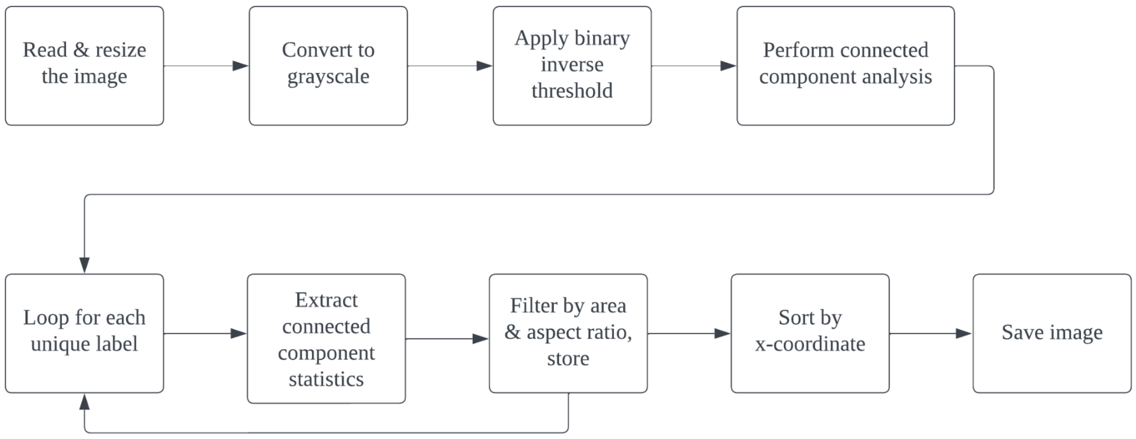
Task Allocation

We split the tasks of this assignment as follows:
Hong Bo: Cropping Images for Training Dataset, Image Preprocessing and Car Plate Segmentation
Chen Xi and Ivan: Training the Neural Network, Testing the Neural Network, Running the Segmentation Algorithm and Neural Network on the Car Plate Dataset

All of us worked on the report together.

