

ANPR USING MACHINE LEARNING TECHNIQUES

MATTHEW CASEY

[CENTRE NUMBER: 64395]
[CANDIDATE NUMBER: 9430]

CONTENTS

Research.....	3
Godalming college	3
Research Methods - Analysis	4
The Interview	5
Questions	5
The Questionnaire	6
The Interview and analysis	6
Transcription	7
Evidence of the Issues	8
Questionnaire – Analysis	9
UK number plate system.....	11
Existing ANPR Systsems	13
Secondary data analysis	15
Neural Networks.....	16
Research.....	16
Structure of a Neural Network.....	16
The Activation Function	17
Analysis	22
IPSO Charts.....	22
Valid Permit System - Pseudocode.....	23
Current System – Flow Chart (Level 0).....	23
Data flow	25
Data dictionary	27
Databases Normalization and E-R Diagram	27
Case Diagram.....	29
College Site Analysis.....	30

Summary of Research and Analysis.....	31
Requirements.....	31
Justification Of Requirements	34
Design.....	36
User Interface	36
Neural Network Design	39
Matrix Pproperties	39
Key Matrix Functions	40
Feed forward algorithm	44
Backpropagation Alogorithm	45
Saving The Network To A File.....	47
Training a neural network	48
Training Dataset.....	50
Image Formatting design.....	51
Database Design	55
Structure Chart	57
Class Diagram.....	2
Testing Strategy	4
Testing outline	4
Section 1 – Validation Of Inputs and User Interface	5
Section 2 – Algorithm/Processes Testing.....	6
Section 3 – Data Storage Testing.....	9
Section 4 – IO and Database Query Testing.....	10
Testing Data (Examples).....	12
Beta Testing Plan	14
Testing	15
Section 1 – Validation Of Input and User Inpterface.....	15
Section 2 – Algorithm/Processes Testing.....	17
Section 3 – Data Storage Testing.....	20
Section 4 – IO and Database Query Testing.....	21
Beta Testing	24
Demonstrating Key Algorithms	26

Backpropagation	26
Evaluation	29
To What Extent Does the Project Meet Its Requirements?	29
Independent Feedback.....	34
Overall Evaluation Of System.....	35
Technical Solution	1
Generating Training Dataset Using Python and Numpy	1
Linear Algebra Class.....	3
Linear Algebra Modules	8
Neural Network Class.....	11
Character Recognition System Class	20
Character Recognition Testing Interface.....	26
Image Formatting Modules.....	27
Character Formatting Modules	32
Image Formatting Testing Interface.....	37
Database Modules.....	39
ANPR Class	49
ANPR System Interface	51
Resources	56

RESEARCH

PROBLEM OUTLINE | CURRENT SITUATION | END-USER

GODALMING COLLEGE

Godalming College is a 6th Form College containing students mainly in the age range of 16-18 (with some older students taking extended or delayed courses). During most students time at college they will learn to drive, so the College provides a 'Student Car Park' to accommodate the parking requirements on the College site.

Address: *Tuesley Lane, Godalming, GU7 1RS*

Phone: *01483 423526*

With an increasing number of students it has become clear that the College struggles with an overcrowded car park and struggles to monitor the flow of traffic in and out of the site. Current systems are in place to restrict access to certain vehicles and attempt to keep the volume of traffic down. A lack of control in this can lead to a number of problems:

- Students being unable to park at certain times of the day.
- Students who have entered the carpark and been unable to park, therefore go on to do so illegally or dangerously.
- Vehicles that are not registered coming onto the site and parking.
- College Staff spending considerable time dealing with complaints of inconsiderate parking

Evidence for these issues was confirmed in an email from John Erasmus (Estate Manager) on the 10/09/2019 at 16:10 when he outlined to all college students that **“we have already received complaints from local residents about cars being parked dangerously and inconsiderately on roads around the College, particularly those in close proximity of the college”**. He continued by focusing on the issues this provides the college: **“College staff spend considerable time handling complaints, it is also disruptive to you as students if we have to get you out of a lesson to move your car.”** He then goes on to talk about the need for a permit to park on site and urges students to consider the need for them to drive to college from both environmental and college capacitance standpoint.

RESEARCH METHODS - ANALYSIS

1. An **Interview** will allow for best assessment of the current situation from a single person. Although time consuming, a well conducted interview can extract in-depth knowledge of the situation at hand – especially if the interviewee is willing and chosen appropriately. Recording data from an interview is more difficult than other areas; answers will be more substantial and complex. The interview could potentially be transcribed to allow for post-interview evaluation of the raised issues, however ensuring the interviewee is aware of this is key.
2. A **Questionnaire** can be used to evaluate a larger number of users. The collected data will often be more quantitative, so easier for analysis purposes. It could be argued that the quality of collected data will decrease as the research method is less formal, which could lead to a large volume of un-useful data.
3. It is likely that **Secondary Data Analysis** will allow me to collect large amounts of useful accurate data that can contribute to the project. This is a viable technique for research with limited time and resources and involves the analysis of data collected by someone else for another primary purpose.

THE INTERVIEW

The interview will be conducted on the 17/09/2019 on the college site with a member of the College Security or Grounds Team. Following are a set of pre-prepared interview questions which will allow me to extract the most possible information from the interview. The responses will be paraphrased (by the interviewer) and recorded as key points under each question heading

QUESTIONS

- “Describe the current system used to control cars being parked on site without a permit? What is your role within this system? ”
 - Allows me to gain an understanding into the current system in place. The requirements that are currently being met will become obvious and so to may their limitations.
- “What would you say the key issues are with the current way in which the college car park is managed?”
 - Identify current key issues to be worked on.
- “What are the key ways in which you are monitoring the usage of the college car park?”
 - Allows me to construct a visual representation describing their current processes.
- “What, to you, would be a crucial part of a successful system?”
- “To what extent do you feel an ANPR system would be useful for the college?”
- “How important is validation and high levels of accuracy in a system to be used in college?”
 - Evaluate their need and requirements.
- “Is it a difficult task to identify invalid cars entering the car park?”
- “Is the car park often full? If so to what extent does this disrupt a student’s journeys into school and the flow of traffic around college?”
 - Assesses the need for an automated system
- “How many spaces are available in the college car park? How many students have a valid permit and how does this vary throughout the year?”
- “To what extent are the college staff and resources being wasted on parking related issues?”
 - Find out the extent of the need for a system to control the situation and some quantitative data behind the system.

- “If a solution was to be produced, when would a specific deadline for this project be?”
 - Find out the time frame for such a project to be completed

THE QUESTIONNAIRE

I feel that supplying a questionnaire to students at Godalming College will provide me with a new perspective on the problem and potentially provide new requirements for a solution. This questionnaire, however, is designed to be given to both staff and students. All questionnaire will be filled out anonymously.

- 1) How satisfied are you with the current system of identifying invalid parking on college premises?
Not satisfied 1 2 3 4 5 6 7 8 9 10 *Very satisfied*
- 2) How often do you see cars parking on premises without a license?
Not often 1 2 3 4 5 6 7 8 9 10 *Very often*
- 3) How useful would a system that monitored whether or not the carpark was full or not be?
Not useful 1 2 3 4 5 6 7 8 9 10 *Very useful*
- 4) How would you rate the current car parking system at college?
Not good 1 2 3 4 5 6 7 8 9 10 *Very good*
- 5) How much of a need do you feel is present for a system which monitors traffic on the college site?
Not needed 1 2 3 4 5 6 7 8 9 10 *Needed*
- 6) How would you feel, from a data protection point of view, about the tracking of your vehicle entering the college premises – and the association of this with data about you?
Not 1 2 3 4 5 6 7 8 9 10 *Very Comfortable*
- 7) How many times a week do you park in college?
 1 2 3 4 5 *Not at all*

THE INTERVIEW AND ANALYSIS

I conducted an interview with Christopher Keegan, a member of the grounds team at Godalming College, on the 24/09/2019. We discussed the current situation at the college and the proposal to update the system.

TRANSCRIPTION

- “Describe the current system used to control cars being parked on site without a permit? What is your role within this system? ”

When a student submits an application form for a parking permit they are given a ‘permit sticker’ to stick to their front windscreen. This sticker needs to be in view from outside the front of the car. At key points in the day, e.g. 8:00-8:45 or 16:15-16:45, a member of the grounds team stands down near the entrance of the college and monitors the cars entering the premises – looking for a permit stuck to the front windscreen. If a car is identified without a sticker it can be turned away. But often this is not a focus of the member of staff, but they are thinking about controlling the flow of traffic and ensuring a safe entry to the college. It is therefore quite likely that a car can enter the premises without a permit.

- “What would you say the key issues are with the current way in which the college car park is managed?”

As mentioned above, it is not the key focus of the member of staff at key times to ensure all cars have a permit. To further that, for the majority of the day there is no member of staff monitoring the entrance at all, so any car can enter the premises.

- “What are the key ways in which you are monitoring the usage of the college car park?”

Currently during the day all we do is occasional checks of the college car park to ensure all cars have a permit and are parked safely and responsibly. If a car is spotted to be parked badly a student will be removed from their lessons to park appropriately. This system is very inefficient on its use of staff time at college.

- “What, to you, would be a crucial part of a successful system?”

Being alerted whenever an invalid vehicle enters the premises would be a vital in helping us to sort out incidents quickly and efficiently. A general log of all cars currently on the premises and when they arrived and left would be of great use for security purposes.

- “To what extent do you feel an ANPR system would be useful for the college?”

A well-working ANPR system would benefit the college hugely and would allow us to be more secure on site and divert resources, which are currently spent on monitoring the car park, to other areas.

- “How important is validation and high levels of accuracy in a system to be used in college?”

A high level of accuracy in determining whether or not a vehicle is allowed to park on the premises would be desirable. If the system consistently reported invalid cars that turned out to be valid it would reduce the effectiveness of it as a resource to save staff time (as it would be wasted by checking up on false detections). A few cars that were valid making it through would be acceptable.

- “Is it a difficult task to identify invalid cars entering the car park?”

Looking for a specific number plate in the car park is not too difficult. The most time is spent by checking every car in the car park for a parking permit. If the system provided us with a list of number plates that should not be parked on the premises this would vastly improve the efficiency of the process.

- “Is the car park often full? If so to what extent does this disrupt a student’s journeys into school and the flow of traffic around college?”

In the earlier parts of the year, when many lower sixth students have not passed their driving tests, the cars par will generally have spaces available throughout the day. Peak times in the day are still very busy however. As the year progresses and more students pass, the car park does become very full and many students will not be able to park. The situation is not helped by the fact that many students waste time by driving in college before determining whether or not there are spaces. They are then rushed to find a space outside college and often park badly.

- “How many spaces are available in the college car park? How many students have a valid permit and how does this vary throughout the year?”
- “To what extent are the college staff and resources being wasted on parking related issues?”

As I mentioned earlier, in a number of areas of the system we currently run the time of college staff is consistently being wasted. Even a small prove to the efficiency of the system could make a large difference.

- “If a solution was to be produced, when would a specific deadline for this project be?”

In theory, a good deadline would be the start of the next college year. This would be before September 2019. Assuming other hardware would need to be set up, this deadline could be earlier to ensure we can set up and test the system.

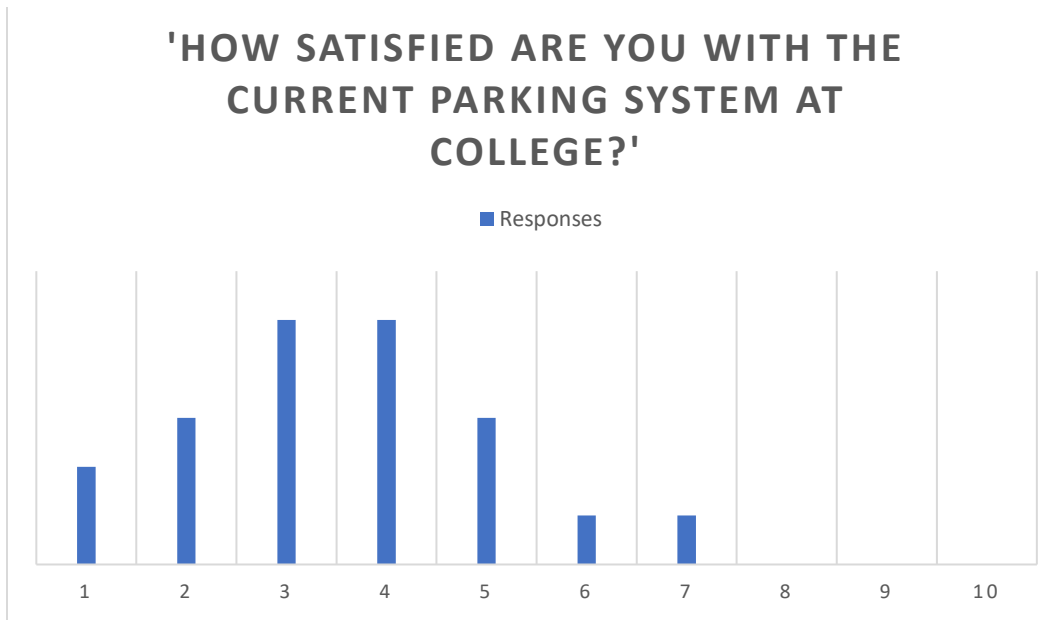
EVIDENCE OF THE ISSUES

These two images were taken on the 23/09/2019 at 16: 19 and show the issues that the college can face with congestion in the car park. This shows how overcrowded the car park can get and the lack of management of who can park. If the project was implemented it should be able to reduce the volumes of cars being parked and therefore reduce congestion at these peak traffic times.

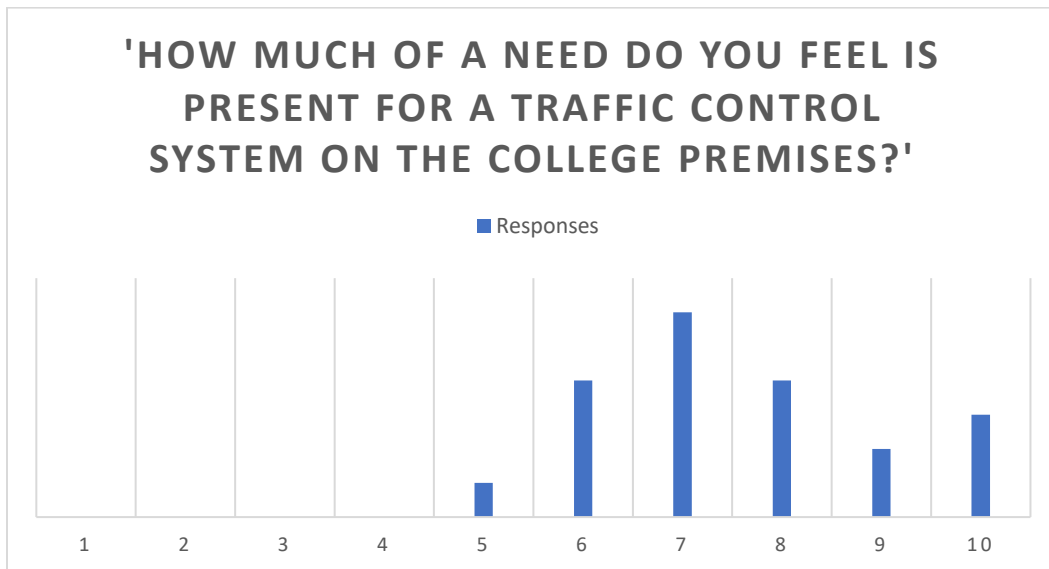


QUESTIONNAIRE – ANALYSIS

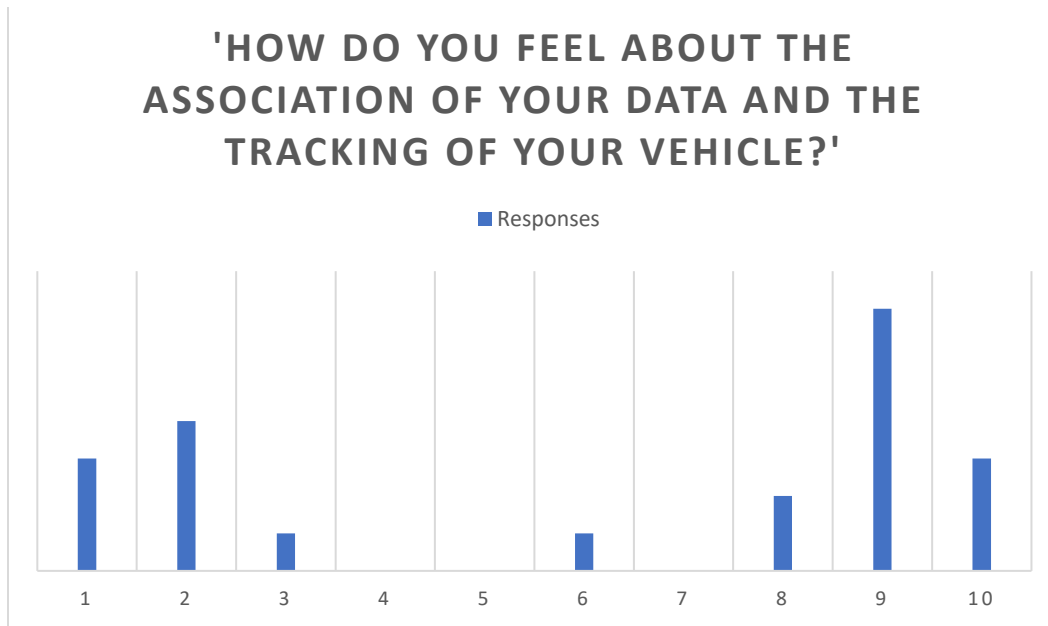
I gave the above questionnaire to 10 upper sixth students at the college. Following is a graphical summary of the responses.



The strong negative skew on this data suggests that students feel the need for a new/updated system. This fails to provide me data on what areas of the current system need changing so that will need to be considered later. It does however strongly hint that the current systems fails to complete the task (of monitoring the parking) to even a satisfactory level. Looking into whether or not this opinion comes from an inability to park in busy times, a lack of knowledge of how full the car park is or simply there often being invalid cars parked in the car park.

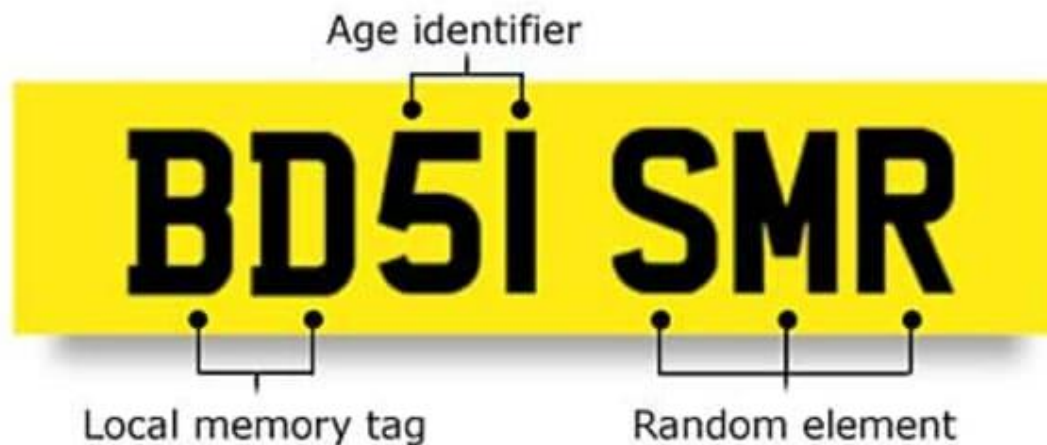


For the positive skew of this data I can extend on the conclusion made above. The fact that a need to control the traffic on college premises is viewed to be strongly required, hints that the problem largely lies with invalid parking on college premises (which needs to be controlled). Its clear that the car park simply being full is not just the issue, but the lack of control in who is parking there, which is causing it to be full.



This last graph shows an interesting split of opinion on how students feel about the monitoring of their vehicle entering college premises (and the association of this data with them as a student entering the premises). I will evaluate the data protection side of the problem later on, but here it is clear there is a divide in opinion. Research will need to be done into what data I can actually associate. The system I provide may need to simply identify number plates and then the association with other data is then performed by the college.

UK NUMBER PLATE SYSTEM



Current System

Introduced in 2001, they are arranged in the format of two letters, two numbers, followed by 3 more letters. The second two numbers are used as 'age identifiers'. They tell you which six month period the car was first registered (either March to August or September to February).

Year	1st March - August End	1st September - February End
2001/02		51
2002/03	02	52
2003/04	03	53
2004/05	04	54
2005/06	05	55
2006/07	06	56
2007/08	07	57
2008/09	08	58
2009/10	09	59
2010/11	10	60
2011/12	11	61
2012/13	12	62
2013/14	13	63
2014/15	14	64
2015/16	15	65
2016/17	16	66
2017/18	17	67
2018/19	and so on	until 50/00 in 2050/51

This system will be in place until 2050. A March to August plate can be identified by a number less than 50 and vice versa.

The first two letters are known as a local memory tag and show where the vehicle was registered. The first letter represents the region and the second the DVLA local office.

The last three letters are chosen randomly and are allocated by the dealership when the car is registered. These will ensure each vehicle has a unique identity.

Some characters are never used due to their similarities to other characters. These include:

- I (Similar to 1)
- O (Similar to 0)
- U (Similar to V)
- Z (Similar to 2)

*It is likely these changes are put in for the benefit of existing ANPR systems to improve accuracy.

The Old System

From 1983, number plates followed the format X111 XXX. The first letter was representing of a year in which the car was registered (e.g. A = 1963).

Pre 1903, dateless, registration plates are in existence and can be purchased to put on new vehicles.

Other number plate information

- Each character must be 79mm high and 50mm wide (except the number 1)
- The width of each character stroke must be 14mm
- There must be 11mm between characters of the same group.
- There must be 33mm between different groups of characters
- They work of one mandatory typeface: 'Charles Wright 2001'
- Hard to read variants of this style are prohibited, however a 3D effect version is permitted.
- Optionally they can display a national emblem
- Colours and reflectivity of the number plates are also specified.
- Front plates must have black characters and white background
- Back plates must have black characters and yellow background
- A coloured non-reflective border is allowed
- No other markings are permitted on the number plate

*Notably these rules are post 2001. Variants on the fonts and spacings used were present before 2001.

EXISTING ANPR SYSTEMS








This technology is used in the UK for law enforcement and to detect, deter and disrupt organised crime. Vehicle movement on UK roads is recorded by a network of 11,000 cameras that submit around 50 million ANPR hits daily to a national database. The ANPR data is stored with associated data for a year and can be analysed and used as evidence in investigations. After a hit the record is often automatically checked against a database of vehicles of interest.

Information on how these systems work is very limited so research into this area is not going to prove to be useful for the project. It can be assumed they follow a standard neural network/machine learning approach. In terms of formatting and triggering it is very unclear. Because of this I have looked into a private ANPR system and the results they have seen.

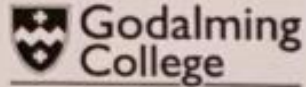
(REF: <https://sensorable.io/articles/anpr-accuracy-test/index.html>)

This article talks about the accuracy of their ANPR system set up at various points throughout the day. They adjusted many factors on their camera including: shutter speed, max vehicle speed and time of day. Many of which may not be required in my solution. A key point they made was how the success

rate dropped rapidly in the evening and through the night as the plates became so obscured by headlights. They noted that their camera fell short in certain scenarios: plates with spaces, short plates and plates with weird combinations. For this reason it could be a good idea to design the system for standard UK plates only, which I believe is a reasonable assumption to be made for the majority of vehicles entering the college car park. They also included a confidence threshold for their system so it would only output when it was above a certain calculated level of confidence. This would naturally cause their accuracy to be higher, but may not be able to be used in this project as getting as many vehicles as possible is crucial. The accuracy for their system came out overall at 88% and that is factoring in their confidence threshold and camera optimisation. This gives an idea of what a very high level system could achieve.

08 Jul 17:06	40km/h	FFP511 
08 Jul 17:05	42km/h	KJB38 
08 Jul 17:04	20km/h	FDE573 
08 Jul 17:03	26km/h	LLY322 
08 Jul 17:03	40km/h	JLY694 
08 Jul 17:02	41km/h	EMF495 
08 Jul 17:01	20km/h	LUC41 

 SECONDARY DATA ANALYSIS


Student Parking Permit Application

Name	
Student Number	
Vehicle Make, Model, and Colour	
Vehicle Registration No.	
Home Address	

Complete and take to the Security Office together with your Test Pass Certificate

Please tick as appropriate:

- I hold a current full driving licence
- My vehicle is insured and has a current Road Fund licence
- My vehicle has a current MOT Certificate (if applicable)
- I understand that disciplinary action will be taken if I am caught driving dangerously or recklessly, or committing anti-social behaviour such as excessive use of the horn, throwing litter, or inconsiderate parking; in College or on local roads whilst travelling to or from College.

As the year progresses the demand for permits exceeds the number of spaces available for student parking on site. Applications for permits will be considered using the following criteria to determine priority. Please tick all those that apply:

- I live more than 1 ½ miles from College
- I live more than 1 mile from a train station (on a direct line to Godalming)
- I am committed to car sharing and will give a lift to students on a regular basis
- I have a disability or medical condition that makes walking or travel by public transport difficult.
- I am an Upper Sixth student

Student signature	Date
-------------------	------

I agree to my son/daughter driving to College and I endorse the above declaration

Parent/Guardian signature	Date
Email	Tel.

This is a scan of the 'Parking Permit Application' which is required to be filled out by all college students who wish to park on the college premises. A notable part of the form is the 'Student Number'. This is unique code given to every student who enrolls at the college (constructed in the form YYNNNN where YY is the year of enrolment and NNNN is their unique exam number). This could become a useful part of identifying students from registration numbers and potentially a useful key field in a database holding records of entry.

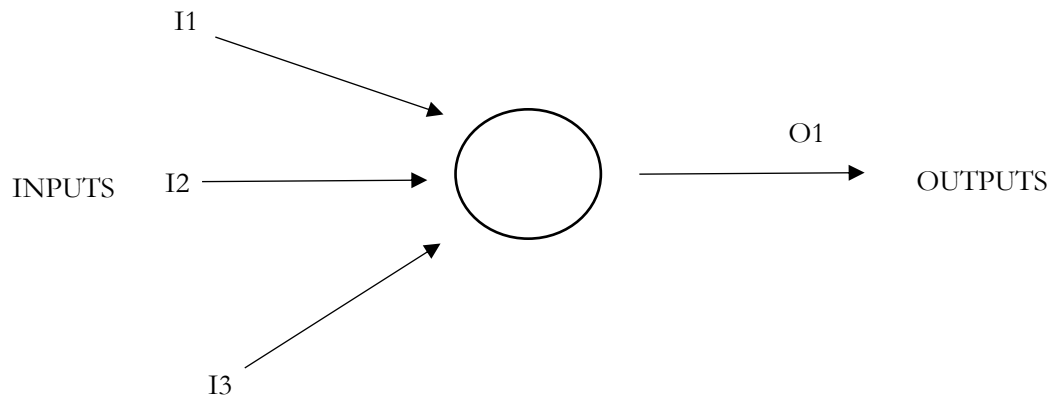
Other parts of the form require you to enter your vehicles colour, model and registration number. The registration number being the key piece of data there. Interestingly the students are required to understand that 'disciplinary action' will be taken if you are caught driving dangerously or recklessly. It is clear that a system to monitor the vehicles entering the premises will allow further enforcement of this idea.

NEURAL NETWORKS

RESEARCH

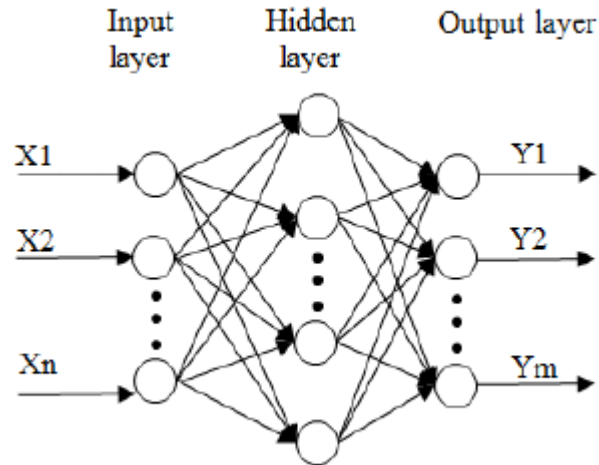
It is clear that to provide an efficient and effective solution to the problem the system will be required to recognize combinations of characters and numbers within a bitmapped image. A current field of research allows this to be achieved using artificial multi-layered neural networks. Here I will discuss the theory behind the structure of an ANN and how they work.

A neural network is composed of layers. Each layer is comprised of multiple nodes (thought of as a location in which computation occurs), which can be thought of as loosely modelling a biological brain. A node combines input data with chosen weights and biases to provide an output, which is then passes on to the next layer.



STRUCTURE OF A NEURAL NETWORK

The networks are structured so that each neuron in the previous layer of the network is connected to each neuron in the current layer. Each of these connections, which scale exponentially, has an associated weight. The value returned from the previous layer is multiplied by this weight in the neuron.



'The Function of a neuron'

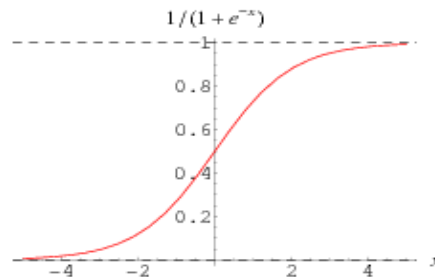
A neuron will take the summation of all the outputs of each of the previous layer nodes multiplied by their respective weights. Let n equal the number of nodes in the previous layer, X equal the previous layer and W equal the weights.

$$O = \sum_{r=1}^{r=n} X_r W_r$$

Repeating this process for every neuron in every layer will **Feedforward** as set of input values in the current network to produce a set of output values.

THE ACTIVATION FUNCTION

It is likely that the previous summation will result in an output far greater than the values returned from the previous layer. This can skew input values and result in determining the 'activation of a neuron' to be difficult. Therefore, after the summation, the output is passed through a function to normalize the value (often between 0 and 1). The logistic sigmoid function is often useful as it will take any input value and return a value between zero and 1 (and for later purposes its derivative is found through a simple process).



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

'How does a network learn?' – Propagating backwards through a network

After a set of input values (often expressed as a column vector) has been passed into a network and fed forward through the system, a set of output values should be produced (also often expressed as a vector). For many new networks the weights and biases between all connections are initialized randomly, therefore a set of inputs will not produce a desired output but – most likely – a random one. There will be a difference in each output value from its target value (the value that output neuron should hold if the network ran correctly). If we let the total error of a network for a given input equal E , O equal the set of outputs and T equal the set of targets.

$$E = \sum_{r=1}^{r=n} |T_r - O_r|$$

It is often far more useful to express this error as a cost for the network. In replace of the modulus function the difference between the target and output is squared (to ensure it is positive), then summated. Therefore, let C equal the cost of the network.

$$C = \sum_{r=1}^{r=n} (T_r - O_r)^2$$

By analysing the cost of a network for a give input we can use **backpropagation** to determine how to minimize the cost.

Derivation of the first derivative of the total error of a single layer perceptron for a single weight that resides in the 2nd layer

Say we want to calculate the change in error for a single weight, x .

$$\frac{dE_{Total}}{dW_x}$$

Using the chain rule,

$$\frac{dE_{Total}}{dW_x} = \frac{dE_{Total}}{dO_x} \times \frac{dO_x}{dO_{xIN}} \times \frac{dO_{xIN}}{dW_x}$$

As,

$$E_{Total} = (T_x - O_x)^2 + (T_{x+1} - O_{x+1})^2 + (T_{x+1} - O_{x+2})^2 \dots$$

$$\frac{dE_{Total}}{dO_x} = -1 \times 2 \times (T_x - O_x)$$

$$\frac{dE_{Total}}{dO_x} = -2(T_x - O_x) = 2(O_x - T_x)$$

As the difference between O_{IN} and O_1 is the sigmoid activation function,

$$\frac{dO_x}{dO_{xIN}} = Sig'(O_{xIN})$$

It is known that the derivative of the sigmoid function ($sig(x)$) is,

$$Sig'(x) = Sig(x)(1 - Sig(x))$$

Therefore,

$$\frac{dO_x}{dO_{xIN}} = Sig(O_{xIN})(1 - Sig(O_{xIN}))$$

Finally,

$$O_{xIN} = H_x \times W_{Hx} + (H_{x+1} \times W_{Hx+1} \dots$$

$$\frac{dO_{xIN}}{dW_{Hx}} = H_x$$

Therefore,

$$\frac{dE_{Total}}{dW_{Hx}} = 2(O_x - T_x)Sig(O_{xIN})(1 - Sig(O_{xIN}))H_x$$

This shows an equation for the rate of change of the total output error for a specific weight in the 2nd layer.

