



FIT3081 Image Processing

Assignment 3 (S1, 2023) - Report

Student 1: Tatiana Sutulova, 30806151

Student 2: Tang Jia Sheng, 32184905

Student 3: Lim Xian Yi, 32454783

1. Task Allocation

1.1 Sutulova Tatiana, 30806151

To begin with, I have implemented the segmentation function that detects separate characters in an image of a car plate, and crops out each character into a separate image. Then, I have worked on implementing the flow of the logic for training and testing. Lastly, I have improved the accuracy of the testing results to the maximum value we could reach.

1.2 Tang Jia Sheng, 32184905

I was responsible for uploading some of the cropped images for training data samples ,implementing part of the functions and testing the most suitable parameter for the model to get the best result. I was also responsible for implementing part of the flow of the logic for training and testing the model.

1.3 Lim Xian Yi, 32454783

I was responsible for testing the correctness of the neural network calculation. My team members were responsible for writing the python code according to 3-layer neural networks architecture taught in week 5 lectures. Since embedding mathematical formulas into python code is not exactly simple and straightforward especially if we wanted to fully utilise the built-in libraries like Numpy, we have to ensure it produces the expected results before the team can proceed to the next step. I was also responsible for drawing the flowchart of our methodology.

2. Methodology



Figure 1. Training & Testing Logic flow chart

2.1 Training of the neural network program using the manually cropped images

The training process begins by manipulating the images in several steps to prepare them for analysis. First, the images are resized to a uniform size of 28 x 28 pixels. Next, they are converted to grayscale and subjected to binary thresholding, which transforms them into a binary representation, simplifying the complexity of the input data. After these initial

transformations, each image is represented as a one-dimensional array of size $1 \times 28 \times 28$, where each value corresponds to an 8-bit integer ranging from 0 to 255. In this representation, 0 represents black, 255 represents white, and values in between represent various shades of grey. To facilitate optimal performance of the sigmoid activation function and mitigate potential numerical issues such as overflow and underflow, each value in the array is then normalized to a range of 0 to 1. This normalization process ensures that the input values fall within the preferred range for the sigmoid function. Furthermore, z-score normalization is applied to the data to ensure appropriate scaling and improve the convergence speed. Z-score normalization is a statistical technique that transforms the data so that it has a mean of zero and a standard deviation of one.

We use a random function to shuffle the image to read in the folder so we can feed the model with data in random order. It is because when feeding the same label data in order, the network may encounter a systematic bias. The network could potentially learn to predict or rely on patterns specific to the order of the data. Shuffling the data ensures that the network sees a mixture of different samples, reducing such biases.

The training starts with initialising the weights and biases with random numbers from -0.5 to 0.5 and from 0 to 1 respectively by calling *Weight_Initialization()*. Then, we forward propagate the image array with the w_{ji} and $bias_j$ into the hidden layer. After that, we forward the output of the hidden layer out_j with w_{kj} and $bias_k$ into the output layer. Then, we check whether the total error is less than the error set to 0.005 or the number of iterations of 300 is reached by calling *Check_for_End()*. If it returns false, we calculate the correction of the weights and biases of the output layer and hidden layer by calling *Weight_Bias_Correction_Output()* and *Weight_Bias_Correction_Hidden()*. After that, we update the weights and biases by calling *Weight_Bias_Update()* and repeat the steps to forward propagate with new weights and biases until *Check_for_End()* returns true. If *Check_for_End()* returns true, we save the current weights and biases and proceed to feed the next training data sample.

2.2 Testing of the neural network program

In order to perform testing on the manually cropped testing images dataset we perform the same image manipulation as described in 2.1, followed by the forward propagation from input to hidden layer and from hidden to the output layer, determining the output values in a form of 1×20 array, where the position of the highest value indicates its classification class: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, B, F, L, M, P, Q, T, U, V, W]

2.3 Segmentation

The segmentation starts with normalising all the images to be of approximately the same size and converting them to grayscale for easier manipulation. The next step is applying the Gaussian blur, adaptive mean thresholding and binary thresholding, which means that the threshold value is calculated locally for each pixel based on its neighbours. It results in a

binary image, where the pixels above the adaptive threshold are set to white (255), whereas pixels below are set to black (0), which helps to segment images into regions of interest based on differences in pixel intensities. Next, we are using the *connectedComponents(..)* function of OpenCV library that performs connected components analysis on the binary image, which allows to identify and label distinct regions in an image (starting from 0 for the background). Next step involves looping through the determined labels and identifying the unique components highlighting them in the image. Each of the identified components are then outlined with the rectangle using *findContours(..)* and *boundingRect(..)* functions of OpenCV library. We are sorting them from the left to the right and getting only those that are bigger in height than 50% and smaller in height than 95% of the original image height.

In order to perform testing on the auto-segmented testing images dataset we perform the same image manipulation as described in 2.1, followed by the forward propagation from input to hidden layer and from hidden to the output layer, determining the output values in a form of 1 x 20 array, where the position of the highest value indicates its classification class: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, B, F, L, M, P, Q, T, U, V, W]

3. Experimental Results

3.1 Training of the neural network program using the manually cropped images

The criteria for training images is that the target outputs must be greater than or equal to 0.9, while remaining 19 outputs should be less than 0.1. Table 1 represents the set of images that did not meet the criteria, since their target output value was 0.88-0.89.

