CHAPTER 3
RESEARCH METHODOLOGY

In this research , the first main goal is to build a working model that have been trained and tested with the prepared dataset. Building the model from preparing the dataset function which consist of loading and preparing the images before being processed by combined feature functions , before it was split for training and testing purpose. Another functions that need to be preapred other than the pre-processing function itself are the neural network that is ResNet50 in this case , because ResNet50 also consist of convolutional and identity block which have their own layers in them. After preparing the functions needed for the model , what was left to do is calling those functions the appropriate input for them. Debugging the coding is also an important part as the training or even the image loading process may take a very long time based on the paramters used and the hardware used to run the coding.

The dataset used are taken from CASIA v2 consisting of image splicing type of forgery or editing an image by adding objects into an image which shape the definition of image forgery on the proposed model. They were made of 8082 images with approximately 42 : 58 ratio for authentic and forged images. During implementation , those image were then divided by 80:20 , where 80% of the dataset was used for training and the rest 20% was used for testing the trained model.

After clearing the first goal , the research proceed to the second and primary goal , to make the working model have a reliable performance. To put a clear boundaries whether the model count as a success or not , I have decided the minimum requirements for reliable performance is 70% or 0,7 in test accuracy. Do note , the reason why the first goal is to build a working model is because I were building ResNet50 from scratch instead of performing transfer learning to a pre-trained ResNet50 model. It was done so that the performance results of the model can be truly measured whether it would be able to reliably distinguish between forged image or not without any other unaccounted external variables from the pre-trained model.

After building a working ResNet50 model , the job here is to increase the performance of the model. In attempt to increase the performance , I tried to adjust various parameter of the model , such as learning rate, image size, epoch, and batch size. However in that very attempt , I found numbers of limitation and even malfunctions that not only hinder the progress , some other malfunctions even reduce the performance of the model. From unsuitable parameter cause the learning process of the neural network hit a stop and cause the resulting performance to stuck and giving the exact same accuracy which was caused by unsuitably high learning rate to a lack of given parameter to a function makes them not working properly despite there are no error induced. Those problems bring us to the previous point which is to debug the program as detailed as possible to avoid such event , especially because some of the problem were solved after some of the data collection were done , despite the fact that the result still gives an information needed it is not a desireable condition for data collecting.

As said previously , that image chroma is more sensitive to change and that statement has been proven in a number of research. Even with that , I also trying to use normal greyscaled image as inputs for the model before going with image chroma. It was done to see the difference in performance between them. The results of the attempt was then recorded to help comparing the performance and decide which one should be used as input for the model in later implementations. The results that were taken and compared are the accuracy of last training epoch and test accuracy.

Finally , for data collection , we compare results based on the difference in parameter with the purpose of finding suitable parameters for the model that were obtained by using evaluate function from tensorflow keras. It was done by changing one specific parameter at a time , such as learning rate, epoch, etc and to see how those change in the parameter would change and affect the result in model performance. The results then were recorded and compared so that informations can be concluded from them to see the better and more suitable parameters to use. Informations that were obtained would help in future development to achieve a better results.

After implementation and data collection by changing parameters , another data collection were done by changing the input. Unlike the previous implementation , not only we must change some of the parameter but also change the coding of data preparation and steps of pre-processing. But the change made toward the coding , were made without changing the basic or general architrctture of the model that was shown in figure 5. The implementation of change in input that does not deviate from the general structure of the model architecture open a new gate of improvement for the model. Change in smaller detail like parameters does give a huge different in result , but with the new opportunity there more possibilities on improving the model as a whole.

