CHAPTER 4
ANALYSIS AND DESIGN

4.1. Analysis

One of this project's goals is to find the performance of the DNN and PCA algorithms in terms of accuracy and speed. The analysis process involves the following steps :

1. Fetch the ORL faces dataset using sklearn and split it into training and test sets
2. Create the DNN model; this model uses ReLU and softmax as its activation functions
3. Train the dataset using the model that has been compiled
4. Calculate and analyze the accuracy of the DNN algorithm with the given circumstance
5. The dataset used for the PCA algorithm is in the image format, so we need to convert them into the ndarray data type
6. Calculate the mean face of the dataset. The mean face is the average feature of the datasets
7. After that, find the normalized faces by subtracting each face in the dataset by the mean face. This normalized face is an extracted or unique feature of each face
8. Calculate the eigenvector using the covariance matrix of the normalized faces matrix
9. Select best eigenvector as much as K, where K is less than the total training images and can represent the whole training set
10. Convert lower-dimensional K eigenvectors to the original face dimensionality
11. Calculate weight vector for each face
12. Calculate the distance between the weight of each input vector and all the weight vectors of the training set
13. Calculate and analyze the accuracy of the PCA algorithm under the given circumstance
14. Compare the accuracy between the DNN and the PCA algorithms

4.2. Design

Python is used as the primary computing language for this project. The reasons for choosing Python are because of its one of the most accessible programming languages available. It has simplified syntax, which gives more emphasis on natural language. Due to its ease and various tools, Python codes can be easily written and executed faster than the other programming languages.

The code of this project is run in the Google Colaboratory or Google Colab. Google Colab is a hosted Jupyter notebook that does not requires setup. The reason of using Google Colab is because it is an excellent tool that provides free access to Google computing resources such as Tensor Processing Unit or TPUs and Graphical Processing Unit or abbreviated as GPUs.

The dataset used for the PCA algorithm is stored in Google Drive, while the dataset used for the DNN algorithm is directly fetched using the sklearn library.

As the first step, we're going to implement the DNN algorithm using the Keras library. Keras is one example of a deep learning API written in Python, running on TensorFlow's machine learning platform. It was developed with a focus on enabling fast experimentation. The core data structures of Keras are layers and models. The simplest type of the model is a linear stack of layers called the Sequential Model.

Before starting the learning process, we need to fetch the dataset using sklearn datasets and split it into a train and test set using the sklearn model selection. Preprocess the train and test set with the help of sklearn preprocessing, which standardizes a dataset along any axis. The next step of the process is to create the model. The model used is a sequential model consisting of 9 layers with softmax and ReLU as the activation functions. Compile the model with the RMSprop optimizer. RMSprop is a gradient-based optimization technique used in neural networks training. Gradients in a very complicated functions, such as neural networks tend to explode or may disappear as the data is propagating through a function. RMSprop addresses this issue by normalizing the gradient using a moving average of the quadratic gradient. After the model is compiled, start the training and evaluate the loss and accuracy of the model using the test data.