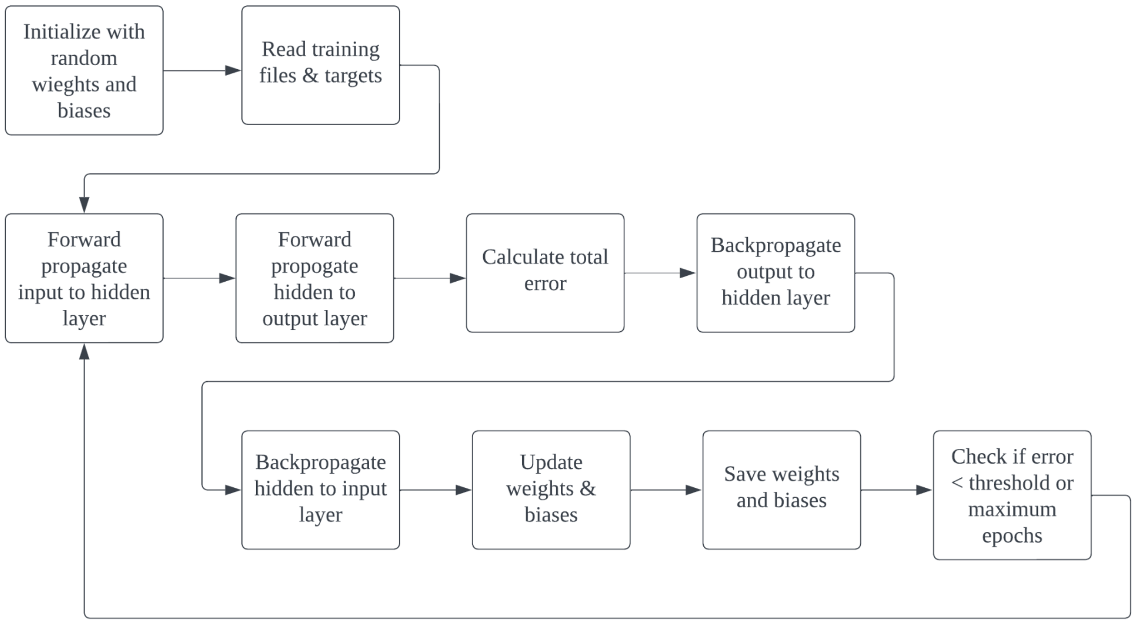
Overall Methodology

Image segmentation flowchart



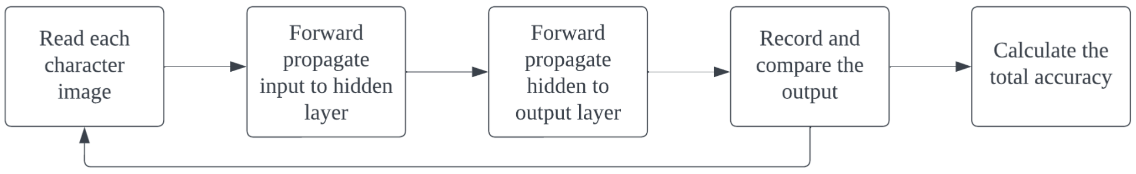
Read & resize the image
Convert to grayscale
Apply binary inverse threshold
Perform connected component analysis
Loop for each unique label
Extract connected component statistics
Filter by area & aspect ratio, store
Sort by x-coordinate
Save image

Neural network training flowchart



Initialize with random wieghts and biases
Read training files & targets
Forward propagate input to hidden layer
Forward propogate hidden to output layer
Calculate total error
Backpropagate output to hidden layer
Backpropagate hidden to input layer
Update weights & biases
Save weights and biases
Check if error < threshold or maximum epochs

Neural network testing flowchart


Read each character image
Forward propagate input to hidden layer
Forward propagate hidden to output layer
Record and compare the output
Calculate the total accuracy

ann_train.py

This code trains a neural network that recognises single characters. It uses 32 x 32 images binarized using Otsu's method represented as a flattened 1 dimensional array and one-hot encoded outputs. The matrix multiplication calculations in the forward propagation is done using the numpy dot function. The error is calculated as the sum of squared differences between the actual and predicted output. The back propagation is also done using numpy multiplication and numpy tiling to repeat *out_j* along rows. After the weights and biases are updated, they are saved to an npz file.

To use the code, place the training images in a folder named 'Train' and should follow the naming convention as mentioned in the Read_Files method. You need to instantiate the Neural_Network class with the desired number of input neurons, hidden neurons, and output neurons. After calling the Read_Files and Training_Neural_Network functions, the weights will be saved to *weights.npz*. During the training process, the global error and the number of epochs will be printed out at each epoch. The final global error will also be printed out upon program completion.

ann_test.py

This code uses pre-trained weights from an .npz file to predict the individual characters for testing. The .npz file contains four arrays *wji, wkj, bias_j, and bias_k* which are weights from the input to the hidden layer, weights from the hidden to the output layer, biases for the hidden layer, and biases for the output layer respectively. The input image is converted to a binary image using Otsu's method then converted into a flattened numpy array. This array is forward propagated through the input layer to the hidden layer to the output layer. The output is a one-hot encoded array which can be converted to a character. This character will be compared against the actual output character to calculate the accuracy.

To use this code, place the *weights.npz* in the same directory as the python file. The testing images should be in a folder named 'Test' and should follow the naming convention as described in the script. After running the script, it will print the predicted outputs and the accuracy.

segmentation.py

This code implements character recognition for car plates using pre-trained weights.

The first part is image preprocessing and character segmentation. The images are converted to grayscale, binarized using Otsu's method. cv2.connectedComponentsWithStats is used to label connected components in the image and collect statistics such as the leftmost (x) coordinate, topmost (y) coordinate, width, height, and area of each bounding box. The bounding boxes are filtered to remove boxes that are too small or too big and boxes that do not meet the aspect ratio condition (height/width > 1.2). The filtered bounding boxes are sorted by x-coordinate to maintain left to right order and saved.

The second part is character recognition using the neural network. The pre-trained weights are loaded from *weights.npz* and the images are resized to 32x32 pixels, converted to grayscale and binarized using Otsu's method. The images are flattened to one-dimensional arrays and passed though the neural network using forward propagation. The output neuron with the maximum activation is selected as the recognition character and the corresponding output is compared against the ground truth to calculate the accuracy.

You use this code, the testing images need to be in the same directory as the script and named as "1.jpg", "2.jpg", "3.jpg", etc. Another folder 'Chars' is used to store the segmented character images. After running the script, the recognized characters will be printed along with the accuracy of the recognition.

Test Results

i. The training images that failed to meet the criteria set for target outputs.
We tested the images using this code block within the *ann_test.py* file:

```
#Checking if any of the outputs fail the criterion
        for k in range(len(out_k)):
            if k == (j-1) % 20:
                if out_k[k] < 0.9:
                    failed_count += 1
                    print("Image " + str(j) + " failed the criterion")
            elif out_k[k] > 0.1:
                failed_count += 1
                print("Image " + str(j) + " failed the criterion")
```

There would be an indication of which image failed the criterion. We also kept a counter for the number of images that failed the criterion, and it would be printed to the screen upon program termination. After testing, all 160 training images passed the criteria set.