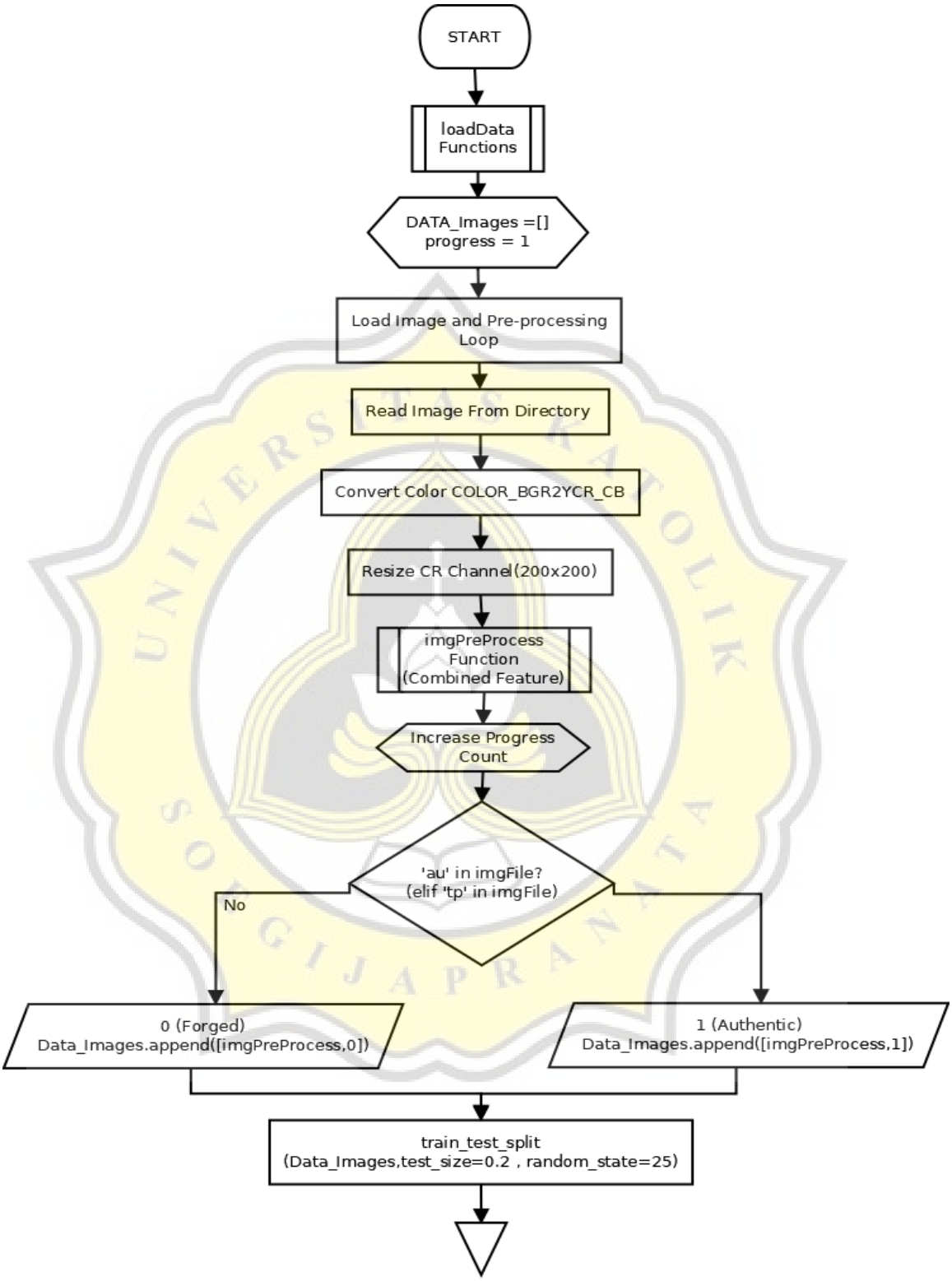

Figure 6: Flowchart Of The Proposed Model Part 1

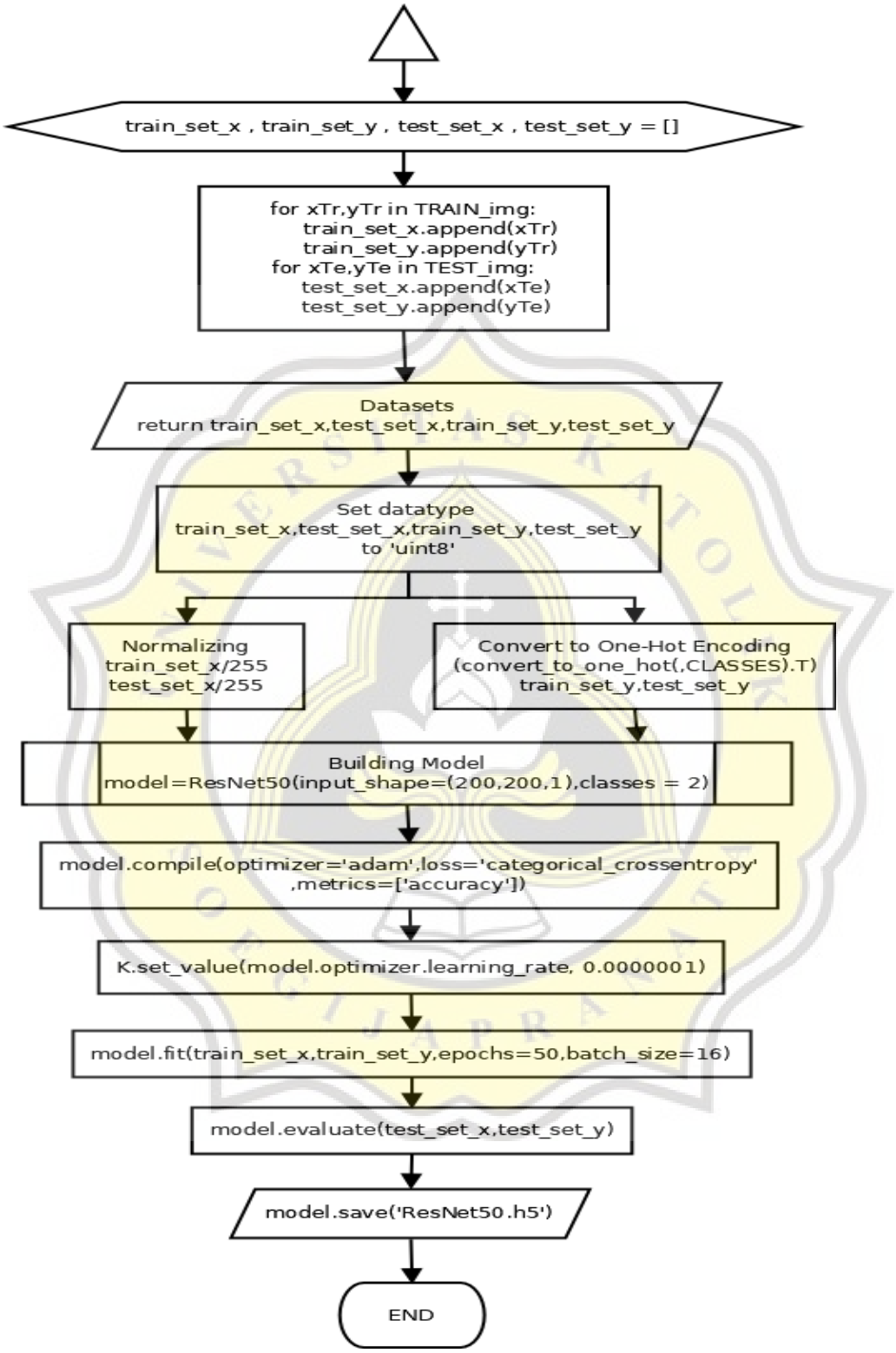

Figure 7: Flowchart Of The Proposed Model Part 2

Figure 6 and 7 above show the flowchart of the proposed model. Outside preparing the functions needed for the model , those are the actual flow of the program. First thing that had to be done is calling the loadData function with directory address of image dataset. Where inside the function , a list container for the images and the label as ground truths in dataset , before starting a loop where all of the images in the directory were read, converted to YCbCr image , and then resized into 200x200 image before being processed through the pre-processing or combined feature function and being added into the previously prepared DATA_Image container along with their label. After all of the image have been processed and added into the container , they were then split using train_test_split function into 80% for training and 20% for testing and not to forget 25 random state. After being split into training and test dataset that is TRAIN_img and TEST_img, number of containers were also prepared so the previously split dataset can be split even further now based on their content that is images and the label or ground truth , the second spliting process produce train_set_x, train_set_y, test_set_x, and test_set_y where set_x represent the image and set_y the label.

After the outputs of loadData function were returned , the next process is to prepare them into a form that can be prepared further and can also be processed and calculated by the neural network. To fulfill those requirement , the datsets were turned into arrays with ‘uint8’ data type. And to make the prepared datasets can be more easily processed by the neural network , there are other process that need to be done. For the