CHAPTER 4
ANALYSIS AND DESIGN

4.1. System Design

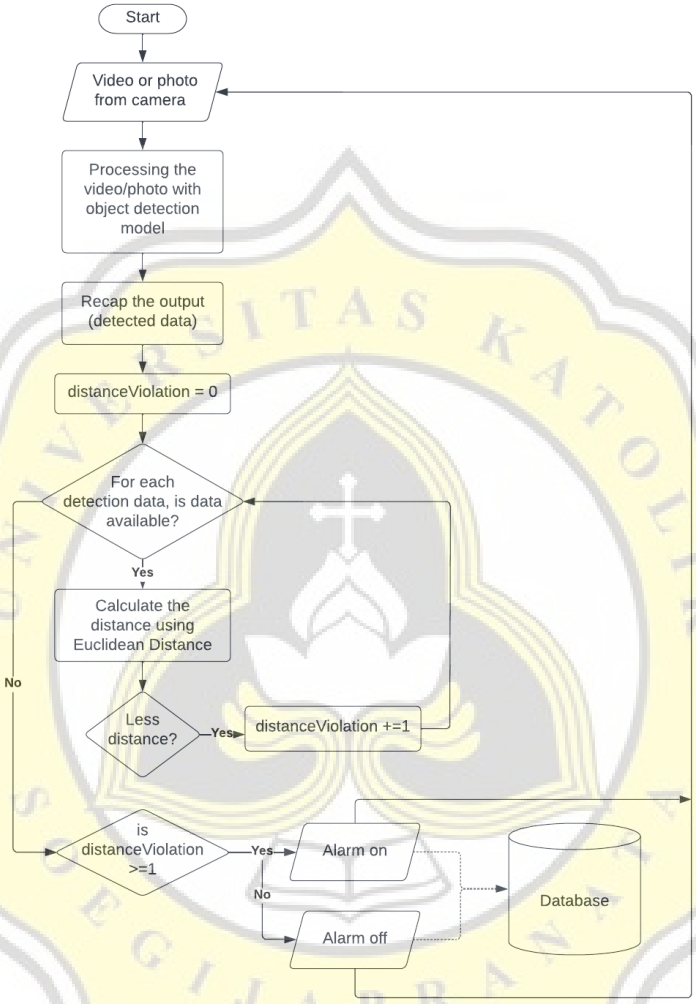

Figure 4.1 System Flowchart

Figure 4.1 displays the system flow using a flowchart. It starts from the input, which is from the video that is taken from the camera, and then, it will be processed with the trained object detection model. This step's output is the detected object's bounding boxes, although it might alternatively be None if the system finds no results.

Then, with the value for variable set to 0, a default variable (example : distanceViolation) is defined to store the total number of distance violations. Next, the system will check to see whether any objects were detected in the previous stage. If any are found,

each data that is detected will be looped over by the system until there data are no longer available.

Then, the distance between the objects in the recent index and the next index will be measured by the system. This research is using Euclidean distance to calculate the distance between two objects. When the object is detected with the bounding boxes, the distance between the two objects will be calculated with the formula of Euclidean Distance (1).

$$(d) = \sqrt[2]{(x2 - x1)^2 + (y2 - y1)^2} \quad (1)$$

(d) is the calculation result of Euclidean Distance, where (x1, x2) and (y1, y2) is the centroid from two objects. In the case where the value of the distance is smaller than the specified value, the distanceViolation variable will be incremented by one value and proceed to the next phase. Otherwise, the system proceeds immediately to the following phases.

After all objects have been processed or when there is no longer any data to process, the system will verify the distanceViolation value. The system will sound the alarm if the value is greater than or equal to 1. Otherwise, the alarm is disabled. Following the upload of the distanceViolation total to the database, the system will repeat the preceding procedure indefinitely (loop) for 24 hours.

4.2. Collecting Data

The dataset for training the model is collected by visiting the Kaggle website and search for face-mask-detection by andrewmvd. It will download a zip file containing the dataset and after it is extracted, the dataset is split into 80% training data and 20% testing data.

4.3. Pre-processing Data

To clean image data for model input, image preprocessing is necessary. Additionally, it could quicken the inference of the model and shorten its training time. If the input images size are huge, reducing its size can speed up model training time greatly without significantly affecting the model performance.

In this research, the first step for preprocessing is to do image labeling to mark the area that will be the object of detection. Image labeling aids in the training of machine learning models to label complete pictures or to identify classes of items within an image. For this research, the labeling process was carried out to detect people's faces. The labeling process is carried out using the labelImg tool which is run using the Python command. The result of the

labeling is a file with the extension .XML. Figure 4.2 shows about how the image labeling is done.

