99.99, 99.99

New prices with outliers removed.
22.99, 25.99, 30, 34.99, 35, 39.99, 40, 40, 44.99,
59.99, 59.99, 64.99, 69.99

Sum of new prices = £73.41

£73.41 ÷ 21 = 27.33

27.33 rounded up to nearest integer = 28

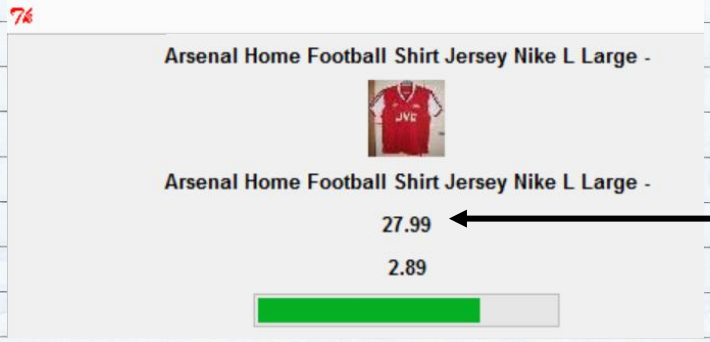
28 − 0.01 = 27.99

So the price I got is 27.99

Therefore, this shows the program correctly identifies outliers because the price I got is the same as the price the program got.

This also shows that the quicksort algorithm is working as it should because to be able to work out outliers, the array has to be ordered correctly first.

Here it shows that the price I calculated is the same as the price the program got, so the program calculates the price correctly.



Quicksort algorithm

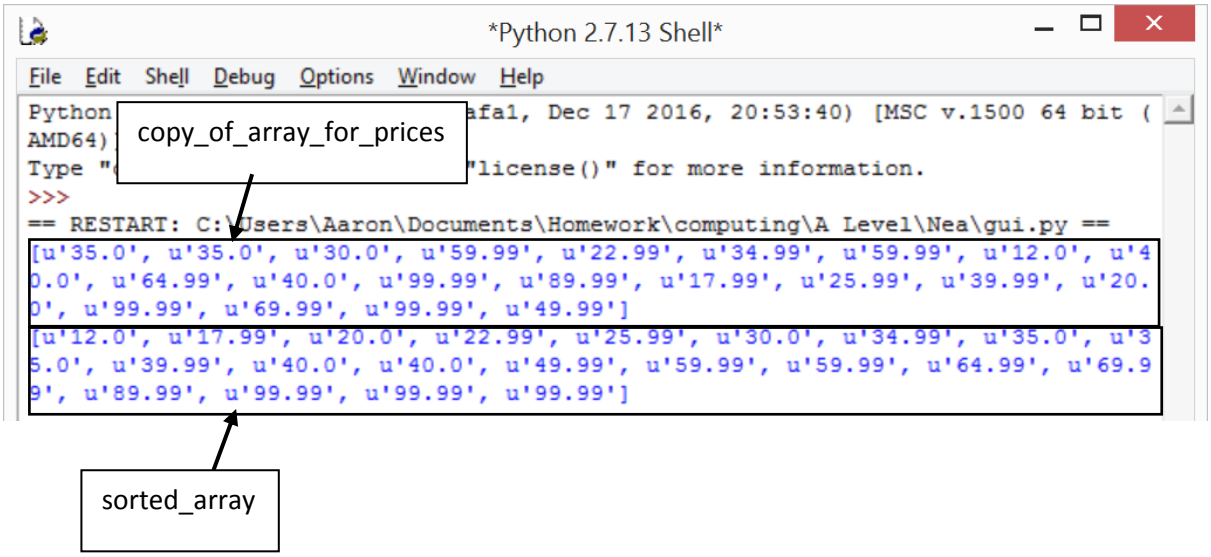
The quicksort algorithm is used to sort the prices of the existing eBay shirts in order, here is the code I used:

```
def quicksort(array, start, end):
    low = start
    high = end
    pivot = array[int((low+high)/2)]
    while low<=high:
        while array[low] < pivot:
            low +=1
        while pivot < array[high]:
            high-=1
        if low <= high:
            temp = array[low]
            array[low] = array[high]
            array[high] = temp
            low+=1
            high-=1
    if start<high:
        quicksort(array, start, high)
    if end > low:
        quicksort(array,low, end)
    return array
```

Here is the code for the calculate_price subroutine:

```
def calculate_price(array_for_prices, year):
    total_price = 0
    copy_of_array_for_prices = []
    for I in range(0,len(array_for_prices)):
        copy_of_array_for_prices.append(array_for_prices[i])
    sorted_array = quicksort(copy_of_array_for_prices, 0, len(copy_of_array_for_pri
ces)-1)
    length_of_array = len(array_for_prices)
    lower_quartile_position = int(math.ceil(length_of_array/4))
    upper_quartile_position = int(math.ceil((length_of_array/4)*3))
    inter_quartile_range = float(sorted_array[int(upper_quartile_position)])-
float(sorted_array[int(lower_quartile_position)])
    if int(year) < 1980:
        difference = 0.05 * inter_quartile_range
    elif int(year) < 2000:
        difference = 0.1* inter_quartile_range
    elif int(year) < 2010:
        difference = 0.2*inter_quartile_range
    elif int(year) < 2015:
        difference = inter_quartile_range
    else:
        difference = 1.5*inter_quartile_range
    lower_bound = float(sorted_array[int(lower_quartile_position)])-difference
    upper_bound = float(sorted_array[int(upper_quartile_position)])+difference
    for I in range(0,len(sorted_array)):
        if float(sorted_array[i]) < lower_bound or float(sorted_array[i])>upper_bou
nd:
            sorted_array[i] = 0
    for I in range(0,len(sorted_array)):
        total_price += float(sorted_array[i])
    price = (math.ceil(total_price / length_of_array))-0.01
    return price
```

To show the quicksort algorithm is working correctly I'm going to print the copy_of_array_for_prices between lines 5 and 6 and then print the sorted_array between lines 6 and 7:



This shows that the quicksort algorithm is working as it should.

Evaluation

Meeting the requirements

Number	Requirement	Met?	Proof
1.1	The program must be able to be used by anyone that wants to list a football shirt on eBay	Yes	Very easy to use
1.2	Must have a good user-friendly interface	Yes	Simple to use and laid out well
1.3	There must be validation to avoid any errors while the program runs	Yes	2 validation videos
2.1	get around 100 images for each club, some clubs may be harder to find lots of images	Yes	Start of run through of code video – 40 seconds
2.2.1	inappropriate images should be deleted from the folders	Yes	Start of run through of code video – 1 min 10 seconds
2.2.2	images should then be renamed in numerical order	Yes	Start of run through of code video – 1 min 20 seconds
3	The neural network should then train	Yes	Training the neural network section in testing and the training the neural network video
3.1	randomly selected images from the training data will be selected along with an output value, which will then be used to train the neural network	Yes	Training the neural network section in testing and the training the neural network video
3.2	the weights from the outcome of the training will be stored in a txt file which will be used later in the program	Yes	Training the neural network section in testing and the training the neural network video
4	User must be able to input an image	Yes	Run through of code video – 3 mins 5 seconds
4.1	the images must be resized to a 50x50 pixel image	Yes	Run through of code video – 3 mins 45 seconds
4.2	the image must then be converted to an array of numbers, 3 numbers representing each pixel in the image	Yes	The images are 50 x 50 pixels, with 3 numbers for each pixel. So 50 x 50 x 3 is the length of the array which is 7500. There are 7500 input nodes for the neural network and as the network has been seen training correctly this show that the image is correctly converted to an array of 7500 elements

4.3	this image must then be able to be passed through the neural network. An output should be displayed of what football club the computer thinks the shirt belongs to using the weights stored in the text file	Yes	Run through of code video – 3 mins 30 seconds
4.4	The user must then be able to confirm whether the computer got the correct club for the picture of the shirt they inputted	Yes	Run through of code video – 3 mins 35 seconds
5.1	The user should input the actual club for the shirt	Yes	Run through of code video – 8 mins 10 seconds
5.2	The neural network should train again using the image inputted by the user, and therefore updating its weights	Yes	Run through of code video – 8 mins 10 seconds
5.3	The program should then continue as it would from point 6	Yes	Run through of code video – 8 mins 45 seconds
6.1.1	Input the year the shirt was from	Yes	Run through of code video – 3 mins 53 seconds
6.1.2	Input their PayPal email	Yes	Run through of code video – 4 mins
6.1.3	Input the size of the shirt	Yes	Run through of code video – 4 mins 10 seconds
6.1.4	Input the weight of the shirt	Yes	Run through of code video – 4 mins 19 seconds
6.2	The program should then use the club, size and the year to search for the item using the eBay API	Yes	Run through of code video – 4 mins 30 seconds and showing the ebay api connection works correctly section
6.3.1	The program should look at currently listed items and sold items to decide on a title for the item	Yes	Run through of code video – 4 mins 30 seconds and checking the title section
6.3.2	Should pull out keywords that occur regularly and format correctly	Yes	Checking the title section
6.4.1	The program should look at currently listed items and sold items to work out a price for the item	Yes	Run through of code video – 4 mins 30 seconds and checking the price section
6.4.2	Any outlier prices should not be used	Yes	Checking the price section
6.4.3	The user should also be able to see recently sold items of their	Yes	Run through of code video – 4 mins 35 seconds and the images section in testing

	particular shirt as requested in the interview in my research		
6.5	The user will then be shown the listing and given the opportunity to make any changes and confirm they are happy for the item to be listed	Yes	Run through of code video – 4 mins 35 seconds
7	User then has option to exit program or list another shirt	Yes	Run through of code video – 5 mins 20 seconds

Possible improvements

From the table above it is shown that all the requirements have been met and the program does everything that it was intended to do, however there are some possible improvements that could be made in the future.

