CHAPTER 3
RESEARCH METHODOLOGY

This chapter describes in detail the steps taken on this project until later in the end find results that match what is done. This research stage discusses the workings of the system developed in this project. Here are some steps to take find the right and correct results.

3.1. Research Process

For the first stage that must be done in this research so that the system built in accordance with the purpose of the study is to correctly predict which leaves have disease and which leaves are normal related to the dataset that has been collected as input.

In this project, the first process that must be done is:

1. Formulate backgrounds, objectives, scope, and problem formulation.
2. Research articles or journals related to topics conducted in this project.
3. Collecting datasets that match the project created, study the algorithms used, and find ways to be able to implement the algorithms used.
4. Analyze the problem and find a solution to the previously mentioned problem, after that build a design that is suitable for the project created.
5. Implementation and analyze the results of the implementation and then make conclusions.

3.2. Collecting Datasets

The dataset used in this project is taken from the *kaggle.com*[1] website, namely PlantVillage Dataset. In this case, the processed dataset will be converted into an array form so that it can be read by the program and the results can be used to detect whether the dataset entered is appropriate or not. The dataset used for the sample was potatoes and tomatoes with a total number of samples used which was 6,652 samples and both were divided into 3 classes, namely Healthy, Early Blight, and Late Blight with the number of samples in each class was different. As shown in the table below.

[1] https://www.kaggle.com/datasets/emmarex/plantdisease

Table 3.1. Samples of Dataset

	Healthy	Early Blight	Late Blight
Potatoes	152	1000	1000
Tomatoes	1591	1000	1909

3.3. Base of Convolutional Neural Network

A Convolutional Neural Network has three layers: a convolutional layer, a pooling layer, and a fully connected layer. Convolutional Neural Network is very useful when working with images. Which scans images from top to bottom and left to right to extract essential features from the image and combine the extracted features to identify the images. It can oversee pictures that have been interpreted, turned, scaled, and alter in viewpoint.

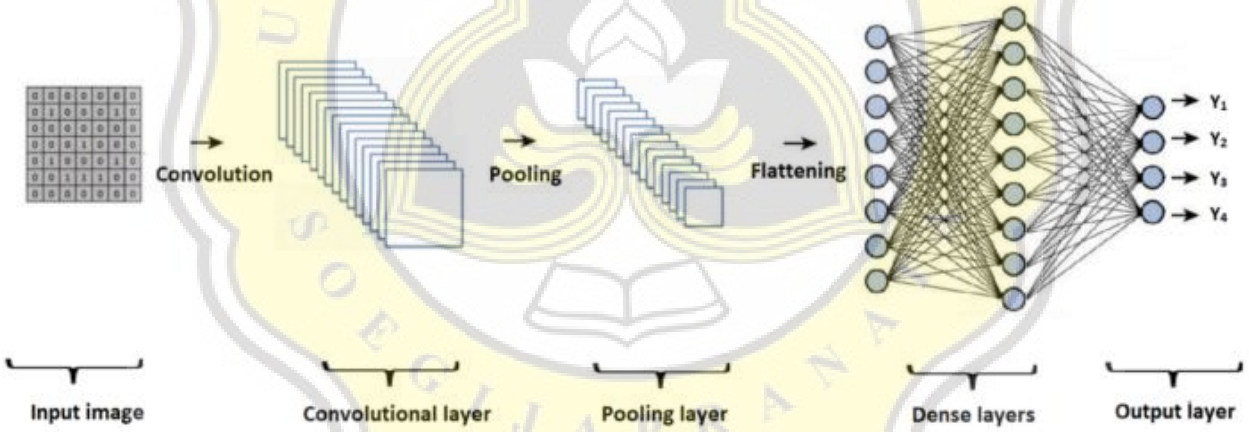

Figure 3.1 : CNN Architecture

3.3.1 Convolution Layer

Convolutional layer produces an activation map by scanning the pictures several pixels at a time using a filter. An example of an input image | 7x7 convolved with a filter k 3x3 with weights of zeros and ones to encode a representation (feature map). The receptive field is highlighted in pink and the corresponding output value for the position is marked in green.