image dataset or set_x , a normalization need to be done , the process was done by dividing the image arrays by 255 , this was done so that if the neural network need to calculate a number in the array into an even higher number there is nothing to worry about memory constraint or other problem caused by it. While in contrast , set_y or the label datasets were turned into one-hot encoding , this process were done so the beural network can calculate even more loosely , but do note that in some cases one-hot encoding may reduce the performance instead of helping to increase them.

Now the datasets are reasy to be used as learning input for neural network , what left to do is to build the neural network model before fitting the model and evaluate the training result. The ResNet50 model was build with 200x200 size just like how the image were resized previously right before pre-processing and also with 1 channel because only 1 channel were taken when the image were resized and the output will be in the form of 2 classes which are 0 for tempered or forged and 1 for authentic image. Then the model must be compiled so it can be trained , the model use ‘adam’ optimizer , ‘categorical cross_entropy’ as the loss function, and ‘accuracy’ metrics. One more thing needs to be done before fitting process that is also very vital to the fitting process itself , that one preparation left is to change the learning rate of the model. Why changing the learning rate is important? Because learning rate that is too high or too low can interfere the fitting process , too low and the model would not be able to learn the dataset properly and must learn the dataset in crawling pace , and too high and the dataset might just jump over the most optimal solution which result in performance that seems like cannot get any higher or stuck despite the fact that the model could perform better.