One improvement is that I could make the neural network work for more than 6 different football teams. Currently I have trained the neural network for 6 football teams (Chelsea, Arsenal, Norwich, Man City, Tottenham and Newcastle). To make the program more useful I could train the neural network again for more football teams, but the more teams you have to train the network for, the longer the network takes to train. By getting the neural network to work for the 6 football clubs, this shows that a neural network can work out the difference between different football clubs, and therefore would be able to for more than 6 clubs with further training.

Another improvement I could make is allowing the user to add more than one image. This would enhance the listing for the user and allow buyers to see more details of the listing instead of just one image. This would be fairly simple to implement and would bring benefits to the seller. This however was not in my requirements and is not necessary for an eBay listing, so I have chosen not to add this in. I could also add a feature that allows the user to crop and rotate the images they input, as requested in my interview from my research. This again is also not in my requirements, not required for an eBay listing and is already available on the eBay website, therefore I have not chosen to implement it. Also continuing with images, I could make it possible for users to input images other than JPEGs, such as PNGs. This would make it simpler for the user because they wouldn't have to make sure that the image they have got is JPEG and can just use any image which is more convenient.

Another improvement I could make is the way that the user inputs the year of the shirt. Currently the user inputs one year, however a football season spans across two different years. For example, if you input the year 2004, you will get shirt results for the season 2003/2004 and 2004/2005. This is a problem because there may be different shirts for these two seasons, thus different prices for the two shirts. This means that the price and title are

not as accurate as they could be. However, the user is shown the shirts and prices of them for recently sold shirts, therefore if they believe the title or price is incorrect they can make changes themself before publishing the item on to eBay.

End user's opinion of the solution

I asked the two end users I interviewed in my research what they thought about the program after running and using it:

What did you like about the program?

End user 1 – I liked how it was quick and easy and compared to sold listings

W. Horsley – I liked the interface and found it easy to navigate around. Also, it is very quick and simple to use. It's useful how it saves time of searching for the title and price yourself, and how the program puts the title words together and in a logical order. Also, I like how It asks for all the key words and details that would maximise the number of views you get on the product when it's on eBay.

What didn't you like about the program?

End user 1 – more choice in football shirts and also would have liked to see the format of the email required.

W. Horsley – I would have preferred it if it took up the whole screen, and a greater choice of football shirts would be useful.

Did you find the program easy to use?

End user 1 – Yes

W. Horsley – yes, the progress bar was useful. It went in a logical order and the buttons and headings were clear

Which features did you like the most?

End user 1 – price comparison and where you input a size

W. Horsley – price comparison

How do you think it could be improved?

End user 1 – wider choice of shirts, maybe remove the background of photos

W. Horsley – full screen and more shirts

Did you find the listing process on the program quicker than using eBay online?

End user 1 – yes

W. Horsley – yes because I didn't have to switch between tabs

Here we can see that both end users like the program and found it easy to use. They both can see improvements like introducing more shirts but ultimately, they both found it quicker to list through the program rather than using the eBay website which was the aim of the project.

Bibliography

A.Mehta. (2015). *digitalvidya.com*. Retrieved from https://www.digitalvidya.com/blog/types-of-neural-networks/

ayearofai. (2016). Retrieved from ayearofai.com: https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b

Codeforwin. (2015, July). Retrieved from codeforwin: https://codeforwin.org/2015/07/c-program-to-subtract-two-matrices.html

Codeforwin. (2017). Retrieved from codeforwin: https://codeforwin.org/2017/12/c-program-add-two-matrix-using-pointers.html

Dataaspirant. (2017). Retrieved from DataAspirant: https://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/

developer.ebay.com. (2020).

eBay. (n.d.). Retrieved from eBay.co.uk.

github, H. . (2018). Retrieved from github: https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.2-Multiplying-Matrices-and-Vectors/

guru99. (n.d.). Retrieved from guru99.com: https://www.guru99.com/backpropogation-neural-network.html

Java67.com. (2016). Retrieved from Java67.com: https://www.java67.com/2016/10/how-to-transpose-matrix-in-java-example.html

machinelearningmastery. (2019). Retrieved from machinelearningmastery: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

maths, I. (n.d.). Retrieved from Inquirymaths.com: http://www.inquirymaths.com/home/algebra-prompts/matrices-inquiry

Mathworld. (n.d.). Retrieved from Mathworld.wolfram: https://mathworld.wolfram.com/HyperbolicTangent.html

Regexr.com. (n.d.). Retrieved from https://regexr.com/

Sharma, S. (2017). Retrieved from towardsdatascience: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Towardsdatascience. (2017). Retrieved from Towards data science: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Towardsdatascience. (2017). Retrieved from Towardsdatascience: https://towardsdatascience.com/machine-learning-fundamentals-ii-neural-networks-f1e7b2cb3eef

Code Appendix

Get training data program

```
from urllib2 import urlopen
import json
import requests
from images import *
from PIL import Image
import os

url = ('https://svcs.ebay.com/services/search/FindingService/v1\
?OPERATION-NAME=findCompletedItems&paginationInput.pageNumber=1&GLOBAL-ID=EBAY-GB&listingType=FixedPrice&SERVICE-VERSION=1.0.0\
&SECURITY-APPNAME=AaronMoo-List-PRD-e4483927e-c89cf935&\
RESPONSE-DATA-FORMAT=JSON&REST-PAYLOAD&keywords=arsenal%20football%20shirt%20home')
#the URL used for searching for completed items using the eBay API
apiresult = requests.get(url)
api_return = apiresult.json()
#gets the result the URL in json format
index = 0
for item in (api_return["findCompletedItemsResponse"][0]["searchResult"][0]["item"]): #searching through each eBay item in the URL
    pic = item["galleryURL"][0] #get the picture of the item
    img = Image.open(urlopen(pic))
    Images(urlopen(pic)).resize("picc" + str(index)) #save the image in specified folder
    index+=1

def rename(): #goes through each picture in specified folder and names the images in numerical order
    i = 0
    for filename in os.listdir("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/Arsenal/"):
        if filename == "Thumbs.db":
            print "not an image"
        else:
            new_name ="pic" + str(i) + ".jpg"
            current_name ='C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/Arsenal/'+ filename
            new_name ='C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/Arsenal/'+ new_name
            os.rename(current_name, new_name)
            i += 1
```

Images.py

```
1.  from PIL import Image
2.  import sys
3.
4.  settings_array = []
5.  try:
6.      settings_file = open("settings.txt", "r") #make sure the settings file exists
7.  except:
8.      print "the settings file doesn't exist, the program will end now"
9.      sys.exit()
10. for line in settings_file:
11.     settings_array.append(line.strip('\n')) #put settings file contents into an array
12.
13. class Images:
14.
15.     '''''manage images'''
16.
17.     def __init__(self, filename):
18.         self.filename = filename
19.
20.     def resize(self, filename2):
21.         img = Image.open(self.filename)
22.         new_img = img.resize((50,50)) #resize the image to 50 x 50 pixels
23.         try:
24.             new_img.save(settings_array[4] + filename2 + ".jpg") #save the resized image in the location passed in
25.         except:
26.             print "File path does not exist in the settings file, please update and then restart"
27.             sys.exit()
28.
29.     def toarray(self): #sub routine for converting the image to an array of numbers
30.         img = Image.open(self.filename, 'r')
31.         w, h = img.size #gets image height and width
32.         pix = list(img.getdata())
33.         x = [pix[n:n+w] for n in range(0, w*h, w)]
34.         arr = []
35.         for i in range(0, len(x)):
36.             for j in range(0, len(x[i])):
37.                 for k in range(0, len(x[i][j])):
38.                     arr.append(round(3*((x[i][j][k]) / float(1000)),5)) #add the 3 rgb numbers for each pixel to the array
39.         return arr
```