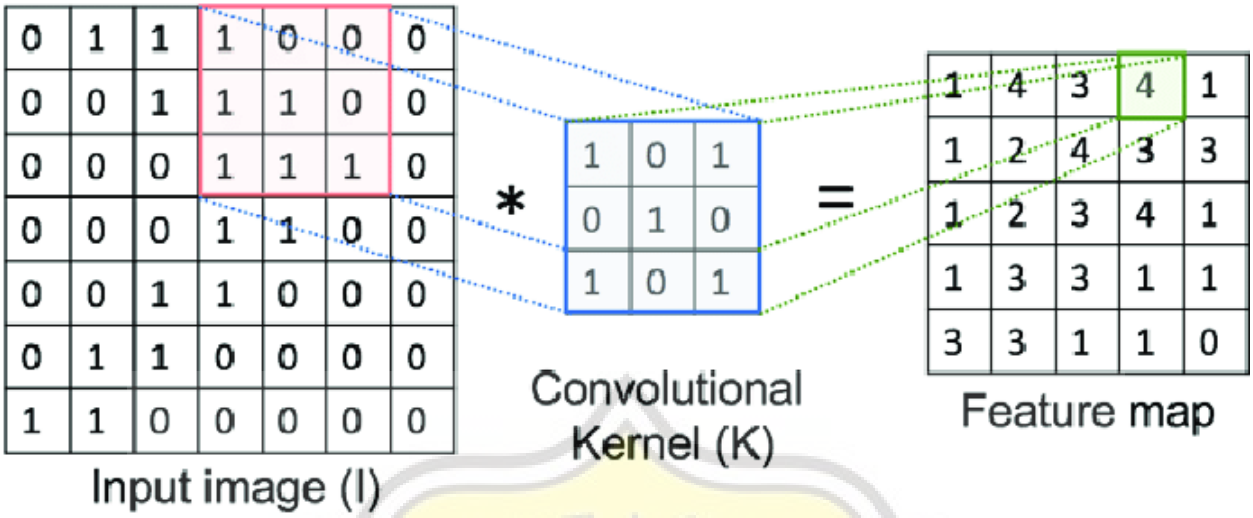

Figure 3.2 : Convolution Layer

3.3.2. Pooling Layer

Pooling layer reduces the amount of data created by the convolutional layer so that it is stored more efficiently. Consists of a filter with a certain size and stride that will shift on the entire activation map. The pooling used is max and average pooling. For example, we use max pooling 2 x 2 with stride 2, then in each filter shift, the maximum value in the 2 x 2 pixel area will be selected, while the average pooling will choose the average value.

