Task Allocation

Task	Contributor
Image Cropping	Kuah Jia Chen Samuel Tai Meng Yao
Cropped Image Preprocessing	
Splitting Dataset	
Neural Network	
Automatic Segmentation	
Report	

Overall Methodology

This section presents a summary of the approach employed for dataset preprocessing, number plate segmentation, and the training and testing of the neural network using a flow chart. To ensure readability, the flow chart is divided into three sections: data pre-processing, automatic segmentation, and the neural network. This division is necessary as the complete workflow cannot fit into a single flow chart without compromising legibility due to the small text size.

Step 1 (Dataset Pre-processing)

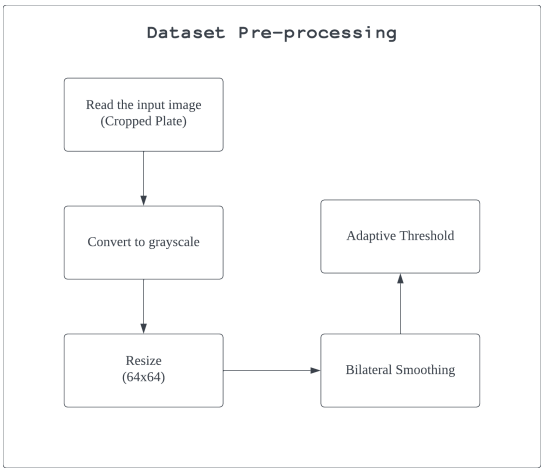

Figure 1: overall methodology for dataset pre-processing

In this section, we present an overview of the methodology used for dataset pre-processing. The process starts by reading the image and then proceeds with converting it to grayscale and resizing it. Next, we apply noise smoothing using the bilateral filter. Finally, we use the adaptive threshold function from cv2 to convert the image into a binary format. As a result, we obtain cropped alphabets and numerals that are filled rather than an outlined image.

Step 2 (Automatic Segmentation)

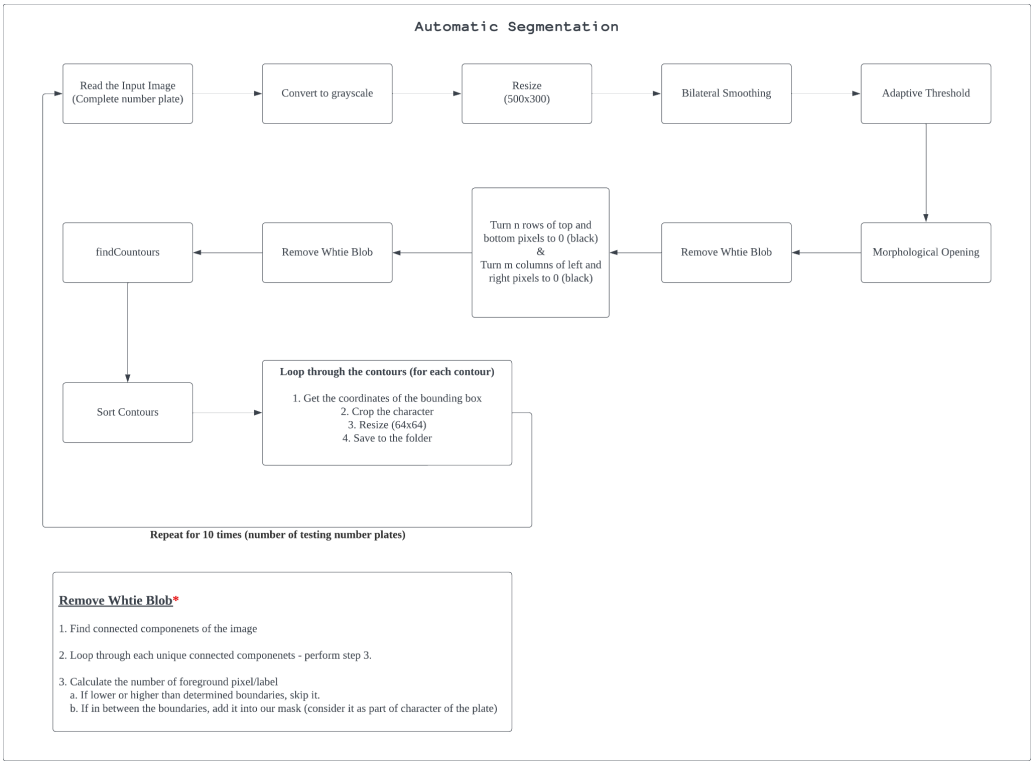

Figure 2: overall methodology for Automatic Segmentation

This section provides an overview of the methodology used for the automatic segmentation of car license plates. The process begins by reading the image and performing the grayscale conversion, followed by resizing it to 500x300 pixels. To reduce noise, a Bilateral Filter is applied to the image. An adaptive threshold is then used to convert the image into a binary format, resembling a filled number plate.

