

CHAPTER 4

ANALYSIS AND DESIGN

Analysis

The Basic of Fuzzy Logic

As said in [13], that the fuzziness is more likely appear as a result of reasoning, The temperature of an object can be said as “too hot” to some people, while some other can describe is as “warm” instead. The other example is that a person living in Asia consider that someone with 170cm height as tall, while the other person who lived in Europe considers 190cm as tall, while 170cm is considered short. These differences in opinions later becomes what is called as the membership of an object. And the membership values of an object can be different from each perspectives.

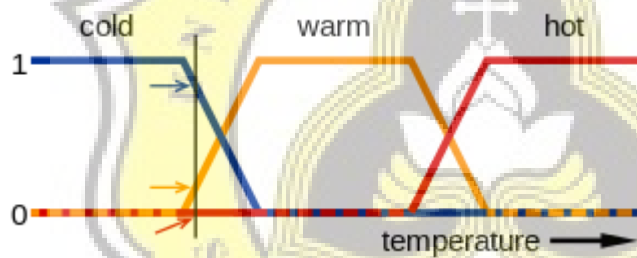


Illustration 4.1. 1: fuzzy graph

The image 4.1.1 above is an example of fuzzy logic graph, also known as trapezoidal graph. It can be seen in the image that there are some points where a temperature falls into 2 category at the same time. Such as “cold” and “warm” or “warm” and “hot” at the same time. The states lie between the value of 1 and 0, which means they’re partially true or partially false.

Considering this, there must lie some point where a condition or state receives the same opinion of reasoning. And that point exists may exist as true, false, or inbetween, as a fuzzy object can be partially true or partially false. Since a computer only considers things as 1 or 0, the partial state of an

object needs to be mapped by using functions called “membership function” in order to obtain the fuzzy output of an object. And the whole process of converting an object to obtain its degree of memberships is called “fuzzification”. After the fuzzification, the memberships will then be integrated to obtain its truth value. And this process is called defuzzification.

Obtaining Image Features and Character Recognition

In order to be able to predict the characters and numbers in the chassis plate, the neural network needs to be feed with datasets to train it. The dataset comes from the character images segmented from a feature map or a decision map, which is the combination of different feature maps extracted from an image.

The steps that will be done in this research are as follows:

1. Image conversion to grayscale

The obtained chassis plate images are first converted into grayscale by creating a new image using the maximum values in each pixels as the new pixel values.

2. Fuzzy Edge extraction

As for the fuzzy edge map, this formula is chosen since it offers a more efficient and precise fuzzy edge detection. The formula was proposed by Hamid [14] and also mentioned in [3]. Firstly we need to obtain 2 membership values for each corresponding cells in the image.

$$\widehat{\mu}_{mn}^1 = \min \left(1, \frac{\max_{i,j \in [1,w]} W(i,j) - \min_{i,j \in [1,w]} W(i,j)}{\Delta_1} \right),$$

Illustration 4.1. 2: first membership

$$\widehat{\mu}_{mn}^2 = 1 - \min \left(1, \frac{g_{mn} - \text{MeanMed } W(i,j)}{\Delta_2} \right),$$

Illustration 4.1. 3: 2nd membership

In the images 4.1.2 and 4.1.3 above, g_{mn} refers to the gray levels in each pixel of the grayscale image. And the max and min are obtained from the grayscale version of the original image. And Δ is replaced with 64, as said in [3], to obtain the best possible result.

After both memberships are obtained, the next step is to normalize the value by using this formula.

$$X' = \bigcup_{m=1}^M \bigcup_{n=1}^N \frac{\min(\widehat{\mu}_{mn}^1, \widehat{\mu}_{mn}^2)}{g_{mn}}.$$

Illustration 4.1. 4: normalization

As seen in the image 4.1.4 above, the final image containing the truth values is obtained by first looping around the pixels of the image and selecting the minimum value between the first membership and the second membership,

and then that minimum value will then be divided by the gray value of the corresponding pixel in the grayscale image.

The resulting fuzzy image will then be segmented manually to obtain and characters and numbers contained in it. To make it easier when recognizing the image and to reduce the complexity when recognizing similar characters, the segmented image will be divided into 2 groups known as “numbers” and “letters” group. With this, the recognition process can be done separately for number and letter characters.