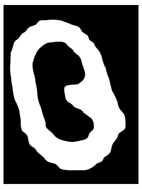
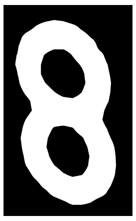
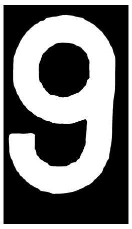
ii. The testing of the neural network program using the manually cropped images
The expected result for the test is:

```
[0,1,2,3,4,5,6,7,8,9,'B','F','L','M','P','Q','T','U','V','W',0,1,2,3,4,
5,6,7,8,9,'B','F','L','M','P','Q','T','U','V','W']
```

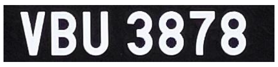
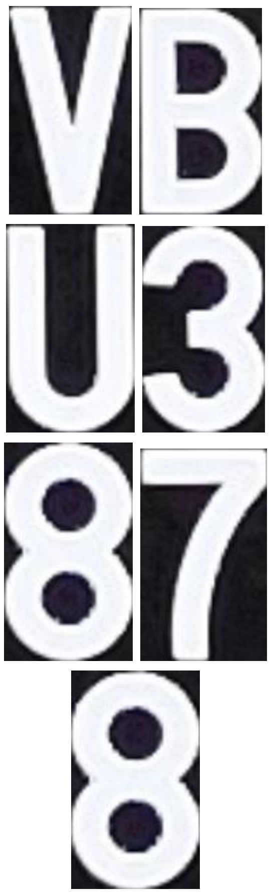
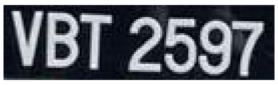
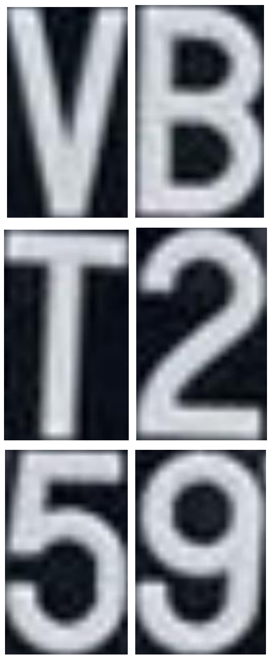
Whereas our result is:

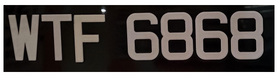
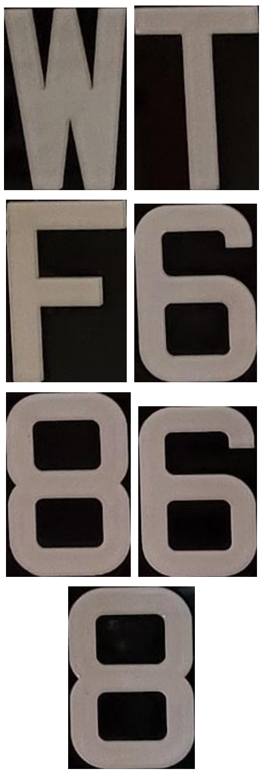
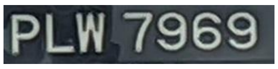
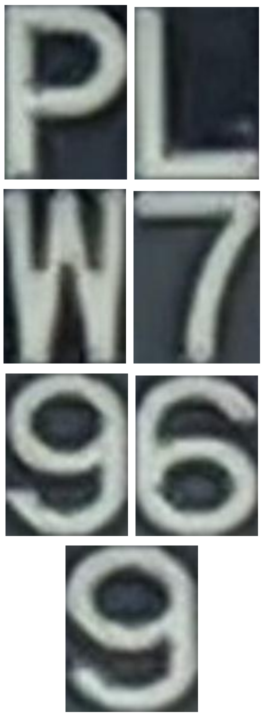
```
[0,1,2,3,4,5,6,7,8,9,'B','F','L','M','P',8,'T','U','V','W',0,1,2,3,4,5,
6,7,'B',8,'B','F','L','M','P','Q','T','U','V','W']
```