Matrix.py

```
1.  import random
2.  import math
3.
4.  class Matrix:
5.
6.      '''''Matrix class'''
7.
8.      def __init__(self, rows, cols):
9.          self.rows = rows
10.         self.cols = cols
11.         self.matrix = []
12.
13.     def MakeMatrix(self):
14.         for i in range(0,self.rows):
15.             self.matrix.append([])
16.             for j in range(0,self.cols):
17.                 self.matrix[i].append(j)
18.                 self.matrix[i][j]=0 #make every element in the matrix have a value of 0 for the default
19.         return self.matrix
20.
21.     def RandomizeMatrix(self):
22.         for i in range(0,self.rows):
23.             for j in range(0,self.cols):
24.                 self.matrix[i][j] = random.uniform(-1,1) #make every element in the matrix have a float value between -1 and 1
25.         return self.matrix
26.
27.     def multiply(self, n):
28.         if isinstance(n, Matrix):
29.             for i in range(0,self.rows):
30.                 for j in range(0,self.cols):
31.                     self.matrix[i][j] *= n.matrix[i][j] #times two matrices together
32.             return self.matrix
33.         else:
34.             for i in range(0,self.rows):
35.                 for j in range(0,self.cols):
36.                     self.matrix[i][j] *= n #times each element in a matrix by a value n
37.             return self.matrix
38.
```

```
39.     def add(self, n):
40.         if isinstance(n, Matrix):
41.             for i in range(0,self.rows):
42.                 for j in range(0,self.cols):
43.                     self.matrix[i][j] += n.matrix[i][j] #add two matrices together
44.             return self.matrix
45.         else:
46.             for i in range(0,self.rows):
47.                 for j in range(0,self.cols):
48.                     self.matrix[i][j] += n #add n to each element in the matrix
49.             return self.matrix
50.
51.     def subtract(self, n):
52.         result = Matrix(self.rows, self.cols)
53.         result.MakeMatrix()
54.         for i in range(0,result.rows):
55.             for j in range(0,result.cols):
56.                 result.matrix[i][j] = self.matrix[i][j] - n.matrix[i][j] #subtract one matrix from another
57.         return result
58.
59.     def dotproduct(self, n): #dot product of two matrices
60.         if isinstance(n, Matrix):
61.             if self.cols != n.rows:
62.                 print "Not equal cols and rows"
63.             else:
64.                 result = Matrix(self.rows, n.cols)
65.                 result.MakeMatrix()
66.                 for i in range(0, result.rows):
67.                     for j in range(0,result.cols):
68.                         total = 0
69.                         for k in range(0, self.cols):
70.                             total += self.matrix[i][k] * n.matrix[k][j]
71.                         result.matrix[i][j] = total
72.                 return result
73.         else:
74.             print "Not matrix"
75.
76.     def transpose(self): #transpose a matrix
77.         result = Matrix(self.cols, self.rows)
78.         result.MakeMatrix()
79.         for i in range(0, self.rows):
80.             for j in range(0, self.cols):
```

```
                result.matrix[j][i] = self.matrix[i][j]
        return result

    def apply_function(self, fun): #apply a function to each element in the matrix
        for i in range(0, self.rows):
            for j in range(0, self.cols):
                val = self.matrix[i][j]
                self.matrix[i][j] = fun(val)
        return self.matrix

    def apply_function_new_matrix(self, fun): #apply a function to each element in the matrix and return a new matrix containing thos
e values
        result = Matrix(self.rows, self.cols)
        result.MakeMatrix()
        for i in range(0, result.rows):
            for j in range(0, result.cols):
                val = self.matrix[i][j]
                result.matrix[i][j] = fun(val)
        return result
```