Derivation of the first derivative of the total error of a single layer perceptron for a single weight that resides in the 1st layer

Say we want to calculate the change in error for a single weight, x

$$\frac{dE_{Total}}{dW_x}$$

As the first derivative of the output error respecting a hidden layer neuron is comprised of the derivative for each output respecting that hidden neuron

$$\frac{dE_{Total}}{dH_x} = \frac{dE_1}{dH_x} + \frac{dE_2}{dH_x} + \frac{dE_3}{dH_x} \dots$$

Therefore using the chain rule,

$$\frac{dE_1}{dH_x} = \frac{dE_1}{dO_{IN}} \times \frac{dO_{IN}}{dH_x}$$

$$\frac{dE_1}{dO_{IN}} = \frac{dE_1}{dO} \times \frac{dO}{dO_{IN}} = 2(O_x - T_x)Sig(O_{xIN})(1 - Sig(O_{xIN}))$$

(*calculated above)

As,

$$O_{IN} = (W_{Hx} \times H_x) + (W_{Hx+1} \times H_{x+1}) + (W_{Hx+2} \times H_{x+2}) \dots$$

$$\frac{dO_{IN}}{dH_x} = W_{Hx}$$

Combining result in,

$$\frac{dE_{Total}}{dH_x} = 2(O_x - T_x)Sig(O_{xIN})(1 - Sig(O_{xIN}))W_{Hx}$$

Using the chain rule again,

$$\frac{dE_{Total}}{dW_x} = \frac{dE_{Total}}{dH_x} \times \frac{dH_x}{dH_{xIN}} \times \frac{dH_{xIN}}{dW_x}$$

Using the derivative of the sigmoid function we know,

$$\frac{dH_x}{dH_{xIN}} = Sig(H_{xIN})(1 - Sig(H_{xIN})) = H_x(1 - H_x)$$

And as,

$$H_{xIN} = (I_x \times W_x) + (I_{x+1} \times W_{x+1}) + (I_{x+2} \times W_{x+2}) \dots$$

$$\frac{dH_{xIN}}{dW_x} = I_x$$

Finally, by combining all of the above,

$$\frac{dE_{Total}}{dW_x} = 2(O_x - T_x)Sig(O_{xIN})(1 - Sig(O_{xIN}))W_{Hx}H_x(1 - H_x)I_x$$

ANALYSIS

DATA FLOW | DECOMPOSITION | ISPO | E-R

IPSO CHARTS

Level 0 IPSO Chart – Whole system

INPUTS Video (Live) of cars entering the college premises	PROCESSES Identify Number Plate
STORAGE Database of records	OUTPUT Record of car entering college

Level 1 IPSO Chart – Number Plate Detection

INPUTS Single frame of video	PROCESSES Isolate number plate within frame
STORAGE -	OUTPUT Unformatted image of number plate

Level 2 IPSO Chart – Image formatting

INPUTS Unformatted image of number plate	PROCESSES Format and decompose into single letters
STORAGE -	OUTPUT Formatted inputs for neural network

Level 3 IPSO Chart – Character Recognition

INPUTS Formatted image of character	PROCESSES Feed forward through neural network
STORAGE Weights and Biases of trained network	OUTPUT Recognized character

 VALID PERMIT SYSTEM - PSEUDOCODE

For a student to be given a valid parking permit they must pass the following criteria described by the following Pseudocode.

FUNCTION Check If Student Is Valid()

IF Hold Driver's License AND Insured AND Road Fund License AND Has Valid MOT AND Agree to Terms THEN

IF Distance from College IS GREATER THAN 1.5 AND Distance from Station IS GREATER THAN 1 THEN

Return TRUE

ENDIF

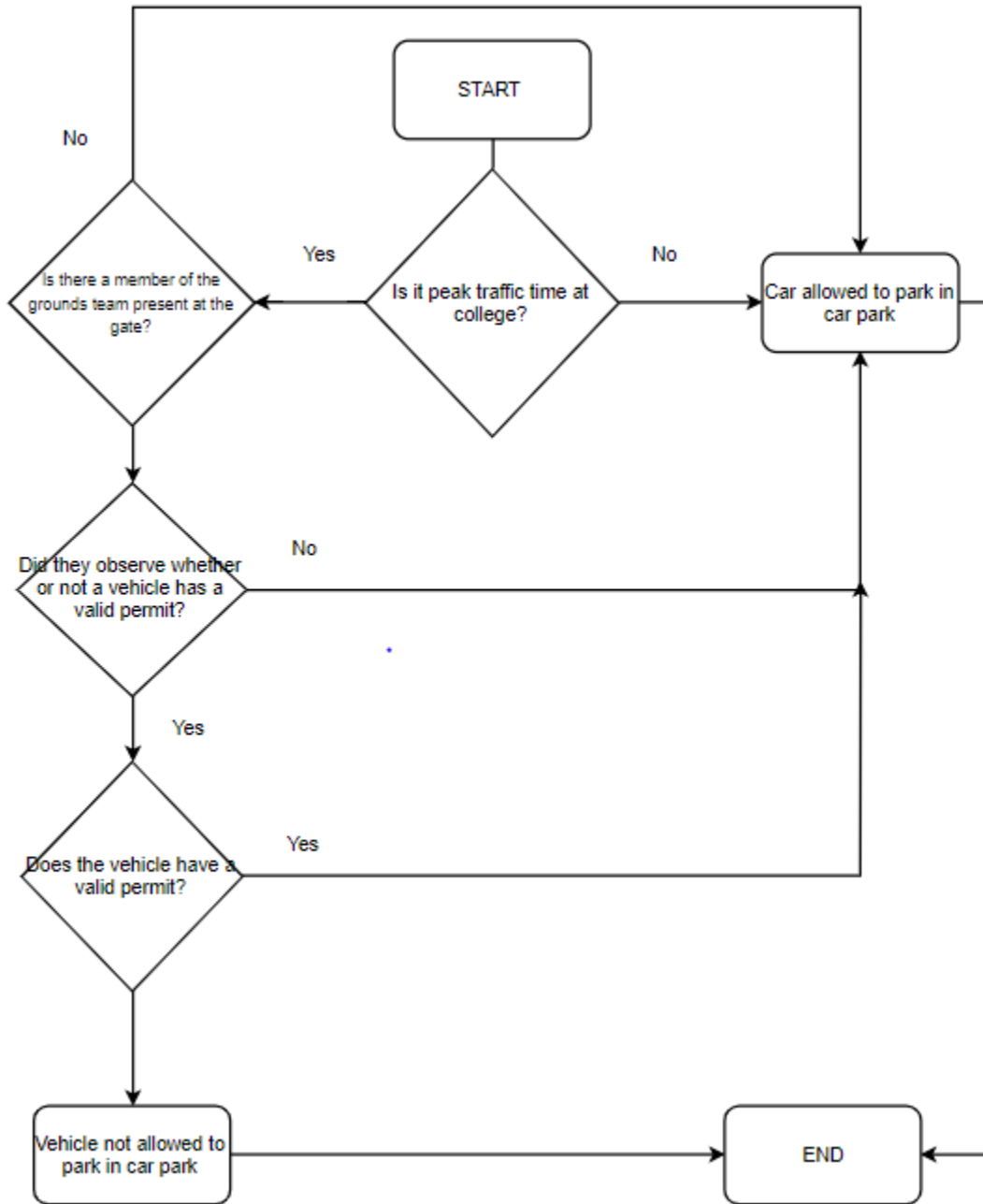
ENDIF

Return FALSE

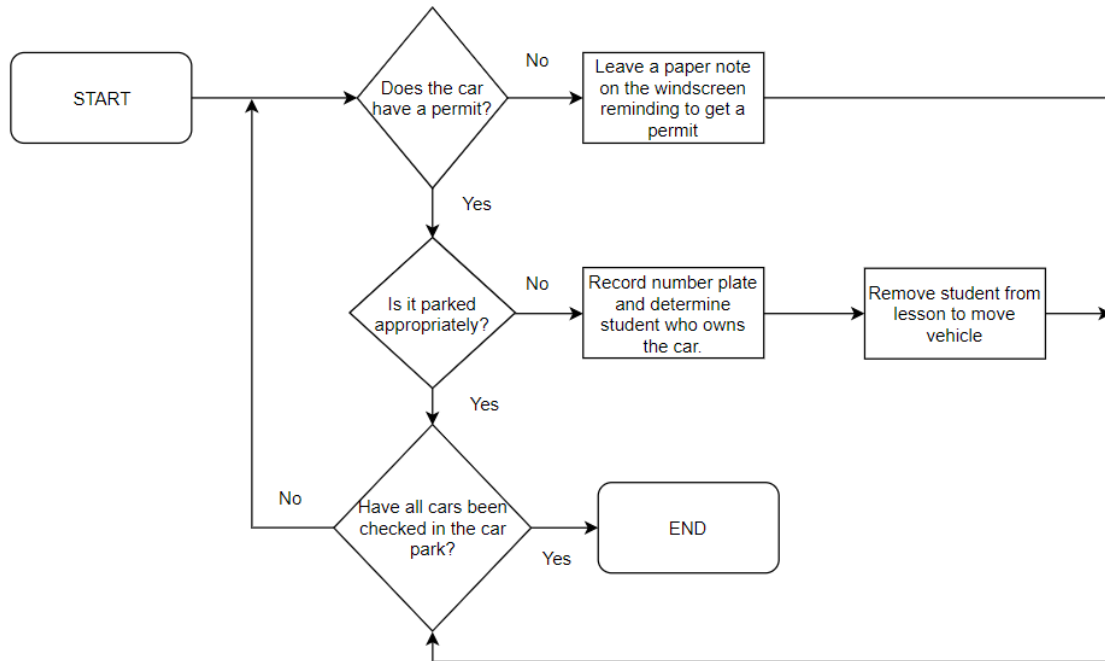
END FUNCTION

 CURRENT SYSTEM – FLOW CHART (LEVEL 0)

This describes the current system in place for when a car enters the college premises and determines whether or not it will be able to park.

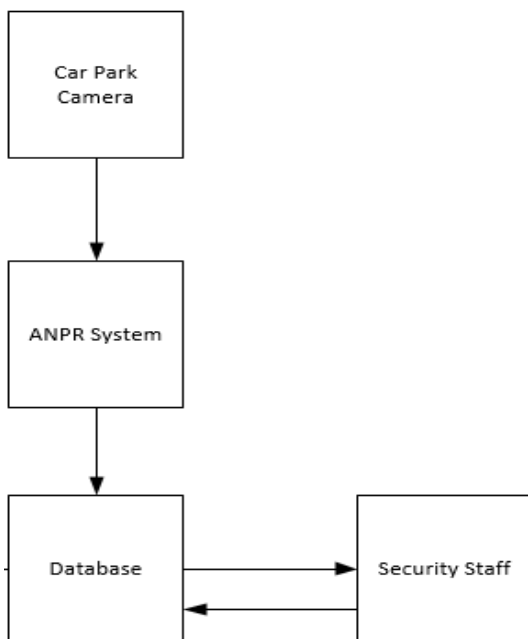


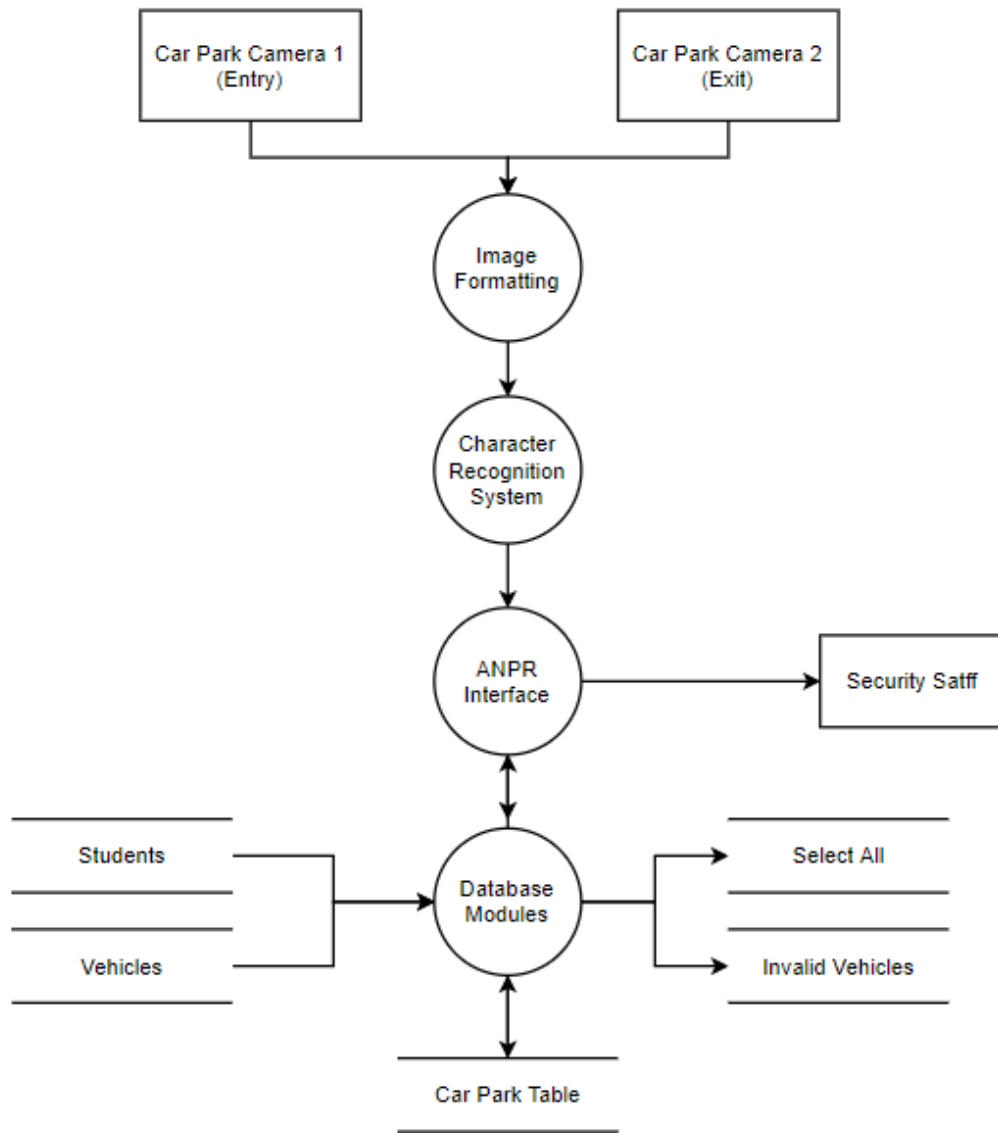
This describes the process a member of the grounds team would go through when checking the car park for invalid parking.



DATA FLOW

This provides a very abstracted view of the flow of data in the proposed system.





Key

- Square = Entity
- Circle = Process
- Double line = Data Store (Sections of Database)

 DATA DICTIONARY

Data Item	Data Type	Data Format	Description	Example
Number Plate	Varchar	LLNN LLL	A collection of letters and numbers that identifies each vehicle.	EN15 SYB
Student ID	Integer	NNNNNN	A unique identifier for each student in the college.	189430
Car Model	Varchar	Brand Model	A description of the vehicle for use in records.	Hyundai i10
Car Colour	Varchar	Colour	A description of the vehicle for use in records	White
First Name	Varchar	Name1 Name2(opt)	To identify the student owner of the car	Matthew
Second Name	Varchar	Name1 Name2(opt)	To identify the student owner of the car (for use by grounds team)	Casey
Email	Varchar	-	Stored in records for use by grounds team.	189430@godalming.ac.uk
Telephone	Varchar	0NNNN NNNNN	A number to contact the owner of the vehicle.	01234 567890
Parking Warnings	Integer	-	A count of the number of warnings the student has been given for inappropriate parking.	2

 DATABASES NORMALIZATION AND E-R DIAGRAM

Database Normalization

The database side of this project is likely to be small, however it will still represent the core end-user side of the system so will need to be well well-structured and robust. The first reason for

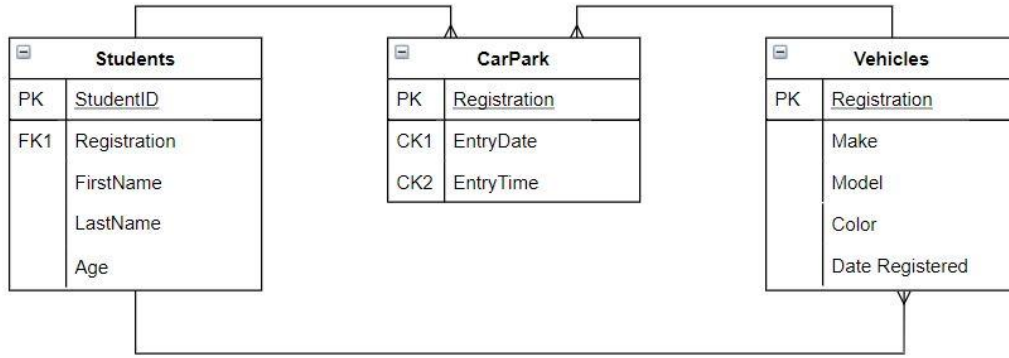
normalizing a database is to minimize duplicate data (which when the database starts to collect data over multiple days) will likely result in a far better use of storage. The second is to minimise modification issues. The database should be able to handle a student changing registration number etc. The third is to simplify queries.

1st Normal Form of a database structure is when information is stored in a relational table with each column containing atomic values and no repeating groups of columns.

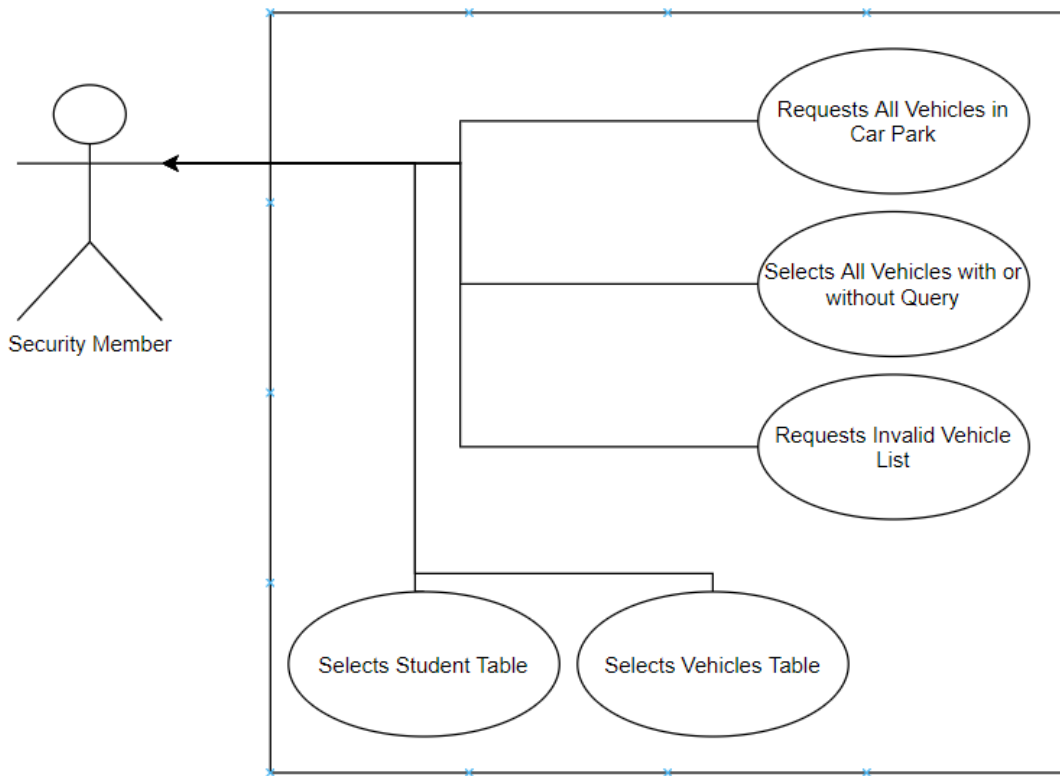
2nd Normal Form of a database structure is when all columns depend on the table's primary key.

3rd Normal Form of a database structure is when all of its columns are not transitively dependant on the primary key.

1st Normal Form (Flat File)	2nd Normal Form	3rd Normal Form
(Flat File)	(Table Students)	(Table Registered Vehicles)
<u>Car Registration Number</u>	<u>Registration</u>	<u>Registration</u>
First Name	First Name	Car Model
Last Name	Last Name	Car Make
Car Model	Car Model	Car Colour
Car Colour	Car Colour	Date Registered
Date		
Entry Time	(Table Records)	(Table Students)
Exit Time	<u>Number Plate</u>	<u>Student ID</u>
Valid Permit	<u>Entry Time</u>	Registration
	<u>Exit Time</u>	First Name
	<u>Date</u>	Last Name
		Age
		(Car Park)
		<u>Registration</u>
		<u>Entry Date</u>
		<u>Entry Time</u>
		Exit Time



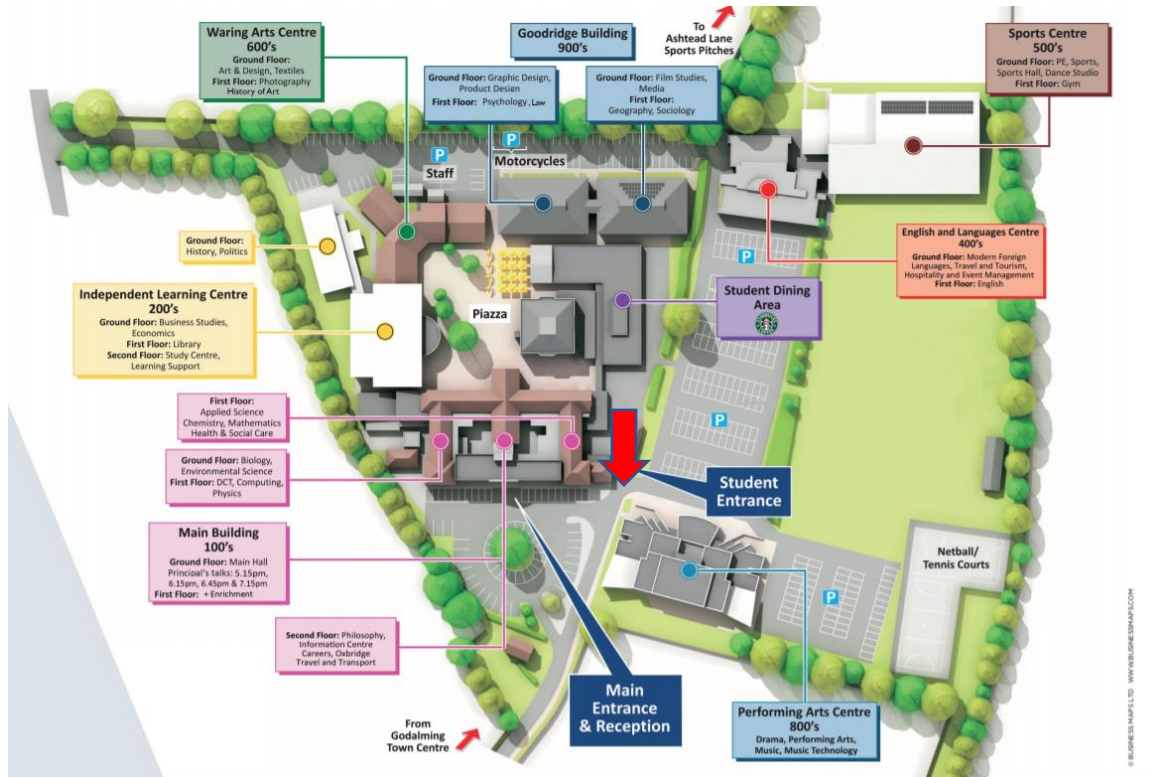
CASE DIAGRAM



There are few use cases for the proposed system and all of which are by one user, however they are outlined in the diagram above.

 COLLEGE SITE ANALYSIS

Godalming College operates on a campus site on Tuesley Ln, Godalming GU7 1RS. The site map is shown below. (REF: <https://www.godalming.ac.uk/docs/default-source/default-document-library/godalming-visitors-guide-2020-print.pdf?sfvrsn=0>)



It can be seen that there are many places to park on site all of which have a variety of different uses. The car parks near the top of the site are for staff use only and cannot be accessed by students. Staff are required to scan their ID card to enter that car park. No cameras would be placed up there as it is already a well-monitored area; no vehicles can park here without a 'permit' and congestion is low. Near the front of the site there is a temporary parking area which is used for drop off etc. I do not believe the traffic in and out of this location needs to be monitored as it is so variable.

It is the two car parks off to the right of the site which are the focus of the project. The larger one nearer the entrance is the main student car park and the smaller one is the 'overflow' car park for when the main one becomes too full. There is no system to limit access to this car park unlike the teachers one, so it would be proposed that the cameras are placed at the entrance/exit to this car park. There is only one way in and out of the car park so the cameras would be placed at the location shown. This entrance is two lanes so an entrance camera could be pointed one way to monitor cars entering and an exit camera the other on the opposite lane.

SUMMARY OF RESEARCH AND ANALYSIS

It became clear from the interview previously conducted that the college wastes a large amount of resources on issues that arise from student parking within college. A solution that could gather and structure data about the car park seemed to benefit the college in multiple aspects. A key area of this was being able to identify which vehicles were entering the college premises at all times. From this I had to research into methods of identifying vehicles – from which automated number plate recognition became clearly the only option. All ANPR systems stem back to individual character recognition which work off multi-layer neural networks. My research into this area has allowed me to conclude that the accuracies provided from a well-designed network for individual characters can be extrapolated up to recognizing number plates at reasonable accuracy. The further research into the British number plate system will allow me to improve the efficiency of my system by making appropriate assumptions about number plates.

Even though the problem may appear to have a simple IO structure, when analyzing far more complexity appeared. I decided to break the IPSO down into levels of complexity, which became useful when looking at each section of the potential solution (i.e. The character recognition system). The dataflow through the system was modelled in a two levels of DFD. The lowest level diagram did not provide enough of a view into the intricacies of system, so I broke it down into a higher level DFD of a proposed system. Since creating that diagram it has become clear that a sperate table of invalid vehicles is inefficient and un-necessary.

Originally I thought that a flat-file database structure would suffice and simple a record would be appended for each vehicle entering the site. However, my analysis of questionnaire responses, combined with the interview, meant I had to rethink the entire output/user end of the system. A reasonably more complex database structure holding far more useful and detailed information appeared to be what a true solution would require. The E-R diagram has now been designed to allow for a user-query based front-end solution, meaning the user can specify what data they want to view and it is retrieved from the network. The CASE diagram showed me that the system requires a login and password.

REQUIREMENTS

Functional Requirements

(INPUTS)

1. This system must be able to operate for at least the entirety of the college day (8:00 to 5:00).
 - 1.1. The system should run automatically once set up, so require no user input to start recording on any given day.
2. The system be able to process image captures of vehicles entering the site. The mechanics behind how the images are extracted from the video are not required.
 - 2.1. Each of these captures should be compressed and formatted before being stored and passed into the main system.
 - 2.2. The User of the system should be able to view the image being processed.

3. The User should be able to add a student and a vehicle to the database from inputs to the main interface.
4. The Users inputs for querying the database should be validated.

(PROCESSING)

5. The system should process each input image to produce a standard formatted output.
 - 5.1. The formatted image should be with resolution x by y.
 - 5.2. The image should be grayscale or black and white.
 - 5.3. Each pixel should have an associated/stored normalized value between 0 and 1 for use in the network.
6. This system must be able to identify a number plate within an image.
 - 6.1. The solution should be able to identify a number plate within an image at a success rate at or above 70%.
7. The system should be able to isolate the number plate within the image.
 - 7.1. It should then decompose the number plate image into images of each character on the number plate.
 - 7.2. It should be able to accurately decompose the image of the number plate into separate character images at a success rate at or above 70%.
8. The system should be able to identify a single character within a 28 by 28 grid of normalized pixels.
 - 8.1. The solution should be able to identify these characters at a success rate above 85%.
9. The system should be able to combine a correct series of recognitions as to enable it to recognise an entire number plate.
 - 9.1. The solution should be able to correctly identify number plates at success rate at or above 70%. This can include number plates in which errors were made but the plate was still identified correctly used adjustments with the database (see 10.2).
10. The system should be able to produce a list of invalid vehicles to enter the premises (this does not have to be able table in itself but can be obtained through querying the database)
 - 10.1. If the same number plate is recognised again (before it has been determined to leave the site) no new record will be created (assumed to be stationary next to camera).
 - 10.2. If the system incorrectly processes an image and produces an output with 2 or fewer errors to any number plate in the database, this will be corrected and added to the database.

(STORAGE)

11. The solution should include a normalised database that contains the tables specified (See E-R Diagram).
 - 11.1. When a number plate is recognised (for a vehicle entering the site) the system should submit a record to the database including: Registration (PK), Entry Date, EntryTime and a Null ExitTime.
 - 11.2. When a number plate is recognised (for a vehicle leaving the site) the system should update the corresponding record in the database to that vehicle with its ExitTime.
 - 11.3. The database should contain a table of student IDs (primary key) and their associated data (see E-R Diagram).
 - 11.4. The database should contain a table of registered vehicles. Registration number (PK), Make, Model, Color and Registration Date.

(OUTPUTS)

12. The user must be able to query the database and search for any record in the database in a simple way.
 - 12.1. The extent of the criteria for a search are described by, but not limited to, the examples listed in section 13.1.
 - 12.1.1. Search by Student Name (First or Surname)
 - 12.1.2. Search by Car Registration
 - 12.1.3. Search by StudentID
13. The system should have a built in process that can query the database in such a way as to output the list of invalid vehicles (past or present) in the car park.
14. The User should be able to view the Students and Vehicles Databases within the interface.

Additional Requirements

(OUTPUTS)

15. The system should be able to provide a count for the number of spaces used and therefore the spaces available in the carpark currently.
 - 15.1. This count should be accurate to within 5 spaces. Note that this is consistent with the accuracy of the system. The college car park would have over 100 cars at times so 70% accuracy (likely more when you consider each vehicle would have multiple captures) would mean that more than 5 would be incorrectly identified. However, they would still have been identified as vehicles so the count of the car park is still correct. The count is only dependant on when a plate is identified not whether it is accurately identified.

JUSTIFICATION OF REQUIREMENTS

(1)

For the system to provide an effective solution it must be active for as long a period as possible (so not to miss any vehicles entering the site). A 24h operating system would seem most appropriate (hence requirement 15). If an aim of the system is to save of staff time and resources it would make sense for the system to run automatically (potentially on a timer).

(2)

Clearly the system would only thoroughly provide accurate data if every vehicle was captured while entering the site. Each capture will likely be on a time (e.g. 2Hz) rather than triggered by a piece of hardware (sensor) as this will make the system cheaper to implement. Due to the volume of images captured per day (on the order of 100,000 if running for 24h) each image has to be compressed and only stored if necessary, this will mean that no excessive storage is required. For this project I will not be dealing with the capturing of images but providing a system to analyze the captured images; This is simply from a hardware and time constraints. In the final user interface it would be useful for the user to be able to view the image being processed for validation purposes (2.2).

(3)

For completeness of the solution a user should be able to run the whole system from within one interface which means being able to add a student and a vehicle to the database. This will be done on a single entry at a time basis. This is because (from my analysis) it seems that students apply on an individual basis for a parking permit so the user would simply be appending students to the database one at a time (not blocks students).

(5)

I gathered from my research into neural networks that standardizing the inputs is key to obtaining even reasonable results. The decision to reduce the resolution of the image has two key benefits. First, the processing required to train a network grows exponentially with the number of inputs to the network (other factors include number of layers and number of nodes in each layer). Therefore (as the inputs to the network will be pixel values from an image) reducing the resolution and therefore number of inputs will be key to being able to train the network will reasonable processing power and time. The decision to greyscale also factors into this by simplifying the image down.

(6)

Rather than attempting to identify a number plate from within an image, it is clear that the number plate needs to be extracted from within an image and then the image resized so contains just the number plate. The threshold of 70% will ensure that the required overall accuracy of the system is met. As it is likely that multiple images of each vehicle will be taken as it enters the site (some error in the identification of a number plate can occur, yet still produce good results).

(7)

As the proposed and researched network only recognizes a single character from within an image, once the number plate is identified it will need to be broken down into sub images of each character. I feel this option is feasible due to the standard nature of number plates; my research into character size and spacing will characters to be drawn from an image of a number plate through a standard process. A threshold of 70% again is placed on this process to ensure a good level of accuracy is met. The combination of these two criteria for image formatting would ensure that at least 50% of images are passed into the network correctly formatted (clearly that is a worst case scenario).