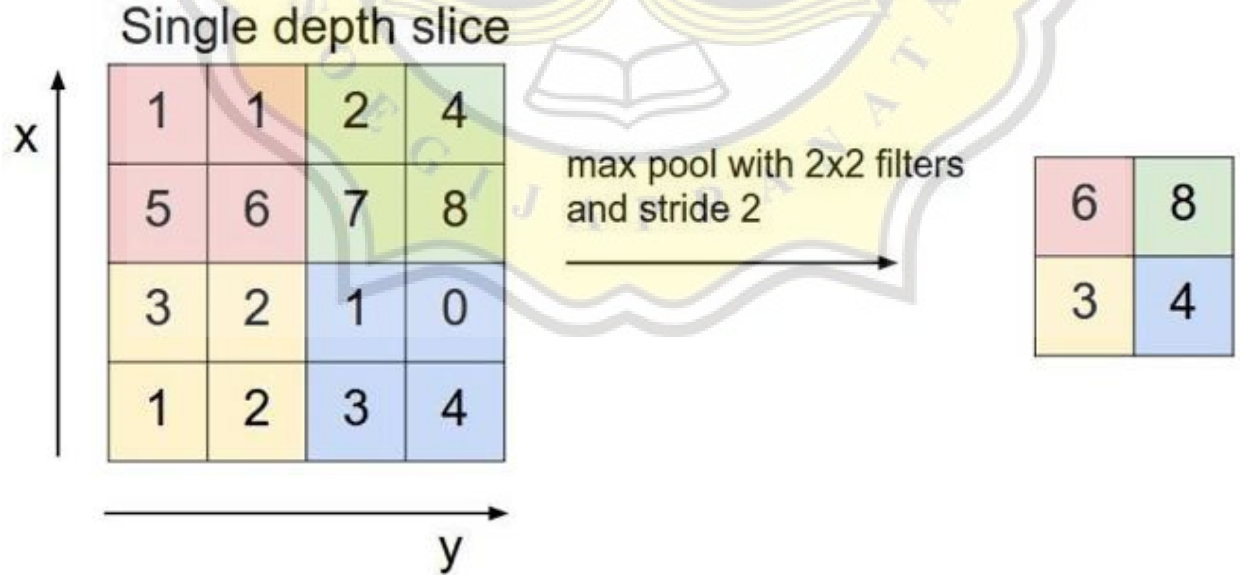

Figure 3.3 : Pooling Layer

3.3.3. Fully Connected Layer

Fully connected input layer – The preceding layers' output is "flattened" and turned into a single vector which is used as an input for the next stage.

The first fully connected layer – adds weights to the inputs from the feature analysis to anticipate the proper label.

Fully connected output layer – offers the probability for each label in the end.