Neuralnetwork.py

```
1.          from matrix import *
2.          import math
3.          from images import *
4.          import sys
5.
6.          settings_array = []
7.          try:
8.              settings_file = open("settings.txt", "r") #make sure settings file exists
9.          except:
10.             print "the settings file doesn't exist, the program will end now"
11.             sys.exit()
12.         for line in settings_file:
13.             settings_array.append(line.strip('\n')) #add contents of settings file to an array
14.
15.         def same(x):
16.             return x
17.
18.         #activation functions
19.         def sigmoid(x):
20.           if x < 0:
21.             return 1- 1 / (1 + math.exp(x))
22.           return 1 / (1 + math.exp(-x))
23.
24.         def dsigmoid(x):
25.           return x * (1 - x)
26.
27.         def relu(x):
28.             if x < 0:
29.                 return x * 0.01
30.             else:
31.                 return x
32.
33.         def drelu(x):
34.           if x < 0:
35.             return 0.01
36.           else:
37.             return 1
38.
```

```
39.         def tanh(x):
40.           t = math.exp(x)
41.           s = math.exp(-x)
42.           return (t - s) / (t + s)
43.
44.         def dtanh(x):
45.           return 1 - (x*x)
46.
47.         def softmax(outputs): #produce softmax of output array
48.           arr = []
49.           denominator = 0
50.           for i in range(0,outputs.rows):
51.             for j in range(0,outputs.cols):
52.               denominator += math.exp(outputs.matrix[i][j])
53.           for i in range(0,outputs.rows):
54.             for j in range(0,outputs.cols):
55.               arr.append((math.exp(outputs.matrix[i][j]))/denominator)
56.           return arr
57.
58.         def softmaxtrain(outputs): #another softmax sub routine
59.           arr = []
60.           denominator = 0
61.           for i in range(0,len(outputs)):
62.             for j in range(0,1):
63.               denominator += math.exp(outputs[i][j])
64.           for i in range(0,len(outputs)):
65.             for j in range(0,1):
66.               arr.append((math.exp(outputs[i][j]))/denominator)
67.           return arr
68.
69.         class neuralNetwork:
70.
71.             '''''neural network'''
72.
73.             def __init__(self, inputnodes, hiddennodes1, hiddennodes2, outputnodes): #instantiation class, take the number of input, h
    1, h2 and output nodes
74.
75.                 self.inodes = inputnodes
76.                 self.hnodes1 = hiddennodes1
77.                 self.hnodes2 = hiddennodes2
78.                 self.onodes = outputnodes
79.
```

```
80.                 self.weights_ih = Matrix(self.hnodes1, self.inodes)
81.                 self.weights_h1h2 = Matrix(self.hnodes2, self.hnodes1)
82.                 self.weights_ho = Matrix(self.onodes, self.hnodes2)
83.
84.                 self.weights_ih.MakeMatrix()
85.                 self.weights_h1h2.MakeMatrix()
86.                 self.weights_ho.MakeMatrix()
87.
88.                 self.weights_ih.RandomizeMatrix()
89.                 self.weights_h1h2.RandomizeMatrix()
90.                 self.weights_ho.RandomizeMatrix()
91.
92.                 self.bias_h = Matrix(self.hnodes1, 1)
93.                 self.bias_h2 = Matrix(self.hnodes2, 1)
94.                 self.bias_o = Matrix(self.onodes, 1)
95.
96.                 self.bias_h.MakeMatrix()
97.                 self.bias_h2.MakeMatrix()
98.                 self.bias_o.MakeMatrix()
99.
100.                self.bias_h.RandomizeMatrix()
101.                self.bias_h2.RandomizeMatrix()
102.                self.bias_o.RandomizeMatrix()
103.
104.                self.learningrate = 0.01
105.
106.            def write_weights_to_file(self, wih, bh1, wh1h2, bh2, wh3o, bo): #subroutine for writing the weights to a text file
107.                f = open(settings_array[5], "w+")
108.                for i in wih:
109.                    for j in i:
110.                        f.write(str(j)+",")
111.                f.write("\n")
112.                for i in bh1:
113.                    for j in i:
114.                        f.write(str(j)+",")
115.                f.write("\n")
116.                for i in wh1h2:
117.                    for j in i:
118.                        f.write(str(j)+",")
119.                f.write("\n")
120.                for i in bh2:
121.                    for j in i:
```

```
                    f.write(str(j)+",")
            f.write("\n")
            for i in wh3o:
                for j in i:
                    f.write(str(j)+",")
            f.write("\n")
            for i in bo:
                for j in i:
                    f.write(str(j)+",")
            f.write("\n")
            f.close()

        def run_with_existing_weights(self, input_array): #run the neural network with weights from txt file
            inputs = Matrix(len(input_array), 1)
            inputs.MakeMatrix()
            for i in range(0, len(input_array)):
                inputs.matrix[i][0] = input_array[i] #convert the input array of the image to type matrix
            try:
                f = open(settings_array[5], "r") #make sure the weigths file exists
            except:
                print "Weights file in settings file does not exist or file path is incorrect, please update and restart"
                sys.exit()
            try: #retrieve the weights from the file
                ih1 = f.readline()
                ih1split = ih1.split(",")
                ih1_weights = []
                for i in range(0,len(ih1split)-1):
                    ih1_weights.append(ih1split[i])
                weights_ih1 = Matrix(self.hnodes1,self.inodes)
                weights_ih1.MakeMatrix()
                count_ih1 = 0
                for i in range(0,self.hnodes1):
                    for j in range(0,self.inodes):
                        weights_ih1.matrix[i][j] = float(ih1_weights[count_ih1])
                        count_ih1 += 1

                bh1 = f.readline()
                bh1split = bh1.split(",")
                bh1_weights = []
                for i in range(0,len(bh1split)-1):
                    bh1_weights.append(bh1split[i])
                weights_bh1 = Matrix(self.hnodes1,1)
```

```
                    weights_bh1.MakeMatrix()
                    count_bh1 = 0
                    for i in range(0,self.hnodes1):
                        for j in range(0,1):
                            weights_bh1.matrix[i][j] = float(bh1_weights[count_bh1])
                            count_bh1 += 1

                    h1h2 = f.readline()
                    h1h2split = h1h2.split(",")
                    h1h2_weights = []
                    for i in range(0, len(h1h2split)-1):
                        h1h2_weights.append(h1h2split[i])
                    weights_h1h2 = Matrix(self.hnodes2,self.hnodes1)
                    weights_h1h2.MakeMatrix()
                    count_h1h2 = 0
                    for i in range(0,self.hnodes2):
                        for j in range(0,self.hnodes1):
                            weights_h1h2.matrix[i][j] = float(h1h2_weights[count_h1h2])
                            count_h1h2 += 1

                    bh2 = f.readline()
                    bh2split = bh2.split(",")
                    bh2_weights = []
                    for i in range(0,len(bh2split)-1):
                        bh2_weights.append(bh2split[i])
                    weights_bh2 = Matrix(self.hnodes2,1)
                    weights_bh2.MakeMatrix()
                    count_bh2 = 0
                    for i in range(0,self.hnodes2):
                        for j in range(0,1):
                            weights_bh2.matrix[i][j] = float(bh2_weights[count_bh2])
                            count_bh2 += 1

                    h3o = f.readline()
                    h3osplit = h3o.split(",")
                    h3o_weights = []
                    for i in range(0, len(h3osplit)-1):
                        h3o_weights.append(h3osplit[i])
                    weights_h3o = Matrix(self.onodes,self.hnodes2)
                    weights_h3o.MakeMatrix()
                    count_h3o = 0
                    for i in range(0,self.onodes):
```

```
206.                        for j in range(0,self.hnodes2):
207.                            weights_h3o.matrix[i][j] = float(h3o_weights[count_h3o])
208.                            count_h3o += 1
209.
210.                    bo = f.readline()
211.                    bosplit = bo.split(",")
212.                    bo_weights = []
213.                    for i in range(0,len(bosplit)-1):
214.                        bo_weights.append(bosplit[i])
215.                    weights_bo = Matrix(self.onodes,1)
216.                    weights_bo.MakeMatrix()
217.                    count_bo = 0
218.                    for i in range(0,self.onodes):
219.                        for j in range(0,1):
220.                            weights_bo.matrix[i][j] = float(bo_weights[count_bo])
221.                            count_bo += 1
222.
223.                    f.close()
224.                except:
225.                    print "Invalid weights file"
226.                    sys.exit()
227.
228.                #feedforward the image inputted
229.                hidden1 = weights_ih1.dotproduct(inputs)
230.                hidden1.add(weights_bh1)
231.                hidden1.apply_function(sigmoid) #get hidden 1 output
232.
233.                hidden2 = weights_h1h2.dotproduct(hidden1)
234.                hidden2.add(weights_bh2)
235.                hidden2.apply_function(sigmoid) #get hidden 2 output
236.
237.                output = weights_h3o.dotproduct(hidden2)
238.                output.add(weights_bo)
239.                output.apply_function(sigmoid) #get final output
240.
241.                return output
242.
243.            def feedforward(self, input_array):
244.
245.                inputs = Matrix(len(input_array), 1)
246.                inputs.MakeMatrix()
247.                for i in range(0, len(input_array)):
```

```
                    inputs.matrix[i][0] = input_array[i]
                #generating hidden output
                hidden = self.weights_ih.dotproduct(inputs)
                hidden.add(self.bias_h)
                #activation function
                hidden.apply_function(sigmoid) #get hidden 1 output

                hidden2 = self.weights_h1h2.dotproduct(hidden)
                hidden2.add(self.bias_h2)
                hidden2.apply_function(sigmoid) #get hidden 2 output

                output = self.weights_ho.dotproduct(hidden2)
                output.add(self.bias_o)
                output.apply_function(sigmoid) #get output

                output = softmax(output) #apply softmax to output array

                return output

            def train(self, inputs_array, targets_array,nn):

                inputs = Matrix(len(inputs_array), 1)
                inputs.MakeMatrix()
                for i in range(0, len(inputs_array)):
                    inputs.matrix[i][0] = inputs_array[i] #convert input image array to type matrix
                #generating hidden output
                hidden = self.weights_ih.dotproduct(inputs)
                hidden.add(self.bias_h)
                #activation function