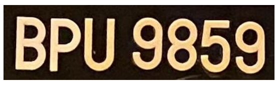
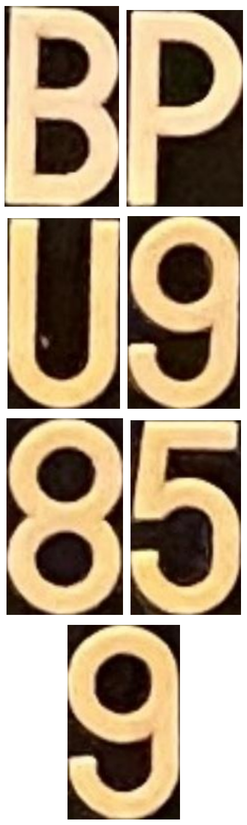
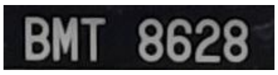
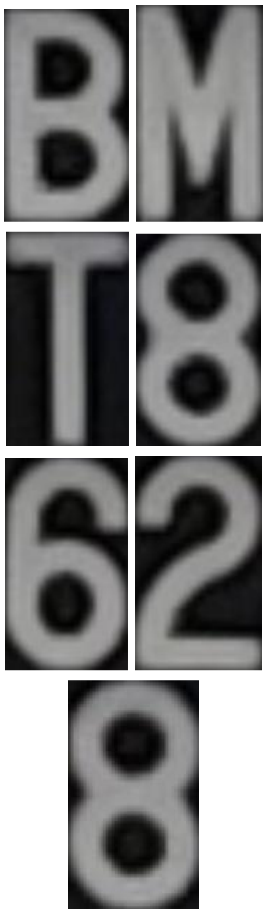
The accuracy of our neural network on the testing of the cropped images is 37/40 ≈ 92.5%.

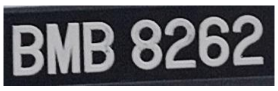
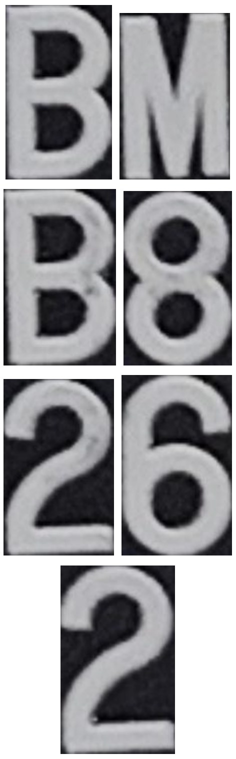
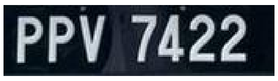
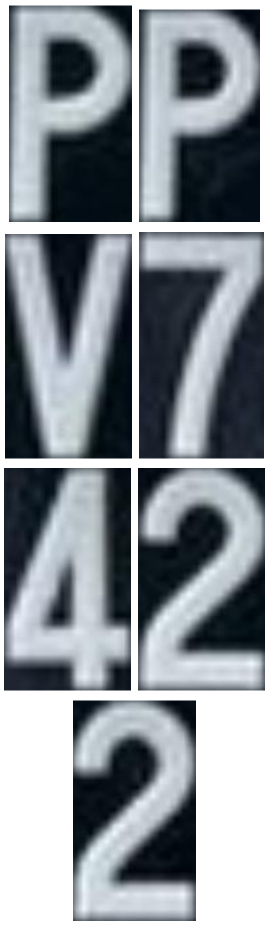
Image	Expected Result	Predicted Result	Explanation
	Q	8	A trait of the capital letter Q, the “tail” part is not distinct in this image, causing it to resemble the number 8 and incorrectly identified by the neural network.
	8	B	The number 8 has similar features as the capital letter B, in which they both have two holes and curves. The neural network may have picked up such features and misclassified the image as B.
	9	8	The number 9 has similar features as the number 8, in which they both have a closed loop at the top and a curved line at the bottom. The “tail” of 9 in this image may have been too curved to the point that the neural network misunderstood it as a closed loop, resembling 8.