To remove small or unwanted objects while preserving the plate's shape, a morphological opening operation is applied. The next step involves removing the white blob within the image. This is achieved by identifying connected components and evaluating the number of pixels within each component against predefined thresholds. This process eliminates undesired blobs that are not part of the number plate's characters. Additionally, blacking out a specified number of pixels at the top, bottom, left, and right sides of the image is performed to further reduce noise. This assumes that the characters are typically located near the image's centre. Another round of removing white blobs is carried out on the image.

After these pre-processing steps, a clean number plate with minimal noise is obtained. The `findContours()` function is utilized to detect the contours of the characters, which are then sorted from left to right to ensure the proper order. Finally, the coordinates of the bounding boxes (representing the characters) are extracted, and the characters are cropped accordingly and saved in their respective folders.

Step 3 (Neural Network for Training)

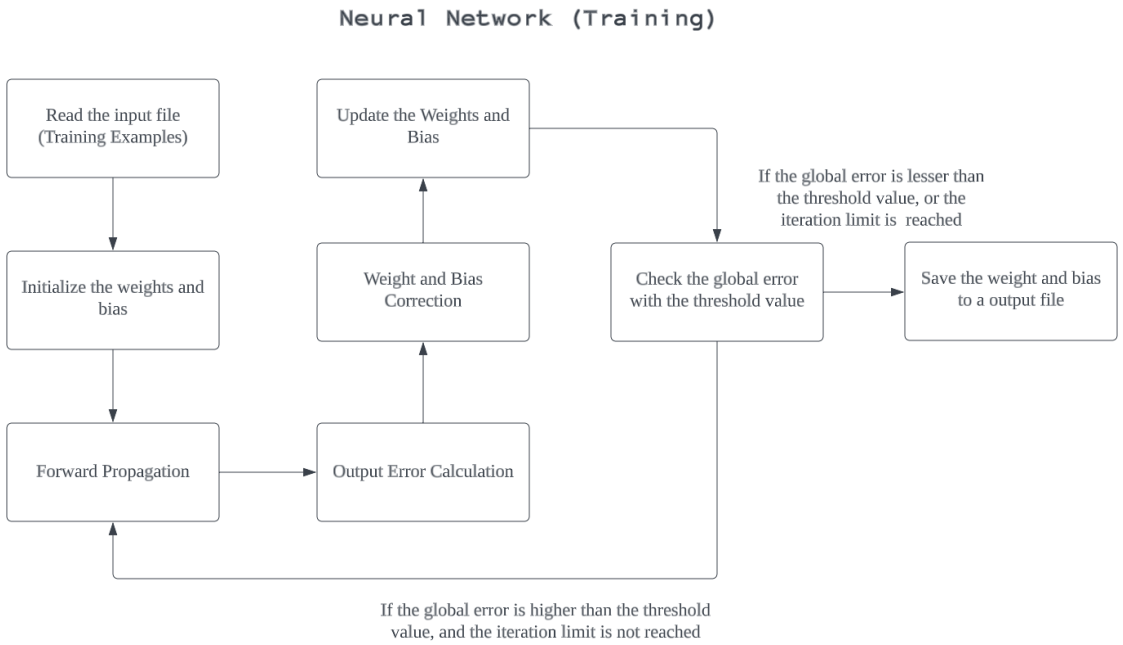

Figure 3: overall methodology for Training Neural Network

The overall methodology for training a neural network involves a series of essential steps.

The training process begins by acquiring a set of training examples, comprising input images that serve as the foundation for training the network. Next, the network's weights and biases are initialised. The training progresses through the utilisation of forward propagation, where the input data is propagated through the network.

Following the forward propagation step, the network's output error is calculated by comparing its predicted output with the expected output. This error measurement quantifies the disparity between the network's performance and the desired outcome. To improve the network's performance, an iterative process called weight and bias correction is employed. This involves adjusting the weights and biases of the network in a manner that minimises the overall error and brings the network's predictions closer to the desired outputs.

Once the weights and biases have been updated, the training process iterates by evaluating the global error against a predefined threshold value. If the global error falls below the threshold or the iteration limit is reached, the training process is considered complete. At this stage, the final weights and biases are saved to an output file, capturing the acquired knowledge of the trained network.

However, if the global error still exceeds the threshold value and the iteration limit has not been reached, the training process continues. The network undergoes another round of forward propagation, refining its predictions based on the updated weights and biases. This iterative process, consisting of forward propagation, error calculation, weight and bias correction, and threshold evaluation, persists until the desired level of accuracy is achieved or the iteration limit is exceeded.