                hidden.apply_function(sigmoid)

                hidden2 = self.weights_h1h2.dotproduct(hidden)
                hidden2.add(self.bias_h2)
                hidden2.apply_function(sigmoid) #get hidden 2 output

                #generate output
                outputs = self.weights_ho.dotproduct(hidden2)
                outputs.add(self.bias_o)
                #activation function
                outputs.apply_function(sigmoid) #get output

                #put targets array into matrix
```

```
290.                targets = Matrix(len(targets_array), 1)
291.                targets.MakeMatrix()
292.                for i in range(0, len(targets_array)):
293.                    targets.matrix[i][0] = targets_array[i]
294.
295.                #calculate output errors
296.                output_errors = targets.subtract(outputs)
297.
298.                #calculate gradients
299.                gradients = outputs.apply_function_new_matrix(dsigmoid)
300.                gradients.multiply(output_errors)
301.                gradients.multiply(self.learningrate)
302.
303.                #calculate deltas
304.                hidden2_transposed = hidden2.transpose()
305.                weight_ho_deltas = gradients.dotproduct(hidden2_transposed)
306.
307.                #adjust ho weights by deltas
308.                self.weights_ho.add(weight_ho_deltas)
309.                #adjust bias by deltas
310.                self.bias_o.add(gradients)
311.
312.                #calculate hidden2 layer errors
313.                weights_ho_transposed = self.weights_ho.transpose()
314.                hidden2_errors = weights_ho_transposed.dotproduct(output_errors)
315.
316.                hidden2_gradient = hidden2.apply_function_new_matrix(dsigmoid)
317.                hidden2_gradient.multiply(hidden2_errors)
318.                hidden2_gradient.multiply(self.learningrate)
319.
320.                hidden1_transposed = hidden.transpose()
321.                weight_h1h2_deltas = hidden2_gradient.dotproduct(hidden1_transposed)
322.
323.                self.weights_h1h2.add(weight_h1h2_deltas)
324.                self.bias_h2.add(hidden2_gradient)
325.
326.                weights_h1h2_transposed = self.weights_h1h2.transpose()
327.                hidden_errors = weights_h1h2_transposed.dotproduct(hidden2_errors) #calculate hidden errors
328.
329.                #calculate hidden gradients
330.                hidden_gradient = hidden.apply_function_new_matrix(dsigmoid)
331.                hidden_gradient.multiply(hidden_errors)
```

```
332.                hidden_gradient.multiply(self.learningrate)
333.
334.                #calculate input to hidden deltas
335.                inputs_transposed = inputs.transpose()
336.                weight_ih_deltas = hidden_gradient.dotproduct(inputs_transposed)
337.
338.                #adjust ih weights
339.                self.weights_ih.add(weight_ih_deltas)
340.                #adjust hidden bias by deltas
341.                self.bias_h.add(hidden_gradient)
342.
343.                #write the weights to the weights file
344.                nn.write_weights_to_file(self.weights_ih.matrix, self.bias_h.matrix, self.weights_h1h2.matrix, self.bias_h2.matrix, se
    lf.weights_ho.matrix, self.bias_o.matrix)
345.
346.            def train_with_existing_weights(self, inputs_array, targets_array,nn):
347.
348.                inputs = Matrix(len(inputs_array), 1)
349.                inputs.MakeMatrix()
350.                for i in range(0, len(inputs_array)):
351.                    inputs.matrix[i][0] = inputs_array[i] #convert the inputs image array to type matrix
352.                try:
353.                    f = open(settings_array[5], "r") #make sure the weights file exists
354.                except:
355.                    print "Weights file in settings file does not exist or file path is incorrect, please update and restart"
356.                    sys.exit()
357.                try: #retrieve the weights from the file
358.                    ih1 = f.readline()
359.                    ih1split = ih1.split(",")
360.                    ih1_weights = []
361.                    for i in range(0,len(ih1split)-1):
362.                        ih1_weights.append(ih1split[i])
363.                    weights_ih1 = Matrix(self.hnodes1,self.inodes)
364.                    weights_ih1.MakeMatrix()
365.                    count_ih1 = 0
366.                    for i in range(0,self.hnodes1):
367.                        for j in range(0,self.inodes):
368.                            weights_ih1.matrix[i][j] = float(ih1_weights[count_ih1])
369.                            count_ih1 += 1
370.
371.                    bh1 = f.readline()
372.                    bh1split = bh1.split(",")
```

```
                    bh1_weights = []
                    for i in range(0,len(bh1split)-1):
                        bh1_weights.append(bh1split[i])
                    weights_bh1 = Matrix(self.hnodes1,1)
                    weights_bh1.MakeMatrix()
                    count_bh1 = 0
                    for i in range(0,self.hnodes1):
                        for j in range(0,1):
                            weights_bh1.matrix[i][j] = float(bh1_weights[count_bh1])
                            count_bh1 += 1

                    h1h2 = f.readline()
                    h1h2split = h1h2.split(",")
                    h1h2_weights = []
                    for i in range(0, len(h1h2split)-1):
                        h1h2_weights.append(h1h2split[i])
                    weights_h1h2 = Matrix(self.hnodes2,self.hnodes1)
                    weights_h1h2.MakeMatrix()
                    count_h1h2 = 0
                    for i in range(0,self.hnodes2):
                        for j in range(0,self.hnodes1):
                            weights_h1h2.matrix[i][j] = float(h1h2_weights[count_h1h2])
                            count_h1h2 += 1

                    bh2 = f.readline()
                    bh2split = bh2.split(",")
                    bh2_weights = []
                    for i in range(0,len(bh2split)-1):
                        bh2_weights.append(bh2split[i])
                    weights_bh2 = Matrix(self.hnodes2,1)
                    weights_bh2.MakeMatrix()
                    count_bh2 = 0
                    for i in range(0,self.hnodes2):
                        for j in range(0,1):
                            weights_bh2.matrix[i][j] = float(bh2_weights[count_bh2])
                            count_bh2 += 1

                    h3o = f.readline()
                    h3osplit = h3o.split(",")
                    h3o_weights = []
                    for i in range(0, len(h3osplit)-1):
                        h3o_weights.append(h3osplit[i])
```

```
                    weights_h3o = Matrix(self.onodes,self.hnodes2)
                    weights_h3o.MakeMatrix()
                    count_h3o = 0
                    for i in range(0,self.onodes):
                        for j in range(0,self.hnodes2):
                            weights_h3o.matrix[i][j] = float(h3o_weights[count_h3o])
                            count_h3o += 1

                    bo = f.readline()
                    bosplit = bo.split(",")
                    bo_weights = []
                    for i in range(0,len(bosplit)-1):
                        bo_weights.append(bosplit[i])
                    weights_bo = Matrix(self.onodes,1)
                    weights_bo.MakeMatrix()
                    count_bo = 0
                    for i in range(0,self.onodes):
                        for j in range(0,1):
                            weights_bo.matrix[i][j] = float(bo_weights[count_bo])
                            count_bo += 1

                    f.close()
                except:
                    print "Invalid weights file"
                    sys.exit()

                inputs = Matrix(len(inputs_array), 1)
                inputs.MakeMatrix()
                for i in range(0, len(inputs_array)):
                    inputs.matrix[i][0] = inputs_array[i] #convert input image array to type matrix

                #feedforward the input matrix
                hidden1 = weights_ih1.dotproduct(inputs)
                hidden1.add(weights_bh1)
                hidden1.apply_function(sigmoid)

                hidden2 = weights_h1h2.dotproduct(hidden1)
                hidden2.add(weights_bh2)
                hidden2.apply_function(sigmoid)

                outputs = weights_h3o.dotproduct(hidden2)
                outputs.add(weights_bo)
```

```
457.                outputs.apply_function(sigmoid)
458.
459.                targets = Matrix(len(targets_array), 1)
460.                targets.MakeMatrix()
461.                for i in range(0, len(targets_array)):
462.                    targets.matrix[i][0] = targets_array[i]
463.
464.                #calculate output errors
465.                output_errors = targets.subtract(outputs)
466.
467.                #calculate gradients
468.                gradients = outputs.apply_function_new_matrix(dsigmoid)
469.                gradients.multiply(output_errors)
470.                gradients.multiply(self.learningrate)
471.
472.                #calculate deltas
473.                hidden2_transposed = hidden2.transpose()
474.                weight_ho_deltas = gradients.dotproduct(hidden2_transposed)
475.
476.                #adjust ho weights by deltas
477.                weights_h3o.add(weight_ho_deltas)
478.                #adjust bias by deltas
479.                weights_bo.add(gradients)
480.
481.                #calculate hidden layer errors
482.                weights_ho_transposed = weights_h3o.transpose()
483.                hidden2_errors = weights_ho_transposed.dotproduct(output_errors)
484.
485.                hidden2_gradient = hidden2.apply_function_new_matrix(dsigmoid)
486.                hidden2_gradient.multiply(hidden2_errors)
487.                hidden2_gradient.multiply(self.learningrate)
488.
489.                hidden1_transposed = hidden1.transpose()
490.                weight_h1h2_deltas = hidden2_gradient.dotproduct(hidden1_transposed)
491.
492.                weights_h1h2.add(weight_h1h2_deltas)
493.                weights_bh2.add(hidden2_gradient)
494.
495.                weights_h1h2_transposed = weights_h1h2.transpose()
496.                hidden_errors = weights_h1h2_transposed.dotproduct(hidden2_errors)
497.
498.                #calculate hidden gradients
```

```
499.                hidden_gradient = hidden1.apply_function_new_matrix(dsigmoid)
500.                hidden_gradient.multiply(hidden_errors)
501.                hidden_gradient.multiply(self.learningrate)
502.
503.                #calculate input to hidden deltas
504.                inputs_transposed = inputs.transpose()
505.                weight_ih_deltas = hidden_gradient.dotproduct(inputs_transposed)
506.
507.                #adjust ih weights
508.                weights_ih1.add(weight_ih_deltas)
509.                #adjust hidden bias by deltas
510.                weights_bh1.add(hidden_gradient)
511.
512.                #write the updated weights to the weights file
513.                nn.write_weights_to_file(weights_ih1.matrix, weights_bh1.matrix, weights_h1h2.matrix, weights_bh2.matrix, weights_h3o.
    matrix, weights_bo.matrix)
```