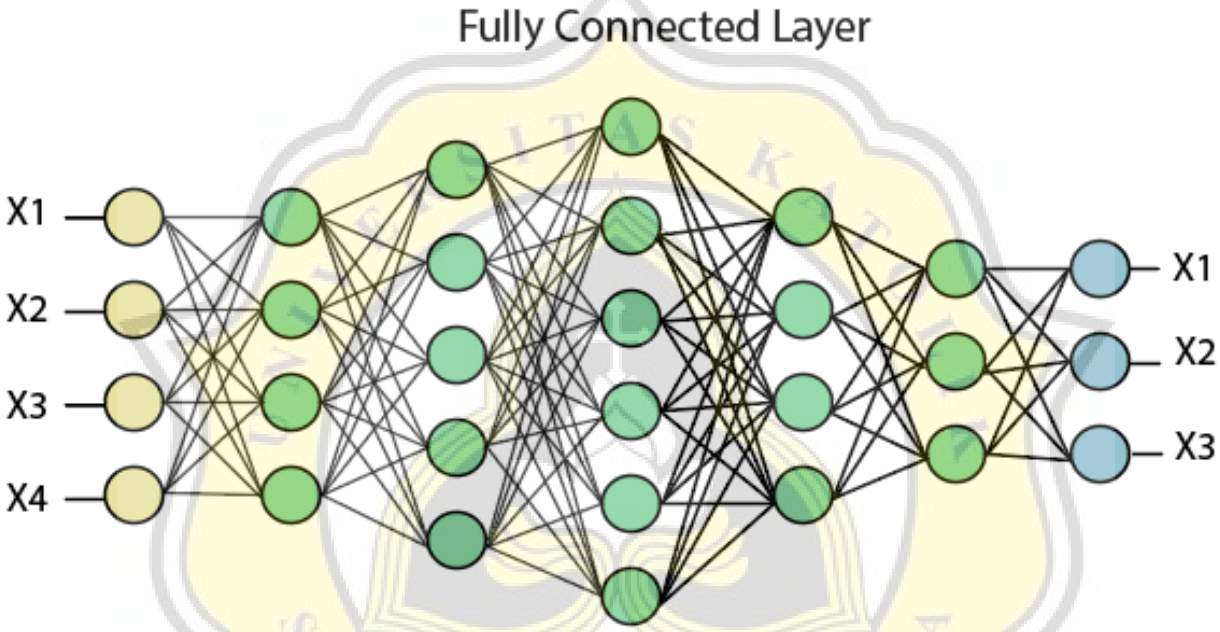

Figure 3.4 : Fully Connected Layer

In the above diagram, the feature map matrix will be converted into the vector such as x1, x2, x3... xn with the help of fully connected layers. We will combine features to create a model and apply the activation function such as softmax or sigmoid to classify the outputs.

3.4. MobileNet Architecture of Convolutional Neural Network

MobileNet is a convolutional neural network, which serves as a deep learning algorithm that is primarily used to classify images. MobileNet is the result of a deep learning research in computer vision which attempts to come up with models that can be run in embedded systems. In order to reduce the number of parameters, MobileNet introduced depth-wise convolutions. The

second iteration of MobileNet implements a system known as “Inverted Residual Block” to help improving the performance of the model. The main goal of MobileNet is to ensure the architecture of the network is streamlined and balanced in terms of latency and accuracy.

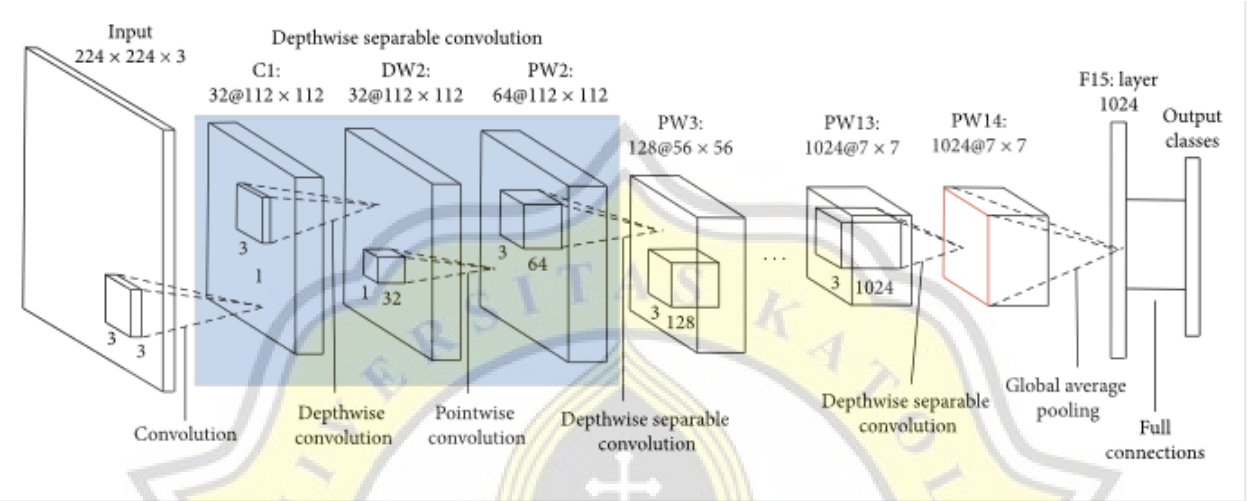

Figure 3.1 : MobileNet Architecture

The secret of MobileNet’s lightweight yet still competitive performance is in its special pooling layer which implements ‘Lite R-ASPP’ (Lite Reduced Atreus Spatial Pyramid). This system deploys the global-average pooling in similar manner compared to Squeeze and Excitation module, in which the model uses a large pooling kernel with large stride to save computational power, and only one 1x1 convolution in the model. The model applies atrous convolution to the last block of its neural network to extract denser features, and add a skip connection from low-level features to capture detailed information.