3. Other approaches

Aside from extracting the fuzzy edge, another approach is also used in which the other kinds of fuzzy maps. Such as Fuzzy Hue, Saturation, and Intensity maps then integrating it with the Fuzzy Edge image to obtain the decision map, based on [1].

The first step in this approach is to first converting the original image into Hue, Saturation, Intensity format.

$$I = \frac{(r + g + b)}{3}$$

$$S = 1 - \frac{\min\{r, g, b\}}{I}$$

$$H = \cos^{-1} \left\{ \frac{(r - g) + (r - b)}{2 [(r - g)^2 + (r - b)(g - b)]^{1/2}} \right\}$$

Illustration 4.1. 5: HSI function

As seen in the image 4.1.5 above, the intensity of the image is obtained by picking the average value of the red, green, and blue of the image. In other words, the intensity is the grayscale version of the original image. As for saturation can be obtained by subtracting 1 with the result of dividing the

minimum pixel value with its average value, and the Hue value can be seen above.

As for the fuzzy maps can be obtained with the following functions:

$$\mu_{\tilde{H}}(h) = u(\mu_r(h), \mu_g(h))$$

Illustration 4.1. 6: hue membership

$$\mu_{\tilde{S}}(h) = \exp(-as).$$

Illustration 4.1. 7: sat membership

$$\mu_{\tilde{I}}(i) = 1 - \exp(-a(i - \bar{i})).$$

Illustration 4.1. 8: int membership

On the functions in images 4.1.6, 4.1.7, and 4.1.8, the α in here is a constant with the value of 1. And h, s, I are the hue, saturation, and intensity respectively.

while μ_r and μ_g are the memberships of red and green of a corresponding hue pixel which can be obtained by using this function:

$$\mu_c(h) = \exp(-a|h - h_c|)$$

Illustration 4.1. 9: pixel membership

The value h_c in the image 4.1.9 above can be replaced by the average of all the hue values in the hue map. And h is the value of the corresponding hue pixel.

The fuzzy HSI maps will then be integrated into decision map g by union:

$$\tilde{g} = u(w_h \tilde{h}, w_s \tilde{s}, w_i \tilde{i})$$

Illustration 4.1. 10: decisionmap g

In image 4.1.10, the process of obtaining the decision map g can be seen. Before being integrated, first the weight of the fuzzy maps is needed for the fuzzy maps h, s, and i , which can be obtained with this function:

$$w_a = 1 - \frac{\left(\sum_{i=1}^M \sum_{j=1}^N b_{ij} \right)}{MN}$$

where

$$b_{ij} = \begin{cases} 1, & \text{if } \tilde{a}_{ij} \geq t \\ 0, & \text{otherwise} \end{cases}$$

Illustration 4.1. 11: weight

In illustration 4.1.11, the value of b_{ij} is obtained by thresholding the value of the corresponding pixels (in this case, \tilde{a}_{ij}) within the fuzzy map. And the threshold value of t can be obtained from the average of the maximum and minimum value of the fuzzy map.

The same will be done to the g map and fuzzy edge map before integrating them into the final decision map m that will be segmented and used as dataset. The function for the decision map is as follows:

$$\tilde{m} = w_g \tilde{g} \pm w_e \tilde{e},$$

Illustration 4.1. 12: decisionmap m

In this case, this method does not seem to be effective. Due to the many similarities in color between the characters engraved in the plate and the other

parts of the plate itself, which are in shades of gray, causing some of the fuzzy maps to appear blank and making the integrated map to appear darker. Some objects inside the image are lost after the conversion.



Illustration 4.1. 13: fuzzyedge vs edge detection

In the image above, the difference in feature quality can be seen. In which the original image can be seen on top, decision map is in the middle, and fuzzy edge map in the bottom. This shows that the decision map method isn't very effective to be used chassis plate.

4. Dataset retrieval

Once the process of creating the decision map is done. The map will then divided into 2 groups for training dataset and testing dataset. This will be done on both the alphabet and numeric images.