Training.py

```
1.  from neuralnetwork import *
2.  from images import *
3.  import sys
4.
5.  #settings file
6.  settings_array = []
7.  try:
8.      settings_file = open("settings.txt", "r")
9.  except:
10.     print "the settings file doesn't exist, the program will end now"
11.     sys.exit()
12. for line in settings_file:
13.     settings_array.append(line.strip('\n'))
14.
15. #clubs file
16. clubs = []
17. try:
18.     clubs_file = open("clubs.txt", "r")
19. except:
20.     print "the clubs file does not exist, the program will end now"
21.     sys.exit()
22. for club in clubs_file:
23.     clubs.append(club.strip('\n'))
24.
25. if int(settings_array[6]) != len(clubs):
26.     print "the txt files do not match up, the program will end now"
27.
28. def softmaxtrain(outputs): #sub routine for softmax
29.   arr = []
30.   denominator = 0
31.   for i in range(0,len(outputs)):
32.     for j in range(0,1):
33.       denominator += math.exp(outputs[i][j])
34.   for i in range(0,len(outputs)):
35.     for j in range(0,1):
36.       arr.append((math.exp(outputs[i][j]))/denominator)
37.   return arr
38.
```

```
training_inputs = [] #training inputs of shirts
for i in range(0, 113):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/chelsea/pic" + str(i) + ".jpg").toarray())
for i in range(0,108):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/arsenal/pic" + str(i) + ".jpg").toarray())
for i in range(0,59):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/Norwich/pic" + str(i) + ".jpg").toarray())
for i in range(0,106):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/mancity/pic" + str(i) + ".jpg").toarray())
for i in range(0,87):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/tottenham/pic" + str(i) + ".jpg").toarray())

for i in range(0,123):
  training_inputs.append(Images("C:/Users/Aaron/Documents/Homework/computing/A Level/Nea/newcastle/pic" + str(i) + ".jpg").toarray())


training_targets = [] #training targets for shirts
for i in range(0,113):
  training_targets.append([1,0,0,0,0,0])#chelsea
for i in range(0,108):
  training_targets.append([0,1,0,0,0,0])#arsenal
for i in range(0,59):
  training_targets.append([0,0,1,0,0,0])#norwich
for i in range(0,106):
  training_targets.append([0,0,0,1,0,0])#man city
for i in range(0,87):
  training_targets.append([0,0,0,0,1,0])#tottenham
for i in range(0,123):
  training_targets.append([0,0,0,0,0,1])#newcastle

nn = neuralNetwork(7500,200,20,settings_array[6]) #instantiate neural network

for i in range(0,1): #train neural network once with random weigths and write to file
  inputs = random.choice(training_inputs)
  index = training_inputs.index(inputs)
  target = training_targets[index]
  nn.train(inputs,target, nn)
  print "trained"

ind = 0
for i in range(0,1000): #train neural network 1000 times using existing weights in file, print out results every 100 iterations
  ind +=1
```

```
  inputs = random.choice(training_inputs)
  index = training_inputs.index(inputs)
  target = training_targets[index]
  nn.train_with_existing_weights(inputs,target,nn)
  print "trained" + str(ind)
  if ind == 100 or ind == 200 or ind == 300 or ind == 400 or ind == 500 or ind == 600 or ind == 700 or ind == 800 or ind == 900 or in
d == 1000:
      print "chelsea = [1,0,0,0,0,0]"
      for i in range(1,11):
          c = nn.run_with_existing_weights(Images("chelseatest" + str(i) + ".jpg").toarray()).matrix
          print softmaxtrain(c)
      print "arsenal = [0,1,0,0,0,0]"
      for i in range(1,11):
          a = nn.run_with_existing_weights(Images("arsenaltest" + str(i) + ".jpg").toarray()).matrix
          print softmaxtrain(a)
      print "norwich = [0,0,1,0,0,0]"
      for i in range(1,11):
          nor = nn.run_with_existing_weights(Images("norwichtest" + str(i) + ".jpg").toarray()).matrix
          print softmaxtrain(nor)
      print "man city = [0,0,0,1,0,0]"
      for i in range(1,11):
          man = nn.run_with_existing_weights(Images("mancitytest" + str(i) + ".jpg").toarray()).matrix
                  print softmaxtrain(man)
             print "tottenham = [0,0,0,0,1,0]"
             for i in range(1,11):
                 w = nn.run_with_existing_weights(Images("tottenham" + str(i) + ".jpg").toarray()).matrix
                 print softmaxtrain(w)
             print "newcastle = [0,0,0,0,0,1]"
             for i in range(1,11):
                 new = nn.run_with_existing_weights(Images("newcastle" + str(i) + ".jpg").toarray()).matrix
                 print softmaxtrain(new)

      for i in range(0,10): #train neural network 10 times printing out the results for each iteration, and also printing the number
of shirts it got correct
         ind +=1
         inputs = random.choice(training_inputs)
         index = training_inputs.index(inputs)
         target = training_targets[index]
         nn.train_with_existing_weights(inputs,target, nn)
         print "trained" + str(ind)

         cc = 0
```

```
        print "chelsea = [1,0,0,0,0,0]"
        for i in range(1,11):
          chelsea = nn.run_with_existing_weights(Images("chelseatest" + str(i) + ".jpg").toarray()).matrix
          cindex = 0
          chigh = 0
          chelseaout = softmaxtrain(chelsea)
          print chelseaout
          for i in range(0,len(chelseaout)):
            if chelseaout[i] > chigh:
              chigh = chelseaout[i]
              cindex= i
          if cindex == 0:
            cc+=1
        print cc

        ac = 0
        print "arsenal = [0,1,0,0,0,0]"
        for i in range(1,11):
          arsenal = nn.run_with_existing_weights(Images("arsenaltest" + str(i) + ".jpg").toarray()).matrix
          aindex = 0
          ahigh = 0
          arsenalout = softmaxtrain(arsenal)
          print arsenalout
          for i in range(0,len(arsenalout)):
            if arsenalout[i] > ahigh:
              ahigh = arsenalout[i]
              aindex= i
          if aindex == 0:
            ac+=1
        print ac

        nc = 0
        print "norwich = [0,0,1,0,0,0]"
        for i in range(1,11):
          norwich = nn.run_with_existing_weights(Images("norwichtest" + str(i) + ".jpg").toarray()).matrix
          nindex = 0
          nhigh = 0
          norwichout = softmaxtrain(norwich)
          print norwichout
          for i in range(0,len(norwichout)):
            if norwichout[i] > nhigh:
              nhigh = norwichout[i]
```

```
            nindex= i
        if nindex == 2:
          nc+=1
      print nc

      mc = 0
      print "man city = [0,0,0,1,0,0]"
      for i in range(1,11):
        mancity = nn.run_with_existing_weights(Images("mancitytest" + str(i) + ".jpg").toarray()).matrix
        mcindex = 0
        mchigh = 0
        mcout = softmaxtrain(mancity)
        print mcout
        for i in range(0,len(mcout)):
          if mcout[i] > mchigh:
            mchigh = mcout[i]
            mcindex= i
        if mcindex == 3:
          mc+=1
      print mc

      tc = 0
      print "tottenham = [0,0,0,0,1,0]"
      for i in range(1,11):
        tottenham = nn.run_with_existing_weights(Images("tottenham" + str(i) + ".jpg").toarray()).matrix
        tcindex = 0
        tchigh = 0
        tcout = softmaxtrain(tottenham)
        print tcout
        for i in range(0,len(tcout)):
          if tcout[i] > tchigh:
            tchigh = tcout[i]
            tcindex= i
        if tcindex == 4:
          tc+=1
      print tc

      nec = 0
      print "newcastle = [0,0,0,0,0,1]"
      for i in range(1,11):
        newcastle = nn.run_with_existing_weights(Images("newcastle" + str(i) + ".jpg").toarray()).matrix
        neindex = 0
```

```
203.            nehigh = 0
204.            newcastleout = softmaxtrain(newcastle)
205.            print newcastleout
206.            for i in range(0,len(newcastleout)):
207.              if newcastleout[i] > nehigh:
208.                nehigh = newcastleout[i]
209.                neindex= i
210.            if neindex == 1:
211.              nec+=1
212.          print nec
213.
214.          if cc > 6 and ac > 6 and nc > 6 and mc > 6 and tc > 6 and nec > 6:
215.              print "70% correct for each"
216.              sys.exit()
```