3.5. Splitting Data Into 80% Data Training and 20% Data Testing

Train/test split is one of the methods that can be used to evaluate the performance of machine learning models. This model evaluation method divides the dataset into two parts, namely the part used for data training and for testing data with a certain proportion. Train data is used to fit machine learning models, while test data is used to evaluate the fit results of the model.

Evaluation of machine learning models with train/test splits is suitable for large datasets. As we know, train/test split divides datasets into train sets and test sets, or in other words, the data used for the training and testing process is a different set of data.

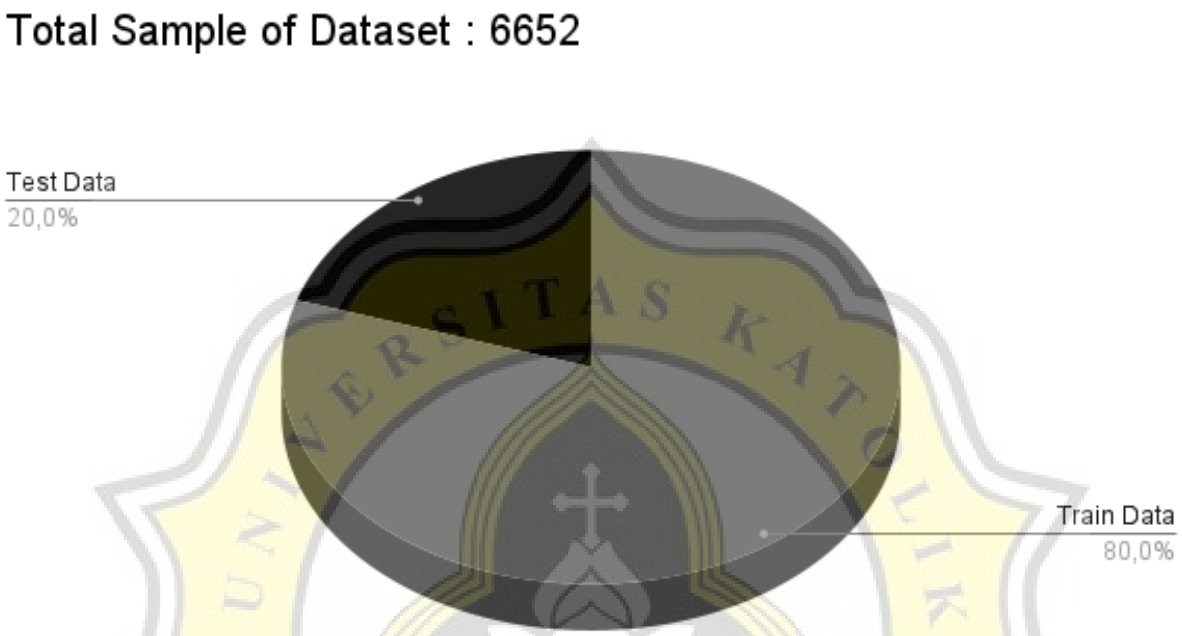

Figure 3.1 Dataset Split Chart

Because the testing data is not used to train the model, the model does not know the outcome of the data. This is called out-of-sample testing. A model is said to be good if it has high accuracy or is good for out-of-sample data, because the main purpose of creating a model is of course to correctly predict data whose outcome is not yet known. In the experiments carried out the dataset used as many as 6652 divided into 80% of training data and 20% of testing data obtained from training data as many as 5321 training data and the rest into testing data.