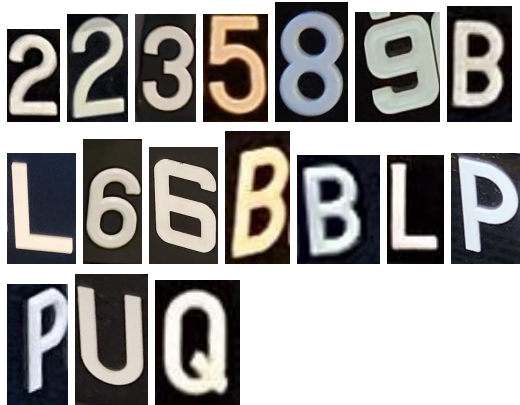
Images	Their corresponding names
	5_01.jpg, 2_04.jpg, 8_07.jpg, 9_11.jpg, L_04.jpg, 2_02.jpg, B_10.jpg, 3_02.jpg, P_03.jpg, 6_02.jpg, Q_08.jpg, P_09.jpg, 6_06.jpg, L_10.jpg, B_02.jpg, U_09.jpg, B_09.jpg

Table 1. Images that do not meet training criteria

3.2 Testing of the neural network program using the manually cropped images

The results obtained for cropped images is 70% accuracy, meaning that 28/40 images are properly classified. Table 2 displays all the successfully classified images.



Images	Their corresponding names
	0_03.jpg, 2_07.jpg, 2_09.jpg, 3_04.jpg, 3_08.jpg, 4_06.jpg, 4_09.jpg, 5_09.jpg, 6_03.jpg, 7_02.jpg, 7_09.jpg, 8_05.jpg, 8_10.jpg, B_01.jpg, F_02.jpg, L_01.jpg, L_09.jpg, M_01.jpg, P_01.jpg, P_06.jpg, Q_02.jpg, T_01.jpg, T_02.jpg, U_01.jpg, V_02.jpg, V_05.jpg, W_08.jpg, W_09.jpg

Table 2. Successful testing cases for manually cropped images

3.3 Segmentation and testing for 10 Malaysian car number plate

The segmentation was successfully performed for all 10 Malaysian car number plates with each of the characters being properly segmented as shown in Table 3. The results obtained for the auto-segmented images is 52% accuracy, meaning that 36/69 images are properly classified.

Images	Their corresponding names
	0_0_B.jpg, 0_1_M.jpg, 0_2_B.jpg, 0_3_8.jpg, 0_4_2.jpg, 0_5_6.jpg, 0_6_2.jpg 1_0_B.jpg, 1_1_M.jpg, 1_2_T.jpg, 1_3_8.jpg, 1_4_6.jpg, 1_5_2.jpg, 1_6_8.jpg 2_0_B.jpg, 2_1_P.jpg, 2_2_U.jpg, 2_3_9.jpg, 2_4_8.jpg, 2_5_5.jpg, 2_6_9.jpg 3_0_B.jpg, 3_1_Q.jpg, 3_2_P.jpg, 3_3_8.jpg, 3_4_1.jpg, 3_5_8.jpg, 3_6_9.jpg 4_0_P.jpg, 4_1_L.jpg, 4_2_W.jpg, 4_3_7.jpg, 4_4_9.jpg, 4_5_6.jpg, 4_6_9.jpg 5_0_P.jpg, 5_1_P.jpg, 5_2_V.jpg, 5_3_7.jpg, 5_4_4.jpg, 5_5_2.jpg, 5_6_2.jpg

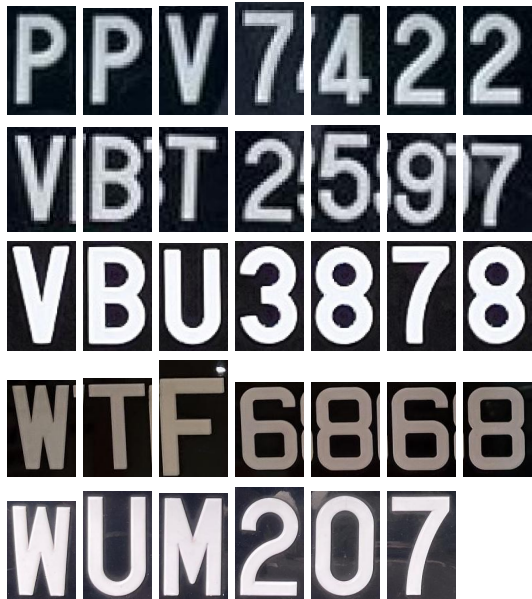
	<p>6_0_V.jpg, 6_1_B.jpg, 6_2_T.jpg, 6_3_2.jpg, 6_4_5.jpg, 6_5_9.jpg, 6_6_7.jpg</p> <p>7_0_V.jpg, 7_1_B.jpg, 7_2_U.jpg, 7_3_3.jpg, 7_4_8.jpg, 7_5_7.jpg, 7_6_8.jpg</p> <p>8_0_W.jpg, 8_1_T.jpg, 8_2_F.jpg, 8_3_6.jpg, 8_4_8.jpg, 8_5_6.jpg, 8_6_8.jpg</p> <p>9_0_W.jpg, 9_1_U.jpg, 9_2_M.jpg, 9_3_2.jpg, 9_4_0.jpg, 9_5_7.jpg</p>
---	--

Table 3. Segmentation results

4. Recommendations on accuracy improvements

In order to improve accuracy of the neural network, more data should be collected for training. Increasing the training dataset provides more diverse examples for models to learn from, allowing the model to capture a broader range of patterns and variations present in the data and, thus, enabling it to generalise better to unseen examples. With more data, the model becomes more robust and less likely to overfit or memorise specific examples from the training dataset. In case there is no data to be added to the dataset, it is possible to perform data augmentation, which allows to artificially increase the size of the training dataset by applying various transformations to the existing dataset, such as adding noise or scaling the images.