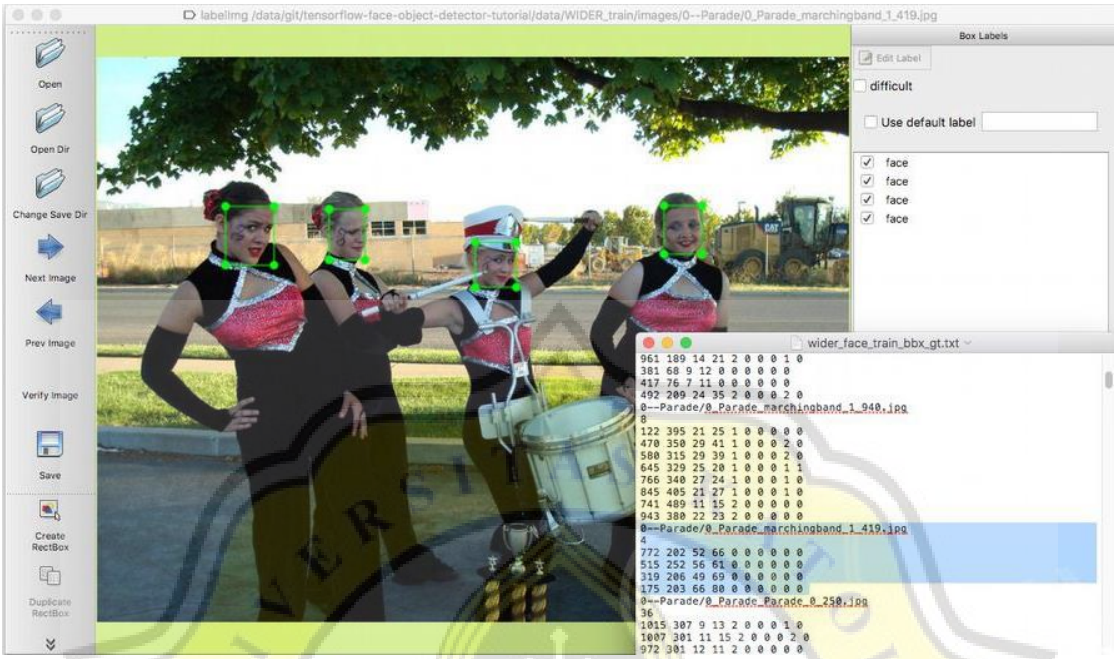

Figure 4.2 Image Labeling

Secondly, a label map is created to be used as a reference in the algorithm for classification with a file extension of .pbtxt. Thirdly, the TFRecord files are created to store binary sequences for files in the training and testing data folder so it can be an input in the training process that can be read by the Tensorflow algorithm. These files are generated using generate_tfrecord.py.

4.4. Training Model

The first step is to configure the pipeline.config that contained a training configuration by changing the hyperparameter and some file paths. The pre-trained SSD MobileNet V2 320x320 model for this research, must then be trained in the next step. An innovative depth-wise separable convolution and a lean network are features of the one-stage object identification model SSD MobileNet V2. It is a model that is often used on low-compute devices with great precision. MobileNet is an architecture that has a role in feature extraction and SSD is an architecture that has a role in classifying and localizing detection objects in the image.

There are two blocks in the architecture of MobileNet v2 which can be seen on Figure 4.3, they are stride 1 blocks and stride 2 blocks. There are three layers in each blocks. The first

layer is a 1x1 point-wise layer of convolution with ReLu 6 and batch normalization. The second layer is similar to first but with a 3x3 depth-wise layer of convolution. The third layer is a 1x1 convolution layer with batch normalization. On an SSD, a regular convolution is performed with strides 1 and 2 to get the detected object with the bounding box and the class name.

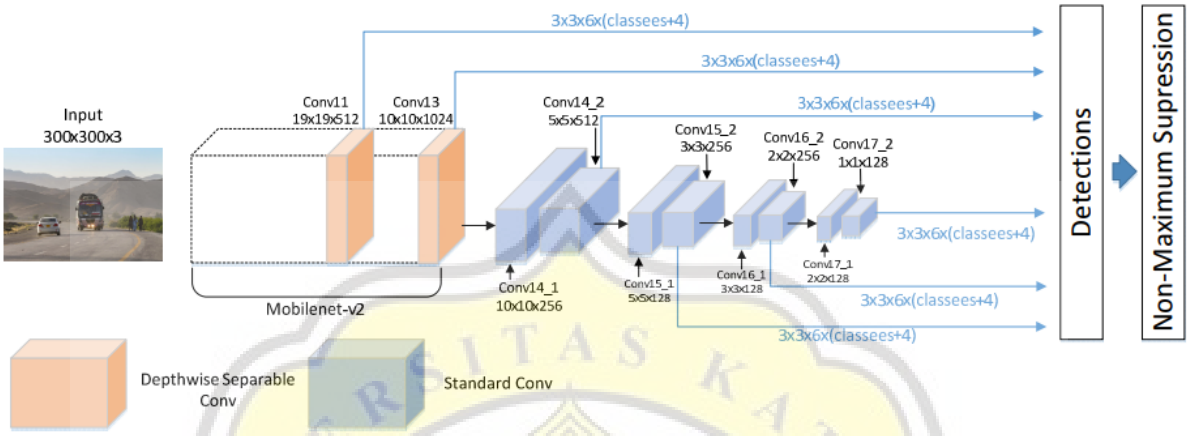

Figure 4.3 SSD MobileNet V2 Architecture [10]

The training process can be started by running a command that generates the model_main_tf2.py along with the pipeline.config. After the training phase is complete, several files are formed to be used for evaluation. The files are events files that contain information on training activities (such as loss) and checkpoint files (training results for each trial). After the training results are obtained, the file is exported using exporter_main_v2.py to save the model for evaluation.