3.6. K-Fold Cross Validation

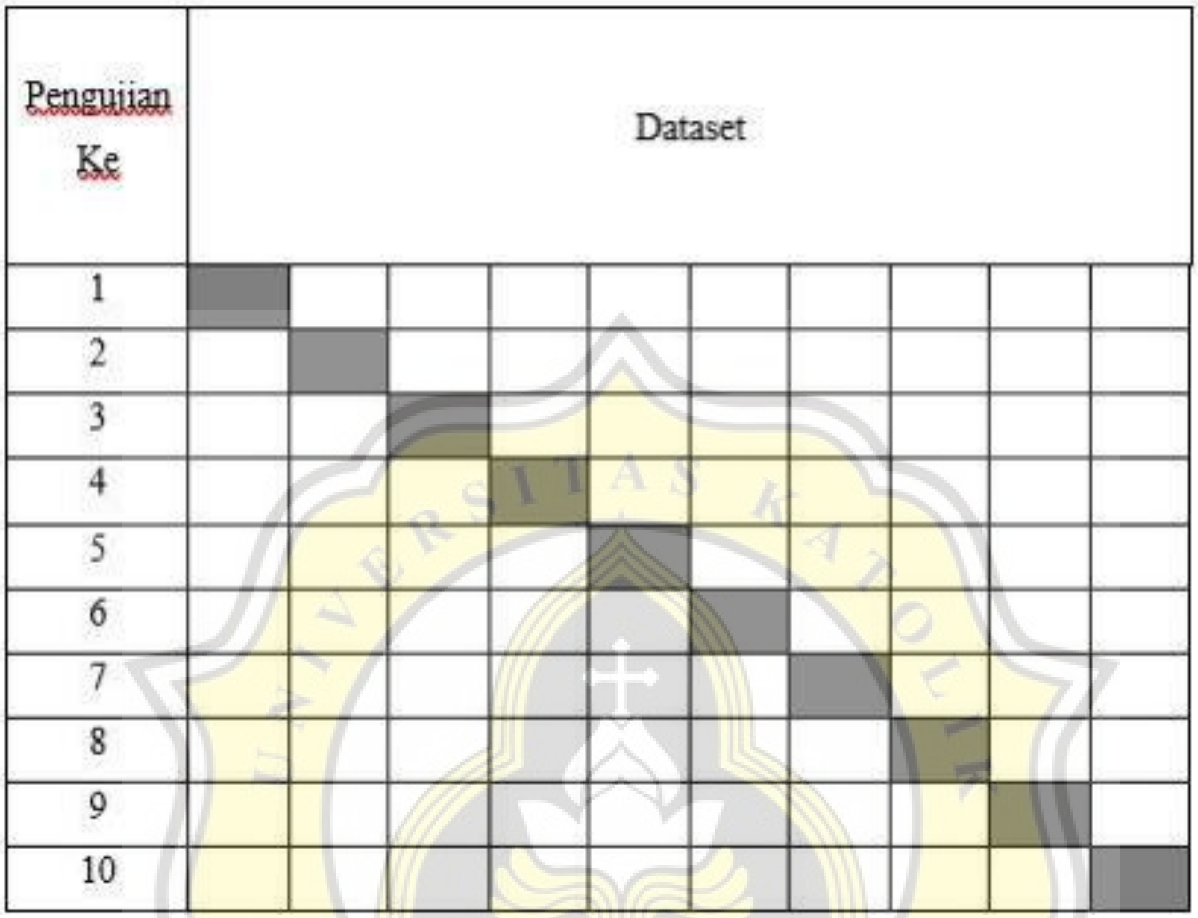

Figure 3.1 : 10 Fold-Cross Validation Illustration

In figure 3.1 the dark part of 10-Fold Validation represents training data and the light part of 10-Fold Validation represents testing data. This method can be used to measure different algorithm parameters, and the selection of several predetermined models, but K-Fold Validation has the disadvantage that performance estimates have a small sample.

The stage to perform K-Fold Validation is as follows, the existing data is divided into segments that are the same size or close, Furthermore, training data for iteration or repetition and validation data is carried out, so that in each iteration different data folds are needed for validation while the remaining k1 folds are used for learning. The data is divided into levels before performing the K-fold process, rearranging the data for each data fold is an appropriate representative of the overall existing data. So every fold that will be done is represented by the

existing overall data, each flip that is done is the best data that has been given based on the percentage of data representatives.