And from those groups, the maps will be stored in a dataset array and labelled for each characters and numbers. And then they'll be saved as text files.

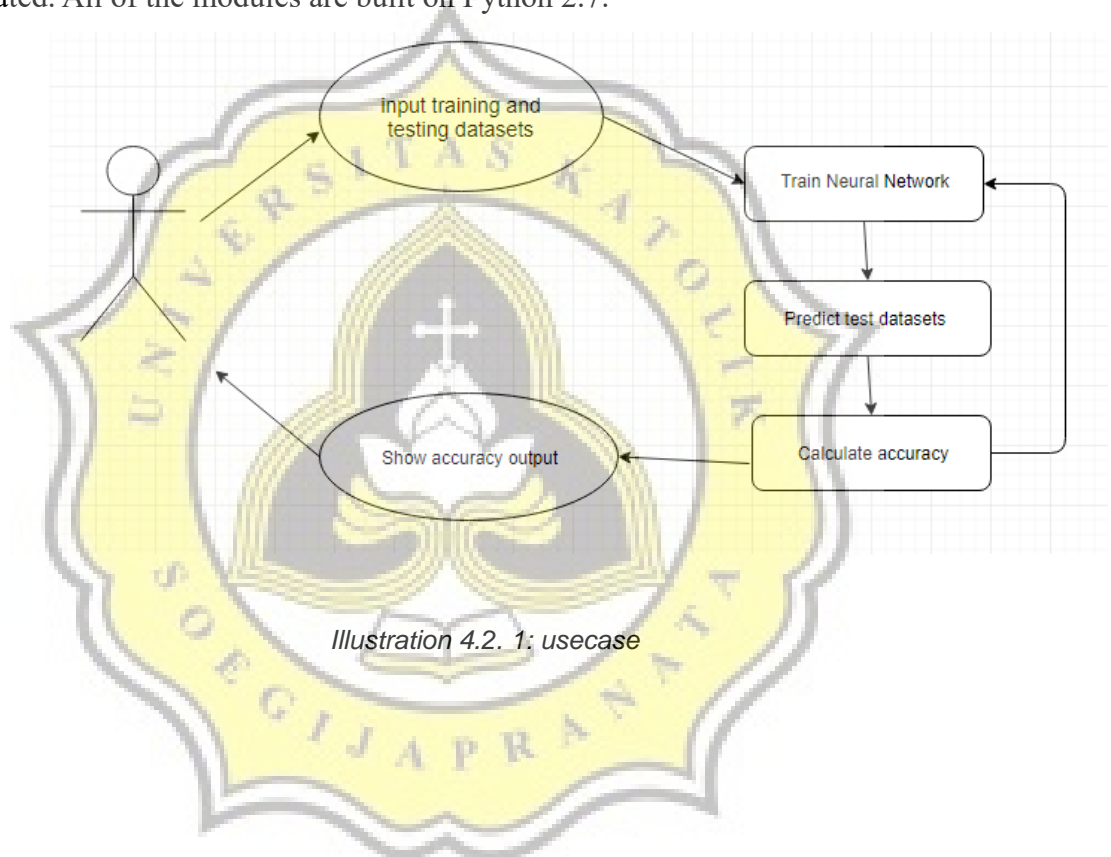
5. Training and Testing with Neural Network

The dataset will be used to train and test the Neural Network. The neural network will use different amount of output nodes to test the alphabet and numeric datasets separately. The accuracy will then be shown each time the training and testing finishes so that it can be analysed. And after the training

session finished, the neural network will begin predicting each numbers and letters in every images in the inputted directory and saving the result.

Desain

This part will explain the design of the programs and modules that are created. All of the modules are built on Python 2.7.