4.5. Hardware Design

This research uses NVIDIA Jetson Nano because it can process several sensors in parallel and execute many contemporary neural networks on each sensor stream, and it supports high-resolution sensors. Additionally, it supports a large number of well-liked AI frameworks, which makes it simple for developers to incorporate their desired models and frameworks into the final result.

The camera is used to capture images that will be the input to the NVIDIA Jetson Nano. A MicroSD card that captures the image of a high resolution for better detection purposes is also needed with a capacity of 64 GB. The MicroSD required more capacity than the recommended 32 GB, because it is used to store images. The device will also be using a Wifi Adapter to capture the Wifi signals and help connect the device to the network wirelessly, the

Wifi adapter needs to be a high gain adapter with a wireless bandwidth of up to 150Mbps, allowing for video streaming.

The tools that are needed can be seen on Figure 4.4. These tools are put together in such a way as to form a unified whole.

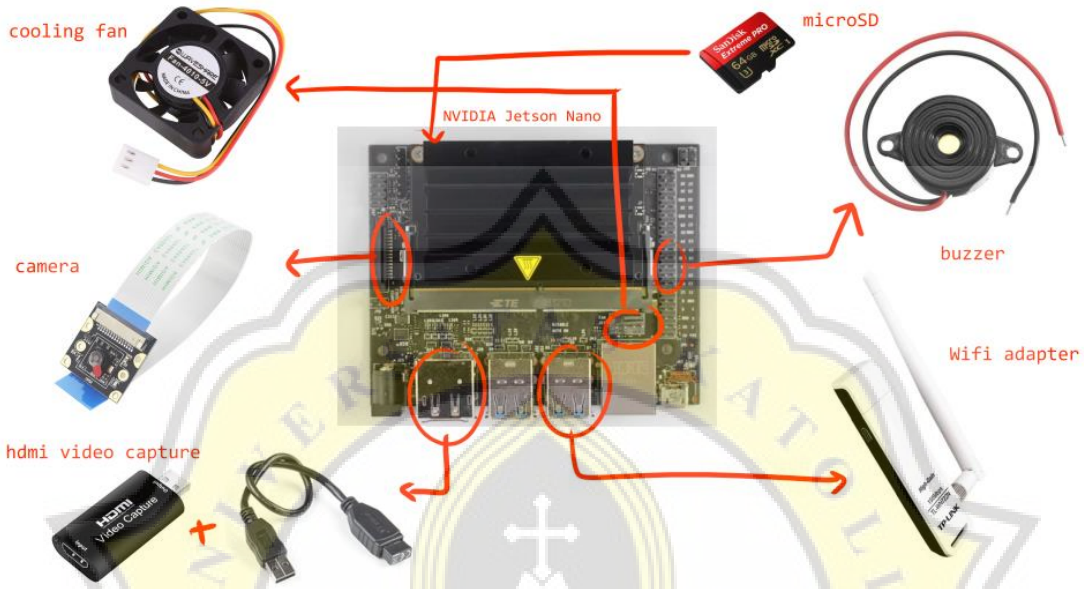

Figure 4.4 Hardware Design

4.6. Implementation

When the model is already exported and the hardware is already prepared, the next step is to code based on the design system that has been created. The previously trained model is then loaded into the code. After the coding is prepared, then, the NVIDIA Jetson Nano is connected to the laptop using HDMI Video Capture and via a Wifi network connection. Next, the code is run after the two are successfully connected.

In this research, the position of the camera on the NVIDIA Jetson Nano is at a height of approximately 75-79 centimeters from the floor because it is put on a table, next to the laptop, with the camera angle of neutral, facing straight on with the subject to improve the detection since the object is clear in that position.

4.7. Evaluation

The evaluation process for this detection system is going to use 10 pictures and put them in front of the camera. In a table, the actual number of violations and non-violations is listed together with the number of violations and non-violations that the system detected from

each image. Then from that data, the performance of the model is obtained by calculating the values of precision, recall, and accuracy.

Precision (1) aims to demonstrate how accurately the machine learning model categorizes the model as being positive. It is calculated by dividing the number of the objects identified correctly as positive (True Positive) by the number of all that is positive (both correctly and incorrectly). Recall (2) is to score the model's ability to identify positive samples. It is defined by dividing the true positives by anything that should have been predicted as positive. Accuracy (3) is to measure the number of predictions that a model made correctly in relation to the total number of predictions made. It is computed by dividing the correct predictions by the total predictions.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

TP (True Positive) is when violators are detected as violators correctly. FP (False Positive) is when non-violators are detected as violators. TN (True Negative) is when non-violators are detected as non-violators correctly. FN (False Negative) is when violators are detected as non-violators.