(8)

From my analysis of existing networks and the accuracies they have achieved on hand-drawn characters, a network which performs recognition at a rate of 85% seems achievable. This accuracy was chosen carefully due to its link with the overall accuracy in the recognition of the number plate. Assuming every number plate contains 7 characters that need to be identified correctly in a row, to achieve the overall accuracy required (9.1) an individual character accuracy above 95% is needed ($0.95^7 = 0.70$ (2SF)). I have allowed a tolerance of 10% here with the network accuracy for single character recognition as there are multiple features with the system that should allow for increased overall accuracy. Taking the worst case scenario: $0.7 \times 0.7 \times 0.85^7 = 0.157$ (3SF) seems very probabilistically low. However say the system only requires getting 5 or more correct in 7 (as adjustments can be made with comparisons with the network), using a binomial distribution model, $P = 0.926 \times 0.49 = 0.454$ (Shown below – most of the error comes from the image formatting probabilities). Then considering only certain characters can be located in certain positions, multiple captures are taken of every image and the fact that the binomial probability distribution used is probably invalid due to a varying probability for each character, it is likely a system with these probabilities could recognize image(s) at a rate of 0.7 or greater (9.1).

let X be the number of successful characters.
 where $p = 0.85$, $n = 7$
 $\therefore X \sim \text{Bin}(7, 0.85)$
 $\therefore P(X \geq 5) = 1 - P(X \leq 4)$
 $= 0.2097$
 $+ 0.3960$
 $+ 0.3206$

 0.926 (8SF)

(9)

As mentioned earlier each vehicle that enter the site will have multiple images processed of it. An accuracy of any given image above 70% should enable extremely high accuracies overall for every car being recognized as it enters the site.

(10)

From my interview this appeared to be a key requirement of the system. Originally I thought a separate table of invalid entries would be most appropriate, however on analyzing the appropriate structure of a database this seems to be unnecessary. I was made aware that at certain times of day there are high levels of traffic in and out of college which leads to vehicles queuing in and out. This would mean they would likely be sat by the camera for longer periods of time. To prevent many

records of the same vehicle being added they must be determined to have left the car park before they can create a new record. As the number of vehicles on the database is quite low (400-700) it can be assumed that most number plates are very different. It is therefore likely that if a number plate is recognized in which most characters match, apart from a few, that it was that original plate. I have set the cut off at 5 or more out of 7 for reasonable accuracy but not too high as to frequently have false positives.

(11)

The database solution proposed draws from the analysis of how to produce a normalized database containing all the required data. Originally it was planned to just record vehicles entering the site however to be able to keep an accurate count of the number of available spaces another camera on the exit would be required. It seemed not efficient to have two tables for entry and exit so simply an exit time would be added to the single car park table (this was also effective in meeting 10.1).

(12)

Querying the database would be a key part of the usefulness of the final solution. There would be large volumes of data on the database so this system would allow for fast access to that. I felt that the criteria of: Student Name (First and Last), Registration and StudentID would be the most useful to be able to search by as most commonly the user would either want to know if a student is in the car park (hence Names and ID) or which student drives a certain car (hence Registration).

(14)

To simply be able to check that a record has been added to the database.

(15)

This appeared to be a feature that would be very useful for the user and can be determined from cars with an entry time and not exit time. The only error that would occur with this count would be if a vehicle left the car park but was not recognized by the system so no exit time was set. As the count would be reset each day an accuracy to within 5 spaces seems reasonable.

DESIGN

USER INTERFACE

The end system will have a user interface in which the user can query the database of vehicle entries. I have decided to provide a query based system (rather than simply providing a flat file of all records) as this will allow the user to filter the large volume of data coming in; it was clear that simply seeing all the data is not that useful but being able to enquire about the data was key to a useful system. The interface will be made as a Windows Forms App running of Visual Basic. A design for this interface is shown below. (This may not be the final design used but should reflect all the key aspects of the interface).

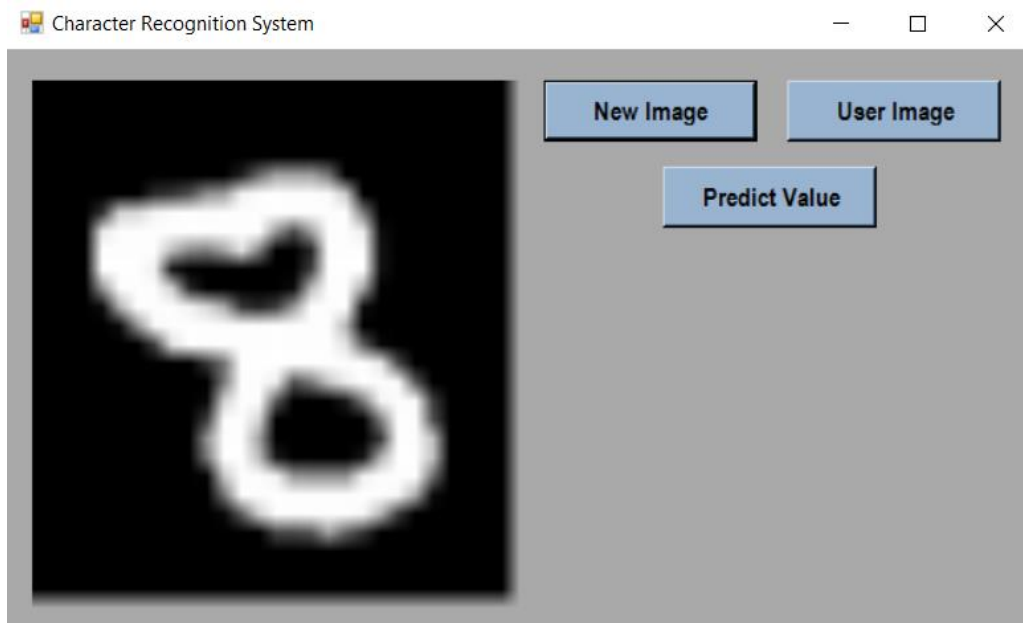
Car Park Database

Godalming College ANPR System Invalid Vehicles

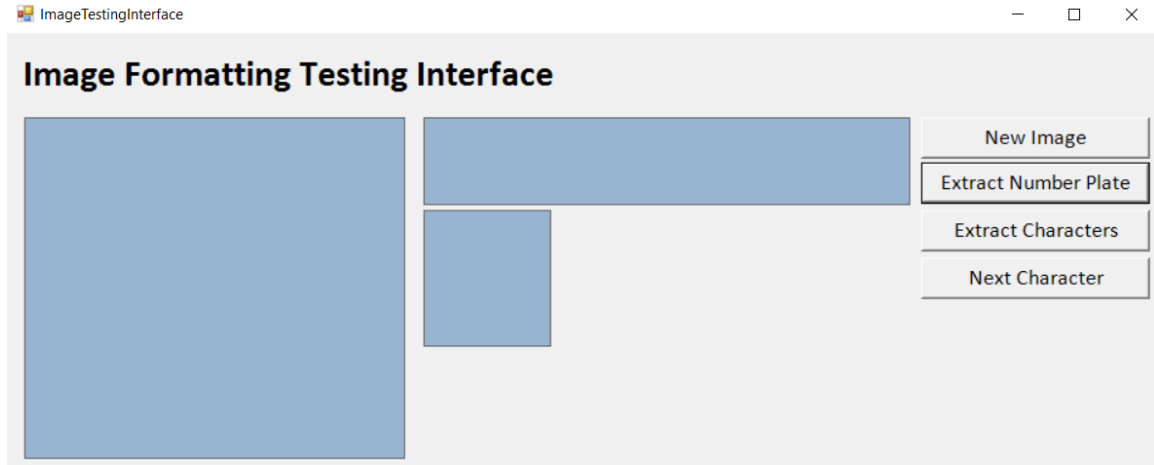
Student Name Car Make Car Model Car Registration Date

 04 November 2019

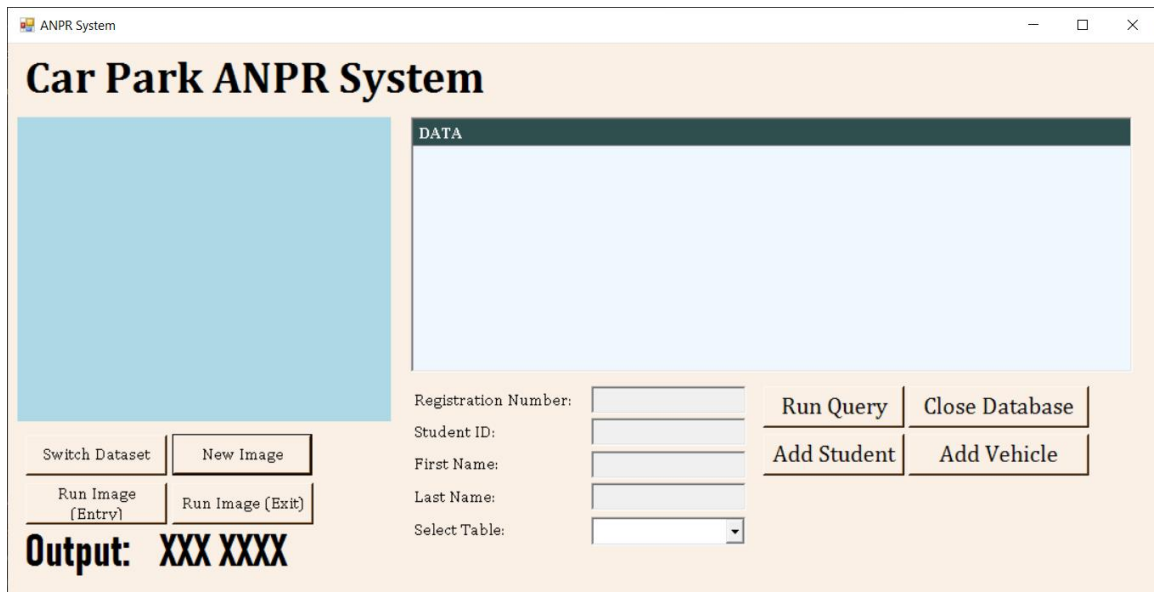
For the development and testing of the solution it is clear two additional testing interfaces will be required: Character Recognition Testing Interface and Image Formatting Testing Interface.



The Image shown above is a design for the character recognition system testing interface. This interface will instantiate the Character Recognition System Class and provide an interface for its methods. The New Image button will allow the user to scroll through randomly chosen character images. The User Image button will load a specific file (saved to the device) which can then be tested. An addition to this interface could be to add a “Test Interface” Button which would evaluate a set of images and provide a percentage accuracy of the system. This interface should determine whether the Character Recognition requirements have been met.



The Image shown above is a design for the Image Formatting testing interface. This interface will run the Image and Character Formatting module and ensure everything works. The New Image Button will scroll through images of number plates (in the left hand box). The Extract Number Plate Button will run the method and paste the result in the top right box. The Extract Characters Button will run the method and paste the first result in the bottom right box, which can then be scrolled through with Next Character. This interface should determine whether the formatting requirements have been met.



The Image shown above is a design for the final user interface of the system. As no access to a video feed is available, the system will be tested using just set images so that side of the interface would change if this system was formally used. This should allow for all requirements of the system to be tested.

NEURAL NETWORK DESIGN

I have chosen to create my neural network by manipulating matrices and the properties of operations of matrices. The network will be structured so that each of the layers of nodes are a X by 1 dimensional matrix and each of the weights are a X by Y dimensional matrix so that the value of each node is stored in an element of a matrix. The sizes of each layer for a simple single layer perceptron with X inputs Y hidden nodes and Z outputs are described below. Matrices are defined in a “rows by columns” format.

INPUT Matrix: X by 1

HIDDEN Matrix: Y by 1

OUTPUT Matrix: Z by 1

WEIGHTS INPUT → HIDDEN: X by Y

WEIGHTS HIDDEN → OUTPUT: Y by Z

For programming purposes these matrices will be stored as two dimensional arrays. The formatting of data in this will allow for efficient feeding of data through the network. The process of this is described later. Matrixes can be added, subtracted, multiplied or transposed. Some of these algorithms are described below.

MATRIX PROPERTIES

Addition and Subtraction

When adding or subtracting matrices you simply add or subtract each element in the matrix from the corresponding element in the subsequent array. This property will be useful for operations in the network such as calculating the error in the output matrix (by subtracting a target matrix)

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} + \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} a + e & b + f \\ c + g & d + h \end{pmatrix}$$

Multiplication (Scalar and Elementwise)

Scalar multiplication of a constant means to multiply each element in the matrix by a constant value (e.g. double every value in a matrix). Elementwise multiplication is similar to addition and subtraction but each element is multiplied by the corresponding element.

$$2 \times \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 2a & 2b \\ 2c & 2d \end{pmatrix}$$

Multiplication (Matrix)

For matrix multiplication to be valid the number of columns in the first matrix must equal the number of rows in the second. The resulting matrix has dimensions equal to the rows of the first by the columns of the second. It is described by the following process.

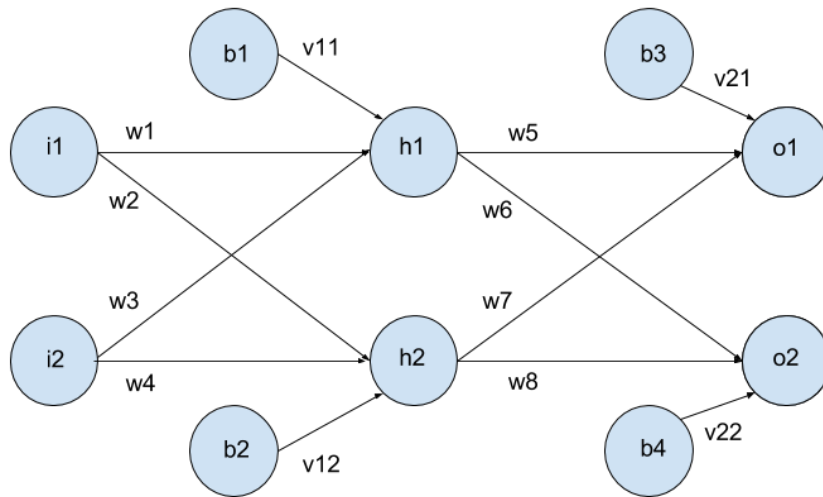
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{pmatrix}$$

The algorithm for multiplying matrices can be used to pass data through a network.

Consider,

$$\begin{pmatrix} W_1 & W_3 \\ W_2 & W_4 \end{pmatrix} \times \begin{pmatrix} I_1 \\ I_2 \end{pmatrix} = \begin{pmatrix} I_1W_1 + I_2W_3 \\ I_1W_2 + I_2W_4 \end{pmatrix}$$

For a network structure as follows,



We know that h1 should be comprised of a weighted sum of the inputs, which would be expressed as $i_1w_1 + i_2w_3$; h2 would be expressed as $i_1w_2+i_2w_4$. It is noticeable that the resultant matrix shown above is comprised of the expected results for the two nodes. Therefore the result of multiplying a weight matrix by the input matrix gives the hidden layer. This property can be repeated to obtain the output layer. Bias matrices can simply be added to the resultant matrices.

KEY MATRIX FUNCTIONS

=====

FUNCTION Matrix Multiplication (Matrix1, Matrix2)

IF Check Dimensions For Multiplication(Matrix1, Matrix2) THEN

For i As Integer <- 0 To Result Rows - 1

For j As Integer <- 0 To Result Cols - 1

Sum <- 0

```

    For k As Integer <- 0 To Matrix1.NumOfArrayCols
        Sum +<- Matrix1.Matrix(i, k) * Matrix2.Matrix(k, j)
    NEXT

    Result Matrix(i, j) <- Sum

NEXT

NEXT

RETURN Result Matrix

ELSE

    PRINT ("Invalid Dimensions of Input Matrix (Multiplication - Module Sub)")

    RETURN False

END IF

END FUNCTION

=====

FUNCTION Check Dimensions For Multiplication ( Matrix1 , Matrix2 )

    IF Matrix1.NumOfCols <- Matrix2.NumOfRows THEN

        RETURN True

    ELSE

        RETURN False

    END IF

END FUNCTION

=====

FUNCTION Check Dimensions For Elementwise (Matrix1 , Matrix2)

    IF Matrix1.NumOfArrayRows <- Matrix2.NumOfArrayRows And Matrix1.NumOfCols <-
Matrix2.NumOfCols THEN

        RETURN True

    ELSE

        RETURN False

```

```
END IF
```

```
END FUNCTION
```

```
=====
```

```
FUNCTION Transpose Matrix ( Matrix )
```

```
For RowsIter As Integer <- 0 To Matrix Num Of Array Rows
```

```
For ColsIter As Integer <- 0 To Matrix Num Of Array Cols
```

```
Result(ColsIter, RowsIter) <- Matrix(RowsIter, ColsIter)
```

```
NEXT
```

```
NEXT
```

```
RETURN Result
```

```
END FUNCTION
```

```
=====
```

```
FUNCTION Generate Random Number() As Single
```

```
Random Number As New Random()
```

```
RETURN Random Number NEXT(0, 1000) / 1000
```

```
END FUNCTION
```

```
=====
```

```
FUNCTION Matrix Addition(Matrix1 , Matrix2 )
```

```
IF Check Dimensions For Elementwise(Matrix1, Matrix2) THEN
```

```
For RowsIter As Integer <- 0 To Matrix1.NumOfArrayCols
```

```
For ColsIter As Integer <- 0 To Matrix1.NumOfArrayRows
```

```
Result Matrix(ColsIter, RowsIter) <- Matrix1.Matrix(ColsIter, RowsIter) +  
Matrix2.Matrix(RowsIter, ColsIter)
```

```
NEXT
```

```
NEXT
```

```
ELSE
```

```
PRINT("Invalid Dimensions of Input Matrix (Addition - Module Sub)")
```

```

END IF

RETURN Result

END FUNCTION

=====

FUNCTION Matrix Subtraction(Matrix1 , Matrix2 )

IF Check Dimensions For Elementwise(Matrix1, Matrix2) THEN

  For RowsIter As Integer <- 0 To Matrix1.NumOfArrayRows

    For ColsIter As Integer <- 0 To Matrix1.NumOfArrayCols

      Result Matrix(RowsIter, ColsIter) <- Matrix1.Matrix(RowsIter, ColsIter) -
Matrix2.Matrix(RowsIter, ColsIter)

    NEXT

  NEXT

ELSE

  PRINT("Invalid Dimensions of Input Matrix (Addition - Module Sub)")

END IF

RETURN Result

END FUNCTION

=====

FUNCTION Return Sigmoid(Matrix )

For RowsIter <- 0 To Matrix Num Of Array Rows

  For ColsIter <- 0 To Matrix Num Of Array Cols

    X <- Matrix Matrix(RowsIter, ColsIter)

    Result Matrix(RowsIter, ColsIter) <- 1 / (1 + e ^(-X))

  NEXT

NEXT

RETURN Result

END FUNCTION

```

```
=====
```

```
FUNCTION Return Derivative Sigmoid(Matrix )
  For RowsIter <- 0 To Matrix Num Of Array Rows
    For ColsIter <- 0 To Matrix Num Of Array Cols
      y <- Matrix Matrix(RowsIter, ColsIter)
      Result Matrix(RowsIter, ColsIter) <- y * (1 - y)
    NEXT
  NEXT
RETURN Result
END FUNCTION
```

```
=====
```

```
FUNCTION Return Transposition(Matrix )
  For RowsIter As Integer <- 0 To Matrix Num Of Array Rows
    For ColsIter As Integer <- 0 To Matrix Num Of Array Cols
      Result Matrix(ColsIter, RowsIter) <- Matrix(RowsIter, ColsIter)
    NEXT
  NEXT
RETURN Result
END FUNCTION
```

```
=====
```

FEED FORWARD ALGORITHM

This algorithm will manipulate the properties of multiplying matrices (described earlier) to allow an efficient passing through of input through the network.

```
FUNCTION FeedForward(InputArray)
  Inputs <-- MatrixFromArray(InputArray)
  Hidden.Matrix <-- MatrixMultiplication(WeightsBetInputHidden, Inputs)
  Hidden.ElementWiseAddition(BiasHidden)
```

```

Hidden.Sigmoid()
Outputs.Matrix <-- MatrixMultiplication(WeightsBetHiddenOutput, Hidden)
Outputs.ElementWiseAddition(BiasOutput)
Outputs.Sigmoid()
OutputsArray <-- ArrayFromMatrix(Outputs)
RETURN OutputsArray
END FUNCTION

```

BACKPROPAGATION ALGORITHM

This is a pseudocode algorithm for propagating backwards through a network to find the errors in each of the weights. This is derived from the mathematical derivation shown earlier. This algorithm will adjust the weights immediately after calculating them. It will be likely that to achieve optimum results I will have to take a stochastic gradient decent approach; this will mean that the weights will have to be summed over a series of backpropagations to then averaged then applied to the weights. For character recognition this will provide a network which provides the best accuracy across the full range of data and not specialize in certain characters (i.e. it can recognise and 'A' and not a 'B').

```

SUB Backpropagation(InputsArray, TargetsArray)
#CONVERT INPUTS TO MATRICES (described earlier)
Inputs <-- MatrixFromArray(InputsArray)
Targets <-- MatrixFromArray(TargetsArray)
#FEED INPUTS THROUGH THE NEXTWORK
Hidden.Matrix <-- MatrixMultiplication(WeightsBetInputHidden, Inputs)
Hidden.ElementWiseAddition(BiasHidden)
Hidden.Sigmoid()
Output.Matrix <-- MatrixMultiplication(WeightsBetHiddenOutput, Hidden)
Output.ElementWiseAddition(BiasOutput)
Output.Sigmoid()
#CALCULATE OUTPUT GRADIENT
OutputError <-- MatrixSubtraction(Targets, Output)

```

```

OutputError.ScalarMultiplicationOfInput(OutputError)
OutputErrorDerivative <-- MatrixSubtraction(Output, Targets)
OutputErrorDerivative.MultiplicationOfSingleValue(2)
OutputSigmoidDerivative <-- ReturnDerivativeSigmoid(Output)
OutputErrorDerivative.ScalarMultiplicationOfInput(OutputSigmoidDerivative)
HiddenTransposed <-- ReturnTransposition(Hidden)
OutputGradient.Matrix<--MatrixMultiplication(OutputErrorDerivative,HiddenTransposed)
OutputBiasGradient <-- OutputErrorDerivative
OutputGradient.MultiplicationOfSingleValue(LearningRate)
OutputBiasGradient.MultiplicationOfSingleValue(LearningRate)

'CACULATE HIDDEN LAYER GRADIENT'
WeightsBetHiddenOutputTransposed <-ReturnTransposition(WeightsBetHiddenOutput)
HiddenErrorDerivative.Matrix<MatrixMultiplication(WeightsBetHiddenOutputTransposed,
OutputErrorDerivative)
HiddenSigmoidDerivative <-- ReturnDerivativeSigmoid(Hidden)
HiddenErrorDerivative.ScalarMultiplicationOfInput(HiddenSigmoidDerivative)
InputsTransposed <-- ReturnTransposition(Inputs)
HiddenGradient.Matrix<--MatrixMultiplication(HiddenErrorDerivative, InputsTransposed)
HiddenBiasGradient <-- HiddenErrorDerivative
HiddenGradient.MultiplicationOfSingleValue(LearningRate)
HiddenBiasGradient.MultiplicationOfSingleValue(LearningRate)

'ADJUST THE WEIGHT MATRIXES BY GRADIENTS'
WeightsBetInputHidden.ElementWiseSubtraction(HiddenGradient)
WeightsBetHiddenOutput.ElementWiseSubtraction(OutputGradient)

```

```
'ADJUST THE BIASES BY GRADIENTS
BiasHidden.ElementWiseSubtraction(HiddenBiasGradient)
BiasOutput.ElementWiseSubtraction(OutputBiasGradient)
BiasOutput.DisplayMatrix()
END SUB
```

SAVING THE NETWORK TO A FILE

In the process of training the neural network (and for use once trained) a system will need to be in place to save the weights and biases of the optimised network. The weights and biases will need to be saved in a structured way to allow for them to both be saved and load to/from a network. I have decided to save a network to a plain text file, where each line of the file contains the value of a weight in the matrix.

For a single layer perceptron with an input layer, hidden layer and output layer they will be saved in the following format.

STRUCTURE:

- Weights between Input and Hidden
- Weights between Hidden and Output
- Biases for Hidden Layer
- Biases for Output Layer

The weights in each of the previous two-dimensional matrices will be saved across then down. This process can be extrapolated for a more complex network.

$$\begin{pmatrix} W_1 & W_2 \\ W_3 & W_4 \end{pmatrix} \xrightarrow{\text{saves}} \begin{cases} W_1 \\ W_2 \\ W_3 \end{cases}$$

From the structure shown above, an algorithm to load a network can be derived.

```
FOR ColsIter IN RANGE 0 to (InputToHiddenWeightsCols - 1) DO
  FOR RowsIter IN RANGE 0 to (InputToHiddenWeightsRows - 1) DO
    InputToHiddenWeights(RowsIter, ColsIter) <-- READFILE
  NEXT
NEXT
NEXT
```



```

FOR ColsIter IN RANGE 0 to (HiddenToOutputWeightsCols - 1) DO
    FOR RowsIter IN RANGE 0 to (HiddenToOutputWeightsRows - 1) DO
        HiddenToOutputWeights(RowsIter, ColsIter) <-- READFILE
    NEXT
NEXT
FOR RowsIter IN RANGE 0 to (HiddenBiasRows - 1) DO
    HiddenBias(RowsIter, 0) <-- READFILE
NEXT
FOR RowsIter IN RANGE 0 to (OutputBiasRows - 1) DO
    OutputBias(RowsIter, 0) <-- READFILE
NEXT

```

TRAINING A NEURAL NETWORK

Neural networks can be trained through various different techniques. For a recognition based system a supervised learning method is most effective, it involves the process of feeding a input through a network and comparing the output with a known target value. The error between these two data points is then used to make changes to the weights in the network (backpropagation). Therefore to use this technique a dataset of images and their associated values is required.

Two different methods for training the network through supervised learning can be used. Either adjustments are made to a network after each input is processed, or a batch of adjustments is averaged after a group of images is passed through the network then the network is adjusted using those. From research into the optimum technique for the system being made, it is clear that the latter is the superior method. It provides a slower but more precise gradient decent and produces a network that works best across all input data as opposed to being biased towards certain outputs. The overall training process is often contained within epochs, these are used to group and evaluate the progress of training a network.

A possible algorithm for training a character recognition system is shown,

```

SUB TrainNetwork(Epochs, Batches, BatchSize)
    FOR EpochIter IN RANGE 0 To Epochs DO
        TotalHiddenBiasGradient.ZeroMatrix()
        TotalHiddenGradient.ZeroMatrix()
        TotalOutputBiasGradient.ZeroMatrix()
    
```

```

    TotalOutputGradient.ZeroMatrix()
    FOR BatchIter IN RANGE 0 To Batches DO
        FOR BatchSizeIter IN RANGE 0 to BatchSize DO
            # SET CURRENT INPUT VALUES
            # SET CURRENT TARGET VALUES
            # BACKPROPAGATION
        NEXT
        ApplyStochasticDecent(N * BatchSize)
    NEXT
    # SAVE NETWORK
NEXT
END SUB

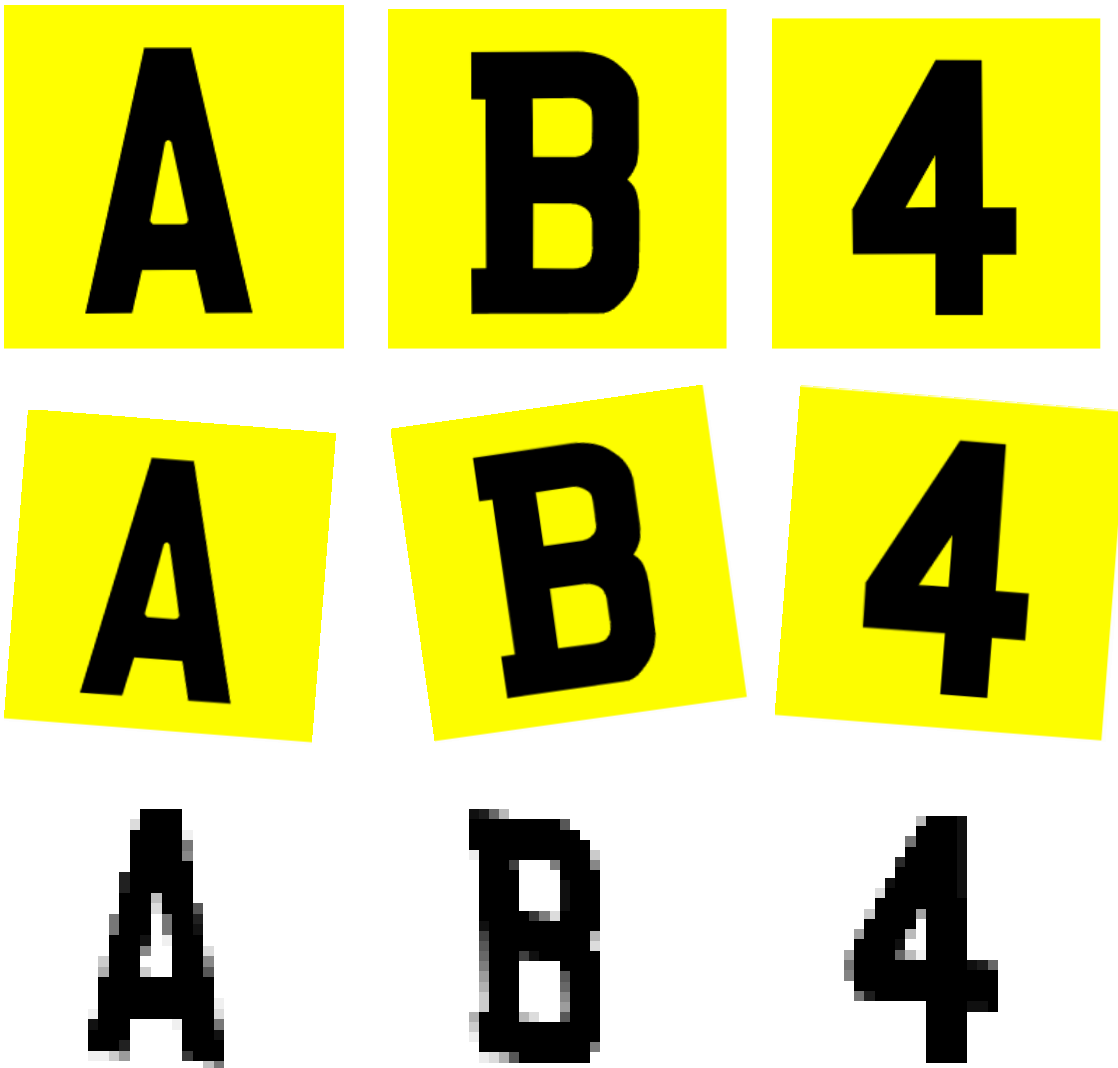
SUB ApplyStochasticDecent(Divisor)
    Multiplier <-- 1 / Divisor
    TotalHiddenBiasGradient.MultiplicationOfSingleValue(Multiplier)
    TotalHiddenGradient.MultiplicationOfSingleValue(Multiplier)
    TotalOutputBiasGradient.MultiplicationOfSingleValue(Multiplier)
    TotalOutputGradient.MultiplicationOfSingleValue(Multiplier)
    InputsToHiddenWeights.ElementWiseSubtraction(TotalHiddenGradient)
    HiddenToOutputsWeights.ElementWiseSubtraction(TotalOutputGradient)
    HiddenBias.ElementWiseSubtraction(TotalHiddenBiasGradient)
    OutputBias.ElementWiseSubtraction(TotalOutputBiasGradient)
END SUB

```

TRAINING DATASET

As is clear from the training system described above a large dataset of training images is required. This dataset would probably have to be on the order of 1000 images of each character (note each image would have to be 'labelled' with the character it shows). After a long period searching for a dataset to use none could be found. There were sets of limited characters (e.g. 0-9) or of whole number plates but none of just the characters A-Z and 0-9 in the number plate character font.

Because number plates are all in a standard font I have decided to create/generate a dataset to use to train using a single source image. A capture will be made of each of the characters in the number plate font. These captures will then, using an image processing library, be rotated in 3 dimensions randomly within set degrees of freedom. Each rotation will form a new image, which can then be formatted as required for training. This demonstrated below,



The image formatting side of VB.net is limited I have decided this will be done in another programming language. Numpy is a scientific computing library for Python which has extensive image formatting features. A notable method in Numpy is the AffineTransform method in the image class which maps an input image to an output image using a 'rotation' matrix. This would allow for 3D rotations

IMAGE FORMATTING DESIGN

Image formatting is a key part of the functionality of the system. Many of the interfaces and classes which run the character recognition system will require images to be inputted before passing in. The Image Formatting will be held in a modular structure as a collection of public (accessed by other classes) and private (accessed by other formatting functions) functions.

The Image Formatting will be split into two Modules: Image Formatting and Character Formatting.

Image Formatting:

- Receives an input image and able to return a standard format and resolution image.
- Receives a standard image and able to return a list of rectangles which contain any number plates in an image.
- Receives a collection of rectangles and a bitmap image and able to return a list of bitmap images each of which is a number plate in the image.

A key part of identifying the number plate is determining which colour pixel will be said to be on a number plate. As the system identifies rear plates this will be done by evaluating when the RGB values of a pixel match the criteria for a 'yellow' number plate pixel. I have determined the a good range for RGB values as shown below, but this can be adjusted on testing.

E.G.