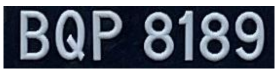
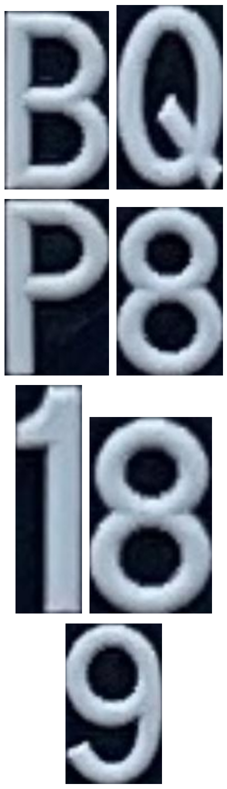
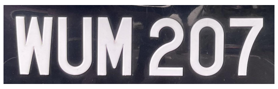
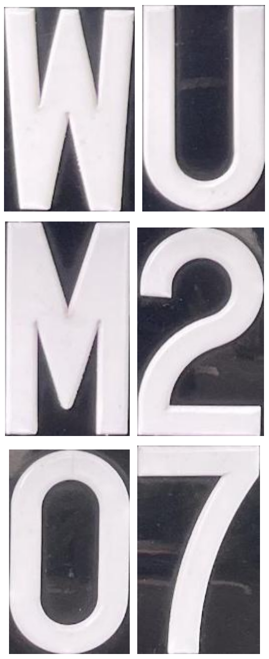
iii. The segmentation and test results obtained for the 10 license plates.

Car Plate	Segmentation	Expected Results	Predicted Results
		V,B,U,3,8,7,8	V,B,U,3,8,7,8
		V,B,T,2,5,9,7	V,B,T,2,5,8,7

		W,T,F,6,8,6,8	W,T,F,3,B,5,2
		P,L,W,7,9,6,9	P,L,W,V,9,6,9

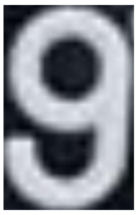
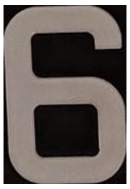
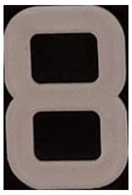
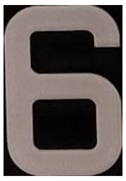
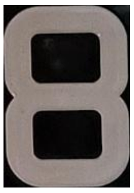
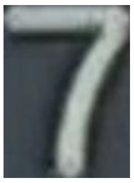
		B,P,U,9,8,5,9	B,P,U,9,8,5,9
		B,M,T,8,6,2,8	B,M,T,8,6,2,8

		B,M,B,8,2,6,2	B,M,B,8,2,6,2
		P,P,V,7,4,2,2	P,P,V,7,4,2,2

		B,Q,P,8,1,8,9	B,Q,P,8,1,8,9
		W,U,M,2,0,7	W,U,M,2,0,7

Our segmentation algorithm correctly segments all 10 car plate images into their respective individual characters.

Out of 69 characters, our neural network correctly recognised 63 of them, having an accuracy of 63/69 ≈ 91.30%.

Image	Expected Result	Predicted Result	Explanation
	9	8	The number 9 has similar features as the number 8, in which they both have a closed loop at the top and a curved line at the bottom. The “tail” of 9 in this image may have been too curved to the point that the neural network misunderstood it as a closed loop, resembling 8.
	6	3	The number 6 and the number 3 have some common features, such as a curved top and bottom, as well as two “hollow” center parts. This may be the cause of the neural network misclassifying this image.
	8	B	The number 8 has similar features as the capital letter B, in which they both have two holes and curves. The neural network may have picked up such features and misclassified the image as B.
	6	5	The number 6 and the number 5 greatly resemble each other. Their only difference is the closing of the bottom loop, therefore this feature might be overlooked by the neural network.
	8	2	This may be due to the bias introduced to the neural network by the training image set. Most images of 8 are distorted in shape whereas images of 2 are straighter and better looking.
	7	V	Both 7 and V have a slanted line, to which this feature may be picked up by the neural network and resulted in a misclassification.

Further Improvement

Recommendation on how the accuracy would have been improved

To further improve the accuracy of character recognition in a neural network, there are a few additional strategies to consider:

1. **Increasing the training data:** For this assignment, we only supply 8 training images to the neural network for each letter and number. Adding more training data can help the neural network learn a wider variety of patterns and improve its ability to generalise. We can gather additional images of the characters we want to recognize and include them in the training set. The larger and more diverse the training set, the better the network's performance is likely to be.

2. **Implementing data augmentation techniques:** Instead of relying solely on gathering more data, we can augment the existing training data by applying transformations such as rotation, scaling, translation, and distortion. These techniques can help increase the variability of the training set and make the network more robust to variations in the input.

3. **Implementing feature extraction techniques:** Feature extraction involves identifying and extracting specific characteristics or features from the input data that are relevant to the recognition task. As an example,we can extract the presence of a tail in the character "Q" to differentiate it from "O" or "0". By explicitly incorporating

such features into the training process, we can provide the network with additional cues to improve its accuracy. This can be done using techniques like image processing algorithms or specialised convolutional neural networks designed for feature extraction.