By following this comprehensive methodology, the neural network progressively improves its performance and acquires the capability to make accurate predictions based on the training data.

Step 4 (Neural Network for Testing)

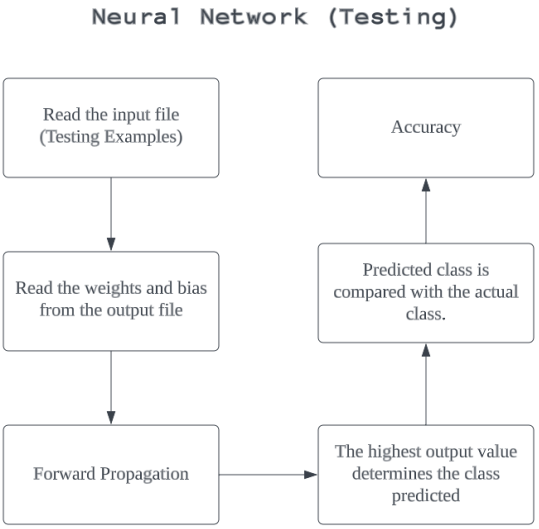

Figure 4: overall methodology for TestingNeural Network

The overall methodology for testing a neural network, whether for the Testing of the Neural Network using Manually Cropped Images or the Testing of Malaysian Car Number Plate Recognition using the Tested Neural Network, can be outlined as follows. It is important to note that both testing processes share a common underlying methodology.

To initiate the testing phase, the input file containing the testing examples is read. In addition, the weights and biases, obtained during the training phase and stored in an output file, are retrieved. Moving forward, the testing commences with forward propagation. The input data from the testing examples is propagated through the network.

Based on the results of forward propagation, the predicted class is determined by identifying the class with the highest output value. This prediction represents the network's evaluation of the given input. To assess the accuracy of the network's predictions, the predicted class is compared to the actual class. This evaluation provides valuable insights into the network's ability to correctly identify and classify the input data.

Ultimately, the accuracy of the neural network is determined by calculating the ratio of correct predictions to the total number of testing examples. This accuracy metric quantifies the network's performance and its ability to generalise predictions to previously unseen data.

By following this overall methodology, the testing phase effectively measures the neural network's predictive capabilities and provides valuable insights into its accuracy and proficiency in classifying new data.

Explanation of:

The training images that failed to meet the criteria set for target outputs

For our Neural Network, all training images have successfully met the criteria set for target outputs, showcasing the effectiveness of the model. The results of the training section highlight the network's ability to learn and recognize the desired features. Specifically, when analysing the target outputs for each training image (totalling 160), we observe that they are consistently greater than or equal to 0.9. This indicates that the network has effectively learned to identify and classify the desired features within the training data.

Furthermore, it is worth noting that the remaining 19 outputs are less than 0.1. This signifies that the network has also learned to differentiate between different classes, as these outputs reflect the network's confidence in excluding certain features that do not align with the desired class.

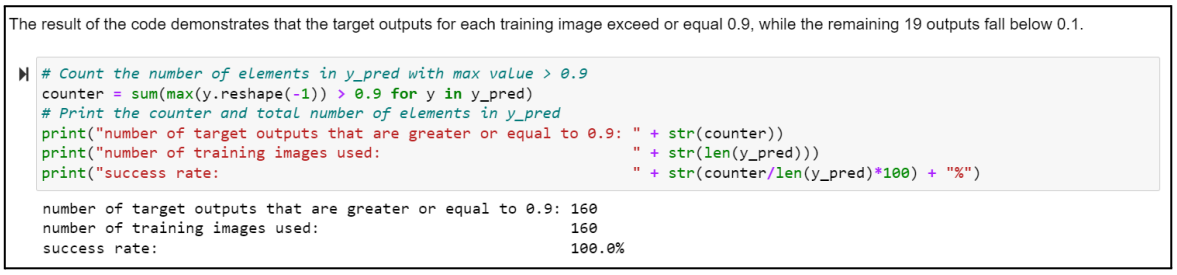

Figure 5: Number of target outputs that meet the criteria

The test results obtained for cropped images

The trained neural network program is subjected to testing using manually cropped images to evaluate its prediction accuracy. The primary goal is to assess the network's performance in making accurate predictions. The testing process produces compelling results, showcasing a remarkable level of accuracy. In particular, the network achieves an impressive accuracy rate surpassing 90%, with the above figure illustrating an outstanding accuracy rate of 95%.