FUNCTION ValidNumberPlatePixel(X as Integer, Y as Integer) AS Boolean

PixelColor → GetPixelColor(X, Y)

IF PixelColor.R < 130 and PixelColor.G < 100 and PixelColor.B > 60

RETURN True

ELSE

RETURN False

END IF

END FUNCTION

The following is a simplified Pseudocode algorithm showing the essence of how a list of number plate rectangles can be identified in an image. Rectangles are an existing class in VB.NET which store a

starting coordinate and a width and height in an object so are used to encapsulate the region of the number plate in an image. Some of the functions used below will be defined later on.

```

FUNCTION GetNumberPlate() AS Rectangle
    FOR ImageX as Integer → 0 to Width
        FOR ImageY as Integer → 0 to Height
            IF PixIsValid(ImageX, ImageY)
                IF NOT CoordinateIsInExistingImageSection(ImageX, ImageY)
                    NumberPlate → CheckIfNumberPlate(ImageX, ImageY)
                    IF 3 < WidthHeightRatio < 6
                        ImageList.ADD(NumberPlate)
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    ENDFOR
    RETURN ImageList
END FUNCTION

```

The CheckIfNumberPlate() Function is used to return a region from a starting point in which the number plate would be if its there. It will recursively check horizontally and vertically for runs of valid number plate pixels. The recursion is designed so the run is incremented on the collapse of the stack once the terminating condition (invalid pixel) is met.

```

FUNCTION CheckIfNumberPlate(Image, StartX, StartY) as Rectangle
    XRun → 0
    YRun → 0
    CheckAdjacentPixel(Image, XRun, StartX, StartY, True)
    CheckAdjacentPixel(Image, YRun, StartX, StartY, False)
    ImageSection → New Rectangle(StartX, StartY, XRun, YRun)
    RETURN ImageSection
END FUNCTION

```

```

SUB CheckAdjacentPixel(Image as Bitmap, (ByRef) Run as Integer, StartX, StartY, Horizontal)

```

```
IF Horizontal
    StartX += 1
ELSE
    StartY += 1
END IF

If StartX < Image.Width and StartY < Image.Height
    If ValidNumberPlatePixel(Image, StartX, StartY)
        CheckAdjacentPixel(Image, Run, StartX, StartY, Horizontal)
        Run += 1
    END IF
END IF

END SUB
```

Character Formatting

- Receives an input image and able to extract a character from a starting point.
- Able to construct a list of rectangles of the locations of characters within an image of a number plate.
- Able to construct a list of Bitmaps of the characters.
- Able to format each of the Bitmap images so they can be passed into the network.

These formatting modules are run after the image formatting so receive an image of just the number plate.

The GetCharacter() Function will return a rectangle of the region containing a character from a starting point in the image. It will be designed recursively in combination with a WidthOfCharacter() Function.

DATABASE DESIGN

The database has already be designed in concept in the analysis section. The required Data Definition Language and Structured Query Language will be outlined here.

I will be using the connection string as follows:

```
"Provider=Microsoft.Jet.OLEDB.4.0;" &
  "Data Source=" & DatabaseName & ".mdb;" &
  "Jet OLEDB:Engine Type=5"
```

(Where DatabaseName is the name of the database being created)

There are three main tables to create and the DDL for those tables is outlined below.

```
CREATE TABLE CarPark(Registration Varchar(20), EntryDate Date, EntryTime
Varchar(8), ExitTime VarChar(8) PRIMARY KEY(Registration, EntryDate, EntryTime))
```

```
CREATE TABLE Students(StudentID Int, Registration Varchar(20), FirstName
VarChar(20), LastName VarChar(20), Age Int, PRIMARY KEY(StudentID, Registration))
```

```
CREATE TABLE Vehicles(Registration Varchar(20) PRIMARY KEY, Make VarChar(20),
Model Varchar(20), Color Varchar(20), DateRegistered Date)
```

Structured Query Language

A key part of the system is to be able to add records to each of the databases. The general syntax for such a query is:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Using this a generalized function to construct a submit query for any set of data into any table can be formed.

```
FUNCTION ConstructSQLForDataSubmit(Table as String, Fields as Array, Data as Array)
```

```
  SqlStr → "INSERT INTO " + Table + "("
```

```
  FOR Iter → 0 TO Fields.Length -1
```

```
    IF Iter <> Fields.Length -1
```

```
      SqlStr +→ Fields(Iter) + ", "
```

```
    ELSE
```

```
      SqlStr +→ Fields(Iter)
```

```
  END IF
```



```

SqlStr +→ “) VALUES (“
FOR Iter → TO Data.Length -1
    IF Iter <> Data.Length -1
        SqlStr += Data(Iter) + “,”
    ELSE
        SqlStr += Data(Iter) + “)”
    END IF
RETURN SqlStr
END FUNCTION

```

Note: All Data values must have single quote marks around the data (SQL Syntax)

Another key part of the system is to be able to select data from tables in the database. As the only purpose is to display the data I will be selecting data and pasting it into a VB Forms DataGrid. The following outlines the general structure for selecting all the data in a query.

```

SUB SelectAllDataFromTable(Table as String, DatabaseConnections...)

```

```

    SELECT Case Table
        ArrayOfFields → “Array containing the fields of the selected table”
    END SELECT
    SqlStr → “SELECT “
    FOR Iter → 0 TO ArrayOfFields.Length - 1
        IF Iter <> ArrayOfFields.Length - 1
            SqlStr +→ “[“ + ArrayOfFields(Iter) + “],”
        ELSE
            SqlStr +→ “[“ + ArrayOfFields(Iter) + “]”
        END IF
    END FOR
    SqlStr +→ “FROM “ + Table
    “Execute SQL and deposit in DataGrid”
END SUB

```

If data is being selected with a query a following SQL can be executed.

```
SELECT CarPark.Registration, CarPark.EntryDate, CarPark.EntryTime,  
CarPark.ExitTime, Students.StudentID, Students.FirstName,  
Students.LastName, Students.Age, Vehicles.Make, Vehicles.Model,  
Vehicles.Color, Vehicles.DateRegistered  
  
FROM ((CarPark INNER JOIN Students ON CarPark.Registration =  
Students.Registration) INNER JOIN Vehicles ON CarPark.Registration =  
Vehicles.Registration)  
  
WHERE "Conditions of Query"
```

If the invalid vehicles in the car park want to be selected the following SQL can be executed.

```
SELECT Registration, EntryDate, EntryTime, ExitTime FROM CarPark  
  
WHERE NOT EXISTS (SELECT Registration FROM Students WHERE  
CarPark.Registration = Students.Registration)
```

Another useful command would be to clear a table of all its data. This can be done as follows.

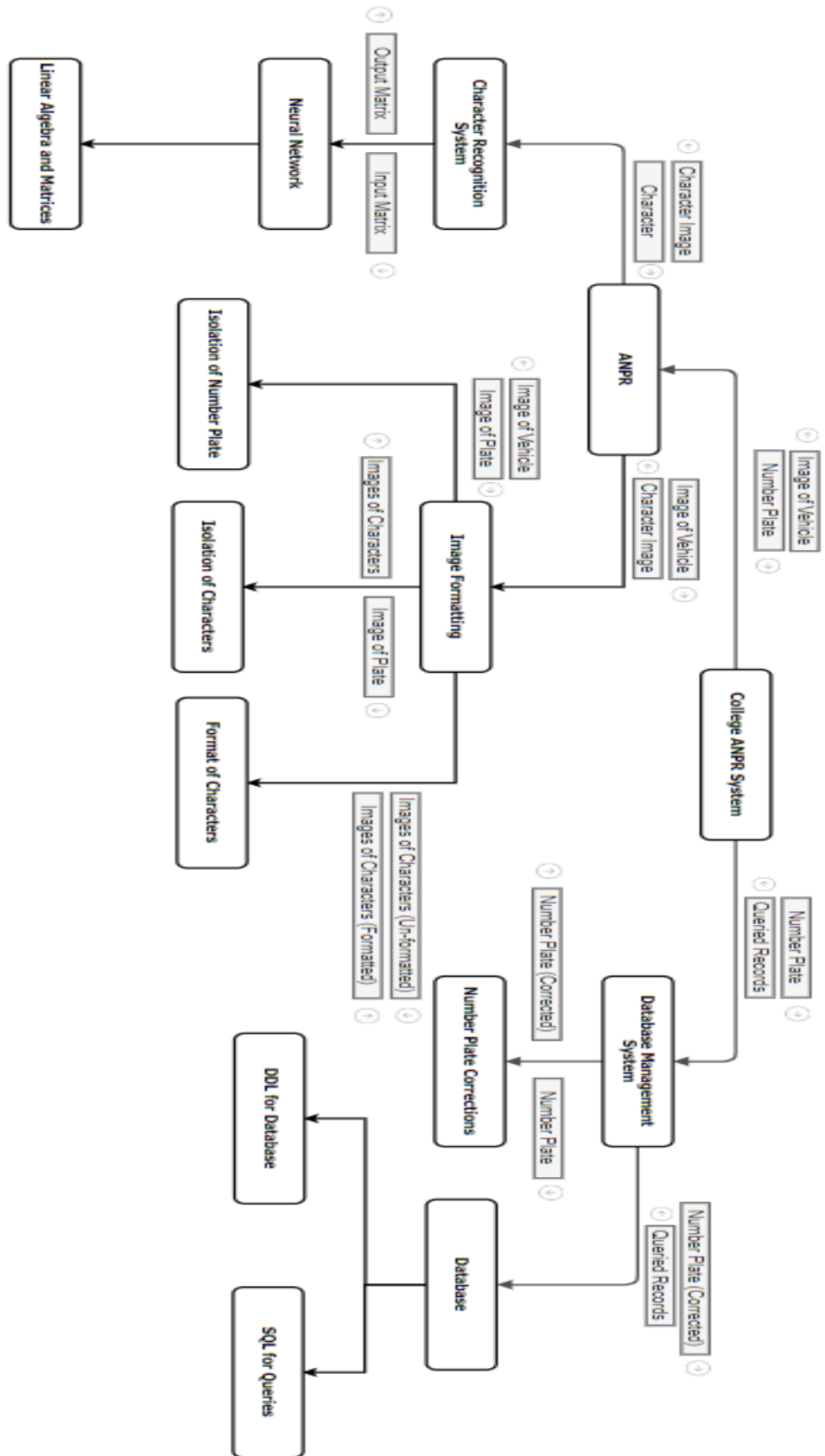
```
DELETE FROM "Table Name" WHERE "Condition"
```

Or to completely delete a table,

```
DROP TABLE "Table Name"
```

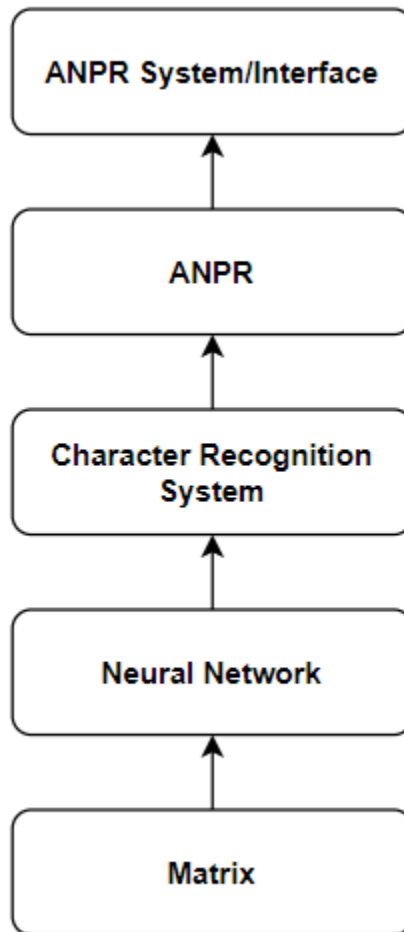
STRUCTURE CHART

This shows a breakdown of the system into its lowest manageable levels. The flow of data around the system is shown. The boxes represent each of the sections of the system and the arrows show how they come together. The labels represent the data that would be passed to and from each of the levels.



CLASS DIAGRAM

The class structure for a system such as this has to be very linear. The image formatting side of the system will be implemented as a module of functions which return the values required from the parameters passed in. These can be called within classes. The same goes for the database side of the system, which will be set up as a module of functions and procedures set up and run queries for connections passed in as parameters. This means that the class set up is linear succession of composition aggregation all the way from the matrix class to the ANPR system. A solid arrow head shows composition.



Although some of these may vary when the technical solution is created, these are the class definitions for each of the classes described above.

(ANPR System/Interface)

ANPRSystem = Class

Public

ANPR: ANPR

DatabaseName: String

DatabaseConnection: Connection (Type TBD)

(ANPR)

ANPR = Class

Public

CharacterRecognitionSystem: CharacterRecognitionSystem

DatasetFileLocation: String

NetworkFileName: String

(Character Recognition System)

CharacterRecognitionSystem = Class

Public

CharacterNeuralNetwork: NeuralNetwork

Inputs(): Real

Targets(): Real

Outputs(): Real

(Neural Network)

NeuralNetwork = Class

Public

InputNodes: Integer

Hidden1Nodes: Integer
Hidden2Nodes: Integer
OutputNodes: Integer
WeightsIH1: Matrix
WeightsH1H2: Matrix
WeightsH2O: Matrix
Hidden1Bias: Matrix
Hidden2Bias: Matrix
OutputBias: Matrix
LearningRate: Real

(Matrix)
Matrix = Class
Public
NumOfRows: Integer
NumOfCols: Integer
Matrix(): Real

TESTING STRATAGY

TESTING OUTLINE

This table describes the format and sections of the testing process.

Test Section	Description	Purpose
1	Validation of Inputs and Interface Testing	Check the Interface is logical and structure and operates in a user-friendly way. Ensure that all inputs to the system can be handled appropriately and no error are thrown.

2	Algorithm/Processes Testing	Check the accuracy of the core of the system including the accuracy of both character and number plate recognition.
3	Data Storage Testing	Check the accuracy and structure of the data stored in the database. The database must be structured as described in design and function properly.
4	IO/Database Query Testing	When querying the database the correct outputs must be produced. These must be displayed in a user friendly way.

SECTION 1 – VALIDATION OF INPUTS AND USER INTERFACE

No	Test	Description	Test Data (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Ref
1.1	General user interface test.	Check that all the user face displays correctly and the layout is as designed and intuitive. All the buttons should function as expected.	N/A *Buttons such as drop downs should be displaying content.	The interface functions properly with all buttons having a clear purpose.		
1.2	Compressing Of Images and User viewing.	Images should be compressed before being processed and stored for both efficiency in processing and storage. The user should be able to view image being processed.	T: 1a, 1b Er: N/A Ex: 1c	All images of valid aspect ratio are converted to a standard resolution.		
1.3	Add a student to the database.	The user should be able to add a student to the student table from	T: 1d Er: 1e	If any invalid input is entered then the		

		within the user interface.	Ex: N/A	user is asked to re-enter all values, else the record is added to the database.		
1.4	Add a vehicle to the database.	The user should be able to add a vehicle to the vehicles table from within the interface.	T: 1f Er: 1g Ex: N/A	If any invalid input is entered then the user is asked to re-enter all values, else the record is added to the database.		

SECTION 2 – ALGORITHM/PROCESSES TESTING

No	Test	Description	Test Data (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Ref
2.1	Single Character Recognition (un-trained network)	Using the testing interface check *using a set of testing data* the accuracy of single character recognition, with a un-trained network.	Single character test set. Neural Network v1.	<5% (This will also produce a cost, C0, for comparison in following tests).		
2.2	Single Character Recognition (partially-	Using the testing interface check *using a set of testing data* the	Single character test set. Neural Network v2	~25%. The key comparison should be		

	trained network – v2).	accuracy of single character recognition, with a partially trained network.	(25% Trained)	made between the costs. C1 << C0		
2.3	Single Character Recognition (partially-trained network – v3).	Using the testing interface check *using a set of testing data* the accuracy of single character recognition, with a partially trained network.	Single character test set. Neural Network v3 (50% trained).	~50%. The key comparison should be made between the costs. C2 < C1		
2.4	Single Character Recognition (partially-trained network – v4).	Using the testing interface check *using a set of testing data* the accuracy of single character recognition, with a partially trained network.	Single character test set. Neural Network v4 (75% trained).	~75%. The key comparison should be made between the costs. C3 < C2		
2.5	Single Character Recognition (partially-trained network – v5).	Using the testing interface check *using a set of testing data* the accuracy of single character recognition, with a trained network.	Single character test set. Neural Network v5 (100% trained).	>85%. The key comparison should be made between the costs. C4 < C3		
2.6	Single Character Recognition (trained network – v5)	Using the testing interface check *using a set of single character testing data* the accuracy of single character recognition, with an trained network.	Single character test set containing only one type of character (Randomly chosen).	>85%		
2.7	Isolation of Number Plates	Using the Image Formatting testing interface check a set of real world images with	Set of Real World Images (Mixed cars	>70%		

		the formatting algorithms.	and backgrounds) e.g. 2a			
2.8	Extreme and Erroneous Test for Isolation of Number Plate.	Using the Image formatting testing interface check a set of random real world images which do NOT contain a number plate.	Set of Real World Images that do not contain a number plate. e.g. 2b	“No Number Plate Found.” Errors are caught and dealt with appropriately.		
2.9	Extraction of Characters from a number plate image.	Using a set of data which can have their number plates extracted in accordance with test 2.7, test to extract the separate character images.	Set of number plate extractable images. e.g. 2c	>70%		
2.10	Standardising of extracted characters.	Check that after a set of character images has been extracted they are formatted and stored in the appropriate way. Do this by saving an example image.	Number plate image which the characters can be extracted. e.g. 2d	A standard resolution greyscale image is produced with the character in the centre of the image.		
2.11	Number Plate Recognition	Using the ANPR system run a set of generated plate images and real world plate images. This test will just evaluate the output of the system (not checks made against the database)	Set (more than 10) of images number plates. e.g. 1a	>70% (Could be less if some images contain only a few errors and can be corrected using the database).		
2.12	Number Plate Recognition Using	Using the ANPR system run a set of generated plate images and real	Set (more than 10) of images	>70%		

	Database Correction.	world plate images. Check the CarPark table for the outputs and any corrections made.	number plates. e.g. 1a			
2.13	General Entry and Exit date and time check on records.	Run through the dataset having the vehicles enter and exit and check the records are updated appropriately.	Set of working images.	All dates and times are entered correctly.		
2.13	Same vehicle entering the premises again before leaving.	Run an image of a number plate (one that provides a valid output). Run the same image again as a vehicle entering the car park.	A working image of a vehicle.	“Vehicle currently in car park”		

SECTION 3 – DATA STORAGE TESTING

No	Test	Description	Test Data (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Ref
3.1	Storage of data in a database.	Check that when a record is submitted the data is correctly sent of to the database and stored in the correct tables.	Number plate Image	The Record is amended to the CarPark Table Correctly (Check in MS Access)		
3.2	General Database Functionality Check	Check the database connections open and close correctly. The database should be structured as	N/A	The database is structured according		




		designed and data flow correctly.		to the design.		
--	--	-----------------------------------	--	----------------	--	--



SECTION 4 – IO AND DATABASE QUERY TESTING

No	Test	Description	Test Data (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Ref
4.1	First Name Query Box Input validation test	Testing with a range of test data the response of the system to different queries under name.	“Matthew” 12247558 “Name...(long)”	All errors are dealt with. For the Erroneous data the query can return no values.		
4.2	Last Name Query Box Input validation test	Testing with a range of test data the response of the system to different queries under name.	“Casey” 12247558 “Name...(long)”	All errors are dealt with. For the Erroneous data the query can return no values.		
4.4	Car Registration Query Box Input validation test	Testing with a range of test data the response of the system to different queries under Registration.	“ENI5 SYB” True “ENI5 SYBX”	All errors are dealt with. For the Erroneous data the query can return no values.		
4.5	StudentID Query Box Input validation test	Testing with a range of test data the response of the system to different queries under StudentID	000001 “Test String” 0000000000001	All errors are dealt with. For the Erroneous data the query can		

				return no values.		
4.5	Test Student Name Query	Using *a testing set of data* check the systems output for queries just under First Name.	“First Name” (In and Not in Database)	All queries return exact correct data.		
4.6	Test Last Name Query	Using *a testing set of data* check the systems output for queries just under the Last Name.	“Student Name” (In and Not In Database)	All queries return exact correct data		
4.7	Test StudentID Query	Using *a testing set of data* check the systems output for queries under the StudentID	0000001 (In and Not In Database)	All queries return exact correct data		
4.8	Test Car Registration Query	Using *a testing set of data* check the systems output for queries just under the car registration.	“Car Registration” (In and Not in Database)	All queries return exact correct data		
4.9	Multiple Queries	Check the system’s ability to handle combinations of the above tests at once (inner join the results)	“First Name” + “Last Name”, StudentID + Registration.	All queries return exact correct data		
5.0	Invalid Parking	Check the Invalid parking query using.	A full database of both valid and invalid vehicles.	All queries return exact correct data		
5.1	Car Park Count Check.	Check system for tracking the number of cars.	Dataset of Images.	An accurate count is kept and displayed.		

TESTING DATA (EXAMPLES)

<p>1a</p>	
<p>1b</p>	
<p>1c</p>	 <p>(Any image with an invalid aspect ratio – the fact the rear number plate is not visible is not relevant).</p>
<p>1d</p>	<p>{000001, “AB12 CDE”, “Tom”, “Smith”, 17}</p>
<p>1e</p>	<p>{“Test”, “AB12 CDE”, “Tom”, “Smith”, 17}</p>

	<p>{000001, 1, "Tom", "Smith", 17}</p> <p>{000001, "AB12 CDE", True, "Smith", 17}</p> <p>{000001, "AB12 CDE", "Tom", "Smith", "Test"}</p>
1f	{ "AB12 CDE", "Hyundai", "i10", "White", "01/01/2019" }
1g	<p>{1, "Hyundai", "i10", "White", "01/01/2019"}</p> <p>{ "AB12 CDE", 100000000000, "i10", "White", "01/01/2019" }</p> <p>{ "AB12 CDE", "Hyundai", 3.465, "White", "01/01/2019" }</p> <p>{ "AB12 CDE", "Hyundai", "i10", "White", 100 }</p>
2a	
2b	

	
2c	
2d	

BETA TESTING PLAN

For the secondary side of testing I will be handing the system over to a member of the security team at college for evaluation. He will be able to personally be able to assess the: interface, system accuracy and storage of the system. The testing will be conducted in a more qualitative way in which they will be asked to complete a survey with a mix of rating and long answer questions. A survey plan is described below.

Questionnaire

1) How would you rate the interface in terms of:

Layout/Design

Not satisfied 1 2 3 4 5 6 7 8 9 10 *Very satisfied*

Justification:

Intuitiveness/Clarity

Not satisfied 1 2 3 4 5 6 7 8 9 10 *Very satisfied*
Justification:

Ease of Use

Not satisfied 1 2 3 4 5 6 7 8 9 10 *Very satisfied*
Justification:

2) How would you rate the accuracy of the ANPR system?

Not Good 1 2 3 4 5 6 7 8 9 10 *Very Good*
Justification:

3) Do you feel the accuracy of the system is sufficient?

Not Enough 1 2 3 4 5 6 7 8 9 10 *Enough*
Justification:

4) Do you feel the system can be implemented into college?

Response:

5) Are there any key features missing from the system that you feel are required?

Response:

6) How do you feel about how the database is structured and the information the system can potentially provide:

Response:

7) Any other comments?

Response:

TESTING

This section will show the testing of the technical solution as described in the Testing Strategy as well as dry-run/step throughs of key algorithms in the solution. The testing will be conducted in a video which will be linked below. The tests will be carried out in sequence in the video and a time stamp placed in the according box of the testing table below (which will be filled out with values from the test).

Video Link: <https://www.youtube.com/watch?v=JmBlfEXpDts&t=19s>

SECTION 1 – VALIDATION OF INPUT AND USER INPTERFACE

No	Test	Description	Test Data (Typical,	Expected Outcome	Actual Outcome	Ref

			Erroneous, Extreme)			
1.1	General user interface test.	Check that all the user face displays correctly and the layout is as designed and intuitive. All the buttons should function as expected.	N/A *Buttons such as drop downs should be displaying content.	The interface functions properly with all buttons having a clear purpose.	The main interface was clear and the layout was simple. All buttons work as expected. This will be further tested in the beta testing.	00:00:13
1.2	Compressing Of Images and User viewing.	Images should be compressed before being processed and stored for both efficiency in processing and storage. The user should be able to view image being processed.	T: 1a, 1b Er: N/A Ex: 1c	All images of valid aspect ratio are converted to a standard resolution. Test 1c image can be distorted.	All testing images were processed with no errors thrown. They were all saved to a file of resolution 260x140. The extreme piece of data was distorted (as expected) but still processed.	00:01:51
1.3	Add a student to the database.	The user should be able to add a student to the student table from within the user interface.	T: 1d Er: 1e Ex: N/A	If any invalid input in entered then the user is asked to re-enter all values, else the record is added to the database.	For a set of valid inputs to all fields a record was sent off to the database. When an error occurred in any of the fields the user was prompted to re-enter all fields. If a repeat record was entered the user was informed of this but not asked to re-enter.	00:04:08

1.4	Add a vehicle to the database.	The user should be able to add a vehicle to the vehicles table from within the interface.	T: 1f Er: 1g Ex: N/A	If any invalid input is entered then the user is asked to re-enter all values, else the record is added to the database.	For a set of valid inputs to all fields a record was sent off to the database. When an error occurred in any of the fields the user was prompted to re-enter all fields. If a repeat record was entered the user was informed of this but not asked to re-enter. Message: "Data Not Inserted. Invalid Input"	00:06:33
-----	--------------------------------	---	----------------------------	--	--	----------

SECTION 2 – ALGORITHM/PROCESSES TESTING

Video Link (Part 1): <https://www.youtube.com/watch?v=KeVViiHCzTk&t=538s>

Video Link (Part 2): https://www.youtube.com/watch?v=HCaFxtV_Vp0&t=16s

No	Test	Description	Test Data (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Ref
2.1	Single Character Recognition (un-trained network)	Using the testing interface check *using a set of testing data* the accuracy of single character recognition, with a un-trained network.	Single character test set. Neural Network v1.	<5% (This will also produce a cost, C0, for comparison in following tests).	Tests: 4.0%, 4.4% Cost: 5852.2	00:01:14
2.2	Single Character Recognition (partially-	Using the testing interface check *using a set of testing data* the	Single character test set. Neural	~25%. The key comparison should	Test: 0.0%, 0.1%	00:03:43

	trained network – v2).	accuracy of single character recognition, with a partially trained network.	Network v2 (25% Trained)	be made between the costs. C1 << C0	Cost: 1034.2	
2.3	Single Character Recognition (partially-trained network – v3).	Using the testing interface check *using a set of testing data* the accuracy of single character recognition, with a partially trained network.	Single character test set. Neural Network v3 (50% trained).	~50%. The key comparison should be made between the costs. C2 < C1	Test: 12.4% Cost: 335.54	00:05:57
2.4	Single Character Recognition (partially-trained network – v4).	Using the testing interface check *using a set of testing data* the accuracy of single character recognition, with a partially trained network.	Single character test set. Neural Network v4 (75% trained).	~75%. The key comparison should be made between the costs. C3 < C2	Test: 23.1% Cost: 287.2	00:07:42
2.5	Single Character Recognition (partially-trained network – v5).	Using the testing interface check *using a set of testing data* the accuracy of single character recognition, with a trained network.	Single character test set. Neural Network v5 (100% trained).	>85%. The key comparison should be made between the costs. C4 < C3	Test: 88.4%, 87.5%, 87.7%, 88.6% Cost: 60.49	00:09:09
2.6	Single Character Recognition (trained network – v5)	Using the testing interface check *using a set of single character testing data* the accuracy of single character recognition, with an trained network.	Single character test set containing only one type of character (Randomly chosen).	>85%	A: 100% 7: 100%	00:11:31
2.7	Isolation of Number Plates	Using the Image Formatting testing interface check a set of real world images with	Set of Real World Images (Mixed cars and	>70%	10 out of 12 images the number plate was fully extracted. The	00:00:56

		the formatting algorithms.	background s) e.g. 2a		other two only partly extracted. 10/12 = 83.3%	
2.8	Extreme and Erroneous Test for Isolation of Number Plate.	Using the Image formatting testing interface check a set of random real world images which do NOT contain a number plate.	Set of Real World Images that do not contain a number plate. e.g. 2b	“No Number Plate Found.” Errors are caught and dealt with appropriately.	Both images without the number plate returned: “Not Found.”. No errors or images were returned (as expected).	00:02:49
2.9	Extraction of Characters from a number plate image.	Using a set of data which can have their number plates extracted in accordance with test 2.7, test to extract the separate character images.	Set of number plate extractable images. e.g. 2c	>70%	6 out of the 10 available images were fully extracted and the other 4 suffered only one error but still extracted most of the characters. 6/10 = 60%	00:03:30
2.10	Standardising of extracted characters.	Check that after a set of character images has been extracted they are formatted and stored in the appropriate way. Do this by saving an example image.	Number plate image which the characters can be extracted. e.g. 2d	A standard resolution greyscale image is produced with the character in the centre of the image.		N/A
2.11	Number Plate Recognition	Using the ANPR system run a set of generated plate images and real world plate images. This test will just evaluate the output of the system (not checks made	Set (more than 10) of images number plates. e.g. 1a	>70% (Could be less if some images contain only a few errors and can be corrected	It fully recognised 4 of the 5 generated images and 2 of the 7 real world images. 6/12 = 50%	00:07:17

		against the database)		using the database).		
2.12	Number Plate Recognition Using Database Correction.	Using the ANPR system run a set of generated plate images and real world plate images. Check the CarPark table for the outputs and any corrections made.	Set (more than 10) of images number plates. e.g. 1a	>70%	It fully recognised 5 of the 5 generated images and 4 of the 6. This equates to 9 out of 12. 9/12 = 75%	00:10:10
2.13	General Entry and Exit date and time check on records.	Run through the dataset having the vehicles enter and exit and check the records are updated appropriately.	Set of working images.	All dates and times are entered correctly.	All entry and exit times are updated as expected. If a car enters after another car but leaves before it, it will still be updated appropriately.	00:12:54
2.13	Same vehicle entering the premises again before leaving.	Run an image of a number plate (one that provides a valid output). Run the same image again as a vehicle entering the car park.	A working image of a vehicle.	“Vehicle currently in car park”	Message Box: “Car Currently in Car Park. Not added to database”. Just running an image as leaving that is not in the database has no effect.	00:14:01

SECTION 3 – DATA STORAGE TESTING

Video Link: <https://www.youtube.com/watch?v=d1a-HI3iqmA&t=26s>

No	Test	Description	Test Data (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Ref

3.1	Storage of data in a database.	Check that when a record is submitted the data is correctly sent off to the database and stored in the correct tables.	Number plate Image	The Record is amended to the CarPark Table Correctly (Check in MS Access)	The record is correctly added to the database in the correct format and saved as seen in MS Access.	00:00:27
3.2	General Database Functionality Check	Check the database connections open and close correctly. The database should be structured as designed and data flow correctly.	N/A	The database is structured according to the design.	All tables can be viewed in the interface and are structured as designed.	00:01:16

SECTION 4 – IO AND DATABASE QUERY TESTING

No	Test	Description	Test Data (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Ref
4.1	First Name Query Box Input validation test	Testing with a range of test data the response of the system to different queries under name.	“Matthew” 12247558 “Name...(long)”	All errors are dealt with. For the Erroneous data the query can return no values.	Typical data returned appropriate record. Erroneous and extreme returned no errors and no records.	00:02:23
4.2	Last Name Query Box Input validation test	Testing with a range of test data the response of the system to different queries under name.	“Casey” 12247558 “Name...(long)”	All errors are dealt with. For the Erroneous data the query can	Typical data returned appropriate record. Erroneous and extreme returned no errors and no records.	00:04:10

				return no values.		
4.4	Car Registration Query Box Input validation test	Testing with a range of test data the response of the system to different queries under Registration.	“ENI5 SYB” True “ENI5 SYBX”	All errors are dealt with. For the Erroneous data the query can return no values.	Typical data returned appropriate record. Erroneous and extreme returned no errors and no records. If any of the letters in the plate were lower case the associated record is returned.	00:04:47
4.5	StudentID Query Box Input validation test	Testing with a range of test data the response of the system to different queries under StudentID	000001 “Test String” 0000000000001	All errors are dealt with. For the Erroneous data the query can return no values.	Typical data returned appropriate record. Erroneous and extreme returned no errors and no records. When a string is entered a message of: “StudentID not Integer”.	00:05:57
4.5	Test Student Name Query	Using *a testing set of data* check the systems output for queries just under First Name.	“First Name” (In and Not in Database)	All queries return exact correct data.	Query data in car park then record is returned, else no record is returned. For a name with two records both are returned.	00:07:07
4.6	Test Last Name Query	Using *a testing set of data* check the systems output for queries just under the Last Name.	“Student Name” (In and Not In Database)	All queries return exact correct data	Query data in car park then record is returned, else no records are returned.	00:09:14

4.7	Test StudentID Query	Using *a testing set of data* check the systems output for queries under the StudentID	0000001 (In and Not In Database)	All queries return exact correct data	Query data in car park then record is returned, else no records are returned.	00:10:16
4.8	Test Car Registration Query	Using *a testing set of data* check the systems output for queries just under the car registration.	“Car Registration” (In and Not in Database)	All queries return exact correct data	Query data in car park then record is returned, else no records are returned.	00:10:42
4.9	Multiple Queries	Check the system’s ability to handle combinations of the above tests at once (inner join the results)	“First Name” + “Last Name”, StudentID + Registration.	All queries return exact correct data	When a query is made which returns multiple records it was refined with a second parameter. The query by first name returned two records which were filtered to one by a last name query. The same could be done for StudentID and registration if required.	00:11:16
5.0	Invalid Parking	Check the Invalid parking query using.	A full database of both valid and invalid vehicles.	All queries return exact correct data	When the Invalid Parking is selected from the drop down an automatic query is made which compares the cars in the car park to those in the database. A series of records is returned.	00:11:48

5.1	Car Park Count Check.	Check system for tracking the number of cars.	Dataset of Images.	An accurate count is kept and displayed.	When a car enters the count is incremented. When a car leaves the count is decremented. The count would be accurate to the level of accuracy provided by the ANPR system.	00:12:53
-----	-----------------------	---	--------------------	--	---	----------

BETA TESTING

The solution was handed over to Christopher Keegan, a member of the Godalming College Security Team, on the 10/03/2020 in Godalming College for beta testing. He was able to test the interface, run images of vehicles through the system and test the database. His responses to the survey are paraphrased below.