Gui.py

```
1.          import Tkinter
2.          import tkMessageBox
3.          import ttk
4.          import tkFileDialog
5.          import tkSimpleDialog as simpledialog
6.          from PIL import ImageTk, Image
7.
8.          import re
9.          from urllib2 import urlopen
10.         import json
11.         import requests
12.         import math
13.          import io
14.         import sys
15.
16.         from ebaysdk.trading import Connection as Trading
17.
18.         from neuralnetwork import *
19.         from images import *
20.
21.         continue_listing = True
22.         #settings file
23.         settings_array = []
24.         try:
25.             settings_file = open("settings.txt", "r") #make sure settings file exists
26.         except:
27.             print "the settings file doesn't exist, the program will end now"
28.             sys.exit()
29.         for line in settings_file:
30.             settings_array.append(line.strip('\n')) #add contents of settings file to an array
31.
32.         #clubs file
33.         clubs = []
34.         try:
35.             clubs_file = open("clubs.txt", "r") #make sure clubs file exists
36.         except:
```

```
37.             print "the clubs file does not exist, the program will end now"
38.             sys.exit()
39.         for club in clubs_file:
40.             clubs.append(club.strip('\n')) #add contents of settings file to an array
41.
42.         def main():
43.             while continue_listing == True:
44.                 main_menu = Tkinter.Tk() #main menu window
45.                 main_menu.title("eBay Lister")
46.                 main_menu.geometry('500x400')
47.                 welcome_label = Tkinter.Label(main_menu, text = "Welcome To The Auto eBay Lister", font=("Arial Bold", 20))
48.                 welcome_label.grid(column=0, row=0)
49.                 input_image_button = Tkinter.Button(main_menu, text="Input image", font=("Arial Bold", 12), command=lambda: open_file(
    main_menu))
50.                 input_image_button.grid(column=0,row=1, pady=50) #input image button
51.                 progress_bar = ttk.Progressbar(main_menu, length=200) #progress bar, makes it look better and more clear for user
52.                 progress_bar.grid(column=0,row=2, pady=50)
53.                 button_exit = Tkinter.Button(main_menu, text="Exit program", font=("Arial Bold", 12), command=lambda: end(main_menu))

54.                 button_exit.grid(column=0,row=3) #exit program button
55.                 main_menu.mainloop()
56.
57.         def end(main_menu): #sub routine which ends the program when called
58.             main_menu.destroy()
59.             sys.exit()
60.
61.         def window_for_info_being_added(file_path, club_name):
62.             input_shirt_details_window = Tkinter.Tk() #make window where user can input details of shirt
63.             input_shirt_details_window.title("Input details for shirt")
64.             input_shirt_details_window.geometry('500x300')
65.             valid_year = False
66.             valid_email = False
67.             label_year = Tkinter.Label(input_shirt_details_window, text="Year of shirt", font=("Arial Bold", 10))
68.             label_year.grid(column=0,row=0)
69.             label_size = Tkinter.Label(input_shirt_details_window, text="Select the size of the shirt", font=("Arial Bold", 10))
70.             label_size.grid(column=0,row=1)
71.             label_email = Tkinter.Label(input_shirt_details_window, text="PayPal email", font=("Arial Bold", 10))
72.             label_email.grid(column=0,row=2)
73.             label_weight = Tkinter.Label(input_shirt_details_window, text="Select the weight of the shirt", font=("Arial Bold", 10))
74.             label_weight.grid(column=0,row=3)
75.             label_year_number = Tkinter.Label(input_shirt_details_window,text="", font=("Arial Bold", 10))
76.             label_year_number.grid(column=1, row=0)
```

```
            combo_size = ttk.Combobox(input_shirt_details_window) #drop down box for selecting size of shirt
            combo_size['values']= ("XXS", "XS", "S", "M", "L", "XL", "XXL", "XXXL") #possible sizes
            combo_size.current(0)
            combo_size.grid(column=1,row=1)
            label_email_display = Tkinter.Label(input_shirt_details_window,text="", font=("Arial Bold", 10))
            label_email_display.grid(column=1, row=2)
            combo_weight = ttk.Combobox(input_shirt_details_window) #drop down box for selecting weight of shirt
            combo_weight['values']= ("w<1","1<w<2","2<w<10","10<w<15") #possible weights
            combo_weight.current(0)
            combo_weight.grid(column=1,row=3)
            progress_bar = ttk.Progressbar(input_shirt_details_window, length=200)
            progress_bar.grid(column=0,row=5) #update progress bar
            progress_bar['value']=30
            button_confirm_text = Tkinter.Button(input_shirt_details_window, text="Confirm", font=("Arial Bold", 10), command=lambda:
    confirm_items(label_year_number.cget("text"), combo_size.get(),label_email_display.cget("text"),combo_weight.get(), input_shirt_detai
    ls_window, file_path, club_name))
            button_confirm_text.grid(column=0,row=4) #button for confirming if user is happy with details they have input
            while valid_year == False:
                try:
                    year = int(simpledialog.askstring('Year','Please input the year for the shirt', parent=input_shirt_details_window)
    )
                    if year < 1900 or year > 2020: #valid range of years for the football shirts
                        tkMessageBox.showinfo('Invalid year', 'Please input a valid year')
                    else:
                        valid_year = True
                except:
                            tkMessageBox.showinfo('Invalid year', 'Please input a valid year')
                   label_year_number.configure(text=str(year)) #display the year to the user

                   while valid_email == False:
                       try:
                           email = str(simpledialog.askstring('Email','Please input your PayPal email', parent=input_shirt_details_win
    dow))
                           regex = re.search("[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-
    ]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?", email)
                           if regex == None: #use regex to check is email is valid
                               tkMessageBox.showinfo('Invalid email', 'Please input a valid email')
                           else:
                               valid_email = True
                       except:
                           tkMessageBox.showinfo('Invalid email', 'Please input a valid email')
                   label_email_display.configure(text=str(email)) #display the email to the user
```

```
114.
115.       def scroll_box(event):
116.           canvas.configure(scrollregion=canvas.bbox("all"),width=500,height=450) #Tkinter scroll box
117.
118.       def confirm_items(year, size, email, weight, window, file_path, club_name):
119.           size_actual =''
120.           if size == "S": #if statement for turning drop down box size to the actual word, used for the title
121.               size_actual = "Small"
122.           elif size == "M":
123.               size_actual = "Medium"
124.           elif size == "L":
125.               size_actual = "Large"
126.           if size_actual == '':
127.               size_actual = size
128.           postage_price = 0
129.           if weight == "w<1": #if statement for turning the drop down box weight to the postage cost
130.               postage_price = 2.89
131.           elif weight == "1<w<2":
132.               postage_price = 4.05
133.           elif weight == "2<w<10":
134.               postage_price = 6.49
135.           else:
136.               postage_price = 8.99
137.           url = (settings_array[0] + club_name + "+shirt+home+" + str(year)+"+"+size_actual) #URL to be searched on eBay to find sim
    ilar items using user's inputted information
138.           internet_connection = False
139.           while internet_connection == False: #make sure user is connected to internet, otherwise won't be able to search for the UR
    L
140.               try:
141.                   apiresult = requests.get(url) #result of searching for the URL using the requests import
142.                   internet_connection = True
143.               except:
144.                   tkMessageBox.showinfo('No Internet Connection or invalid URL', 'Please connect to the internet or chnage the URL i
    n the settings file and restart the program')
145.           try:
146.               json_format_of_listings = apiresult.json() #put URL return in json format
147.               array_for_prices =[]
148.               array_for_images =[]
149.               array_for_titles =[]
150.               for item in (json_format_of_listings["findCompletedItemsResponse"][0]["searchResult"][0]["item"]): #search through eac
    h listing
151.                   picture_of_listing = item["galleryURL"][0] #get each picture
```

```
                    ebay_title = item["title"][0] #get each title
                    array_for_titles.append(ebay_title) #add title to titles array
                    array_for_images.append(picture_of_listing) #add images to images array
                    price_of_shirt = item['sellingStatus'][0]["convertedCurrentPrice"][0]['__value__'] #get each price
                    array_for_prices.append(price_of_shirt) #add price to prices array
                window.destroy()
                price = calculate_price(array_for_prices, year) #calculate the price of the item using calculate_price sub routine
                words = []
                word_array = []
                word_count_array = []
                if len(array_for_titles) > 0: #make sure there are titles in the array
                    for title in range(0,len(array_for_titles)):
                        split_title = array_for_titles[title].split() #split each title into words by spaces
                        for word in split_title:
                            word = word.lower()
                            words.append(word) #add each word in the titles to the words array
                    for word in words: #search through each word in the words array
                        if len(word_array) == 0:
                            word_array.append(word) #add the first word to the word array
                            word_count_array.append(1) #increase the count of that word by 1
                        else:
                            if word in word_array: #if the word already is in the word_array increase the count of that word by 1
                                index = word_array.index(word)
                                word_count_array[index] = word_count_array[index] + 1
                            else: #otherwise add the word to the words_array and add a new count for that word
                                word_array.append(word)
                                word_count_array.append(1)
                    title_words = [] #array for containing the words for the title
                    index = 0
                    for num in word_count_array:
                        if num >= len(array_for_titles)/2.75:
                            title_words.append(word_array[index]) #if the word occurs regularly then add the word to the title_words a
rray
                        index +=1
                    for title_word in range(0,len(title_words)):
                        title_words[title_word] = title_words[title_word].capitalize() #make each word have a capital at start
                    title =  " ".join(title_words) #join the words together to make the title
                else:
                    title = club_name + " Football Soccer Home Shirt Year " + year + " Size UK " + size_actual + " Good Condition" #de
fault title
                show_listing_window = Tkinter.Tk() #window which shows the user's listing and previously sold items similar
                show_listing_window.title("Listing")
```

```
192.                 show_listing_window.geometry('1000x1000')
193.                 show_listings(title, price, postage_price, show_listing_window, email, file_path)
194.             except:
195.                 title = club_name + " Football Soccer Home Shirt Year " + year + " Size UK " + size_actual + " Good Condition" #defaul
    t title
196.                 valid_price = False
197.                 price = 0
198.                 while valid_price == False:
199.                     try:
200.                         price = float(simpledialog.askstring('Price','No listings found, please input a price', parent=window)) #make
    sure user inputs a valid price
201.                         if price < 0.99:
202.                             tkMessageBox.showinfo('Invalid price', 'Please input a valid price')
203.                         else:
204.                             valid_price = True
205.                     except:
206.                         tkMessageBox.showinfo('Invalid price', 'Please input a valid price')
207.                 window.destroy()
208.                 show_listing_window = Tkinter.Tk() #window which shows the user's listing and previously sold items similar
209.                 show_listings(title, price, postage_price, show_listing_window, email, file_path)
210.             if len(array_for_images)>0: #make sure there are images in the array
211.                 scroll_window=Tkinter.Frame(show_listing_window,relief=Tkinter.GROOVE,width=500,height=500,bd=1)
212.                 scroll_window.grid(row=8,column=0)#making a scroll bar section with tkinter
213.                 global canvas
214.                 canvas=Tkinter.Canvas(scroll_window)
215.                 scroll_frame=Tkinter.Frame(canvas)
216.                 myscrollbar=Tkinter.Scrollbar(scroll_window,orient="vertical",command=canvas.yview)
217.                 canvas.configure(yscrollcommand=myscrollbar.set)
218.                 myscrollbar.pack(side="right",fill="y")
219.                 canvas.pack(side="left")
220.                 canvas.create_window((0,0),window=scroll_frame,anchor='nw')
221.                 scroll_frame.bind("<Configure>",scroll_box)
222.                 for i in range(0,len(array_for_prices)): #display each image of recently sold items in scrol bar
223.                     img = ImageTk.PhotoImage(Image.open(urlopen(array_for_images[i])))
224.                     panel = Tkinter.Label(scroll_frame, image = img)
225.                     panel.image = img
226.                     panel.grid(column=0,row=i)
227.                     Tkinter.Label(scroll_frame,text=str(array_for_prices[i])).grid(row=i,column=2) #display the price next to the imag
    e
228.
229.         def quicksort(array, start, end): #quicksort algorithm to order the prices
230.             low = start
```

```
            high = end
            pivot = array[int((low+high)/2)]
            while low<=high:
                while array[low] < pivot:
                    low +=1
                while pivot < array[high]:
                    high-=1
                if low <= high:
                    temp = array[low]
                    array[low] = array[high]
                    array[high] = temp
                    low+=1
                    high-=1
            if start<high:
                quicksort(array, start, high) #recursively using the quicksort subroutine
            if end > low:
                quicksort(array,low, end) #recursively using the quicksort subroutine
            return array

        def calculate_price(array_for_prices, year): #sub routine for calculating the price of the item
            total_price = 0
            copy_of_array_for_prices = []
            for i in range(0,len(array_for_prices)):
                copy_of_array_for_prices.append(array_for_prices[i]) #make a copy of the original array so the original array can be u
    sed later on
            sorted_array = quicksort(copy_of_array_for_prices, 0, len(copy_of_array_for_prices)-
    1) #order the array using the quicksort sub routine
            length_of_array = len(array_for_prices) #get the number of prices
            lower_quartile_position = int(math.ceil(length_of_array/4)) #find lower quartile postition
            upper_quartile_position = int(math.ceil((length_of_array/4)*3)) #finf the upper quartile position
            inter_quartile_range = float(sorted_array[int(upper_quartile_position)])-
    float(sorted_array[int(lower_quartile_position)]) #work out the inter quartile range
            if int(year) < 1980: #the older the shirt, the greater the interquartile range usually due to some very expensive shirts,
    so the difference between the bounds needs to be smaller for older shirts, otherwise there will be no outliers
                difference = 0.05 * inter_quartile_range