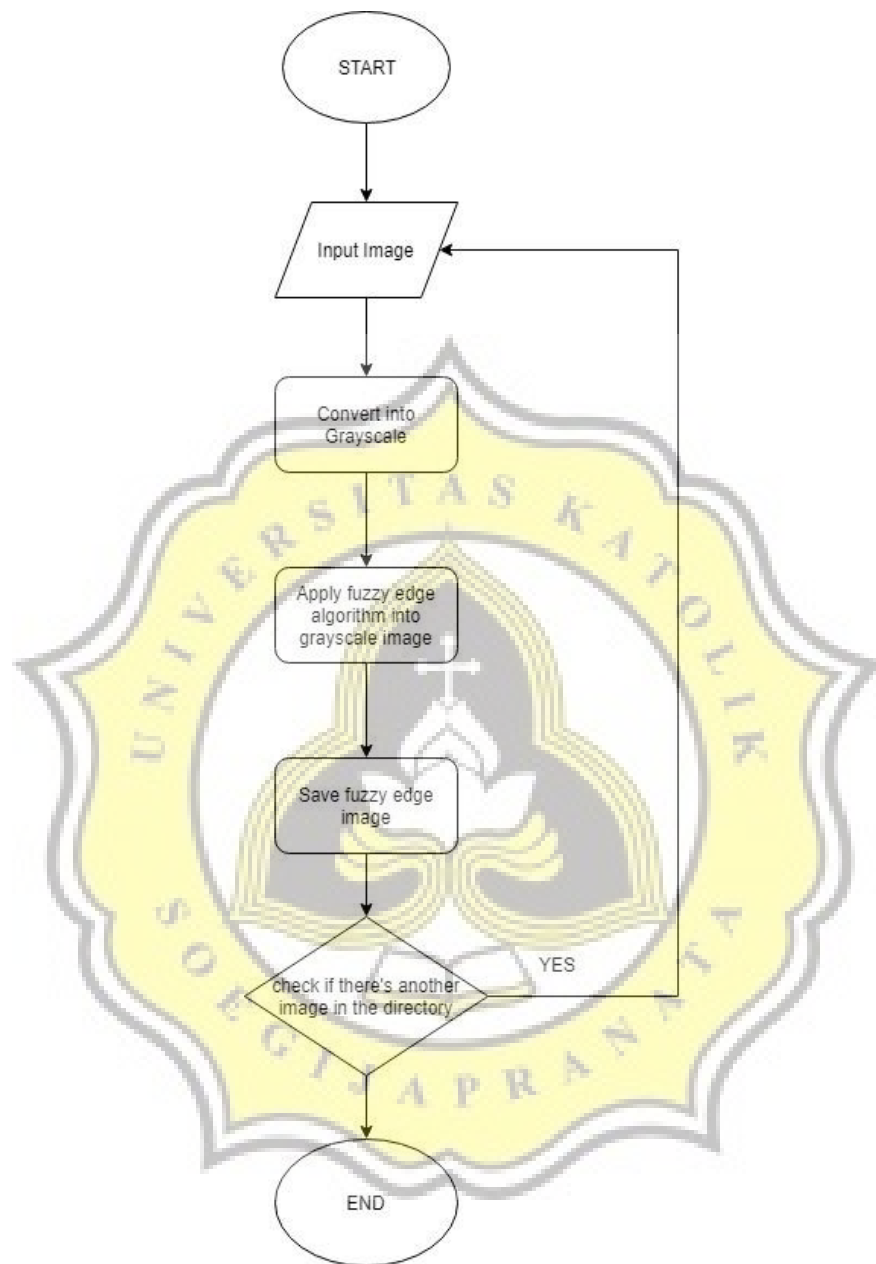


As seen on the diagram above, the user will need to input the datasets needed. Using the dataset, the neural network will train itself and do a prediction for each of the datasets. After the predicting process ends, the accuracy of the prediction will be calculated and shown to the user.

The neural network used in this research uses the OCR neural network made by Nielsen [6].

The designs of other modules in this research are as follows

Obtaining fuzzy edge diagram

*Illustration 4.2. 2: flowchart1*

Based on the diagram above, the fuzzy edge program will first convert the input image into grayscale before applying the fuzzy edge algorithm to extract its features. The features of the image will then be saved as new image, and the program will loop through the directory to look for other images inside the directory to be inputted as the next image and repeat the process until the features of all the images inside the directory are extracted. Once all the features are extracted, the loop will be stopped.