Questionnaire

1) How would you rate the interface in terms of:

Layout/Design

Not satisfied 1 2 3 4 5 6 7 8 9 10 *Very satisfied*

“The interface was clean and efficient. The design was definitely of high quality and the only comment to make would be the interface is slightly small especially when considering the volumes of data which may need to be displayed.”

Intuitiveness/Clarity

Not satisfied 1 2 3 4 5 6 7 8 9 10 *Very satisfied*

“Some of the buttons etc required explaining for exactly what they do however this is sort of expected. The querying of the database seemed to be intuitive however ‘select all’ is a bit ambiguous and for those not familiar with a database system ‘run query’ may not be that obvious.”

Ease of Use

Not satisfied 1 2 3 4 5 6 7 8 9 10 *Very satisfied*

“The system seemed to function smoothly and I came across no issues in terms of the running/flow of the interface. It seemed fairly easy to use except for the small points mentioned earlier.”

- 2) How would you rate the accuracy of the ANPR system?

Not Good 1 2 3 4 5 6 7 8 9 10 *Very Good*

“Most of the images were recognised (even with the database but that seems reasonable). There were definitely a range of images that could be recognised with a variety of backgrounds. There were some it did not manage to get which is why it is not given a perfect score but the accuracy is still very high.”

- 3) Do you feel the accuracy of the system is sufficient?

Not Enough 1 2 3 4 5 6 7 8 9 10 *Enough*

“As I just said the accuracy of the system I feel is very high so do believe that is it definitely sufficient to implement here at college. The images tested may have even been more complex/detailed (in terms of backgrounds etc) than that which would be processed at college. Many here would be captured with limited background etc from a nice angle, so I would assume the accuracy would be even greater.”

- 4) Do you feel the system can be implemented into college?

“The foundation is definitely there for a system which can be implemented in college. The manual input of images would obviously have to be changed for an automated one, but it would seem reasonable that some trigger could be placed at the entrance to the car park which triggers and image capture. Other than that the processing side of the system seems there.”

- 5) Are there any key features missing from the system that you feel are required?

“If a system was used where plates were recognised from an image extracted from a video feed It would have been good to see an example of that being implemented (essentially recognition from a video feed). Also a feature to be able to remove a student/vehicle from the database may be nice.”

- 6) How do you feel about how the database is structured and the information the system can potentially provide:

“As you have realised there would be large volumes of data coming through this system. The database seems to be structured efficiently. I like the system for adding students and vehicles to the database as it fits perfectly with the current system. It really nicely displays all the key information and I like how it all fits together.”

- 7) Any other comments?

“I am definitely very pleased with the system provided and seems to meet all the requirements. With a few changes it could definitely be implemented.”

 DEMONSTRATING KEY ALGORITHMS

BACKPROPAGATION

To demonstrate the functionality of my backpropagation algorithm and therefore the validity of the neural network I will be dry running a simple version of the network. For obvious reasons I will not be able to dry run the real network as this contains 900 inputs each of which would have to be processed. Instead a simplified network comprised of two inputs, two hidden nodes and one output node will be tested. A network such as this can be trained to perform logical operations on two inputs (e.g. AND, OR).

The network will be defined as,

```
Dim ANDNeuralNet As New NeuralNetworkSLP(2, 4, 1)
```

With a set of possible inputs,

```
Dim Inputs1() As Integer = {0, 0}
Dim Inputs2() As Integer = {1, 0}
Dim Inputs3() As Integer = {0, 1}
Dim Inputs4() As Integer = {1, 1}
```

And possible outputs,

```
Dim Outputs(1) As Single
Dim Output(0) As Single
```

Then using the backpropagation method the network will be trained,

```
For iter As Integer = 0 To 1000
    ANDNeuralNet.Backpropagation(Inputs1, Outputs1)
    ANDNeuralNet.Backpropagation(Inputs2, Outputs1)
    ANDNeuralNet.Backpropagation(Inputs3, Outputs1)
    ANDNeuralNet.Backpropagation(Inputs4, Outputs2)
Next
```

The user is then asked to provide an input and the system gives an output,

```
Console.Write("Input 1: ")
Inputs(0) = Console.ReadLine()
Console.Write("Input 2: ")
Inputs(1) = Console.ReadLine()

Output = ANDNeuralNet.FeedForward(Inputs)
Console.WriteLine("Output: " & Output(0))

If Output(0) > 0.5 Then
    Console.WriteLine("Output: 1")
Else
```

```

    Console.WriteLine("Output: 0")
End If
Console.ReadLine()

```

I will be dry-running the system with the inputs (1, 1). First I will check that the correct outputs are produced when the inputs are fed forward.

$$\begin{pmatrix} W1 & W2 \\ W3 & W4 \end{pmatrix} \times \begin{pmatrix} I1 \\ I2 \end{pmatrix} = \begin{pmatrix} W1I1 + W2I2 \\ W3I1 + W4I2 \end{pmatrix} = \begin{pmatrix} H1 \\ H2 \end{pmatrix}$$

To test the system I will be using the random weight and bias matrices of,

$$\text{WeightsBetweenInputHidden} = \begin{pmatrix} 0.46 & 0.97 \\ 0.32 & 0.65 \end{pmatrix}$$

$$\text{WeightsBetweenHiddenOutput} = \begin{pmatrix} 0.12 \\ 0.78 \end{pmatrix}$$

$$\text{BiasHidden} = \begin{pmatrix} 0.54 \\ 0.77 \end{pmatrix}$$

$$\text{BiasOutput} = (0.18)$$

Therefore the Hidden Matrix should be,

$$\text{Hidden} = \begin{pmatrix} 0.46 \times 1 + 0.97 \times 1 \\ 0.32 \times 1 + 0.65 \times 1 \end{pmatrix} = \begin{pmatrix} 1.43 \\ 0.97 \end{pmatrix} + \text{Hidden Bias} = \begin{pmatrix} 1.43 \\ 0.97 \end{pmatrix} + \begin{pmatrix} 0.54 \\ 0.77 \end{pmatrix} = \begin{pmatrix} 1.97 \\ 1.74 \end{pmatrix}$$

Then passed through sigmoid,

$$\text{Hidden} = \begin{pmatrix} 0.877 \\ 0.851 \end{pmatrix}$$

When a breakpoint was set after the hidden layer was calculated the Hidden matrix held the values,

Hidden	{Neural_Net_Library.Matrix}	Neural_Net_Library.Matrix
Matrix	{Length=2}	Single()
(0, 0)	0.8776111	Single
(1, 0)	0.8506871	Single
NumOfArrayCols	0	Integer
NumOfArrayRows	1	Integer
NumOfCols	1	Integer
NumOfRows	2	Integer

The Output Matrix is calculated by,

$$\begin{pmatrix} W5 & W6 \end{pmatrix} \times \begin{pmatrix} H1 \\ H2 \end{pmatrix} = \begin{pmatrix} W5H1 + W6H2 \end{pmatrix} = (O)$$

Therefore Output is given by,

$$\begin{aligned} \text{Output} &= (0.12 \times 0.877 + 0.78 \times 0.851) = (0.769) + \text{Output Bias} \\ &= (0.769) + (0.18) = (0.949) \end{aligned}$$

Then passed through sigmoid,

Output = (0.721)

When a breakpoint is set after the output is calculated the output can be seen,

Output	{Neural_Net_Library.Matrix}	Neural_Net_Library.Matrix
Matrix	{Length=1}	Single()
(0, 0)	0.720883667	Single
NumOfArrayCols	0	Integer
NumOfArrayRows	0	Integer
NumOfCols	1	Integer
NumOfRows	1	Integer

Using this output we can backpropagate the error to hand calculate the changes to the weights and compare these to the values calculated by the solution.

As the rate of change of the total error for a weight in the 2nd layer is described by,

$$\frac{dE_{Total}}{dW_{Hx}} = 2(O_x - T_x)Sig(O_{xIN})(1 - Sig(O_{xIN}))H_x$$

For this simple version the changes to the second layer weights expressed as a matrix would be calculated as such.

$$\begin{pmatrix} \Delta W_5 \\ \Delta W_6 \end{pmatrix} = 2(0.721 - 1)Sig(0.721)(1 - Sig(0.721)) \begin{pmatrix} 0.877 \\ 0.851 \end{pmatrix}$$

Therefore,

$$\begin{pmatrix} \Delta W_5 \\ \Delta W_6 \end{pmatrix} = -0.123 \times \begin{pmatrix} 0.877 \\ 0.851 \end{pmatrix} = \begin{pmatrix} -0.0986 \\ -0.0956 \end{pmatrix}$$

Note these values are all rounded to 3SF but the calculations held exact values all the way through.

Setting a breakpoint after the deltas are calculated shows,

OutputGradient	{Neural_Net_Library.Matrix}	Neural_Net_Library.Matrix
Matrix	{Length=2}	Single()
(0, 0)	-0.0985752344	Single
(0, 1)	-0.0955510661	Single
NumOfArrayCols	1	Integer
NumOfArrayRows	0	Integer
NumOfCols	2	Integer
NumOfRows	1	Integer

As the rate of change of the total error for a weight in the 1st layer is described by,

$$\frac{dE_{Total}}{dW_x} = 2(O_x - T_x)Sig(O_{xIN})(1 - Sig(O_{xIN}))W_{Hx}H_x(1 - H_x)I_x$$

The first part has been previously calculated,

$$\frac{dE_{Total}}{dW_x} = -0.123 \times \begin{pmatrix} W_5 \\ W_6 \end{pmatrix} \times H_x(1 - H_x)I_x$$

$$\begin{pmatrix} \Delta W1 & \Delta W2 \\ \Delta W3 & \Delta W4 \end{pmatrix} = \begin{pmatrix} 0.12 \\ 0.78 \end{pmatrix} (-0.123) \times (scalar) \begin{pmatrix} 0.877(1 - 0.877) \\ 0.851(1 - 0.851) \end{pmatrix} \times (1 \quad 1)$$

Note dimensionally this produces a 2x2 matrix.

Therefore,

$$\begin{pmatrix} \Delta W1 & \Delta W2 \\ \Delta W3 & \Delta W4 \end{pmatrix} = \begin{pmatrix} -0.01476 \\ -0.09594 \end{pmatrix} \times (scalar) \begin{pmatrix} 0.1079 \\ 0.1268 \end{pmatrix} \times (1 \quad 1)$$

$$\begin{pmatrix} \Delta W1 & \Delta W2 \\ \Delta W3 & \Delta W4 \end{pmatrix} = \begin{pmatrix} -1.5 \times 10^{-3} & -1.5 \times 10^{-3} \\ -0.012 & -0.012 \end{pmatrix}$$

The changes are given to 2SF and note that large rounding error could be present.

When a breakpoint is set after the changes to the first layer weights are calculated the values are given.

HiddenGradient	(Neural_Net_Library.Matrix)	Neural_Net_Library.Matrix
Matrix	(Length=4)	Single()
(0, 0)	-0.00144774164	Single
(0, 1)	-0.00144774164	Single
(1, 0)	-0.0111282663	Single
(1, 1)	-0.0111282663	Single

The tracking of these values through the backpropagation algorithm show that the implementation is working as expected and producing the correct values.

EVALUATION

TO WHAT EXTENT DOES THE PROJECT MEET ITS REQUIREMENTS?

Functional Requirements

(INPUTS)

1. This system must be able to operate for at least the entirety of the college day (8:00 to 5:00).

For this requirement it is difficult to assess whether or not it has been met. The solution was not made for recognition within a video but it is clear from the solution that it can easily be adapted into it. Currently, for testing purposes, the interface has user input but that is clear not required so this target has been met in that the system is set up to allow for this to be met.

- 1.1. The system should run automatically once set up, so require no user input to start recording on any given day.

As with the previous requirement the system has been designed to allow for this to be the case but for obvious reasons it is not currently able to do this (for hardware constraints).

2. The system be able to process image captures of vehicles entering the site. The mechanics behind how the images are extracted from the video are not required.

- 2.1. Each of these captures should be compressed and formatted before being stored and passed into the main system.

The images being used by the system are stored in a standard resolution and format. The 30x30 input dimensions for the network have worked well. This requirement has been fully met.

- 2.2. The User of the system should be able to view the image being processed.

The image is clearly on display in the interface.

3. The User should be able to add a student and a vehicle to the database from inputs to the main interface.

The system for adding records to the database works well and efficiently and seems to suit the individual style to which students apply for a parking permit. The requirement has been fully met. There are two potential improvements one would be to be able to add multiple students at once, but this does not seem to be too essential. The other would be to be able to delete records from the database. That was not required however and would have meant compromising on the compact feel of the interface. When testing no errors occurred in adding to the database and any erroneous inputs were caught and the user was asked to re-enter.

4. The Users inputs for querying the database should be validated.

All query inputs are validated and any erroneous pieces of data are caught and dealt with appropriately. The query system is set out clearly and includes all key criteria fields.

(PROCESSING)

5. The system should process each input image to produce a standard formatted output.

- 5.1. The formatted image should be with resolution x by y.

The images are formatted to a 30x30 resolution which worked well as a compromise between quality of image (and therefore detail to analyse) and processing rate/efficiency. Any larger and training the network may have taken too long. Any smaller and the quality could have been insufficient.

- 5.2. The image should be grayscale or black and white.

The images were set to grayscale for reduction in file size which allow for the large volumes of training data to be formed.

- 5.3. Each pixel should have an associated/stored normalized value between 0 and 1 for use in the network.

The normalizing of pixel values worked well. The range of 0 to 1 worked well with the training algorithm and the activation function.

6. This system must be able to identify a number plate within an image.
 - 6.1. The solution should be able to identify a number plate within an image at a success rate at or above 70%.

As 83.3% of random number plates were extracted in the testing of the system it is clear that this target has been met. Just due to natural variation in the images being tested it is hard to say that this could be much more. The recursive algorithm used to extract the number was very precise in terms of finding the borders of the plate. It could be improved slightly by having a tolerance in which if a single pixel is found in a run which is not a number plate pixel it allows it and continues to search; it only stops when it reaches a run of invalid pixels. This would mean that the system would work when the plate contained small errors e.g. mud. The success rate provided however is clearly sufficient and this requirement has been met.

7. The system should be able to isolate the number plate within the image.
 - 7.1. It should then decompose the number plate image into images of each character on the number plate.
 - 7.2. It should be able to accurately decompose the image of the number plate into separate character images at a success rate at or above 70%.

In the testing it managed a 60% full character extracting rate which is below the requirement of 70%. I would argue that multiple failed on just one character, so if you consider passing that into the system and the fact that it can have up to two errors and be recognised with the database, the success rate of the system as a whole may not be affected too badly. Many of the errors occurred with small irregularities in the plate, which could be corrected if a tolerance system was put in place. The recursive algorithm used was clearly precise and effective and overall I would say the requirement has been met when considering the effectiveness across the whole system.

8. The system should be able to identify a single character within a 28 by 28 grid of normalized pixels.
 - 8.1. The solution should be able to identify these characters at a success rate above 85%.

In the end a 30x30 image was used for quality of image reasons. When the system was tested randomly 4 times the average accuracy of the system came out to be 88.1% which is above the required level. Some individual characters even tested up to 100% accuracy. The key way I feel that that could be improved is if the training dataset was more varied. As no external dataset could be found one had to be generated to this may have been a limiting factor. In terms of the structure of the network (multi-layer perceptron) and the backpropagation/training algorithms used the network has been created perfectly and exactly to plan. From research even the best MLP Character recognition systems will only hit about 90% so the execution of the implementation of the network could not really be improved but minor improvements could be made if other external factors were different.

9. The system should be able to combine a correct series of recognitions as to enable it to recognise an entire number plate.
 - 9.1. The solution should be able to correctly identify number plates at success rate at or above 70%. This can include number plates in which errors were made but the plate was still identified correctly used adjustments with the database (see 10.2).

The 75% accuracy of the overall ANPR system clearly meets this requirement. This does include a set of generated plates as well as real world plates, however from my beta testing its clear that the images passed in if the system was used at college would be far simpler with less background information etc. So I would say these effects cancel and the accuracy has been truly measured. This accuracy appears to be sufficient for the college to implement – obviously the greater the better.

10. The system should be able to produce a list of invalid vehicles to enter the premises (this does not have to be able table in itself but can be obtained through querying the database)

The invalid vehicles are in a separate drop down of the select table box and work well on the premise that any vehicle in the car park that is not in the vehicles database is invalid. An issue arises when a plate is incorrectly identified and it is then added to the invalid vehicles even though its valid. However I feel this error is limited and could not be improved by any means except improving the network.

- 10.1. If the same number plate is recognised again (before it has been determined to leave the site) no new record will be created (assumed to be stationary next to camera).

This works well and the same plate can be run multiple times with no errors. This would be useful when a car is sat by the camera at exit times.

- 10.2. If the system incorrectly processes an image and produces an output with 2 or fewer errors to any number plate in the database, this will be corrected and added to the database.

This system works well. The condition of 2 errors seems appropriate as it allows for plates with just natural small errors to still pass but those when there is just too much of a difference to fail.

(STORAGE)

11. The solution should include a normalised database that contains the tables specified (See E-R Diagram).

- 11.1. When a number plate is recognised (for a vehicle entering the site) the system should submit a record to the database including: Registration (PK), Entry Date, EntryTime and a Null ExitTime.

- 11.2. When a number plate is recognised (for a vehicle leaving the site) the system should update the corresponding record in the database to that vehicle with its ExitTime.

- 11.3. The database should contain a table of student IDs (primary key) and their associated data (see E-R Diagram).

- 11.4. The database should contain a table of registered vehicles. Registration number (PK), Make, Model, Colour and Registration Date.

The database system has been structured according to the design. Records are submitted successfully when a vehicle enters the car park. Times are updated and stored correctly. The

overall flow of data in the database is efficient and has no problems. There were no ways no improve this that have come to light in the testing of the system.

(OUTPUTS)

12. The user must be able to query the database and search for any record in the database in a simple way.

The query system as a whole works well and returns all records by the criteria. The fields that can be queried by are limited by comprehensive. One improvement could be to increase the number of fields that can be queried by. For example including, Date of Registration, Car Make and Car Model. These could have been added but did not seem to be required by the user, so for simplicity of the interface they were left out.

12.1. The extent of the criteria for a search are described by, but not limited to, the examples listed in section 13.1.

12.1.1. Search by Student Name (First or Surname)

12.1.2. Search by Car Registration

12.1.3. Search by StudentID

13. The system should have a built in process that can query the database in such a way as to output the list of invalid vehicles (past or present) in the car park.

14. The User should be able to view the Students and Vehicles Databases within the interface.

All other key points about the database were discussed in the storage section above.

Additional Requirements

(OUTPUTS)

15. The system should be able to provide a count for the number of spaces used and therefore the spaces available in the carpark currently.

15.1. This count should be accurate to within 5 spaces.

This count works well and will be accurate to the accuracy of the image formatting system. It will only not add a vehicle when no plate could be extracted from the image.

INDEPENDENT FEEDBACK

The system was passed over to James Hancock, a student at Godalming College, on the 20/03/2020 along with the requirements so he could provide independent feedback on the system and how well it meets those requirements. He is a student who drives into college so would make use of such as system, also he is a Computer Science Student so would be able to analyse the technical side of the system (e.g. the success of the neural network) with a realistic nature. His evaluation of the system is provided below.

The problem being attempted has many layers of complexity, many of which are very hard to fully achieve, however the requirements set out would provide a comprehensive solution to the problem. To start, the interface is very clean and well-structured and easy to use. There were no faults that I could find and even through thorough testing could not cause any errors/exceptions to be thrown. I like the system for showing the difference between a vehicle entering and leaving the site however it would be nice to see the system working in its entirety meaning plates identified from a video feed. I can see why this has not been done for testing purposes and it is only a small extension to the core system which is already in place.

The neural network is clearly well designed and implemented correctly so that it can work for the task at hand. The structure of the network seems appropriate and the choice for two-dimensional data structures for manipulation as the network is definitely quick and efficient. This has allowed the network to be trained to a high level – easily sufficient for the solution. There is no improvement which would be made to the network.

The algorithm used to extract the number plate is in theory very quick, precise and efficient. It is clear though on implementation that it is common for some small deviation/irregularity to cause issue in the algorithm. In the real world these would be fairly common. Therefore a bit more robustness in extracting of the number plate could be useful. The results seen clearly meet the requirements for the system so the implementation clearly works to a high level.

The algorithm used to extract the characters can fall to the same issues that the one to extract the number plate does: small irregularities in the image can cause issues. This is definitely less pronounced in the character extracting as more conditions have been put in place to make the process more robust. The extracting of characters appears to work at a level easily sufficient for the project which meets the requirements.

In terms of the project as a whole and how the collection of this data leads to it being stored in a database, this system appears to have very few to no faults. The database is set out in a normalised way that allows data to flow nicely in and out of the database. The updating of the times of vehicles in and out of the database works very well and I could not find any way to cause an error/exception in this system.

If I was to summarise what went well with the implementation I would definitely say the structure/design and implementation of the neural network, the design suited the problem and it functioned perfectly. To further that the data management and the final user interface worked really well and complemented the underlying system perfectly. Looking at what could be improved overall, it would just be the formatting systems to extract the character images. Although the algorithms were well designed and even implemented, they did not so much consider the natural variation which can occur

Note: The evaluation of the independent feedback is included in the overall critical evaluation of the system below.

OVERALL EVALUATION OF SYSTEM

There are three key different components which combine to form the system as a whole: Character Recognition/Neural Network, Image Formatting and the Database. Each of those will be analysed in turn here followed by an analysis of the resulting system which they formed.

The character recognition using a neural network formed the majority of the complex processing of the system.; It was clearly the only way to approach solving the problem. The choice of using matrices in the form of two dimensional arrays for the implementation of the network worked well and allowed for a fully working neural network to be formed. The success rates of above 85% show how well the network was able to work and are close to the maximum possible success rates for a network of that topology. A limitation described earlier was the training dataset, which I do not feel contained enough variation as is often present in characters on a number plate (due to a variety of factors). This may have limited the accuracy of the network but was not a factor that was in my control. If the project was attempted again then it could be a good idea for gather a dataset of real world plate characters; I cannot suggest where this would be found and in the research for this project this was attempted and none could be found. To evaluate the neural network as a whole it seemed to work completely as expected and was a strong component of the final system.

The Image Formatting encompasses both the isolation of the number plate within the image and the splitting of that image into individual characters. To start with the isolation of the number plate, the recursive algorithm of checking for runs of valid pixels was theoretically complex but very efficient. The implementation clearly worked but small intricacies in the image could cause it to fail. If this part of the system was attempted again then I would definitely still use a recursive algorithm however a tolerance would be added which would ensure that the algorithm doesn't break as soon as a small difference is found. Essentially the algorithm could be made more layered with more conditions that have to be met. In terms of splitting up into characters another recursive algorithm was used. This worked very well, precisely and consistently. There was even a few conditions placed on this such as a black pixel has to be in the middle 4/5th of the height to ensure none are just found at the top where the plate borders the car. This part was potentially the strongest of the image formatting section and worked very consistently. Continuing this the formatting the characters for the network worked very well especially when the morphing error was resolved when the image was resized; this part of the system was handled as expected. To evaluate the formatting system as a whole if it was definitely very effective and worked very well to meet the requirements set. If it was attempted again I could potentially consider creating a system which would be able to extract plates at different angles (even though that is quite rare); this would require potentially completely rethinking the system for very slight accuracy improvements.

The database side of the project was perhaps the least complex from a processing point of view but clear efficient execution was key. The structure of the database worked very well for the types and volumes of data that the system would experience. All queries to the database were formed well and very hard to cause errors with from the interface. The database as a whole met the requirements entirely with the only potential comment to be to increase the number of possible query fields by adding; car model, car make etc. This would not be that essential and could easily be done.

All three of those components combined to form the system as a whole. Going through the requirements it appears the system as a whole seems to have been successful. Clearly it could not be implemented in its current state and small changes would have to be made for that to be the case, but that was not required on my part for obvious reasons. When analysing the system it was clear the compound effect that the accuracies of each part had on the overall accuracy. With essentially three

components (network, plate, characters) that contain an associated accuracy, these multiplied to give an accuracy of the system. That why the overall 75% accuracy definitely shows the quality of the execution of the solution; it could definitely be put into practice if required.

TECHNICAL SOLUTION

This is a print of the entire code of the system. The total line count for the entire system is **2091**.

GENERATING TRAINING DATASET USING PYTHON AND NUMPY

```
1. import cv2
2. import numpy as np
3. import math
4. from random import randint
5. import os
6.
7. def CreateDataset(ASCIICode, NumberOfImages):
8.     character = chr(ASCIICode)
9.     filename = "C:\\Users\\matth\\OneDrive\\Documents\\College\\Year 2\\Computing\\NEA\\Processed Dataset\\" + character +
    "\\\"
10.     try:
11.         os.mkdir(filename)
12.     except:
13.         pass
14.     for imageiter in range(NumberOfImages):
15.
16.         filename = "C:\\Users\\matth\\OneDrive\\Documents\\College\\Year 2\\Computing\\NEA\\Raw Dataset\\" + character + ".
    PNG"
17.         img = cv2.imread(filename)
18.         img = cv2.resize(img, (100, 100))
19.         img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
20.
21.         x1 = randint(25, 30)
22.         x2 = randint(70, 75)
23.         y1 = randint(20, 25)
24.         y2 = randint(20, 25)
25.
26.         for x in range(100):
27.             for y in range(100):
28.                 if img[x,y] < 75:
```

```
29.             img[x, y] = 0
30.         else:
31.             img[x,y] = 255
32.
33.         pts1 = np.float32([[50,75],[25,25],[75,25]])
34.         pts2 = np.float32([[50,75],[x1,y1],[x2,y2]])
35.
36.         M = cv2.getAffineTransform(pts1,pts2)
37.
38.         img = cv2.warpAffine(img, M, (100, 100), borderMode=cv2.BORDER_REPLICATE)
39.
40.         for x in range(100):
41.             for y in range(100):
42.                 if img[x, y] == 214:
43.                     img[x, y] = 255
44.
45.         img = cv2.resize(img, (30, 30))
46.         filename = "C:\\Users\\matth\\OneDrive\\Documents\\College\\Year 2\\Computing\\NEA\\Processed Dataset\\" + character + "\\\"
47.         filename += character
48.         filename += str(imageiter)
49.         filename += ".BMP"
50.         cv2.imwrite(filename, img)
51.
52.
53. def CreateAlphabet(Datapoints):
54.     for charIter in range(65, 91):
55.         CreateDataset(charIter, Datapoints)
56.
57. def CreateNumbers(Datapoints):
58.     for charIter in range(48, 58):
59.         CreateDataset(charIter, Datapoints)
60.
61. CreateAlphabet(1000)
62. CreateNumbers(1000)
```


LINEAR ALGEBRA CLASS

```
1. Public Class Matrix
2.
3.     'This class will contain all the required data and methods for manipulation when creating the neural network.
4.     Public Property NumOfRows As Integer
5.     Public Property NumOfCols As Integer
6.     Public Property NumOfArrayRows As Integer
7.     Public Property NumOfArrayCols As Integer
8.
9.     Public Matrix(,) As Single
10.
11.     Public Sub New(ByVal NumberOfRows As Integer, ByVal NumberOfCols As Integer)
12.         Try
13.             Me.NumOfRows = NumberOfRows
14.             Me.NumOfCols = NumberOfCols
15.             Me.NumOfArrayRows = NumberOfRows - 1
16.             Me.NumOfArrayCols = NumberOfCols - 1
17.
18.             ReDim Matrix(NumOfArrayRows, NumOfArrayCols)
19.
20.         Catch ex As Exception
21.             Console.WriteLine("Invalid Rows or Columns")
22.         End Try
23.     End Sub
24.
25.     Public Sub DisplayMatrix()
26.         'Useful when testing to see if values match expected.
27.         Console.WriteLine()
28.         For RowsIter As Integer = 0 To NumOfArrayRows
29.             For ColsIter As Integer = 0 To NumOfArrayCols
30.                 Console.Write(Matrix(RowsIter, ColsIter) & vbTab)
31.             Next
32.             Console.WriteLine()
33.             Console.WriteLine()
34.         Next
35.         Console.WriteLine("=====")
```

```
36. End Sub
37.
38. Public Sub IdentityMatrix()
39.     'Sets the matrix to the equivalence of 1.
40.     For RowsIter As Integer = 0 To NumOfArrayRows
41.         For ColsIter As Integer = 0 To NumOfArrayCols
42.             If ColsIter = RowsIter Then
43.                 Matrix(RowsIter, ColsIter) = 1
44.             Else
45.                 Matrix(RowsIter, ColsIter) = 0
46.             End If
47.         Next
48.     Next
49. End Sub
50.
51. Public Sub Randomize()
52.     'Randomizes ever cell to a real number: -1 < x < 1
53.     For RowsIter As Integer = 0 To NumOfArrayRows
54.         For ColsIter As Integer = 0 To NumOfArrayCols
55.             Matrix(RowsIter, ColsIter) = GenerateRandomNumber() * 2 - 1
56.         Next
57.     Next
58. End Sub
59.
60. Public Sub RandomizeByRange(Range)
61.     For RowsIter As Integer = 0 To NumOfArrayRows
62.         For ColsIter As Integer = 0 To NumOfArrayCols
63.             Matrix(RowsIter, ColsIter) = Math.Floor(GenerateRandomNumber() * Range)
64.         Next
65.     Next
66. End Sub
67.
68. Public Sub ElementWiseAddition(ByRef InputMatrix As Matrix)
69.     If CheckDimensionsForElementWise(InputMatrix) Then
70.         For RowsIter As Integer = 0 To NumOfArrayCols
71.             For ColsIter As Integer = 0 To NumOfArrayRows
72.                 Matrix(ColsIter, RowsIter) += InputMatrix.Matrix(ColsIter, RowsIter)
73.             Next
74.         Next
75.     Else
76.         Console.WriteLine("Invalid Dimensions of Input Matrix (Addition - Class Sub)")
```

```
77.     End If
78. End Sub
79.
80. Public Sub ElementWiseSubtraction(ByRef InputMatrix As Matrix)
81.     If CheckDimensionsForElementWise(InputMatrix) Then
82.         For RowsIter As Integer = 0 To NumOfArrayRows
83.             For ColsIter As Integer = 0 To NumOfArrayCols
84.                 Matrix(RowsIter, ColsIter) -= InputMatrix.Matrix(RowsIter, ColsIter)
85.             Next
86.         Next
87.     Else
88.         Console.WriteLine("Invalid Dimensions of Input Matrix (Subtraction - Class Sub)")
89.     End If
90. End Sub
91.
92. Public Function CheckDimensionsForElementWise(ByRef InputMatrix As Matrix)
93.     If NumOfCols = InputMatrix.NumOfCols And NumOfRows = InputMatrix.NumOfRows Then
94.         Return True
95.     Else
96.         Return False
97.     End If
98. End Function
99.
100. Public Function CheckDimensionsForMultiplication(ByRef Matrix As Matrix)
101.     If Me.NumOfCols = Matrix.NumOfRows Then
102.         Return True
103.     Else
104.         Return False
105.     End If
106. End Function
107.
108. Public Sub MatrixMultiplicationOfInput(ByRef InputMatrix As Matrix)
109.     If CheckDimensionsForMultiplication(InputMatrix) Then
110.         Dim ResultRows As Integer = NumOfArrayRows
111.         Dim ResultCols As Integer = InputMatrix.NumOfArrayCols
112.         Dim Result(ResultRows, ResultCols) As Single
113.         Dim CurrentProduct As Single
114.         For ResultRowsIter As Integer = 0 To ResultRows
115.             For ResultColsIter As Integer = 0 To ResultCols
116.                 CurrentProduct = 0
117.                 For ProductIterator As Integer = 0 To ResultRows - 1
```

```
118.             CurrentProduct += Matrix(ResultRowsIter, ProductIterator) * InputMatrix.Matrix(ProductIterator,
    ResultColsIter)
119.                 Next
120.             Result(ResultColsIter, ResultRowsIter) = CurrentProduct
121.         Next
122.     Next
123.     Me.Matrix = Result
124.     Me.NumOfCols = ResultCols + 1
125.     Me.NumOfRows = ResultRows + 1
126.     Me.NumOfArrayCols = ResultCols
127.     Me.NumOfArrayRows = ResultRows
128. Else
129.     Console.WriteLine("Invalid Dimensions of Input Matrix (Multiplication - Class Sub)")
130. End If
131. End Sub
132.
133. Public Sub ScalarMultiplicationOfInput(Input As Matrix)
134.     If CheckDimensionsForElementWise(Input) Then
135.         For i As Integer = 0 To Me.NumOfArrayRows
136.             For j As Integer = 0 To Me.NumOfArrayCols
137.                 Me.Matrix(i, j) *= Input.Matrix(i, j)
138.             Next
139.         Next
140.     Else
141.         Console.WriteLine("Invalid Dimensions of Input Matrix (Scalar Multiplication - Class Sub)")
142.     End If
143.
144. End Sub
145.
146. Public Sub Transpose()
147.     Dim Result(NumOfArrayCols, NumOfArrayRows) As Single
148.     For RowsIter As Integer = 0 To NumOfArrayRows
149.         For ColsIter As Integer = 0 To NumOfArrayCols
150.             Result(ColsIter, RowsIter) = Matrix(RowsIter, ColsIter)
151.         Next
152.     Next
153.     ReDim Matrix(NumOfArrayCols, NumOfArrayRows)
154.     Matrix = Result
155.     NumOfCols = NumOfArrayRows + 1
156.     NumOfRows = NumOfArrayCols + 1
157.     NumOfArrayCols = NumOfCols - 1
```

```
158.         NumOfArrayRows = NumOfRows - 1
159.     End Sub
160.
161.     Public Sub Sigmoid()
162.         'Applies the activation function to every item.
163.         For i As Integer = 0 To NumOfArrayRows
164.             For j As Integer = 0 To NumOfArrayCols
165.                 Dim x As Single = Matrix(i, j)
166.                 Matrix(i, j) = 1 / (1 + Math.Exp(-x))
167.             Next
168.         Next
169.     End Sub
170.
171.     Public Sub MultiplicationOfSingleValue(Value As Single)
172.         For i As Integer = 0 To Me.NumOfArrayRows
173.             For j As Integer = 0 To Me.NumOfArrayCols
174.                 Me.Matrix(i, j) *= Value
175.             Next
176.         Next
177.     End Sub
178.
179.     Public Sub ZeroMatrix()
180.         For RowsIter As Integer = 0 To NumOfArrayRows
181.             For ColsIter As Integer = 0 To NumOfArrayCols
182.                 Matrix(RowsIter, ColsIter) = 0
183.             Next
184.         Next
185.     End Sub
186.
187.     Public Function CostSummationOfElements()
188.         'Useful when evaluating the effectiveness of the training of the network.
189.         Dim Sum As Single
190.         For RowsIter As Integer = 0 To NumOfArrayRows
191.             For ColsIter As Integer = 0 To NumOfArrayCols
192.                 Sum += (Matrix(RowsIter, ColsIter)) ^ 2
193.             Next
194.         Next
195.         Return Sum
196.     End Function
197.
198.
```