            elif int(year) < 2000:
                difference = 0.1* inter_quartile_range
            elif int(year) < 2010:
                difference = 0.2*inter_quartile_range
            elif int(year) < 2015:
                difference = inter_quartile_range
            else:
```

```
269.                difference = 1.5*inter_quartile_range
270.            lower_bound = float(sorted_array[int(lower_quartile_position)])-difference #work out lower bound for outliers
271.            upper_bound = float(sorted_array[int(upper_quartile_position)])+difference #work out upper bound for outliers
272.            for i in range(0,len(sorted_array)): #get rid of any values in array that are lower than the lower bound or higher than th
    e upper bound
273.                if float(sorted_array[i]) < lower_bound or float(sorted_array[i])>upper_bound:
274.                    sorted_array[i] = 0
275.            for i in range(0,len(sorted_array)):
276.                total_price += float(sorted_array[i]) #add all the items in the array together
277.            price = (math.ceil(total_price / length_of_array))-
    0.01 #get average of prices, round up and take off 0.01 to get a 99p at the end
278.            return price
279.
280.        def show_listings(title, price, postage_price, show_listing_window, email, file_path): #sub routine for displaying the user's
    listing
281.            label_title = Tkinter.Label(show_listing_window, text=title, font=("Arial Bold", 10)) #contains lots of labels and buttons
     showing information
282.            label_title.grid(column=0,row=0)
283.            image_of_shirt = ImageTk.PhotoImage(Image.open(settings_array[2]))
284.            panel = Tkinter.Label(show_listing_window, image = image_of_shirt)
285.            panel.image = image_of_shirt
286.            panel.grid(column=0,row=1)
287.            label_description = Tkinter.Label(show_listing_window, text=title, font=("Arial Bold", 10))
288.            label_description.grid(column=0,row=2)
289.            label_price = Tkinter.Label(show_listing_window, text=str(price), font=("Arial Bold", 10))
290.            label_price.grid(column=0,row=3)
291.            label_postage_price = Tkinter.Label(show_listing_window, text=str(postage_price), font=("Arial Bold", 10))
292.            label_postage_price.grid(column=0,row=4)
293.            label_sub_heading = Tkinter.Label(show_listing_window, text="Recently sold shirts", font=("Arial Bold", 10))
294.            label_sub_heading.grid(column=0,row=7)
295.            button_change_title = Tkinter.Button(show_listing_window, text="Change title", font=("Arial Bold", 10), command=lambda: ch
    ange_title(show_listing_window, label_title))
296.            button_change_title.grid(column=1,row=0)
297.            button_change_description = Tkinter.Button(show_listing_window, text="Change description", font=("Arial Bold", 10), comman
    d=lambda: change_description(show_listing_window, label_description))
298.            button_change_description.grid(column=1,row=2)
299.            button_change_price = Tkinter.Button(show_listing_window, text="Change price", font=("Arial Bold", 10), command=lambda: ch
    ange_price(show_listing_window, label_price))
300.            button_change_price.grid(column=1,row=3)
301.            button_change_postage_price = Tkinter.Button(show_listing_window, text="Change postage price", font=("Arial Bold", 10), co
    mmand=lambda: change_postage_price(show_listing_window, label_postage_price))
302.            button_change_postage_price.grid(column=1,row=4)
```

```
303.            button_confirm_listing = Tkinter.Button(show_listing_window, text="List", bg="green", font=("Arial Bold", 10), command=lam
    bda: list_item(label_title.cget('text'),label_description.cget('text'),label_price.cget('text'), label_postage_price.cget('text'), sh
    ow_listing_window, email, file_path))
304.            button_confirm_listing.grid(column=1,row=5)
305.            progress_bar = ttk.Progressbar(show_listing_window, length=200)
306.            progress_bar.grid(column=0,row=5)
307.            progress_bar['value']=75
308.
309.        def list_item(title, description, price, postage_price,window, email, file_path): #sub routine for listing an item on eBay
310.            api = Trading(config_file="ebay.yaml", siteid=3)
311.            with Image.open(file_path) as user_image: #upload the user's image to eBay so can be accessed by the API
312.                user_image.thumbnail((1600,1600))
313.                with io.BytesIO() as image:
314.                    user_image.save(image, "JPEG")
315.
316.                    files = {'file': ('EbayImage', image.getvalue())}
317.                    pictureData = {
318.                        "WarningLevel": "High",
319.                        "PictureSet":'Supersize',
320.                        "PictureName": "Test"
321.                    }
322.                    internet = False
323.                    while internet == False:
324.                        try: #make sure there is a connection to the internet
325.                            response = api.execute('UploadSiteHostedPictures', pictureData, files=files)
326.                            picture = (response.reply.SiteHostedPictureDetails.FullURL)
327.                            internet = True
328.                        except:
329.                            tkMessageBox.showinfo('No internet connection or invalid eBay token', 'Please connect to the internet or g
    et new eBay token')
330.
331.            api_request = { #information for the listing
332.                "Item": {
333.                    "Title": title,
334.                    "Country": "GB",
335.                    "Location": "GB",
336.                    "Site": "UK",
337.                    "ConditionID": "3000",
338.                    "PaymentMethods": "PayPal",
339.                    "PayPalEmailAddress": email,
340.                    "PictureDetails": {"PictureURL": [picture]},
341.                    "PrimaryCategory": {"CategoryID": "123490"},
```

```
                    "Description": description,
                    "ListingType": "FixedPriceItem",
                    "ListingDuration": "GTC",
                    "StartPrice": price,
                    "Currency": "GBP",
                    "ReturnPolicy": {
                        "ReturnsAcceptedOption": "ReturnsAccepted",
                        "RefundOption": "MoneyBack",
                        "ReturnsWithinOption": "Days_30",
                        "ShippingCostPaidByOption": "Buyer"
                    },
                    "ShippingDetails": {
                        "ShippingServiceOptions": {
                            "FreeShipping": "False",
                            "ShippingService": "UK_myHermesDoorToDoorService",
                            "ShippingServiceCost": postage_price
                        }
                    },
                    "DispatchTimeMax": "2"
                }
            }
            try:
                api.execute("AddItem", api_request) #list the item on eBay
                tkMessageBox.showinfo('Listing complete', 'Your item has been published on eBay')
            except:
                tkMessageBox.showinfo('Invalid listing', 'This listing already exists')
            window.destroy()

        def change_price(window, existing_label): #subroutine for when user wants to change the price
            valid_price = False
            while valid_price == False:
                try:
                    price = float(simpledialog.askstring('Change price','Please input a price', parent=window))
                    if price < 0.99:
                        tkMessageBox.showinfo('Invalid price', 'Please input a valid price')
                    else:
                        valid_price = True
                except:
                    tkMessageBox.showinfo('Invalid price', 'Please input a valid price')
            existing_label.configure(text=str(price))

        def change_title(window, existing_label): #subroutine for when user wants to change the title
```

```
384.            valid_title = False
385.            while valid_title == False:
386.                try:
387.                    title = simpledialog.askstring('Change title','Please input a title', parent=window)
388.                    if title != '':
389.                        existing_label.configure(text=title)
390.                        valid_title=True
391.                except:
392.                    tkMessageBox.showinfo('Invalid title', 'Please input a valid title')
393.
394.        def change_description(window, existing_label): #subroutine for when user wants to change the description
395.            valid_description = False
396.            while valid_description == False:
397.                try:
398.                    description = simpledialog.askstring('Change description','Please input a description', parent=window)
399.                    if description != '':
400.                        existing_label.configure(text=description)
401.                        valid_description=True
402.                except:
403.                    tkMessageBox.showinfo('Invalid description', 'Please input a valid description')
404.
405.        def change_postage_price(window, existing_label): #subroutine for when user wants to change the postage price
406.            valid_price = False
407.            while valid_price == False:
408.                try:
409.                    price = float(simpledialog.askstring('Change postage price','Please input postage price', parent=window))
410.                    if price < 0.01:
411.                        tkMessageBox.showinfo('Invalid postage price', 'Please input a valid postage price')
412.                    else:
413.                        valid_price = True
414.                except:
415.                    tkMessageBox.showinfo('Invalid postage price', 'Please input a valid postage price')
416.            existing_label.configure(text=str(price))
417.
418.        def open_file(main_menu):
419.            if len(clubs) != int(settings_array[6]): #number of clubs in clubs text file should match the number of output nodes for t
    he neural network
420.                tkMessageBox.showinfo('Problem with clubs or settings file', 'The number of clubs and number of output nodes do not ma
    tch in the settings and club files')
421.                end(main_menu)
422.            valid_file = False
423.            while valid_file == False:
```

```
                file_path = tkFileDialog.askopenfilename(filetypes = [('Jpeg Files', '*.jpg')]) #window for inputting an image
                if file_path != '': #checks file is actually inputted
                    img = Image.open(file_path)
                    width, height = img.size
                    if width >= 500 and height >= 500: #checks image is of an appropriate size
                        valid_file = True
                    else:
                        tkMessageBox.showinfo('Invalid image', 'Your image is too small, please input another')
                        resize = tkMessageBox.askquestion('Resize?', 'Would you like your image to be resized?\nIt is recommended you
    get a better quality image')
                        if resize == 'yes':
                            img = Image.open(file_path)
                            width, height = img.size
                            new_img = img.resize((500, 500))
                            new_img.save(file_path) #resize image to a suitable size for eBay
                            valid_file = True

            Images(file_path).resize(settings_array[1]) #resize image to 50x50 so can be fed through the neural network
            nn = neuralNetwork(7500,200,20,int(settings_array[6])) #instantiate the neural network class
            try:
                image = nn.run_with_existing_weights(Images(settings_array[2]).toarray()).matrix #pass the image through the neural ne
    twork, which produces an output array
            except:
                tkMessageBox.showinfo('Problem with settings file', 'The number of output nodes in the settings file is incorrect or t
    he file name is incorrect, please correct this and restart the program')
                end(main_menu)
            highest_index = 0
            highest = 0
            club_name = ""
            image_softmax = softmaxtrain(image)
            for i in range(0,len(image_softmax)):
              if image_softmax[i] > highest: #work out the index with the highest value, each club has an index associated with it (ch
    elsea is index 0, etc)
                highest = image_softmax[i]
                highest_index= i
            if highest > 0.18: #if lower than 0.18, then the prediction of the club not clear
                club_name = clubs[highest_index] #gets the club from the clubs.txt file
                main_menu.destroy()
                check_shirt_window = Tkinter.Tk() #window for confirming if the club predicted by the network is correct
                check_shirt_window.title("club correct?")
                check_shirt_window.geometry('200x200')
                try:
```

```
462.                     image_shirt = ImageTk.PhotoImage(Image.open(settings_array[2])) #display the image inputted by the user
463.                 except:
464.                     tkMessageBox.showinfo('Problem with settings file', 'The file path in the settings file does not exist, please upd
    ate the settings file and then restart the program')
465.                     check_shirt_window.destroy()
466.                     sys.exit()
467.                 panel_shirt = Tkinter.Label(check_shirt_window, image = image_shirt)
468.                 panel_shirt.image = image_shirt
469.                 panel_shirt.grid(column=0,row=0, padx=50,pady=50)
470.                 progress_bar = ttk.Progressbar(check_shirt_window, length=200)
471.                 progress_bar.grid(column=0,row=2)
472.                 progress_bar['value']=20 #update progress bar
473.                 correct_shirt = tkMessageBox.askquestion('Shirt', club_name + '?')
474.                 if correct_shirt == 'yes':
475.                     check_shirt_window.destroy()
476.                     window_for_info_being_added(file_path, club_name) #if program gets the right club go to sub routine where user inp
    uts information
477.                 elif correct_shirt == 'no': #if the program gets the wrong club
478.                     valid_title = False
479.                     while valid_title == False:
480.                         try:
481.                             club_name = simpledialog.askstring('Input actual club','Please input the actual club name of the shirt', p
    arent=check_shirt_window) #user inputs actual club name
482.                             if club_name != '': #make sure user inputs something
483.                                 if club_name in clubs: #check if the club inputted is one of the valid ones in the clubs.txt file
484.                                     valid_title = True
485.                                 else:
486.                                     tkMessageBox.showinfo('Invalid club', 'Please input a valid club')
487.                             else:
488.                                 tkMessageBox.showinfo('Invalid club', 'Please input a valid club')
489.                         except:
490.                             tkMessageBox.showinfo('Invalid club', 'Please input a valid club')
491.                     try:
492.                         inputs = Images(settings_array[3]).toarray() #input for training neural network with image inputted by user
493.                     except:
494.                         tkMessageBox.showinfo('Problem with settings file', 'The file path in the settings file does not exist, please
     update the settings file and then restart the program')
495.                         check_shirt_window.destroy()
496.                         sys.exit()
497.                     target = []
498.                     for club in clubs:
499.                         if club == club_name:
```

```
                            club_index = clubs.index(club) #get the club index by finding its position in the clubs.txt file
                    for i in range(0,int(settings_array[6])):
                        target.append(0) #make an array of the length of the number of clubs
                    target[club_index] = 1 #make the club index a 1 for the target for the neural network
                    nn.train_with_existing_weights(inputs,target,nn) #train the neural network with the input and target
                    check_shirt_window.destroy()
                    window_for_info_being_added(file_path, club_name) #go to the sub routine where the user inputs details of shirt
            else:
                tkMessageBox.showinfo('Unknown shirt', 'Unsure what club this is, please try another photo')

        main()
```