Predicting the character and number outputs:

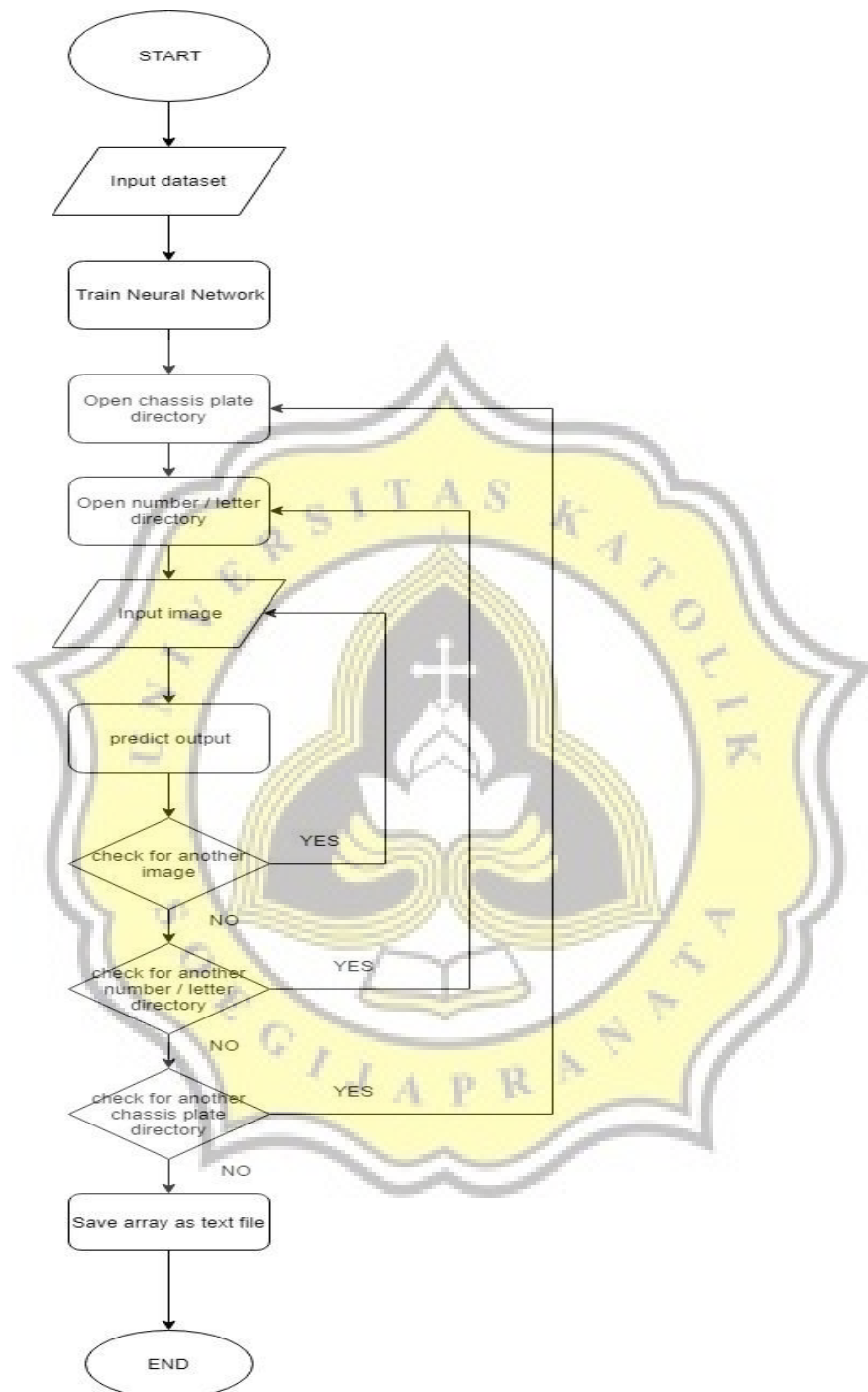


Illustration 4.2. 3: flowchart2

In the diagram above, the program will first take the dataset and use it to train the neural network. The training process will repeat for as many times the user desires, and after the training process is finished, the program will then open the chassis plate directory in search for the image that is located inside the number or letter directory and predict the images based on which directory it is placed. And the images inside the number or letter directory will each be predicted using separate neural networks to avoid complexity.