199. End Class

LINEAR ALGEBRA MODULES

```
1. Module MatrixOperations
2.
3.     'Many methods that appeared in the class are repeated as a static function to suit for different situations when creati
ng the neural network.
4.
5.     Public Function MatrixMultiplication(ByRef Matrix1 As Matrix, ByRef Matrix2 As Matrix)
6.         If CheckDimensionsForMultiplication(Matrix1, Matrix2) Then
7.             Dim ResultRows As Integer = Matrix1.NumOfRows
8.             Dim ResultCols As Integer = Matrix2.NumOfCols
9.             Dim Result As New Matrix(ResultRows, ResultCols)
10.            Dim Sum As Single
11.            For i As Integer = 0 To ResultRows - 1
12.                For j As Integer = 0 To ResultCols - 1
13.                    Sum = 0
14.                    For k As Integer = 0 To Matrix1.NumOfArrayCols
15.                        Sum += Matrix1.Matrix(i, k) * Matrix2.Matrix(k, j)
16.                    Next
17.                    Result.Matrix(i, j) = Sum
18.                Next
19.            Next
20.            Return Result.Matrix
21.        Else
22.            Console.WriteLine("Invalid Dimensions of Input Matrix (Multiplication - Module Sub)")
23.            Return False
24.        End If
25.
26.    End Function
27.
28.    Public Function CheckDimensionsForMultiplication(ByRef Matrix1 As Matrix, ByRef Matrix2 As Matrix)
29.        If Matrix1.NumOfCols = Matrix2.NumOfRows Then
30.            Return True
31.        Else
32.            Return False
33.        End If
```

```
34.
35.     End Function
36.
37.     Public Function CheckDimensionsForElementwise(Matrix1 As Matrix, Matrix2 As Matrix)
38.         If Matrix1.NumOfArrayRows = Matrix2.NumOfArrayRows And Matrix1.NumOfCols = Matrix2.NumOfCols Then
39.             Return True
40.         Else
41.             Return False
42.         End If
43.     End Function
44.
45.     Public Function TransposeMatrix(ByRef Matrix As Matrix)
46.         Dim Result(Matrix.NumOfArrayCols, Matrix.NumOfArrayRows) As Single
47.         For RowsIter As Integer = 0 To Matrix.NumOfArrayRows
48.             For ColsIter As Integer = 0 To Matrix.NumOfArrayCols
49.                 Result(ColsIter, RowsIter) = Matrix.Matrix(RowsIter, ColsIter)
50.             Next
51.         Next
52.         Return Result
53.     End Function
54.
55.     Public Function MatrixFromArray(Input As Array)
56.         Dim Mat As New Matrix(Input.Length, 1)
57.         For i As Integer = 0 To Input.Length - 1
58.             Mat.Matrix(i, 0) = Input(i)
59.         Next
60.         Return Mat
61.     End Function
62.
63.     Public Function ArrayFromMatrix(Input As Matrix)
64.         Dim arr As New List(Of Single)
65.         For i As Integer = 0 To Input.NumOfArrayRows
66.             For j As Integer = 0 To Input.NumOfArrayCols
67.                 arr.Add(Input.Matrix(i, j))
68.             Next
69.         Next
70.         Return arr.ToArray()
71.     End Function
72.
73.     Public Function GenerateRandomNumber() As Single
74.         Static Random_Number As New Random()
```

```
75.         Return Random_Number.Next(0, 1000) / 1000
76.     End Function
77.
78.     Public Function MatrixAddition(Matrix1 As Matrix, Matrix2 As Matrix)
79.         Dim Result As New Matrix(Matrix1.NumOfRows, Matrix1.NumOfCols)
80.         If CheckDimensionsForElementwise(Matrix1, Matrix2) Then
81.             For RowsIter As Integer = 0 To Matrix1.NumOfArrayCols
82.                 For ColsIter As Integer = 0 To Matrix1.NumOfArrayRows
83.                     Result.Matrix(ColsIter, RowsIter) = Matrix1.Matrix(ColsIter, RowsIter) + Matrix2.Matrix(RowsIter, ColsI
ter)
84.                 Next
85.             Next
86.         Else
87.             Console.WriteLine("Invalid Dimensions of Input Matrix (Addition - Module Sub)")
88.         End If
89.         Return Result
90.     End Function
91.
92.     Public Function MatrixSubtraction(Matrix1 As Matrix, Matrix2 As Matrix)
93.         Dim Result As New Matrix(Matrix1.NumOfRows, Matrix1.NumOfCols)
94.         If CheckDimensionsForElementwise(Matrix1, Matrix2) Then
95.             For RowsIter As Integer = 0 To Matrix1.NumOfArrayRows
96.                 For ColsIter As Integer = 0 To Matrix1.NumOfArrayCols
97.                     Result.Matrix(RowsIter, ColsIter) = Matrix1.Matrix(RowsIter, ColsIter) - Matrix2.Matrix(RowsIter, ColsI
ter)
98.                 Next
99.             Next
100.        Else
101.            Console.WriteLine("Invalid Dimensions of Input Matrix (Addition - Module Sub)")
102.        End If
103.        Return Result
104.    End Function
105.
106.    Public Function ReturnSigmoid(Matrix As Matrix)
107.        Dim Result As New Matrix(Matrix.NumOfRows, Matrix.NumOfCols)
108.        For RowsIter = 0 To Matrix.NumOfArrayRows
109.            For ColsIter = 0 To Matrix.NumOfArrayCols
110.                Result.Matrix(RowsIter, ColsIter) = 1 / (1 + Math.Exp(Matrix.Matrix(RowsIter, ColsIter)))
111.            Next
112.        Next
113.        Return Result
```



```

114.     End Function
115.
116.     Public Function ReturnDerivativeSigmoid(Matrix As Matrix)
117.         Dim Result As New Matrix(Matrix.NumOfRows, Matrix.NumOfCols)
118.         Dim y As Single
119.         For RowsIter = 0 To Matrix.NumOfArrayRows
120.             For ColsIter = 0 To Matrix.NumOfArrayCols
121.                 y = Matrix.Matrix(RowsIter, ColsIter)
122.                 Result.Matrix(RowsIter, ColsIter) = y * (1 - y)
123.             Next
124.         Next
125.         Return Result
126.     End Function
127.
128.     Public Function ReturnTransposition(Matrix As Matrix)
129.         Dim Result As New Matrix(Matrix.NumOfCols, Matrix.NumOfRows)
130.         For RowsIter As Integer = 0 To Matrix.NumOfArrayRows
131.             For ColsIter As Integer = 0 To Matrix.NumOfArrayCols
132.                 Result.Matrix(ColsIter, RowsIter) = Matrix.Matrix(RowsIter, ColsIter)
133.             Next
134.         Next
135.         Return Result
136.     End Function
137.
138. End Module

```

NEURAL NETWORK CLASS

```

1. Public Class NeuralNetworkMLP
2.
3.     Public Property InputNodes As Integer
4.     Public Property Hidden1Nodes As Integer
5.     Public Property Hidden2Nodes As Integer
6.     Public Property OutputNodes As Integer
7.
8.     'Weight Matrixes notated as XY meaning Between X and Y
9.     Public Property WeightsIH1 As Matrix
10.    Public Property WeightsH1H2 As Matrix

```

```
11.     Public Property WeightsH20 As Matrix
12.
13.     'Biases
14.     Public Property Hidden1Bias As Matrix
15.     Public Property Hidden2Bias As Matrix
16.     Public Property OutputBias As Matrix
17.
18.     'Learning Rate
19.     Public Property LearningRate As Single
20.
21.     'Properties for Stochastic Training
22.     Public Property TotHidden1Gradient As Matrix
23.     Public Property TotHidden2Gradient As Matrix
24.     Public Property TotOutputGradient As Matrix
25.     Public Property TotHidden1BiasGradient As Matrix
26.     Public Property TotHidden2BiasGradient As Matrix
27.     Public Property TotOutputBiasGradient As Matrix
28.
29.     Public Sub New(InputNodes As Integer, Hidden1Nodes As Integer, Hidden2Nodes As Integer, OutputNodes As Integer)
30.
31.         Me.InputNodes = InputNodes
32.         Me.Hidden1Nodes = Hidden1Nodes
33.         Me.Hidden2Nodes = Hidden2Nodes
34.         Me.OutputNodes = OutputNodes
35.
36.         Me.WeightsIH1 = New Matrix(Hidden1Nodes, InputNodes)
37.         Me.WeightsH1H2 = New Matrix(Hidden2Nodes, Hidden1Nodes)
38.         Me.WeightsH2O = New Matrix(OutputNodes, Hidden2Nodes)
39.
40.         Me.Hidden1Bias = New Matrix(Hidden1Nodes, 1)
41.         Me.Hidden2Bias = New Matrix(Hidden2Nodes, 1)
42.         Me.OutputBias = New Matrix(OutputNodes, 1)
43.
44.         Me.LearningRate = 0.1
45.
46.         WeightsIH1.Randomize()
47.         WeightsH1H2.Randomize()
48.         WeightsH2O.Randomize()
49.         Hidden1Bias.Randomize()
50.         Hidden2Bias.Randomize()
51.         OutputBias.Randomize()
```

```
52.
53.     'Instantiate the Matrix for Stochastic Decent Training
54.     Me.TotHidden1Gradient = New Matrix(Hidden1Nodes, InputNodes)
55.     Me.TotHidden2Gradient = New Matrix(Hidden2Nodes, Hidden1Nodes)
56.     Me.TotOutputGradient = New Matrix(OutputNodes, Hidden2Nodes)
57.     Me.TotHidden1BiasGradient = New Matrix(Hidden1Nodes, 1)
58.     Me.TotHidden2BiasGradient = New Matrix(Hidden2Nodes, 1)
59.     Me.TotOutputBiasGradient = New Matrix(OutputNodes, 1)
60.
61. End Sub
62.
63. Public Sub Backpropagation(InputsArray As Array, TargetsArray As Array, StochasticDecent As Boolean)
64.
65.     'Convert Input Arrays to Matrixes
66.     Dim Inputs As New Matrix(InputNodes, 1)
67.     Dim Targets As New Matrix(OutputNodes, 1)
68.     Inputs = MatrixOperations.MatrixFromArray(InputsArray)
69.     Targets = MatrixOperations.MatrixFromArray(TargetsArray)
70.
71.     'Feed Input Forward through existing network
72.     Dim Hidden1 As New Matrix(Hidden1Nodes, 1)
73.     Dim Hidden2 As New Matrix(Hidden2Nodes, 1)
74.     Dim Output As New Matrix(OutputNodes, 1)
75.
76.     Hidden1.Matrix = MatrixOperations.MatrixMultiplication(WeightsIH1, Inputs)
77.     Hidden1.ElementWiseAddition(Hidden1Bias)
78.     Hidden1.Sigmoid()
79.
80.     Hidden2.Matrix = MatrixOperations.MatrixMultiplication(WeightsH1H2, Hidden1)
81.     Hidden2.ElementWiseAddition(Hidden2Bias)
82.     Hidden2.Sigmoid()
83.
84.     Output.Matrix = MatrixOperations.MatrixMultiplication(WeightsH2O, Hidden2)
85.     Output.ElementWiseAddition(OutputBias)
86.     Output.Sigmoid()
87.
88.     'CALCULATE OUTPUT GRADIENT
89.     Dim OutputError As New Matrix(OutputNodes, 1)
90.     Dim OutputErrorDerivative As New Matrix(OutputNodes, 1)
91.     Dim OutputSigmoidDerivative As New Matrix(OutputNodes, 1)
92.     Dim OutputGradient As New Matrix(OutputNodes, Hidden2Nodes)
```

```
93.         Dim OutputBiasGradient As New Matrix(OutputNodes, 1)
94.         Dim Hidden2Transposed As New Matrix(1, Hidden2Nodes)
95.
96.         OutputError = MatrixOperations.MatrixSubtraction(Targets, Output)
97.         OutputError.ScalarMultiplicationOfInput(OutputError)
98.
99.         OutputErrorDerivative = MatrixOperations.MatrixSubtraction(Output, Targets)
100.        OutputErrorDerivative.MultiplicationOfSingleValue(2)
101.
102.        OutputSigmoidDerivative = MatrixOperations.ReturnDerivativeSigmoid(Output)
103.
104.        OutputErrorDerivative.ScalarMultiplicationOfInput(OutputSigmoidDerivative)
105.
106.        Hidden2Transposed = MatrixOperations.ReturnTransposition(Hidden2)
107.        OutputGradient.Matrix = MatrixOperations.MatrixMultiplication(OutputErrorDerivative, Hidden2Transposed)
108.
109.        OutputBiasGradient = OutputErrorDerivative
110.
111.        OutputGradient.MultiplicationOfSingleValue(LearningRate)
112.        OutputBiasGradient.MultiplicationOfSingleValue(LearningRate)
113.
114.        'CALCULATE HIDDEN2 LAYER GRADIENT
115.        Dim Hidden2ErrorDerivative As New Matrix(Hidden2Nodes, 1)
116.        Dim Hidden2SigmoidDerivative As New Matrix(Hidden2Nodes, 1)
117.        Dim Hidden2Gradient As New Matrix(Hidden2Nodes, Hidden1Nodes)
118.        Dim WeightsH2OT As New Matrix(OutputNodes, Hidden2Nodes)
119.        Dim Hidden2BiasGradient As New Matrix(Hidden2Nodes, 1)
120.        Dim Hidden1Transposed As New Matrix(1, Hidden1Nodes)
121.
122.        WeightsH2OT = MatrixOperations.ReturnTransposition(WeightsH2O)
123.        Hidden2ErrorDerivative.Matrix = MatrixOperations.MatrixMultiplication(WeightsH2OT, OutputErrorDerivative)
124.
125.        Hidden2SigmoidDerivative = ReturnDerivativeSigmoid(Hidden2)
126.
127.        Hidden2ErrorDerivative.ScalarMultiplicationOfInput(Hidden2SigmoidDerivative)
128.
129.        Hidden1Transposed = MatrixOperations.ReturnTransposition(Hidden1)
130.        Hidden2Gradient.Matrix = MatrixOperations.MatrixMultiplication(Hidden2ErrorDerivative, Hidden1Transposed)
131.
132.        Hidden2BiasGradient = Hidden2ErrorDerivative
133.
```

```

134.         Hidden2Gradient.MultiplicationOfSingleValue(LearningRate)
135.         Hidden2BiasGradient.MultiplicationOfSingleValue(LearningRate)
136.
137.         'CACULATE HIDDEN1 LAYER GRADIENT
138.         Dim Hidden1ErrorDerivative As New Matrix(Hidden1Nodes, 1)
139.         Dim Hidden1SigmoidDerivative As New Matrix(Hidden1Nodes, 1)
140.         Dim Hidden1Gradient As New Matrix(Hidden1Nodes, InputNodes)
141.         Dim WeightsH1H2T As New Matrix(Hidden2Nodes, Hidden1Nodes)
142.         Dim Hidden1BiasGradient As New Matrix(Hidden1Nodes, 1)
143.         Dim InputsTransposed As New Matrix(1, InputNodes)
144.
145.         WeightsH1H2T = MatrixOperations.ReturnTransposition(WeightsH1H2)
146.         Hidden1ErrorDerivative.Matrix = MatrixOperations.MatrixMultiplication(WeightsH1H2T, Hidden2ErrorDerivative)
147.
148.         Hidden1SigmoidDerivative = ReturnDerivativeSigmoid(Hidden1)
149.
150.         Hidden1ErrorDerivative.ScalarMultiplicationOfInput(Hidden1SigmoidDerivative)
151.
152.         InputsTransposed = MatrixOperations.ReturnTransposition(Inputs)
153.         Hidden1Gradient.Matrix = MatrixOperations.MatrixMultiplication(Hidden1ErrorDerivative, InputsTransposed)
154.
155.         Hidden1BiasGradient = Hidden1ErrorDerivative
156.
157.         Hidden1Gradient.MultiplicationOfSingleValue(LearningRate)
158.         Hidden1BiasGradient.MultiplicationOfSingleValue(LearningRate)
159.
160.         'Apply Gradients in some way either now or add to stochastic totals
161.
162.         If StochasticDecent Then
163.             TotHidden1Gradient.ElementWiseAddition(Hidden1Gradient)
164.             TotHidden1BiasGradient.ElementWiseAddition(Hidden1BiasGradient)
165.             TotHidden2Gradient.ElementWiseAddition(Hidden2Gradient)
166.             TotHidden2BiasGradient.ElementWiseAddition(Hidden2BiasGradient)
167.             TotOutputGradient.ElementWiseAddition(OutputGradient)
168.             TotOutputBiasGradient.ElementWiseAddition(OutputBiasGradient)
169.         Else
170.             WeightsIH1.ElementWiseSubtraction(Hidden1Gradient)
171.             WeightsH1H2.ElementWiseSubtraction(Hidden2Gradient)
172.             WeightsH2O.ElementWiseSubtraction(OutputGradient)
173.             Hidden1Bias.ElementWiseSubtraction(Hidden1BiasGradient)
174.             Hidden2Bias.ElementWiseSubtraction(Hidden2BiasGradient)

```

```
175.         OutputBias.ElementWiseSubtraction(OutputBiasGradient)
176.     End If
177.
178. End Sub
179.
180. Public Function Feedforward(InputsArray As Array) As Array
181.     'Feed Input Forward through existing network
182.
183.     Dim Inputs As New Matrix(InputNodes, 1)
184.     Inputs = MatrixOperations.MatrixFromArray(InputsArray)
185.
186.     Dim Hidden1 As New Matrix(Hidden1Nodes, 1)
187.     Dim Hidden2 As New Matrix(Hidden2Nodes, 1)
188.     Dim Output As New Matrix(OutputNodes, 1)
189.
190.     Hidden1.Matrix = MatrixOperations.MatrixMultiplication(WeightsIH1, Inputs)
191.     Hidden1.ElementWiseAddition(Hidden1Bias)
192.     Hidden1.Sigmoid()
193.
194.     Hidden2.Matrix = MatrixOperations.MatrixMultiplication(WeightsH1H2, Hidden1)
195.     Hidden2.ElementWiseAddition(Hidden2Bias)
196.     Hidden2.Sigmoid()
197.
198.     Output.Matrix = MatrixOperations.MatrixMultiplication(WeightsH2O, Hidden2)
199.     Output.ElementWiseAddition(OutputBias)
200.     Output.Sigmoid()
201.
202.     Dim OutputsArray(OutputNodes - 1) As Single
203.     OutputsArray = MatrixOperations.ArrayFromMatrix(Output)
204.     Return OutputsArray
205.
206. End Function
207.
208. Public Sub SaveNetworkToFile(FileName As String)
209.
210.     'STRUCTURE
211.     '- Weights Between Input and Hidden1
212.     '- Weights Between Hidden1 and Hidden2
213.     '- Weights Between Hidden2 and Output
214.     '- Bias for Hidden1 layer
215.     '- Bias for Hidden2 layer
```

```
216.     '- Bias for Output layer
217.     '(All formatted across then down)
218.
219.     FileName += ".txt"
220.
221.     Dim StreamWriter As StreamWriter = New StreamWriter(FileName)
222.
223.     For Citer As Integer = 0 To WeightsIH1.NumOfArrayCols
224.         For Riter As Integer = 0 To WeightsIH1.NumOfArrayRows
225.             StreamWriter.WriteLine(CStr(WeightsIH1.Matrix(Riter, Citer)))
226.         Next
227.     Next
228.
229.     For Citer As Integer = 0 To WeightsH1H2.NumOfArrayCols
230.         For Riter As Integer = 0 To WeightsH1H2.NumOfArrayRows
231.             StreamWriter.WriteLine(CStr(WeightsH1H2.Matrix(Riter, Citer)))
232.         Next
233.     Next
234.
235.     For Citer As Integer = 0 To WeightsH2O.NumOfArrayCols
236.         For Riter As Integer = 0 To WeightsH2O.NumOfArrayRows
237.             StreamWriter.WriteLine(CStr(WeightsH2O.Matrix(Riter, Citer)))
238.         Next
239.     Next
240.
241.     For iter As Integer = 0 To Hidden1Bias.NumOfArrayRows
242.         StreamWriter.WriteLine(CStr(Hidden1Bias.Matrix(iter, 0)))
243.     Next
244.
245.     For iter As Integer = 0 To Hidden2Bias.NumOfArrayRows
246.         StreamWriter.WriteLine(CStr(Hidden2Bias.Matrix(iter, 0)))
247.     Next
248.
249.     For iter As Integer = 0 To OutputBias.NumOfArrayRows
250.         StreamWriter.WriteLine(CStr(OutputBias.Matrix(iter, 0)))
251.     Next
252.
253.     StreamWriter.Close()
254.
255. End Sub
256.
```

```
257.     Public Sub LoadNetwork(fileName As String)
258.
259.         fileName += ".txt"
260.
261.         Dim StreamReader As StreamReader = New StreamReader(fileName)
262.
263.         For Citer As Integer = 0 To WeightsIH1.NumOfArrayCols
264.             For Riter As Integer = 0 To WeightsIH1.NumOfArrayRows
265.                 WeightsIH1.Matrix(Riter, Citer) = CSng(StreamReader.ReadLine())
266.             Next
267.         Next
268.
269.         For Citer As Integer = 0 To WeightsH1H2.NumOfArrayCols
270.             For Riter As Integer = 0 To WeightsH1H2.NumOfArrayRows
271.                 WeightsH1H2.Matrix(Riter, Citer) = CSng(StreamReader.ReadLine())
272.             Next
273.         Next
274.
275.         For Citer As Integer = 0 To WeightsH2O.NumOfArrayCols
276.             For Riter As Integer = 0 To WeightsH2O.NumOfArrayRows
277.                 WeightsH2O.Matrix(Riter, Citer) = CSng(StreamReader.ReadLine())
278.             Next
279.         Next
280.
281.         For iter As Integer = 0 To Hidden1Bias.NumOfArrayRows
282.             Hidden1Bias.Matrix(iter, 0) = CSng(StreamReader.ReadLine())
283.         Next
284.
285.         For iter As Integer = 0 To Hidden2Bias.NumOfArrayRows
286.             Hidden2Bias.Matrix(iter, 0) = CSng(StreamReader.ReadLine())
287.         Next
288.
289.         For iter As Integer = 0 To OutputBias.NumOfArrayRows
290.             OutputBias.Matrix(iter, 0) = CSng(StreamReader.ReadLine())
291.         Next
292.
293.         StreamReader.Close()
294.
295.     End Sub
296.
297.     Public Sub ApplyStochasticDecent(Divisor As Integer)
```



```
298.  
299.         Dim Multiplier As Single = 1 / Divisor  
300.  
301.         'Average The Decent Gradients  
302.         TotHidden1BiasGradient.MultiplicationOfSingleValue(Multiplier)  
303.         TotHidden1Gradient.MultiplicationOfSingleValue(Multiplier)  
304.         TotHidden2BiasGradient.MultiplicationOfSingleValue(Multiplier)  
305.         TotHidden2Gradient.MultiplicationOfSingleValue(Multiplier)  
306.         TotOutputBiasGradient.MultiplicationOfSingleValue(Multiplier)  
307.         TotOutputGradient.MultiplicationOfSingleValue(Multiplier)  
308.  
309.         'ADJUST THE WEIGHT MATRIXES BY GRADIENTS  
310.         WeightsIH1.ElementWiseSubtraction(TotHidden1Gradient)  
311.         WeightsH1H2.ElementWiseSubtraction(TotHidden2Gradient)  
312.         WeightsH2O.ElementWiseSubtraction(TotOutputGradient)  
313.  
314.         'ADJUST THE BIASES BY GRADIENTS  
315.         Hidden1Bias.ElementWiseSubtraction(TotHidden1BiasGradient)  
316.         Hidden2Bias.ElementWiseSubtraction(TotHidden2BiasGradient)  
317.         OutputBias.ElementWiseSubtraction(TotOutputBiasGradient)  
318.  
319.         End Sub  
320.  
321.         Public Sub ZeroStochasticDecentTotals()  
322.             TotHidden1BiasGradient.ZeroMatrix()  
323.             TotHidden2BiasGradient.ZeroMatrix()  
324.             TotHidden1Gradient.ZeroMatrix()  
325.             TotHidden2Gradient.ZeroMatrix()  
326.             TotOutputBiasGradient.ZeroMatrix()  
327.             TotOutputGradient.ZeroMatrix()  
328.         End Sub  
329.  
330.  
331.     End Class
```

CHARACTER RECOGNITION SYSTEM CLASS

```

1. Public Class CharacterRecognitionSystem
2.
3.     'This class is composed of the neural network class.
4.     'It brings together its methods With the procedures required To train And run the network in a useful way.
5.
6.     'Declaration of Network Structure
7.     Public CharacterNeuralNetwork As NeuralNetworkMLP
8.     Public Inputs() As Single
9.     Public Targets() As Single
10.    Public Outputs() As Single
11.
12.    Public Sub New(ByRef InputNodes As Integer, ByRef Hidden1Nodes As Integer, ByRef Hidden2Nodes As Integer, ByRef OutputNodes As Integer)
13.        'Dimensionalising each of the data structures
14.        CharacterNeuralNetwork = New NeuralNetworkMLP(InputNodes, Hidden1Nodes, Hidden2Nodes, OutputNodes)
15.        ReDim Inputs(InputNodes - 1)
16.        ReDim Targets(OutputNodes - 1)
17.        ReDim Outputs(OutputNodes - 1)
18.    End Sub
19.
20.    Public Sub SetCurrentInputValues(ByRef CurrentCharacter As Char, ByRef CurrentImage As Integer)
21.        'This routine will extract all of the data from a bitmap image file.
22.        'It will normalise the greyscale pixel values between 0 and 1.
23.        'Then assign the result to the corresponding position in the inputs array.
24.        Dim ActivationValue As Single
25.        Dim ReadFileName As String
26.        ReadFileName = "C:/Users/matth/OneDrive/Documents/College/Year 2/Computing/NEA/Processed Dataset/"
27.        ReadFileName += CurrentCharacter & "/" & CurrentCharacter & CStr(CurrentImage) & ".BMP"
28.        Dim FileImage As Image = Image.FromFile(ReadFileName)
29.        Dim FileBitmap As New Bitmap(FileImage)
30.        Dim CurrentColor As Color
31.        For xiter As Integer = 0 To FileBitmap.Width - 1
32.            For yiter As Integer = 0 To FileBitmap.Height - 1

```

```
33.         CurrentColor = FileBitmap.GetPixel(xiter, yiter)
34.         ActivationValue = CurrentColor.B 'The chosen RGB value does not matter.
35.         ActivationValue /= 255 'Normalizes the values between 0 and 1
36.         ActivationValue = 1.0 - ActivationValue 'As images were black on white this makes black pixels come out as
1 and white as 0 (not visa versa).
37.         Inputs((30 * xiter) + yiter) = ActivationValue
38.     Next
39. Next
40. End Sub
41.
42. Public Sub SetCurrentTargetValues(ByRef CurrentCharacter As Char)
43.     'Sets the values in the Targets array to 0 except for the value at the target position which is set to 1.
44.     'This process is executed in terms of ASCII character codes for iteration purposes
45.     'In the targets array the order is 0-9 then A-Z.
46.     Dim ASCIICode As Integer
47.     ASCIICode = Asc(CurrentCharacter)
48.     For TargetIter As Integer = 0 To 9
49.         If ASCIICode = TargetIter + 48 Then
50.             Targets(TargetIter) = 1
51.         Else
52.             Targets(TargetIter) = 0
53.         End If
54.     Next
55.     For TargetIter As Integer = 10 To 35
56.         If ASCIICode = TargetIter + 55 Then
57.             Targets(TargetIter) = 1
58.         Else
59.             Targets(TargetIter) = 0
60.         End If
61.     Next
62. End Sub
63.
64. Public Sub TrainNetwork(ByRef Epochs As Integer, ByRef NumberOfBatches As Integer, ByRef BatchSize As Integer, ByRef Fi
leName As String)
65.     'This procedure will train the network using stochastic decent.
66.     'It will collect and average the gradients for a 'BatchSize' number of datapoints, then apply them.
67.     'Epochs are used to monitor training progress.
68.     Dim OriginalCostOfNetwork As Single
69.     LoadNetwork(FileName)
70.     For EpochIter As Integer = 1 To Epochs
71.         OriginalCostOfNetwork = CostOfNetwork(990, 999)
```

```

72. Console.WriteLine("Epoch Run")
73. Console.WriteLine("Cost: " & Str(OriginalCostOfNetwork))
74. For BatchIter As Integer = 1 To NumberOfBatches
75.     CharacterNeuralNetwork.ZeroStochasticDecentTotals()
76.     For BatchSizeIter As Integer = 1 To BatchSize
77.         For CharIter As Integer = 48 To 57
78.             SetCurrentInputValues(Chr(CharIter), GenerateRandomNumber() * 1000) 'Picks a random image in the da
taset.
79.             SetCurrentTargetValues(Chr(CharIter))
80.             CharacterNeuralNetwork.Backpropagation(Inputs, Targets, True)
81.         Next
82.         For CharIter As Integer = 65 To 90
83.             SetCurrentInputValues(Chr(CharIter), GenerateRandomNumber() * 1000)
84.             SetCurrentTargetValues(Chr(CharIter))
85.             CharacterNeuralNetwork.Backpropagation(Inputs, Targets, True)
86.         Next
87.     Next
88.     CharacterNeuralNetwork.ApplyStochasticDecent((Outputs.Length() * BatchSize))
89. Next
90. If OriginalCostOfNetwork > CostOfNetwork(990, 999) Then 'Only overwrites the network if its an improvement on th
e last version.
91.     SaveNetwork(FileName)
92. End If
93. Next
94. End Sub
95.
96. Private Function CostOfNetwork(ByRef StartDatapoint As Integer, EndDatapoint As Integer)
97.     'Useful for numerically observing the effect of training the network
98.     'This returns a value associated with the square of the error produced by feeding data through the network
99.     Dim IndividualCost, TotalCost As Single
100.    Dim OutputsMatrix, TargetsMatrix As Matrix
101.    TotalCost = 0
102.    For DataPointIter As Integer = StartDatapoint To EndDatapoint
103.        For CharIter As Integer = 48 To 57
104.            SetCurrentInputValues(Chr(CharIter), DataPointIter)
105.            SetCurrentTargetValues(Chr(CharIter))
106.            Outputs = CharacterNeuralNetwork.Feedforward(Inputs)
107.            OutputsMatrix = MatrixOperations.MatrixFromArray(Outputs)
108.            TargetsMatrix = MatrixOperations.MatrixFromArray(Targets)
109.            OutputsMatrix.ElementWiseSubtraction(TargetsMatrix)
110.            IndividualCost = OutputsMatrix.CostSummationOfElements()