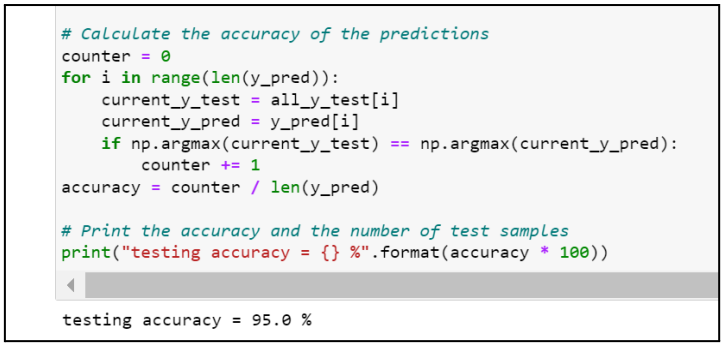

Figure 6: the test results obtained for cropped images

The segmentation and test results obtained for the 10 licence plates

The segmentation results achieved for the 10 license plates are almost flawless, as all of the segmented outputs are clear and accurate. These segmentation results serve as clear evidence of the effectiveness of our segmentation algorithm. However, there is room for improvement in the pre-processing stage of the algorithm to better preserve the shape of the characters without any interference from image noise. In summary, all 10 license plates were successfully segmented, showcasing the overall success of our approach.

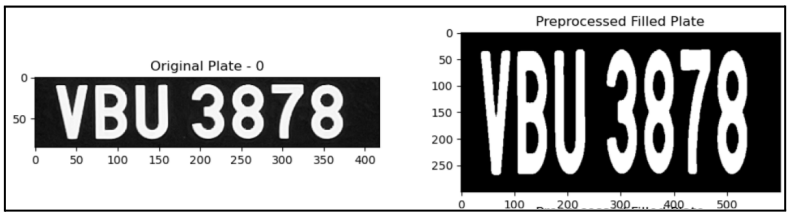

Figure 7: Preprocessed Malaysian Number Plate

Figure 8: Result of automatic segmentation

The trained neural network program underwent evaluation by applying it to recognize 10 Malaysian car number plates in real-world scenarios, with the objective of assessing its performance on actual data. The testing results clearly demonstrate the network's remarkable achievement, as it achieved a high accuracy rate of 91% in correctly recognizing and classifying the car number plates, as shown in the above figure. Out of a total of 69 characters evaluated, the network successfully classified 63 characters correctly.

```
# Print the actual and predicted number plates
print("actual number plates: ")
print(final_actual)
print("predicted number plates:")
print(final_predicted)
```

```
actual number plates:
['VBU3878', 'WUM207', 'VBT2597', 'WTF6868', 'PLW7969', 'BPU9859', 'BMT8628', 'BMB8262', 'PPV7422', 'BQP8189']
predicted number plates:
['V8U3878', 'WUM207', 'V8T2597', 'WTF6868', 'PLW7969', '8PU9859', '8MT8628', 'UMB8262', 'PPV7422', 'UQP8189']
```

Figure 9: the actual and predicted number plates

```
# Calculate and display the accuracy
accuracy = counter / len(y_pred)
print("accuracy = {} % ({} / {})".format(accuracy * 100, str(counter), str(len(all_y_test))))
```

```
accuracy = 91.30434782608695 % (63 / 69)
```

Figure 10: the test results obtained for the 10 licence plates

Recommendation on how the accuracy would have been improved

However, the results obtained from our neural network indicate challenges in accurately classifying the character "B," as it is often misidentified as "8" or "U." This discrepancy highlights an area that requires improvement in our network's performance. To address this issue, one possible approach is to train the network using higher quality images that specifically emphasise the character "B." By providing the network with more representative and distinct examples of this character, we can enhance its ability to correctly recognize and classify "B" in future predictions.

Moreover, improving the overall accuracy of the network can be achieved by implementing more effective preprocessing techniques. Preprocessing plays a crucial role in preparing the input data for the network, and by refining this stage, we can potentially minimise noise, enhance image quality, and optimise feature extraction. Employing advanced preprocessing methods such as image enhancement, noise reduction, and normalisation can contribute significantly to improving the network's overall accuracy.

Therefore, by combining targeted training with higher-quality images for the character "B" and implementing robust preprocessing techniques, we can enhance the performance and accuracy of our neural network, enabling it to make more precise and reliable predictions.

```
# Print the actual and predicted labels for each misclassified data.
for data in visualize_misclassified_data:
    print("Actual label: " + data[0])
    print("Predicted label: " + data[1])
    print()
```

```
Actual label: B
Predicted label: 8

Actual label: B
Predicted label: 8

Actual label: B
Predicted label: 8

Actual label: B
Predicted label: 8

Actual label: B
Predicted label: U

Actual label: B
Predicted label: U
```

Figure 11: the actual and predicted labels for each misclassified data