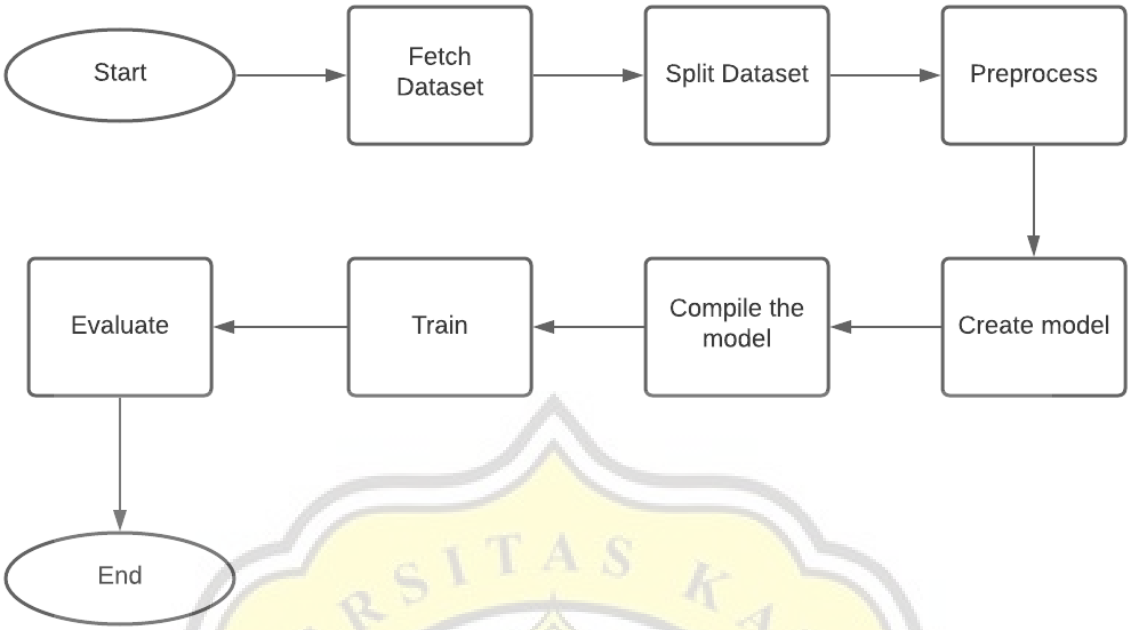

Figure 4.1: DNN Implementation Flowchart

Next, implement the PCA algorithm. The image dataset is stored on Google Drive, so we need to integrate the Google Colab by mounting the Google Drive. Convert all of the image datasets into ndarray using the NumPy library. NumPy is a Python library used to manipulate arrays. We need to find the mean face of the dataset to remove all the dataset's common features. By subtracting each face image from the mean face, we get each image's unique feature, called normalized images. PCA is done by decompose the covariance matrix, so compute the covariance of the normalized images and find the eigenfaces. Then, find a K number of significant eigenface that can represent the whole training set. They must not leave out any important information about the data that we have. Then, calculate the weight of each eigenface. These weights are the proportions of each eigenface to make up each person's face in the dataset. Then we analyze the accuracy by giving the algorithm a test set.

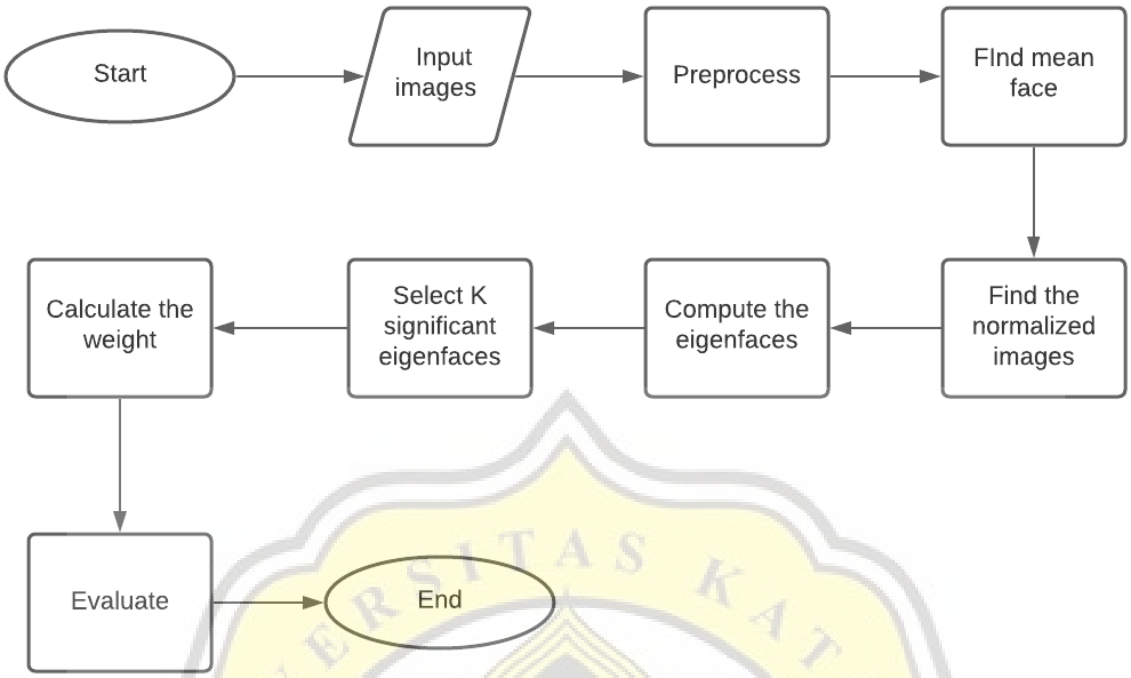

Figure 4.2: PCA Implementation Flowchart

Now, after we implement the DNN and PCA algorithm, compare and analyze these two algorithms in terms of time and accuracy in given circumstances.