```

```

111.         TotalCost += IndividualCost
112.     Next
113.     For CharIter As Integer = 65 To 90
114.         SetCurrentInputValues(Chr(CharIter), DataPointIter)
115.         SetCurrentTargetValues(Chr(CharIter))
116.         Outputs = CharacterNeuralNetwork.Feedforward(Inputs)
117.         OutputsMatrix = MatrixOperations.MatrixFromArray(Outputs)
118.         TargetsMatrix = MatrixOperations.MatrixFromArray(Targets)
119.         OutputsMatrix.ElementWiseSubtraction(TargetsMatrix)
120.         IndividualCost = OutputsMatrix.CostSummationOfElements()
121.         TotalCost += IndividualCost
122.     Next
123. Next
124. Return TotalCost
125. End Function
126.
127. Public Sub LoadNetwork(ByRef FileName As String)
128.     CharacterNeuralNetwork.LoadNetwork(FileName)
129. End Sub
130.
131. Public Sub SaveNetwork(ByRef FileName As String)
132.     CharacterNeuralNetwork.SaveNetworkToFile(FileName)
133. End Sub
134.
135. Public Function TestNetwork(ByRef NumberOfTests As Integer, Optional TestCharacter As Char = "", Optional ShowOutput
As Boolean = False) As String
136.     'This procedure evaluates how many of a random sample of images the network can guess correctly.
137.     'It depends on a set of parameters to allow for a variety of tests.
138.     Dim CorrectTests, TestImageNumber As Integer
139.     Dim TestForSpecificCharacter As Boolean
140.     Dim Character, CharacterGuess As Char
141.     Dim MaxValue, PercentageCorrect As Single
142.     If TestCharacter <> Nothing Then
143.         Character = TestCharacter
144.         TestForSpecificCharacter = True
145.     End If
146.     CorrectTests = 0
147.     For TestIter As Integer = 1 To NumberOfTests
148.         If Not TestForSpecificCharacter Then
149.             Character = Chr(65 + (Math.Floor(GenerateRandomNumber() * 26)))
150.         End If

```

```
151.         TestImageNumber = GenerateRandomNumber() * 1000
152.         SetCurrentInputValues(Character, TestImageNumber)
153.         SetCurrentTargetValues(Character)
154.         Outputs = CharacterNeuralNetwork.Feedforward(Inputs)
155.         If ShowOutput Then
156.             For NumIter As Integer = 0 To 9
157.                 Console.WriteLine(NumIter & ": " & Outputs(NumIter))
158.             Next
159.             For CharIter As Integer = 10 To 35
160.                 Console.WriteLine(Chr(55 + CharIter) & ": " & Outputs(CharIter))
161.             Next
162.             Console.ReadLine()
163.         End If
164.         MaxValue = Outputs.Max()
165.         For NumGuessIter As Integer = 0 To 9
166.             If Outputs(NumGuessIter) = MaxValue Then
167.                 CharacterGuess = Chr(NumGuessIter + 48)
168.             End If
169.         Next
170.         For CharGuessIter As Integer = 10 To 35
171.             If Outputs(CharGuessIter) = MaxValue Then
172.                 CharacterGuess = Chr(CharGuessIter + 55)
173.             End If
174.         Next
175.         If Character = CharacterGuess Then
176.             CorrectTests += 1
177.         End If
178.     Next
179.     PercentageCorrect = (CorrectTests / NumberOfTests) * 100
180.     Return Str(PercentageCorrect)
181. End Function

182.
183. Public Function FeedForwardFromDataset(ByRef Character As Char, ByRef ImageNumber As Integer) As Char
184.     Dim MaxValue As Single
185.     Dim CharacterCode As Integer
186.     SetCurrentInputValues(Character, ImageNumber)
187.     SetCurrentTargetValues(Character)
188.     Outputs = CharacterNeuralNetwork.Feedforward(Inputs)
189.     MaxValue = Outputs.Max()
190.     For NumIter As Integer = 0 To 9
191.         If Outputs(NumIter) = MaxValue Then
```

```
192.         CharacterCode = NumIter + 48
193.     End If
194. Next
195. For CharIter As Integer = 10 To 35
196.     If Outputs(CharIter) = MaxValue Then
197.         CharacterCode = CharIter + 55
198.     End If
199. Next
200. Return Chr(CharacterCode)
201. End Function
202.
203. Public Function FeedForwardExtractedCharacter(ByVal ImageOfChar As Bitmap)
204.     Dim CurrentColor As Color
205.     Dim ActivationValue As Single
206.     Dim MaxValue As Single
207.     Dim CharacterCode As Integer
208.     For xiter As Integer = 0 To ImageOfChar.Width - 1
209.         For yiter As Integer = 0 To ImageOfChar.Height - 1
210.             CurrentColor = ImageOfChar.GetPixel(xiter, yiter)
211.             ActivationValue = CurrentColor.B
212.             ActivationValue /= 255
213.             ActivationValue = 1.0 - ActivationValue
214.             Inputs((30 * xiter) + yiter) = ActivationValue
215.         Next
216.     Next
217.     Outputs = CharacterNeuralNetwork.Feedforward(Inputs)
218.     MaxValue = Outputs.Max()
219.     For NumIter As Integer = 0 To 9
220.         If Outputs(NumIter) = MaxValue Then
221.             CharacterCode = NumIter + 48
222.         End If
223.     Next
224.     For CharIter As Integer = 10 To 35
225.         If Outputs(CharIter) = MaxValue Then
226.             CharacterCode = CharIter + 55
227.         End If
228.     Next
229.     Return Chr(CharacterCode)
230. End Function
231.
232. End Class
```

CHARACTER RECOGNITION TESTING INTERFACE

```
1. Public Class CharacterRecognitionSystemTestingInterface
2.
3.     'Creates an interface to test the character recognition system.
4.     Public Property CharacterRecognitionSystem As New CharacterRecognitionSystem(900, 600, 200, 36)
5.     Public Property CurrentCharacter As Char
6.     Public Property CurrentImageNumber As Integer
7.
8.     Private Sub CharacterRecognitionSystemTestingInterface_Load(sender As Object, e As EventArgs) Handles MyBase.Load
9.         CharacterRecognitionSystem.LoadNetwork("Neural Network")
10.    End Sub
11.
12.    Private Sub NewImageButton_Click(sender As Object, e As EventArgs) Handles NewImageButton.Click
13.        Dim FileName As String
14.        FileName = GenerateRandomImagePath()
15.        Dim Image As Image = Image.FromFile(FileName)
16.        CharacterImage.Image = Image
17.    End Sub
18.
19.    Private Function GenerateRandomImagePath()
20.        'Images are just stored in folders grouping the characters they represent and then each with a specific index.
21.        'E.G. .../A/0.bmp
22.        Dim CharacterCode As Integer
23.        Dim FileName
24.        FileName = "C:/Users/matth/OneDrive/Documents/College/Year 2/Computing/NEA/Processed Dataset/"
25.        CharacterCode = Math.Floor(GenerateRandomNumber() * 36)
26.        If CharacterCode <= 9 Then
27.            CharacterCode += 48
28.        ElseIf CharacterCode > 9 Then
29.            CharacterCode += 55
30.        End If
31.        CurrentImageNumber = GenerateRandomNumber() * 1000
32.        FileName += Chr(CharacterCode) & "/" & Chr(CharacterCode) & CurrentImageNumber & ".BMP"
33.        CurrentCharacter = Chr(CharacterCode)
34.        Return FileName
35.    End Function
```



```

36.
37.     Private Sub RunImageButton_Click(sender As Object, e As EventArgs) Handles RunImageButton.Click
38.         Dim CharacterGuess As Char
39.         Dim CharImage As New Bitmap(CharacterImage.Image)
40.         CharacterGuess = CharacterRecognitionSystem.FeedForwardExtractedCharacter(CharImage)
41.         OutputLabel.Text = CharacterGuess
42.     End Sub
43.
44.     Private Sub TestNetworkButton_Click(sender As Object, e As EventArgs) Handles TestNetworkButton.Click
45.         Dim PercentageCorrect As String
46.         PercentageCorrect = CharacterRecognitionSystem.TestNetwork(1000, "7")
47.         OutputLabel.Text = PercentageCorrect
48.     End Sub
49.
50. End Class

```

IMAGE FORMATTING MODULES

```

1. Module ImageFormatting
2.
3.     'A collection of functions and procedures that can be applied to images extract a number plate from them.
4.
5.     Public Function GetImage(ByVal FileName As String, ByVal XSize As Integer, ByVal YSize As Integer) As Bitmap
6.         'This procedure is used to standardise any input jpeg image so the resolution and filetype are the same
7.         Dim OriginalImage As Image
8.         Dim ImageSize As New Size(XSize, YSize)
9.         OriginalImage = Image.FromFile(FileName)
10.        Using BitmapImage As New Bitmap(OriginalImage, ImageSize)
11.            Return BitmapImage
12.        End Using
13.    End Function
14.
15.    Public Function GetNumberPlate(ByVal BitmapImage As Bitmap, ByVal ResizedImageWidth As Integer, ByVal ResizedImageHeight As Integer) As Rectangle
16.        'This function returns a rectangle of coordinates that should contain the number plate within an image.
17.        'The yellow color on a rear number plate is: R > 180, G > 180, B < 10.
18.        'The function will search to find a yellow pixel, then recursively search surrounding pixels to identify the width and height

```

```

19.         'of a run of possible number plate colors (black or yellow). By evaluating the ratio of the with and height runs it
           can be determined if this is a number plate.
20.         Dim ResizedBitmapImage As Bitmap
21.         Dim Size As New Size(ResizedImageWidth, ResizedImageHeight)
22.         Dim PixelColor As Color
23.         Dim ImageSections(10) As Rectangle
24.         Dim CurrentImageSection As Rectangle
25.         Dim ImageSectionCount As Integer = 0
26.         Dim CoordinatesNotInAnyExistingImageSection As Boolean
27.         Dim ImageSectionRatio As Single
28.         Dim Rect1Area, Rect2Area As Integer
29.         Dim SectionReplaced As Boolean
30.         ResizedBitmapImage = New Bitmap(BitmapImage, Size) 'Scales down the image for efficiency.
31.         ResizedBitmapImage.Save("C:\Users\matth\OneDrive\Documents\College\Year 2\Computing\NEA\Example Images\Current Image (Compressed).bmp")
32.         For Xiter As Integer = 0 To Size.Width - 1
33.             For Yiter As Integer = 0 To Size.Height - 1
34.                 PixelColor = ResizedBitmapImage.GetPixel(Xiter, Yiter)
35.                 If ValidNumberPlatePixel(PixelColor, False) Then 'Search until it finds a valid pixel.
36.                     CoordinatesNotInAnyExistingImageSection = True
37.                     For Each Section In ImageSections 'Evaluates whether or not that pixel is already contained in another
                       plate (improves efficiency as not all pixels have to be checked once the plate is found).
38.                         If CoordinateInRectangle(Section, Xiter, Yiter) And SectionNotNull(Section) Then
39.                             CoordinatesNotInAnyExistingImageSection = False
40.                         End If
41.                     Next
42.                     If CoordinatesNotInAnyExistingImageSection Then
43.                         CurrentImageSection = CheckIfNumberPlate(ResizedBitmapImage, Xiter, Yiter)
44.                         'It was often the case that sub rectangles would form in previously identified rectangles.
45.                         'This prevents that from occurring and assumes that the larger of the two (by area) is the correct location of the plate.
46.                         SectionReplaced = False
47.                         For SectionIter As Integer = 0 To ImageSections.Length - 1
48.                             If RectanglesOverlap(ImageSections(SectionIter), CurrentImageSection) And SectionNotNull(ImageSections(SectionIter)) Then
49.                                 Rect1Area = ImageSections(SectionIter).Width * ImageSections(SectionIter).Height
50.                                 Rect2Area = CurrentImageSection.Width * CurrentImageSection.Height
51.                                 If Rect2Area > Rect1Area Then
52.                                     ImageSections(SectionIter) = CurrentImageSection
53.                                     SectionReplaced = True
54.                                 End If

```

```
55.             End If
56.         Next
57.             ImageSectionRatio = CurrentImageSection.Width / CurrentImageSection.Height
58.             If ImageSectionRatio > 3 And ImageSectionRatio < 6 And Not SectionReplaced And CurrentImageSection.
Width > 15 Then 'Assesses if the ratio of width to height fits that of a number plate.
59.                 ImageSections(ImageSectionCount) = CurrentImageSection
60.                 ImageSectionCount += 1
61.             End If
62.         End If
63.     End If
64. Next
65. Next
66. 'With this routine it is possible for multiple plates to be identified.
67. 'Only the largest rectangle is returned as this is likely to be the target of the ANPR system.
68. Dim LargestRectangle As New Rectangle(0, 0, 0, 0)
69. For Each Rect In ImageSections
70.     If Rect.Width > LargestRectangle.Width Then
71.         LargestRectangle = Rect
72.     End If
73. Next
74. Return ScaleRectangle((BitmapImage.Width / ResizedImageWidth), LargestRectangle)
75. End Function
76.
77. Public Function SectionNotNull(ByVal Rect As Rectangle) As Boolean
78.     'Evaluates whether or not a rectangle has zero width, thus null.
79.     If Rect.Width = 0 Or Rect.Height = 0 Then
80.         Return False
81.     Else
82.         Return True
83.     End If
84. End Function
85.
86. Private Function ScaleRectangle(ByVal Factor As Single, ByVal InputRectangle As Rectangle) As Rectangle
87.     'Used to scale the section of the resized image back up to the size of the original image.
88.     Dim ScaledRectangle As Rectangle
89.     Dim Left, Top, Width, Height As Integer
90.     Left = InputRectangle.Left * Factor
91.     Top = InputRectangle.Top * Factor
92.     Width = InputRectangle.Width * Factor
93.     Height = InputRectangle.Height * Factor
94.     ScaledRectangle = New Rectangle(Left, Top, Width, Height)
```

```

95.         Return ScaledRectangle
96.     End Function
97.
98.     Private Function CheckIfNumberPlate(ByVal Image As Bitmap, ByVal StartX As Integer, ByVal StartY As Integer) As Rectangle
99.         'Once a valid number plate pixel is found its coordinates are passed to this function.
100.        'It recursively checks the adjacent pixels from the starting ones and evaluates runs of valid pixels.
101.        Dim XRun, YRun As Integer
102.        Dim ImageSection As Rectangle
103.        XRun = 0
104.        YRun = 0
105.        CheckAdjacentPixel(Image, XRun, StartX, StartY, True)
106.        CheckAdjacentPixel(Image, YRun, StartX, StartY, False)
107.        ImageSection = New Rectangle(StartX, StartY, XRun, YRun)
108.        Return ImageSection
109.    End Function
110.
111.    Private Sub CheckAdjacentPixel(ByRef Image As Bitmap, ByRef Run As Integer, ByVal StartX As Integer, ByVal StartY As Integer, ByRef Horizontal As Boolean)
112.        'A recursive subroutine to check adjacent pixels from a starting point.
113.        Dim PixelColor As Color
114.        If Horizontal Then
115.            StartX += 1
116.        Else
117.            StartY += 1
118.        End If
119.        If StartX < Image.Width And StartY < Image.Height Then 'Ensures runs don't extend off the image.
120.            PixelColor = Image.GetPixel(StartX, StartY)
121.            If ValidNumberPlatePixel(PixelColor, True) Then
122.                CheckAdjacentPixel(Image, Run, StartX, StartY, Horizontal)
123.                Run += 1 'Increments when the stack collapses on the exit condition (invalid pixel).
124.            End If
125.        End If
126.    End Sub
127.
128.    Private Function ValidNumberPlatePixel(ByRef PixelColor As Color, ByRef CurrentlyInRun As Boolean) As Boolean
129.        'Evaluates a pixels RGB values and determines if it is a valid rear number plate color.
130.        Dim ValidPixel As Boolean
131.        ValidPixel = True
132.        If PixelColor.R() < 130 Or PixelColor.G() < 100 Or PixelColor.B() > 60 Then
133.            ValidPixel = False

```

```
134.         End If
135.         If PixelColor.R < 60 And PixelColor.G < 60 And PixelColor.B < 60 And CurrentlyInRun Then
136.             ValidPixel = True
137.         End If
138.         Return ValidPixel
139.     End Function
140.
141.     Private Function CoordinateInRectangle(ByRef Rectangle As Rectangle, ByRef XPosition As Integer, ByRef YPosition As
Integer) As Boolean
142.         Dim InRectangle As Boolean
143.         InRectangle = True
144.         If XPosition < Rectangle.Left Or XPosition > (Rectangle.Left + Rectangle.Width) Then
145.             InRectangle = False
146.         ElseIf YPosition < Rectangle.Top Or YPosition > (Rectangle.Top + Rectangle.Height) Then
147.             InRectangle = False
148.         End If
149.         Return InRectangle
150.     End Function
151.
152.     Public Sub GetColor(BitmapImage As Bitmap, ByRef XPosition As Integer, ByRef YPosition As Integer)
153.         'Useful when adjusting the RGB conditions for a valid plate pixel.
154.         Dim Color As Color
155.         Color = BitmapImage.GetPixel(XPosition, YPosition)
156.         Console.WriteLine("R: " & Color.R)
157.         Console.WriteLine("B: " & Color.B)
158.         Console.WriteLine("G: " & Color.G)
159.     End Sub
160.
161.     Private Function RectanglesOverlap(ByVal Rect1 As Rectangle, ByVal Rect2 As Rectangle) As Boolean
162.         'This function is used to prevent two sub rectangles forming within a numberplate.
163.         If (Rect1.Left < Rect2.Right And Rect1.Right > Rect2.Left And Rect1.Top > Rect2.Bottom And Rect1.Bottom < Rect2.
Top) Then
164.             Return True
165.         Else
166.             Return False
167.         End If
168.     End Function
169.
170.     Public Function ExtractNumberPlate(ByVal OriginalBitmap As Bitmap) As Bitmap
171.         'This routine extracts a sub image from the original image.
```

```

172.         'It extracts a region defined by a Rectangle of coordinates determined to contain the number plate. Source --
> Destination.
173.         Dim ImgSection As Rectangle
174.         ImgSection = GetNumberPlate(OriginalBitmap, 240, 160)
175.         Dim NumberPlateBitmapImage As Bitmap
176.         NumberPlateBitmapImage = New Bitmap(ImgSection.Width, ImgSection.Height)
177.         'Copies a section of the original image to a bitmap.
178.         Using Graph As Graphics = Graphics.FromImage(NumberPlateBitmapImage)
179.             Dim srcRectangle As New Rectangle(ImgSection.Left, ImgSection.Top, ImgSection.Width, ImgSection.Height)
180.             Dim dstRectangle As New Rectangle(0, 0, ImgSection.Width, ImgSection.Height)
181.             Graph.DrawImage(OriginalBitmap, dstRectangle, srcRectangle, GraphicsUnit.Pixel)
182.         End Using
183.         NumberPlateBitmapImage.Save("C:\Users\matth\OneDrive\Documents\College\Year 2\Computing\NEA\Example Images\Extra
cted Number Plate.bmp")
184.         Return NumberPlateBitmapImage
185.     End Function
186.
187. End Module

```

CHARACTER FORMATTING MODULES

```

1. Module CharacterFormatting
2.
3.     'A collection of functions and methods to extract the individual characters from a number plate image.
4.     'These procedures will also format the images of the characters after they've been extracted so they can be passed into
the neural network.
5.
6.     Public Function GetCharacter(ByVal Image As Bitmap, ByVal StartX As Integer) As Rectangle
7.         'Extracts a character from an image of a plate starting from STARTX
8.         Dim Width As Integer
9.         Width = 0
10.        Do
11.            WidthOfCharacter(Image, StartX, Width)
12.            StartX += 1
13.        Loop Until Width <> 0 Or StartX >= Image.Width
14.        Return New Rectangle(StartX - 2, 3, Width + 4, Image.Height - 3)
15.    End Function
16.

```

```
17. Private Function BlackPixelInColumn(ByVal Image As Bitmap, ByVal Column As Integer)
18.     'Checks to see if a column contains a black pixel.
19.     'If true it is assumed the character is in continuation.
20.     Dim Height, StartY, EndY As Integer
21.     Dim BlackPixelFound As Boolean
22.     Dim CurrentPixel As Color
23.     Height = Image.Height
24.     StartY = Math.Floor(Height / 5) 'Only checks over the middle 4/5th of the image as often black pixels were found at
the image boundary (from cropping errors).
25.     EndY = StartY * 4
26.     BlackPixelFound = False
27.     If Column < Image.Width Then
28.         For PixelIter As Integer = StartY To EndY
29.             CurrentPixel = Image.GetPixel(Column, PixelIter)
30.             If BlackPixel(CurrentPixel) Then
31.                 BlackPixelFound = True
32.             End If
33.         Next
34.         Return BlackPixelFound
35.     Else
36.         Return BlackPixelFound
37.     End If
38.
39. End Function
40.
41. Private Function BlackPixelInRow(ByVal Image As Bitmap, ByVal Row As Integer)
42.     Dim Width As Integer
43.     Dim BlackPixelFound As Boolean
44.     Dim CurrentPixel As Color
45.     Width = Image.Width
46.     BlackPixelFound = False
47.     For PixelIter As Integer = 0 To Width - 1
48.         CurrentPixel = Image.GetPixel(PixelIter, Row)
49.         If BlackPixel(CurrentPixel) Then
50.             BlackPixelFound = True
51.         End If
52.     Next
53.     Return BlackPixelFound
54. End Function
55.
56. Public Function BlackPixel(ByVal Color As Color) As Boolean
```

```

57.         If Color.R < 80 And Color.G < 80 And Color.B < 80 Then
58.             Return True
59.         Else
60.             Return False
61.         End If
62.     End Function
63.
64.     Private Sub WidthOfCharacter(ByVal Image As Bitmap, ByVal StartX As Integer, ByRef Width As Integer)
65.         'Recursive routine which checks if theres a black pixel (therefore a character is still in continuation) then recur
        sively checks the next one until the exit condition is met.
66.         If BlackPixelInColumn(Image, StartX) Then
67.             WidthOfCharacter(Image, StartX + 1, Width)
68.             Width += 1 'Incremented on the collapse of the stack.
69.         End If
70.     End Sub
71.
72.     Public Function ExtractCharacters(ByVal NumberPlateBitmapImage As Bitmap) As List(Of Bitmap)
73.         'Returns a list of bitmap images containing each of the characters in the input image.
74.         Dim Xiter, CharImageCount As Integer
75.         Dim CharacterImageRectList As New List(Of Rectangle)
76.         Dim CharacterImageList As New List(Of Bitmap)
77.         Xiter = 0
78.         CharImageCount = -1
79.         Do
80.             CharImageCount += 1
81.             CharacterImageRectList.Add(GetCharacter(NumberPlateBitmapImage, Xiter))
82.             Xiter = CharacterImageRectList(CharImageCount).Left + CharacterImageRectList(CharImageCount).Width 'Increments
        start so next charcter is found from where the last one ends.
83.         Loop Until Xiter > NumberPlateBitmapImage.Width Or CharacterImageRectList(CharImageCount).Width = 0 'Loops until th
        e end of the image or no character is found.
84.
85.         'Copies each of the character rectangles to bitmap images and forms the list.
86.         For Each srcRect In CharacterImageRectList
87.             If SectionNotNull(srcRect) And srcRect.Width > 6 Then
88.                 Dim CurrentBitmap As New Bitmap(srcRect.Width, srcRect.Height)
89.                 Using Graph As Graphics = Graphics.FromImage(CurrentBitmap)
90.                     Dim dstRect As New Rectangle(0, 0, srcRect.Width, srcRect.Height)
91.                     Graph.DrawImage(NumberPlateBitmapImage, dstRect, srcRect, GraphicsUnit.Pixel)
92.                 End Using
93.                 CharacterImageList.Add(CurrentBitmap)
94.             End If

```



```

95.     Next
96.     CharacterImageList(0).Save("C:\Users\matth\OneDrive\Documents\College\Year 2\Computing\NEA\Example Images\Extracted
Character (Pre-format).bmp")
97.     FormatCharactersForNetwork(CharacterImageList)
98.     Return CharacterImageList
99. End Function
100.
101. Public Function FormatCharactersForNetwork(ByVal CharacterImages As List(Of Bitmap)) As List(Of Bitmap)
102.     'Formats character images so they can be passed into the network.
103.     Dim Size As New Size(30, 30) 'Standard size for input image into network.
104.     Dim FormattedImage As Bitmap
105.     Dim CurrentColor As Color
106.     Dim StandardBlack As Color = Color.FromArgb(0, 0, 0)
107.     Dim StandardWhite As Color = Color.FromArgb(255, 255, 255)
108.     Dim FormattedImages As New List(Of Bitmap)
109.     Dim CharacterBounds As Rectangle
110.     For ImageIter As Integer = 0 To CharacterImages.Count - 1
111.         For Xiter As Integer = 0 To CharacterImages(ImageIter).Width - 1
112.             For Yiter As Integer = 0 To CharacterImages(ImageIter).Height - 1
113.                 CurrentColor = CharacterImages(ImageIter).GetPixel(Xiter, Yiter)
114.                 If CurrentColor.R < 70 And CurrentColor.G < 70 And CurrentColor.B < 70 Then 'Sets all dark pixels to
black and light to white. This removes a lot of noise from the image.
115.                     CharacterImages(ImageIter).SetPixel(Xiter, Yiter, StandardBlack)
116.                 Else
117.                     CharacterImages(ImageIter).SetPixel(Xiter, Yiter, StandardWhite)
118.                 End If
119.             Next
120.         Next
121.         FormattedImage = ResizeCharacterImage(CharacterImages(ImageIter))
122.         CharacterBounds = GetBoundsOfCharacter(FormattedImage)
123.         Dim ShiftedBitmap As New Bitmap(30, 30)
124.         For Xiter As Integer = 0 To 29
125.             For Yiter As Integer = 0 To 29
126.                 ShiftedBitmap.SetPixel(Xiter, Yiter, Color.White)
127.             Next
128.         Next
129.         Using Graph As Graphics = Graphics.FromImage(ShiftedBitmap)
130.             Dim dstRect As New Rectangle(6, 3, 16, 23)
131.             Graph.DrawImage(FormattedImage, dstRect, CharacterBounds, GraphicsUnit.Pixel)
132.         End Using
133.         FormattedImages.Add(ShiftedBitmap)

```

```
134.         Next
135.         FormattedImages(0).Save("C:\Users\matth\OneDrive\Documents\College\Year 2\Computing\NEA\Example Images\Extracted
Character (Post-format).bmp")
136.         Return FormattedImages
137.     End Function
138.
139.     Private Function ResizeCharacterImage(ByVal CharacterImage As Bitmap) As Bitmap
140.         'This routine is used to standardize the size of the image to 30x30 for the network.
141.         'The resize method is not used as it morphs the character and makes it less similar to a standard character.
142.         Dim VerticalResize As New Size(CharacterImage.Width, 30) 'Creates a new image of correct height. This can be don
e as vertical morphing is not too severe.
143.         Dim VerticallyResizedCharacter As New Bitmap(CharacterImage, VerticalResize)
144.         Dim CurrentPixel As Color
145.         Dim Buffer As Integer
146.         'This routine takes a cut of the character from the original image as pastes it on a blank 30x30 image.
147.         If CharacterImage.Width < 30 Then 'Only works if character is less than 30 in width (else it wont fit in 30x30).

148.             Dim FullyResizedCharacter As New Bitmap(30, 30)
149.             Buffer = Math.Ceiling((30 - CharacterImage.Width) / 2)
150.             For Xiter As Integer = Buffer To (29 - Buffer)
151.                 For Yiter As Integer = 0 To 29
152.                     CurrentPixel = VerticallyResizedCharacter.GetPixel(Xiter - Buffer, Yiter)
153.                     FullyResizedCharacter.SetPixel(Xiter, Yiter, CurrentPixel)
154.                 Next
155.             Next
156.             For Xiter As Integer = 0 To Buffer - 1
157.                 For Yiter As Integer = 0 To 29
158.                     FullyResizedCharacter.SetPixel(Xiter, Yiter, Color.White)
159.                 Next
160.             Next
161.             For Xiter As Integer = 30 - Buffer To 29
162.                 For Yiter As Integer = 0 To 29
163.                     FullyResizedCharacter.SetPixel(Xiter, Yiter, Color.White)
164.                 Next
165.             Next
166.             Return FullyResizedCharacter
167.         Else 'It can be assumed morphing wont be too bad as character is already large.
168.             Dim FullyResizedCharacter As New Bitmap(CharacterImage, 30, 30)
169.             Return FullyResizedCharacter
170.         End If
171.     End Function
```

```
172.  
173.     Private Function GetBoundsOfCharacter(ByVal CharacterImage As Bitmap) As Rectangle  
174.         'Gets the start and end positions of the character within the character image.  
175.         Dim XMin, YMin, XMax, YMax As Integer  
176.         XMin = CharacterImage.Width  
177.         YMin = CharacterImage.Height  
178.         XMax = 0  
179.         YMax = 0  
180.         For Xiter As Integer = 0 To CharacterImage.Width - 1  
181.             If BlackPixelInColumn(CharacterImage, Xiter) Then  
182.                 If Xiter < XMin Then  
183.                     XMin = Xiter  
184.                 End If  
185.                 If Xiter > XMax Then  
186.                     XMax = Xiter  
187.                 End If  
188.             End If  
189.         Next  
190.         For Yiter As Integer = 0 To CharacterImage.Height - 1  
191.             If BlackPixelInRow(CharacterImage, Yiter) Then  
192.                 If Yiter < YMin Then  
193.                     YMin = Yiter  
194.                 End If  
195.                 If Yiter > YMax Then  
196.                     YMax = Yiter  
197.                 End If  
198.             End If  
199.         Next  
200.         Return New Rectangle(XMin, YMin, XMax - XMin, YMax - YMin)  
201.     End Function  
202.  
203. End Module
```

IMAGE FORMATTING TESTING INTERFACE

```
1. Public Class ImageFormattingTestingInterface  
2.  
3.     'Used to test the image formatting modules.
```

```
4.     Public Property FileName As String
5.     Public Property ImageCount As Integer
6.     Public Property NumberPlateBitmapImage As Bitmap
7.     Public Property CurrentCharacterCount As Integer
8.     Public Property CharacterImageList As New List(Of Bitmap)
9.
10.    Public Sub New(ByRef ImagesLocation As String)
11.
12.        ' This call is required by the designer.
13.        InitializeComponent()
14.
15.        ' Add any initialization after the InitializeComponent() call.
16.        FileName = ImagesLocation
17.        ImageCount = 0
18.        CurrentCharacterCount = 0
19.    End Sub
20.
21.    Private Sub NewImageButton_Click(sender As Object, e As EventArgs) Handles NewImageButton.Click
22.        'Loads a new image into the OriginalImage PictureBox.
23.        'Images in the selected folder should be held under integer names incremented from 0.
24.        Dim CurrentImage As String
25.        Dim BitmapImage As Bitmap
26.        Dim ImageSize As New Size(1200, 800)
27.        CurrentImage = FileName & "/" & ImageCount & ".jpg"
28.        BitmapImage = New Bitmap(Image.FromFile(CurrentImage), ImageSize)
29.        OriginalImage.Image = BitmapImage
30.        ImageCount += 1
31.    End Sub
32.
33.    Private Sub ExtractNumberPlateButton_Click(sender As Object, e As EventArgs) Handles ExtractNumberPlateButton.Click
34.        Try
35.            Dim OriginalBitmap As New Bitmap(OriginalImage.Image)
36.            NumberPlateBitmapImage = ExtractNumberPlate(OriginalBitmap)
37.            NumberPlateImage.Image = NumberPlateBitmapImage
38.        Catch ex As Exception
39.            MsgBox("Not Found")
40.        End Try
41.    End Sub
42.
43.    Private Sub ExtractCharactersButton_Click(sender As Object, e As EventArgs) Handles ExtractCharactersButton.Click
44.        CharacterImageList = ExtractCharacters(NumberPlateBitmapImage)
```

```

45.     CurrentCharacterCount = 0
46.     CharImage.Image = CharacterImageList(0)
47.     End Sub
48.
49.     Private Sub NextCharButton_Click(sender As Object, e As EventArgs) Handles NextCharButton.Click
50.         CurrentCharacterCount += 1
51.         If CurrentCharacterCount = CharacterImageList.Count Then
52.             CurrentCharacterCount = 0
53.         End If
54.         CharImage.Image = CharacterImageList(CurrentCharacterCount)
55.     End Sub
56.
57. End Class