After building the model and all of the preparation was made , the model can proceed to the fitting or training process. The inputs for the fitting proccess would be train_set_x and train_set_y as the labels , while the parameter used are 50 training epoch that as said in the previous part should be higher but due to some limitation either by hardware or time 50 training epoch was taken and also 16 for the batch size which means the model will learn 16 input images and summarized and comparing them before going to the next 16 input images. After the fitting process were done , the model should be evaluated to see how the model would perform with unfamiliar input , and obviously the input for the evaluation process are test_set_x and test_set_y. And finally after the model were evaluated , the model that have been trained should be saveed so the model can be used for future use , either predicting inputs or further training in the future.

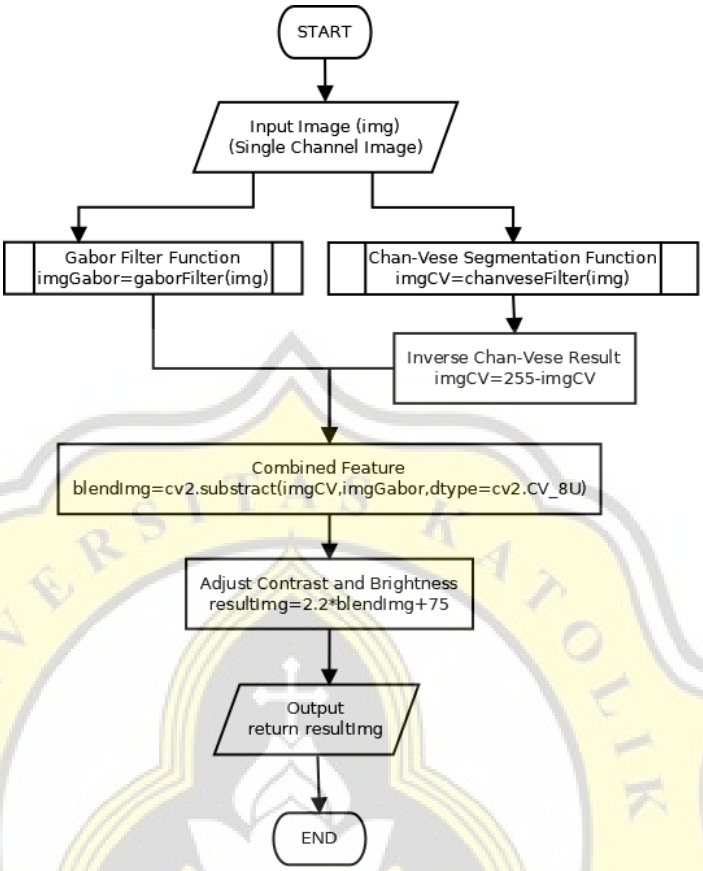

Figure 8: Flowchart Of Combined Feature

To look further down to detail , figure 8 above shows a short and simple flowchart of the process in pre-processing in other words the combined feature. An image with only a single color channel like greyscale or simply an individual color channel was needed as the input combined feature function. The image were then processed through 2 different processing , Gabor filtering function and Chan-Vese segmentation function. Only after the two results were obtained the preocess could continue. But before combining the two results a preparation must be done by inversing the result of the Chan-Vese segmentation , this was done so the result of the combined would not collide and return a result where the only thing can be seen was black color throughout the entire image with no object of interest or anything that can be used to determine whether the image is forged or not. After doing so , the results were combined by using substract function , also as a note do not forget to put data type parameter into the function or the function would not work properly. Finally after the combined feature of the result were made , an adjustment toward brightness and contrast of the combined feature were made. The contrast of

the results were increased by multiplying them by 2.2 and the brightness of the image were increased by 75. These adjustment were done so the neural network would be able to discern the difference easier and more clearly. The adjustment mark the end of the pre-processing and the result of the process was returned.

The two flowchart above explain the flow of the program , from the flow of the main structure and a more detailed of the pre-processing. Though , there are numbers of process that does not mentioned in greater detail in those flowchart , namely the individual pre-processing Gabor filter and Chan-Vese segmentation and also the structure of the neural network ResNet50. The reason why they were not explained in greater detail is because the closely follow the the structure of how they must be build or used especially with ResNet50. As for the indvidual pre-processing the parameter used were not the main focus of project , as it was barely touched and diuscussed but it was a undisputable fact that the parameter of Gabor filter and Chan-Vese segmentation are important for the result performance of the model.