```

DATABASE MODULES

```

1. Module DataBase_Modules
2.
3.     Public Sub OpenDataBase(ByVal DatabaseName As String, ByRef Connection As OleDbConnection)
4.         'Opens a database connection under the database name.
5.         Try
6.             Connection = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & DatabaseName & ".mdb")
7.             Connection.Open()
8.         Catch ex As Exception
9.             MsgBox("Database not opened: " & ex.Message)
10.        End Try
11.    End Sub
12.
13.    Public Sub CloseDatabase(ByVal Connection As OleDbConnection)
14.        Connection.Close()
15.    End Sub
16.
17.    Public Sub AddRecordToCarPark(ByVal Registration As String, ByVal Connection As OleDbConnection, ByVal Command As OleDb
Command)
18.        Dim RecordDate As New Date
19.        Dim Values(2) As String
20.        Dim Fields(2) As String
21.        Dim SQLString As String

```

```

22.     RecordDate = Date.Now
23.     CheckRegistrationAgainstDatabase(Registration, Connection, Command)
24.     Values(0) = Registration
25.     Values(1) = RecordDate.ToShortDateString
26.     Values(2) = RecordDate.ToLongTimeString
27.     Fields = {"Registration", "EntryDate", "EntryTime"}
28.     SQLString = ConstructSQLStrForDataSubmit("CarPark", Fields, Values)
29.     If Not CheckIfCarInCarPark(Registration, Connection, Command) Then
30.         ExecuteDataSubmit(SQLString, Connection, Command)
31.     Else
32.         MsgBox("Car currently in Car Park (Not added to database).")
33.     End If
34. End Sub
35.
36. Private Function CheckIfCarInCarPark(ByVal Registration As String, ByVal Connection As OleDbConnection, ByVal Command As OleDbCommand) As Boolean
37.     'This function is used to ensure that a car is not added multiple times on one entry (e.g. if it stopped by the camera).
38.     Dim SQLStr As String
39.     Dim Reader As OleDbDataReader
40.     Dim DataTable As New DataTable
41.     Dim Rows() As DataRow
42.     Dim CarInCarPark As Boolean
43.     SQLStr = "SELECT * FROM CarPark WHERE CarPark.Registration = " & "" & Registration & "" 'SQL Query
44.     Command = New OleDbCommand(SQLStr, Connection) 'Creates Command
45.     Reader = Command.ExecuteReader() 'Executes Command
46.     DataTable.Load(Reader) 'Loads Data to Datatable
47.     Rows = DataTable.Select() 'Selects all rows from DataTable to an array of DataRows
48.     CarInCarPark = False
49.     For Each Row In Rows
50.         If IsDBNull(Row("ExitTime")) Then 'Checks if the value in ExitTime is Null (Therefore the car is in the car park).
51.             CarInCarPark = True
52.         End If
53.     Next
54.     Return CarInCarPark
55. End Function
56.
57. Public Sub AddExitTimeToCarParkRecord(ByVal Registration As String, ByVal Connection As OleDbConnection, ByVal Command As OleDbCommand)
58.     Dim SQLStr As String

```

```

59.     Dim RecordDate As New Date
60.     RecordDate = Date.Now
61.     CheckRegistrationAgainstDatabase(Registration, Connection, Command)
62.     SQLStr = "UPDATE CarPark SET ExitTime = " & "'" & RecordDate.ToLongTimeString & "'" & " WHERE CarPark.Registration
= " & "'" & Registration & "'"
63.     SQLStr += " AND CarPark.ExitTime IS NULL"
64.     ExecuteDataSubmit(SQLStr, Connection, Command)
65. End Sub
66.
67. Private Sub CheckRegistrationAgainstDatabase(ByRef Registration As String, ByVal Connection As OleDbConnection, ByVal C
ommand As OleDbCommand)
68.     Dim SQLStr = "SELECT Registration FROM Vehicles;" 'Query used to extract all registrations.
69.     Command = New OleDbCommand(SQLStr, Connection)
70.     Dim Reader As OleDbDataReader = Command.ExecuteReader 'Runs query and stored to reader.
71.     Reader.Read() 'Reader has to be read once before data can be accessed.
72.     Dim DatabaseRegistration As String
73.     Dim EndOfStream As Boolean = False
74.     Dim NumberOfErrors As Integer
75.     Dim TextRead As TextReader
76.     Do
77.         Try
78.             TextRead = Reader.GetTextReader(0)
79.             DatabaseRegistration = TextRead.ReadLine 'Extracts first number plate from database.
80.             NumberOfErrors = NumberOfErrorsBetweenRegistrations(Registration, DatabaseRegistration) 'Compares extracted
plate with registration.
81.             If NumberOfErrors < 3 Then 'It is assumed that if there are less than 3 errors they are the same number pla
te and an error has occurred along the way.
82.                 Registration = DatabaseRegistration
83.             End If
84.             Reader.Read() 'Moves to next record.
85.         Catch ex As Exception
86.             EndOfStream = True
87.         End Try
88.     Loop Until EndOfStream = True 'Loops through all records.
89. End Sub
90.
91. Private Function NumberOfErrorsBetweenRegistrations(ByVal Reg1 As String, ByVal Reg2 As String) As Integer
92.     'Returns the number of differences between the records plate and the extracted number plate.
93.     Dim Count As Integer
94.     Count = 0
95.     For CharIter As Integer = 0 To Reg1.Length - 1

```

```

96.         If Reg1(CharIter) <> Reg2(CharIter) Then
97.             Count += 1
98.         End If
99.     Next
100.    Return Count
101. End Function
102.
103. Public Sub AddRecordToStudents(ByVal StudentID As Integer, ByVal Registration As String, ByVal FirstName As String,
ByVal LastName As String, ByVal Age As Integer, ByVal Connection As OleDbConnection, ByVal Command As OleDbCommand)
104.     Dim Fields() As String = {"StudentID", "Registration", "FirstName", "LastName", "Age"}
105.     Dim Values() As String = {StudentID, Registration, FirstName, LastName, Age}
106.     Dim SQLSTR As String
107.     SQLSTR = ConstructSQLStrForDataSubmit("Students", Fields, Values)
108.     ExecuteDataSubmit(SQLSTR, Connection, Command)
109. End Sub
110.
111. Public Sub AddRecordToVehicles(ByVal Registration As String, ByVal Make As String, ByVal Model As String, ByVal Color
As String, ByVal DateRegistered As Date, ByVal Connection As OleDbConnection, ByVal Command As OleDbCommand)
112.     Dim Fields() As String = {"Registration", "Make", "Model", "Color", "DateRegistered"}
113.     Dim Values() As String = {Registration, Make, Model, Color, DateRegistered}
114.     Dim SQLSTR As String
115.     SQLSTR = ConstructSQLStrForDataSubmit("Vehicles", Fields, Values)
116.     ExecuteDataSubmit(SQLSTR, Connection, Command)
117. End Sub
118.
119. Public Function ConstructSQLStrForDataSubmit(Table As String, Fields As Array, Values As Array)
120.     Dim CurrentString As String
121.     'Puts ' around all items in values (SQL syntax)
122.     For iter As Integer = 0 To Values.Length - 1
123.         CurrentString = ""
124.         CurrentString += Values(iter)
125.         CurrentString += ""
126.         Values(iter) = CurrentString
127.     Next
128.     'Constructs the SQL string
129.     Dim SQLStr As String
130.     SQLStr = "INSERT INTO " & Table & " ("
131.     For iter As Integer = 0 To Fields.Length - 1
132.         If iter <> Fields.Length - 1 Then
133.             SQLStr += Fields(iter) & ", "
134.         Else

```



```
135.         SQLStr += Fields(iter)
136.     End If
137. Next
138. SQLStr += ") VALUES ("
139. For iter As Integer = 0 To Values.Length - 1
140.     If iter <> Values.Length - 1 Then
141.         SQLStr += Values(iter) & ","
142.     Else
143.         SQLStr += Values(iter)
144.     End If
145. Next
146. SQLStr += ")"
147. Return SQLStr
148. End Function
149.
150. Public Sub ExecuteDataSubmit(ByVal SQLSTR As String, ByVal Connection As OleDbConnection, ByVal Command As OleDbComm
and)
151.     'Used for no query executions (e.g. data submits)
152.     Try
153.         Command = New OleDbCommand(SQLSTR, Connection)
154.         Command.ExecuteNonQuery()
155.     Catch ex As Exception
156.         MsgBox("Data not inserted. Invalid Input.")
157.     End Try
158. End Sub
159.
160. Public Sub GetTablesForDropDown(ByVal DropDown As ComboBox, ByVal Connection As OleDbConnection)
161.     'Fills the combo box used to select the table with the identifiers for each of the tables.
162.     Dim Tables(0) As String
163.     Tables = GetAllTableNames(Connection)
164.     DropDown.Items.Clear()
165.     For Each item In Tables
166.         DropDown.Items.Add(item)
167.     Next
168.     DropDown.Items.Add("Invalid Registration") 'Adds an Invalid Vehicles option.
169.     DropDown.Items.Add("Select All") 'Adds a select all option.
170. End Sub
171.
172. Private Function GetAllTableNames(ByVal Connection As OleDbConnection)
173.     'Used to get all the table names in the database.
174.     Dim SchemaTable As DataTable = Connection.GetOleDbSchemaTable(OleDbSchemaGuid.Tables,
```

```
175.                                     New Object() {Nothing, Nothing, Nothing, "TABLE"})
176.     Dim Cols() As DataRow = SchemaTable.Select()
177.     Dim ArrayOfNames(Cols.Length - 1) As String
178.     For i As Integer = 0 To Cols.Length - 1
179.         ArrayOfNames(i) = Cols(i)("table_name")
180.     Next
181.     Return ArrayOfNames
182. End Function
183.
184. Private Sub CreateTableCommand(DDLStr As String, ByVal Command As OleDbCommand, ByVal Connection As OleDbConnection)
185.     Try
186.         Command = New OleDbCommand(DDLStr, Connection)
187.         Command.ExecuteNonQuery()
188.     Catch ex As Exception
189.         MsgBox("Table Not Created: " & ex.Message)
190.     End Try
191. End Sub
192.
193. Private Sub CreateFieldCommand(DDLStr As String, ByVal Command As OleDbCommand, ByVal Connection As OleDbConnection)
194.     Try
195.         Command = New OleDbCommand(DDLStr, Connection)
196.         Command.ExecuteNonQuery()
197.         MsgBox("Field Created")
198.     Catch ex As Exception
199.         MsgBox("Field Not Created: " & ex.Message)
200.     End Try
201. End Sub
202.
203. Public Sub ResetTable(ByVal Command As OleDbCommand, ByVal Connection As OleDbConnection, ByVal Table As String)
204.     'Clears a table in the database.
205.     Dim SQLSTR As String
206.     SQLSTR = "DELETE FROM " & Table
207.     Try
208.         Command = New OleDbCommand(SQLSTR, Connection)
209.         Command.ExecuteNonQuery()
210.     Catch ex As Exception
211.         MsgBox("Database Not Refreshed: " & ex.Message)
212.     End Try
```

```

213.         End Sub
214.
215.         Public Sub CreateDatabase(ByVal Command As OleDbCommand, ByVal Connection As OleDbConnection, ByVal DatabaseName As
String)
216.             'The DDL used to create all the tables and fields in the database.
217.             Dim Cata As Catalog = New Catalog()
218.             Try
219.                 Cata.Create("Provider=Microsoft.Jet.OLEDB.4.0;" &
220.                     "Data Source=" & DatabaseName & ".mdb;" &
221.                     "Jet OLEDB:Engine Type=5")
222.             Catch ex As Exception
223.                 MsgBox("Database Creation failed " & ex.Message)
224.             End Try
225.             OpenDataBase(DatabaseName, Connection)
226.             Dim DDLStrTable As String
227.             DDLStrTable = "CREATE TABLE CarPark(Registration Varchar(20), EntryDate Date, EntryTime Varchar(8), ExitTime Var
Char(8) PRIMARY KEY(Registration, EntryDate, EntryTime))"
228.             CreateTableCommand(DDLStrTable, Command, Connection)
229.             DDLStrTable = "CREATE TABLE Students(StudentID Int, Registration Varchar(20), FirstName VarChar(20), LastName Va
rChar(20), Age Int, PRIMARY KEY(StudentID, Registration))"
230.             CreateTableCommand(DDLStrTable, Command, Connection)
231.             DDLStrTable = "CREATE TABLE Vehicles(Registration Varchar(20) PRIMARY KEY, Make VarChar(20), Model Varchar(20),
Color Varchar(20), DateRegistered Date)"
232.             CreateTableCommand(DDLStrTable, Command, Connection)
233.             CloseDatabase(Connection)
234.         End Sub
235.
236.         Public Sub SelectAllDataFromTable(ByVal SelectedTable As String, ByVal Command As OleDbCommand, ByVal Connection As
OleDbConnection, ByRef DataGrid As DataGrid)
237.             Dim Table As New DataTable
238.             Dim SQLStr As String = "SELECT"
239.             Dim ArrayOfFields() As String
240.             Select Case SelectedTable 'Sets the array of fields to appropriate field names so can be used when constructing
the sql.
241.                 Case "CarPark"
242.                     ArrayOfFields = {"Registration", "EntryDate", "EntryTime", "ExitTime"}
243.                 Case "Students"
244.                     ArrayOfFields = {"StudentID", "Registration", "FirstName", "LastName", "Age"}
245.                 Case "Vehicles"
246.                     ArrayOfFields = {"Registration", "Make", "Model", "Color", "DateRegistered"}
247.             End Select

```

```

248.         'Constucts SQL
249.         Try
250.             For iter As Integer = 0 To ArrayOfFields.Length - 1
251.                 If iter <> ArrayOfFields.Length - 1 Then
252.                     SQLStr += "[" & ArrayOfFields(iter) & "],"
253.                 Else
254.                     SQLStr += "[" & ArrayOfFields(iter) & "]"
255.                 End If
256.             Next
257.             SQLStr += " FROM " & SelectedTable
258.             Command = New OleDbCommand(SQLStr, Connection)
259.             Table.Load(Command.ExecuteReader)
260.             Command.Dispose()
261.             DataGridView.DataSource = Table 'Deposits the selected data to a datatable to be used in the form.
262.         Catch ex As Exception
263.             MsgBox("Failed to select: " & ex.Message)
264.         End Try
265.     End Sub
266.
267.     Public Sub SelectInvalidVehicles(ByVal Command As OleDbCommand, ByVal Connection As OleDbConnection, ByRef DataGridView
    As DataGridView)
268.         'Uses an EXISTS database query to select number plates which exist in the car park and not in students.
269.         Dim SQLStr = "SELECT Registration, EntryDate, EntryTime, ExitTime FROM CarPark WHERE NOT EXISTS (SELECT Registra
    tion FROM Students WHERE CarPark.Registration = Students.Registration)"
270.         Dim Table As New DataTable
271.         Try
272.             Command = New OleDbCommand(SQLStr, Connection) 'Creates and executes the command
273.             Table.Load(Command.ExecuteReader)
274.             Command.Dispose()
275.             DataGridView.DataSource = Table 'Adds data to datagrid
276.         Catch ex As Exception
277.             MsgBox("Failed to select.")
278.         End Try
279.     End Sub
280.
281.     Public Sub SelectDataWithQuery(ByVal Command As OleDbCommand, ByVal Connection As OleDbConnection, ByVal DataGridView As
    DataGridView, ByVal RegCondition As String, ByVal StudentIDCondition As String, ByVal FirstNameCondition As String, ByVal Last
    NameCondition As String, ByVal ConditionCount As Integer)
282.         'Used to select data when a query has been made in the query boxes.
283.         Dim SQLSTR As String
284.         Dim Table As New DataTable

```

```
285.         'Constructs SQL
286.         SQLSTR = "SELECT CarPark.Registration, CarPark.EntryDate, CarPark.EntryTime, CarPark.ExitTime, Students.StudentID, Students.FirstName, Students.LastName, Students.Age, Vehicles.Make, Vehicles.Model, Vehicles.Color, Vehicles.DateRegistered "
287.         SQLSTR += "FROM ((CarPark INNER JOIN Students ON CarPark.Registration = Students.Registration) INNER JOIN Vehicles ON CarPark.Registration = Vehicles.Registration)"
288.         If ConditionCount <> 0 Then
289.             SQLSTR += "WHERE "
290.         End If
291.         If RegCondition <> Nothing Then
292.             SQLSTR += "CarPark.Registration = " & "" & RegCondition & ""
293.             If ConditionCount > 1 Then
294.                 SQLSTR += " AND "
295.                 ConditionCount -= 1
296.             End If
297.         End If
298.         If StudentIDCondition <> Nothing Then
299.             SQLSTR += "Students.StudentID = " & CStr(StudentIDCondition)
300.             If ConditionCount > 1 Then
301.                 SQLSTR += " AND "
302.                 ConditionCount -= 1
303.             End If
304.         End If
305.         If FirstNameCondition <> Nothing Then
306.             SQLSTR += "Students.FirstName = " & "" & FirstNameCondition & ""
307.             If ConditionCount > 1 Then
308.                 SQLSTR += " AND "
309.                 ConditionCount -= 1
310.             End If
311.         End If
312.         If LastNameCondition <> Nothing Then
313.             SQLSTR += "Students.LastName = " & "" & LastNameCondition & ""
314.             If ConditionCount > 1 Then
315.                 SQLSTR += " AND "
316.                 ConditionCount -= 1
317.             End If
318.         End If
319.         Try
320.             Command = New OleDbCommand(SQLSTR, Connection)
321.             Table.Load(Command.ExecuteReader)
322.             Command.Dispose()
```

```
323.         Datagrid.DataSource = Table 'Deposits selected data in datatable.
324.     Catch ex As Exception
325.         MsgBox("Failed to select: " & ex.Message)
326.     End Try
327. End Sub
328.
329. Private Function GetColumnNamesInTable(ByVal Connection As OleDbConnection, ByVal TableName As String) As Array
330.     'Used to get all the field names from a table in an array
331.     Dim Restrictions As String() = New String() {Nothing, Nothing, TableName, Nothing}
332.     Dim DataTable As DataTable = Connection.GetSchema("Columns", Restrictions)
333.     Dim Cols() As DataRow = DataTable.Select()
334.     Dim ArrayOfNames(Cols.Length) As String
335.     For i As Integer = 0 To Cols.Length - 1
336.         ArrayOfNames(i) = Cols(i)("column_name")
337.     Next
338.     Return ArrayOfNames
339. End Function
340.
341. Public Function GetNumberOfCarsInCarPark(ByVal Connection As OleDbConnection, ByVal Command As OleDbCommand) As Integer
342.     Dim SQLStr As String
343.     Dim Reader As OleDbDataReader
344.     Dim DataTable As New DataTable
345.     Dim DataRows() As DataRow
346.     SQLStr = "Select * FROM CarPark WHERE CarPark.ExitTime IS NULL" 'SQL Query for vehicles who have not left the car park.
347.     Command = New OleDbCommand(SQLStr, Connection)
348.     Reader = Command.ExecuteReader() 'Executes the command.
349.     DataTable.Load(Reader) 'Loads to a DataTable for analysis.
350.     DataRows = DataTable.Select()
351.     Return DataRows.Count() 'Number of selected items (cars in carpark).
352. End Function
353.
354. End Module
```

ANPR CLASS

```
1. Public Class ANPR
2.
3.     'Composed of the Character Recognition Class.
4.     'It combines the methods of this class with the functions and procedures of the Database Modules.
5.
6.     Public Property CharacterRecognitionSystem As CharacterRecognitionSystem
7.     Public Property DataSetLocaton As String
8.     Public Property NetworkName As String
9.
10.    Sub New(ByVal NetworkName As String, ByVal DataSetLocation As String)
11.        'Instantiates the Character Recognition System.
12.        CharacterRecognitionSystem = New CharacterRecognitionSystem(900, 600, 200, 36)
13.        CharacterRecognitionSystem.LoadNetwork(NetworkName)
14.        DataSetLocaton = DataSetLocation
15.    End Sub
16.
17.    Public Function RunImage(ByVal Image As Image)
18.        'Runs the sequence of formatting modules followed by passing the images into the network to produce an output.
19.        Dim OriginalBitmap As New Bitmap(Image)
20.        Dim NumberPlateBitmapImage As Bitmap
21.        Dim CharacterImageList As List(Of Bitmap)
22.        Dim FormattedCharacters As New List(Of Bitmap)
23.        Dim Output As String
24.        NumberPlateBitmapImage = ExtractNumberPlate(OriginalBitmap)
25.        CharacterImageList = ExtractCharacters(NumberPlateBitmapImage)
26.        FormattedCharacters = FormatCharactersForNetwork(CharacterImageList)
27.        Output = RunListOfCharacters(FormattedCharacters) 'Compressed to a Private Function in the class.
28.        Return Output
29.    End Function
30.
31.    Private Function RunListOfCharacters(ByVal Characters As List(Of Bitmap)) As String
32.        'Runs each of the character images in a list of images to produce an output string.
33.        Dim Output As String
34.        Dim Character As Char
35.        Output = ""
36.        For CharacterImageIter As Integer = 0 To Characters.Count - 1
```

```
37.         Character = CharacterRecognitionSystem.FeedForwardExtractedCharacter(Characters(CharacterImageIter))
38.         CorrectCharacter(Character, CharacterImageIter + 1)
39.         If CharacterImageIter = 4 Then 'Puts a space after the 4th character - standard for UK plates.
40.             Output += " " & Character
41.         Else
42.             Output += Character
43.         End If
44.     Next
45.     Return Output
46. End Function
47.
48. Private Sub CorrectCharacter(ByRef Character As Char, ByVal Position As Integer)
49.     'For standard UK plates only certain characters can be used in certain positions.
50.     '1 and 2 have to be Characters.
51.     '3 and 4 have to be Numbers.
52.     '5, 6, 7 are a combination of all.
53.     'Therefore certain characters can be swapped for their counterpart if predicted incorrectly by the network.
54.     If Position = 1 Or Position = 2 Then
55.         Select Case Asc(Character)
56.             Case 48
57.                 Character = "0" '0 looks like 0
58.             Case 49
59.                 Character = "I" 'I looks like 1
60.             Case 50
61.                 Character = "Z" 'Z looks like 2
62.             Case 53
63.                 Character = "S" 'S looks like 5
64.             Case 56
65.                 Character = "B" 'B looks like 8
66.             Case 82
67.                 Character = "N"
68.         End Select
69.     End If
70.
71.     If Position = 3 Or Position = 4 Then
72.         Select Case Asc(Character)
73.             Case 66
74.                 Character = "8" '8 looks like B
75.             Case 81
76.                 Character = "0" '0 looks like 0
77.             Case 87
```



```

78.         Character = "1" '1 looks like I
79.     End Select
80. End If
81.
82.     If Position = 5 Or Position = 6 Or Position = 7 Then
83.         Select Case Asc(Character)
84.             Case 48
85.                 Character = "0" '0 looks like 0
86.             Case 49
87.                 Character = "I" 'I looks like 1
88.             Case 50
89.                 Character = "Z" 'Z looks like 2
90.             Case 53
91.                 Character = "S" 'S looks like 5
92.             Case 56
93.                 Character = "B" 'B looks like 8
94.         End Select
95.     End If
96. End Sub
97.
98. End Class

```

ANPR SYSTEM INTERFACE

```

1. Public Class ANPR_System
2.
3.     'The final interface of the system which instantiates all of the required classes.
4.     Public Property ANPRSystem As ANPR
5.     Public Property ImageNumber As Integer
6.     Public Property DatasetLocations As New List(Of String) 'Holds the string locations of the folders which contain images
7.     Private Property CurrentDatasetLocation As Integer 'Holds the index in the list of the current dataset location.
8.     Public Property DatabaseName As String
9.     Public Property Connection As OleDbConnection
10.    Public Property Command As OleDbCommand
11.    Public Property DataGrid As New DataGrid
12.
13.    Private Sub ANPR_Testing_Interface_Load(sender As Object, e As EventArgs) Handles MyBase.Load

```

```
14.     DatasetLocations.Add("C:\Users\matth\OneDrive\Documents\College\Year 2\Computing\NEA\Example Images\Generated Number Plates\") 'Adds the two current dataset locations to the list.
15.     DatasetLocations.Add("C:\Users\matth\OneDrive\Documents\College\Year 2\Computing\NEA\Example Images\Real World Number Plates\") 'More can be added if required.
16.     DatasetLocations.Add("C:\Users\matth\OneDrive\Documents\College\Year 2\Computing\NEA\Example Images\Other Testing Images\")
17.     CurrentDatasetLocation = 0 'Starts on the first dataset.
18.     ANPRSystem = New ANPR("Neural Network", DatasetLocations(CurrentDatasetLocation)) 'Instantiates ANPR.
19.     ImageNumber = 0
20.     DatabaseName = "CarParkDatabase"
21.     'CreateDatabase(Command, Connection, DatabaseName) - FOR USE ONLY WHEN DATABASE IS BEING CREATED
22.     OpenDataBase(DatabaseName, Connection)
23.     ResetTable(Command, Connection, "CarPark") 'Refreshes the carpark table.
24.     LoadDataGrid() 'Creates the datagrid to display selected data from the database.
25.     GetTablesForDropDown(SelectTableBox, Connection) 'Adds each of the table names to the select table combo box.
26. End Sub
27.
28. Private Sub NewImageButton_Click(sender As Object, e As EventArgs) Handles NewImage.Click
29.     'Images are stored as indexed file names from 0.
30.     Dim CurrentImageFilePath As String
31.     CurrentImageFilePath = DatasetLocations(CurrentDatasetLocation) & ImageNumber & ".jpg"
32.     Dim OriginalBitmapImage As Bitmap
33.     Dim ImageSize As New Size(1200, 800)
34.     'Error is caught when the index is too great and the file does not exist. So its reset.
35.     Try
36.         OriginalBitmapImage = New Bitmap(Image.FromFile(CurrentImageFilePath), ImageSize)
37.         OriginalImage.Image = OriginalBitmapImage
38.     Catch ex As Exception
39.         MsgBox("No more images.")
40.         ImageNumber = -1
41.     End Try
42.     ImageNumber += 1
43. End Sub
44.
45. Private Sub CloseDatabaseButton_Click(sender As Object, e As EventArgs) Handles CloseDatabaseButton.Click
46.     CloseDatabase(Connection)
47. End Sub
48.
49. Private Sub RunQuery_Click(sender As Object, e As EventArgs) Handles RunQuery.Click
50.     Dim ConditionCount As Integer
51.     If SelectTableBox.Text <> "Select All" Then
```

```
52.         If SelectTableBox.Text = "Invalid Registration" Then
53.             SelectInvalidVehicles(Command, Connection, DataGrid)
54.         Else
55.             SelectAllDataFromTable(SelectTableBox.Text, Command, Connection, DataGrid)
56.         End If
57.         'Queries cannot be made on any other tables except the Select All.
58.         RegNumberText.Enabled = False 'Users cannot interact with input.
59.         StudentIDText.Enabled = False
60.         FirstNameText.Enabled = False
61.         LastNameText.Enabled = False
62.     Else
63.         RegNumberText.Enabled = True 'Users can now make inputs to query.
64.         StudentIDText.Enabled = True
65.         FirstNameText.Enabled = True
66.         LastNameText.Enabled = True
67.         If RegNumberText.Text <> Nothing Then 'Sums the number of conditions made to the query.
68.             ConditionCount += 1
69.         End If
70.         If StudentIDText.Text <> Nothing Then
71.             ConditionCount += 1
72.         End If
73.         If FirstNameText.Text <> Nothing Then
74.             ConditionCount += 1
75.         End If
76.         If LastNameText.Text <> Nothing Then
77.             ConditionCount += 1
78.         End If
79.         SelectDataWithQuery(Command, Connection, DataGrid, RegNumberText.Text, StudentIDText.Text, FirstNameText.Text,
        LastNameText.Text, ConditionCount)
80.     End If
81.
82. End Sub
83.
84. Private Sub LoadDataGrid()
85.     'Instantiates the datagrid.
86.     'This could not be done using the designer so had to be written in.
87.     DataGrid.Location = New Point(380, 70)
88.     DataGrid.RowHeaderWidth = AutoSizeMode
89.     DataGrid.Size = New Size(680, 240)
90.     DataGrid.CaptionText = "DATA"
91.     DataGrid.CaptionForeColor = Color.White
```

```
92.     DataGridView.BackgroundColor = Color.DarkSlateGray
93.     DataGridView.BackgroundColor = Color.AliceBlue
94.     DataGridView.BorderStyle = BorderStyle.Fixed3D
95.     Dim StandardFont As Font
96.     StandardFont = RegNumberLabel.Font
97.     DataGridView.Font = StandardFont
98.     DataGridView.PreferredColumnWidth = 100
99.     Me.Controls.Add(DataGrid)
100.    End Sub
101.
102.    Private Sub AddStudent_Click(sender As Object, e As EventArgs) Handles AddStudent.Click
103.        'Adds a student to the Students table.
104.        Dim FirstName, LastName, Registration As String
105.        Dim StudentID, Age As Integer
106.        Dim ValidInputs As Boolean
107.        Do 'Evaluates valid inputs.
108.            ValidInputs = True
109.            Try
110.                StudentID = CInt(TextBox("StudentID: ")) 'Checks input was an integer.
111.                Registration = TextBox("Registration: ")
112.                FirstName = TextBox("First Name: ")
113.                LastName = TextBox("Last Name: ")
114.                Age = TextBox("Age: ")
115.            Catch ex As Exception
116.                MsgBox("Invalid Input.")
117.                ValidInputs = False
118.            End Try
119.        Loop Until ValidInputs = True
120.        AddRecordToStudents(StudentID, Registration, FirstName, LastName, Age, Connection, Command)
121.    End Sub
122.
123.    Private Sub AddVehicle_Click(sender As Object, e As EventArgs) Handles AddVehicle.Click
124.        'Adds a record to the vehicles database.
125.        Dim Make, Model, Color, Registration As String
126.        Dim DateRegistered As Date
127.        Dim ValidInputs As Boolean
128.        Do 'Repeats until all inputs are valid.
129.            ValidInputs = True
130.            Try
131.                Registration = TextBox("Registration: ")
132.                Make = TextBox("Make: ")
```

```
133.         Model = InputBox("Model: ")
134.         Color = InputBox("Color: ")
135.         DateRegistered = CDate(InputBox("Date Registered: "))
136.         Catch ex As Exception
137.             MsgBox("Invalid Input.")
138.             ValidInputs = False
139.         End Try
140.     Loop Until ValidInputs = True
141.     AddRecordToVehicles(Registration, Make, Model, Color, DateRegistered, Connection, Command)
142. End Sub
143.
144. Private Sub SwitchDatasetButton_Click(sender As Object, e As EventArgs) Handles SwitchDatasetButton.Click
145.     'Changes the current dataset location.
146.     If CurrentDatasetLocation <> DatasetLocations.Count - 1 Then
147.         CurrentDatasetLocation += 1
148.     Else
149.         CurrentDatasetLocation = 0
150.     End If
151.     ImageNumber = 0
152. End Sub
153.
154. Private Sub EntryRunImage_Click(sender As Object, e As EventArgs) Handles EntryRunImage.Click
155.     Try
156.         Output.Text = ANPRSystem.RunImage(OriginalImage.Image)
157.         AddRecordToCarPark(Output.Text, Connection, Command)
158.         UpdateCarParkCount()
159.     Catch ex As Exception
160.         MsgBox("Number Plate Not Found.")
161.     End Try
162. End Sub
163.
164. Private Sub ExitRunImage_Click(sender As Object, e As EventArgs) Handles ExitRunImage.Click
165.     Try
166.         Output.Text = ANPRSystem.RunImage(OriginalImage.Image)
167.         AddExitTimeToCarParkRecord(Output.Text, Connection, Command)
168.         UpdateCarParkCount()
169.     Catch ex As Exception
170.         MsgBox("Number Plate Not Found.")
171.     End Try
172. End Sub
173.
```

```
174.     Private Sub UpdateCarParkCount()  
175.         Dim Count As Integer  
176.         Count = GetNumberOfCarsInCarPark(Connection, Command)  
177.         CarParkCount.Text = "Car Park Count: " & Str(Count)  
178.     End Sub  
179.  
180. End Class
```

RESOURCES

These were the two locations used for research into Neural Networks and Backpropagation:

<https://skymind.ai/wiki/neural-network>

<https://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf>