Minor Project Report

on

**READ-HAND**

Static Hand Gesture Recognition

Submitted in partial fulfilment of the requirements for the award of degree of

**Bachelor of Technology**

by

**Shubham Kachroo (CE-4121-2K16)**

**Sarthak Shah (CE-4133-2K16)**

Under supervision of

**Shikha Gupta**



**Department of Computer Engineering**

**J. C. BOSE UNIVERSITY OF SCIENCE & TECHNOLOGY, YMCA**

**FARIDABAD-121006**

**May 2019**

**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in this project report titled of **“READ-HAND”** in fulfilment of the requirement for the degree of **Bachelor of Technology** and submitted to “**J. C. Bose** **University of Science and Technology, YMCA, Faridabad**”*,* is an authentic record of my own work carried out under the supervision of Shikha Gupta.

The work contained in this report has not been submitted to any other University or Institute for the award of any other degree or diploma by me.

**Shubham Kachroo (CE-4121-2K16)**

**Sarthak Shah (CE-4133-2K16)**

**CERTIFICATE**

This is to certify that the project report titled **“READ-HAND”** submitted by **Shubham Kachroo and Sarthak Shah** to “**J. C. Bose** **University of Science and Technology, YMCA, Faridabad**” for the award of the degree of Bachelor of Technology is a record of bonafide work carried out by them under my supervision. In my opinion, the work has reached the standards of fulfilling the requirements of the regulations to the degree.

Shikha Gupta

(Supervisor)

Assistant Professor

Department of Computer Engineering

J. C. Bose University of Science and Technology, YMCA, Faridabad

**INDEX**

***Chapter***

1. *Brief overview of Project* 
   1. *Introduction*
   2. *Problem Statement*
   3. *Motivation*
   4. *Steps followed in the project*
   5. *Technology Stack*
   6. *Objective and Scope*
   7. *Software and Hardware requirements*
2. *Proposed Methodology*
3. *Introduction to various technologies used in project.*
4. *Guidelines to use the application.*
5. *Code Snippet.*
6. *Conclusion and Results.*
7. *References.*

**Chapter 1**

**Brief overview of Project**

* 1. **Introduction**

Gesture recognition is an open problem in the area of machine vision, a field of computer science that enables systems to emulate human vision.

Gesture recognition has many applications in improving human-computer interaction, and one of them is in the field of Sign Language Translation, wherein a video sequence of symbolic hand gestures is translated into natural language.

**Read-Hand** is an application which uses machine learning algorithm for image classification to classify the captured input image area block to deduce the output sign from a given image dataset which the user wants to use and do the required function.

Initially, the dataset is limited to only a few sign hand signs, when we will achieve high accuracy we will shift to different and more complex dataset of more hand signs.

Steps as to how to make this project: -

1. Locate the input hand in the raw image, conversion to its vectorized form and feeding it to the gesture recognizer.

2. Building a binary classifier for detecting the section with the hand and building a multi-class classifier for identifying the gesture using given data sets.

For this project we will be using different kinds of data sets like to identify if the input is hand or not, and identify the hand sign gesture from the input hand sign.

**1.2 Problem Statement**

Gesture recognition is an open problem in the area of machine vision, a field of computer science that enables systems to emulate human vision.

Read-Hand is an application which uses machine learning algorithm for image classification to classify the captured input image area block to deduce the output sign from a given image dataset which the user wants to use and do the required function.

**1.3 Motivation**

*“The advance of technology is based on making it fit in so that you don't really even notice it, so it’s part of everyday life.”*

-Bill Gates

Our main motive was to make peoples’ life easier and less self-dependent. We want technology to replace the need of complex operations and convert them into simple gestures.

**1.4 Steps followed**

* Selecting the dataset.
* Data preprocessing.
* Identifying the algorithm and building the model.
* Fitting the model to the dataset to achieve accuracy.
* First, selection of proper dataset to work on was the most difficult task since we will test our algorithm on that data, so we need to select and classify the data in the most appropriate manner.

The dataset format is patterned to match closely with the classic MNIST.

Each training and test case represent a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions). The training data (27,455 cases) and test data (7172 cases) are approximately half the size of the standard MNIST but otherwise similar with a header row of label, pixel1, pixel2....pixel784 which represent a single 28x28 pixel image with grayscale values between 0-255.

* Secondly, as the dataset has already given csv values for images, we did not need to do much pre-processing.

If dataset of image was in raw format, we have to convert them in csv format arrays before doing any of the further operations.

Still we perform following steps:

* Separate features (784-pixel columns) and output (result label)
* Reshape the features
* One Hot Encoding on result
* Third, select the algorithm using which we had to classify the image properly and then fit the algorithm on the dataset to achieve the maximum accuracy.

We will use Keras to build the simple **CNN** (Convolutional Neural Network).

There are total **7 layers** in the CNN:

1. 1st Convolutional Layer with relu
2. 1st Max Pooling
3. 2nd Convolutional Layer with relu
4. 2nd Max Pooling
5. Flattening
6. First Full Layer with relu
7. Output Layer with sigmoid

* Fourth, fit the selected algorithm on the dataset and then train the model onto the dataset to achieve the maximum accuracy so as to check the algorithm and also how accurate our project.

**1.4 Technology Stack**

1. Python 3.x
2. Keras with TensorFlow
3. OpenCV 3.4
4. Skicit-learn
5. h5py
6. NumPy
7. Pandas
8. Jupyter Notebook

**1.5 Objectives and Scope**

This project would be focused on the Hand Detection and Classification module where we would perform research on the techniques and methodology to detect and recognise particular gestures. Thus, this project is to come up with a solution that detects hand gesture effectively and record it down. The project will give a prototype to detect and recognize the hand sign gesture with high accuracy.

Scope of this project is mainly for those people who are visually impaired and need special attention for doing numerous daily life tasks.

Our target audience of this project was to address those people who are visually impaired or people who are new to the technology world or people who cannot apply complex operations on computers.

Hand Gesture Detection has a very wide area of applications. People can use this detection project in real life situations like blind people can effectively and efficiently use this hand detection to communicate easily.

**1.6 Software and Hardware requirements**

Software:

* Python
* Jupyter Notebook
* Windows/Linux
* Anaconda Prompt
* Chrome

Hardware:

* CPU

**Chapter 2**

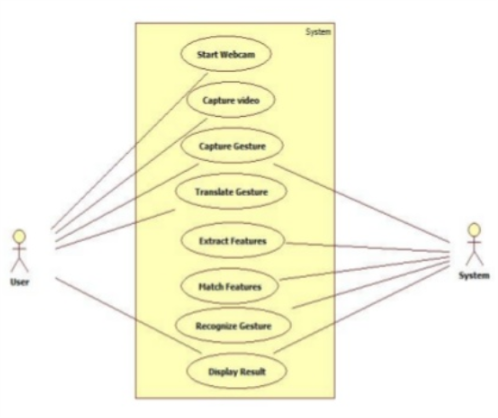
**Proposed Methodology**

**Use case:**

Use case diagram for the proposed solution is given below.

Here, we have two actors and objects where we have to capture the image and then extra features from it.

We match the features, translate the gestures, test it with our dataset and then display the result and show our algorithm accuracy.



**Flow Diagram:**

Below flow diagram is the solution for the proposed project.

In this, we find the sequence which is followed by the project to display the desired output.



**Chapter 3**

**Introduction to various technologies used**

**3.1 Python 3.x**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by Meta programming and Meta objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and acycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter(), map(), and reduce() functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

Beautiful is better than ugly

Explicit is better than implicit

Simple is better than complex

Complex is better than complicated

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a compliment in the Python culture." Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favor of "there should be one—and preferably only one—obvious way to do it".

**2.2 TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

It is used for both research and production at Google. ‍

TensorFlow architecture works in three parts:

1. Pre-processing the data
2. Build the model
3. Train and estimate the model

It is called TensorFlow because it takes input as a multi-dimensional array, also known as tensors.

You can construct a sort of flowchart of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

TensorFlow is the best library of all because it is built to be accessible for everyone. TensorFlow library incorporates different API to build at scale deep learning architecture like CNN or RNN. TensorFlow is based on graph computation; it allows the developer to visualize the construction of the neural network with Tensorboad. This tool is helpful to debug the program. Finally, TensorFlow is built to be deployed at scale. It runs on CPU and GPU.

TensorFlow attracts the largest popularity on GitHub compare to the other deep learning framework.

Among the applications for which TensorFlow is the foundation, are automated image-captioning software, such as DeepDream.

RankBrain now handles a substantial number of search queries, replacing and supplementing traditional static algorithm-based search results.

**2.3 Keras**

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) and its primary author and maintainer is François Chollet, a Google engineer.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

## Advantages of Keras:

* **Fast Deployment and Easy to understand**
* **Large Community Support**
* **Have multiple Backends**
* Cross-Platform and Easy Model Deployment

**2.4 NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**2.5 Pandas**

Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the NumPy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Features:

* Data Frame object for data manipulation with integrated indexing.
* Tools for reading and writing data between in-memory data structures and different file formats.
* Data alignment and integrated handling of missing data.
* Reshaping and pivoting of data sets.
* Label-based slicing, fancy indexing, and sub setting of large data sets.
* Data structure column insertion and deletion.

**2.6 Jupyter Notebook**

Notebook documents (or “notebooks”, all lower case) are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc..) as well as executable documents which can be run to perform data analysis.

A notebook kernel is a “computational engine” that executes the code contained in a Notebook document. The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels).

The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files.

**Chapter 4**

**Guidelines for Using the Project**

Steps to be followed to run the project:

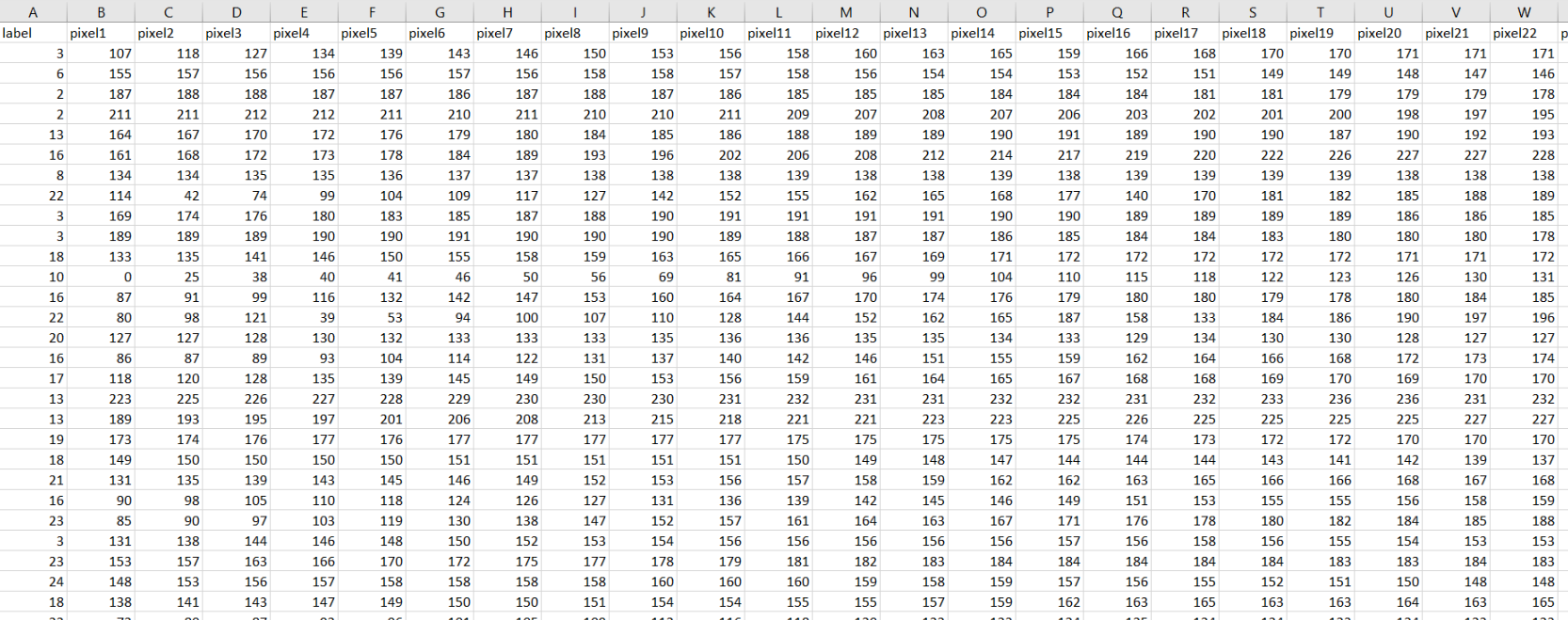
1. Install Python 3.x and Anaconda software on the operating system for Jupyter Notebook and running the project.
2. After installation of Anaconda and Python, open Jupyter Notebook and then browse to the code.ipynb file.
3. Install various modules like TensorFlow, Keras and Skicit-learn using the Anaconda Prompt.
4. After installation of all the required module, we need to simply run the code.ipynb file in the Jupyter Notebook.
5. After a successful run, we will get the accuracy of the algorithm which we implemented to the dataset.
6. We can then apply the algorithm to perform various operations and functions.

**Chapter 5**

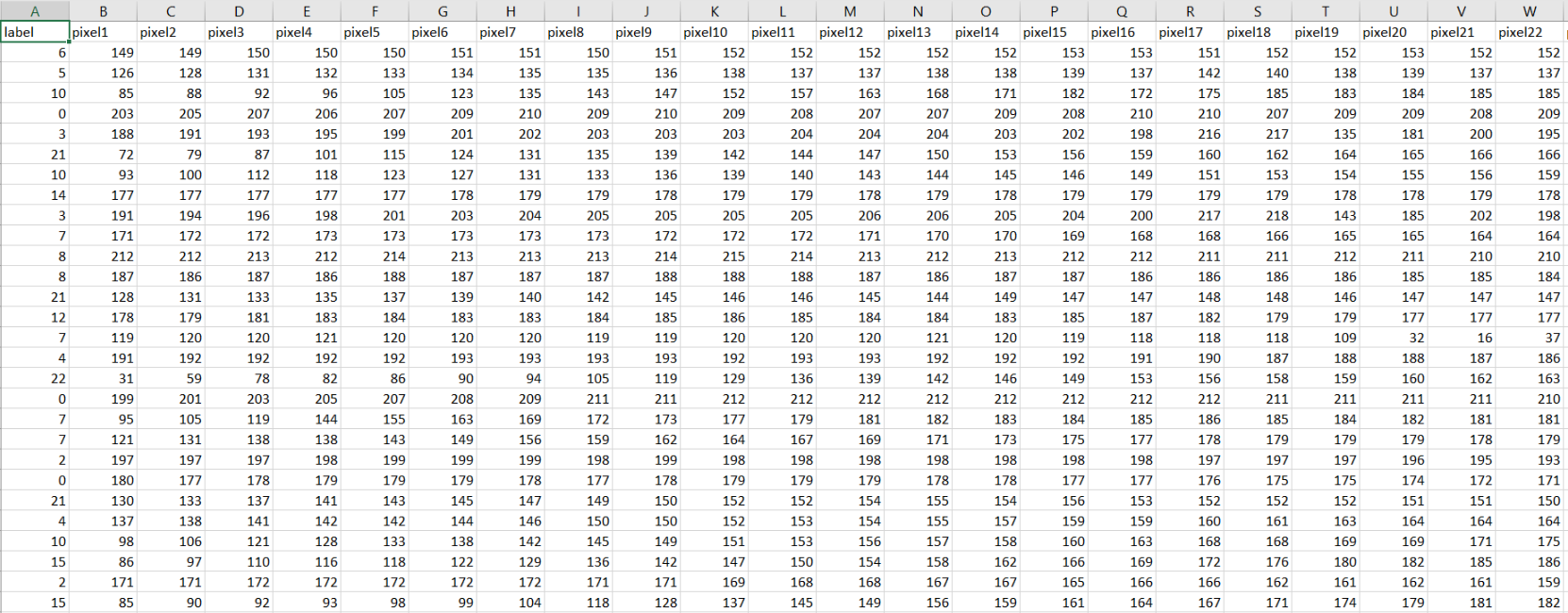
**Code Snippets**

1. **Dataset:**

Train Dataset Screenshot

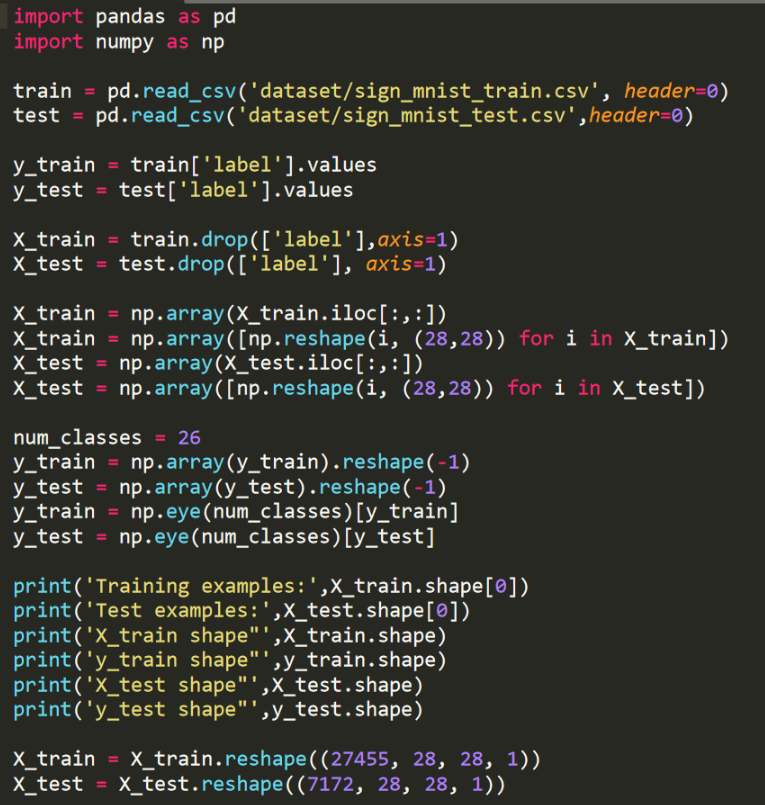


Test Dataset Screenshot



1. **Code Snippets in Sublime:**

Data Pre-processing

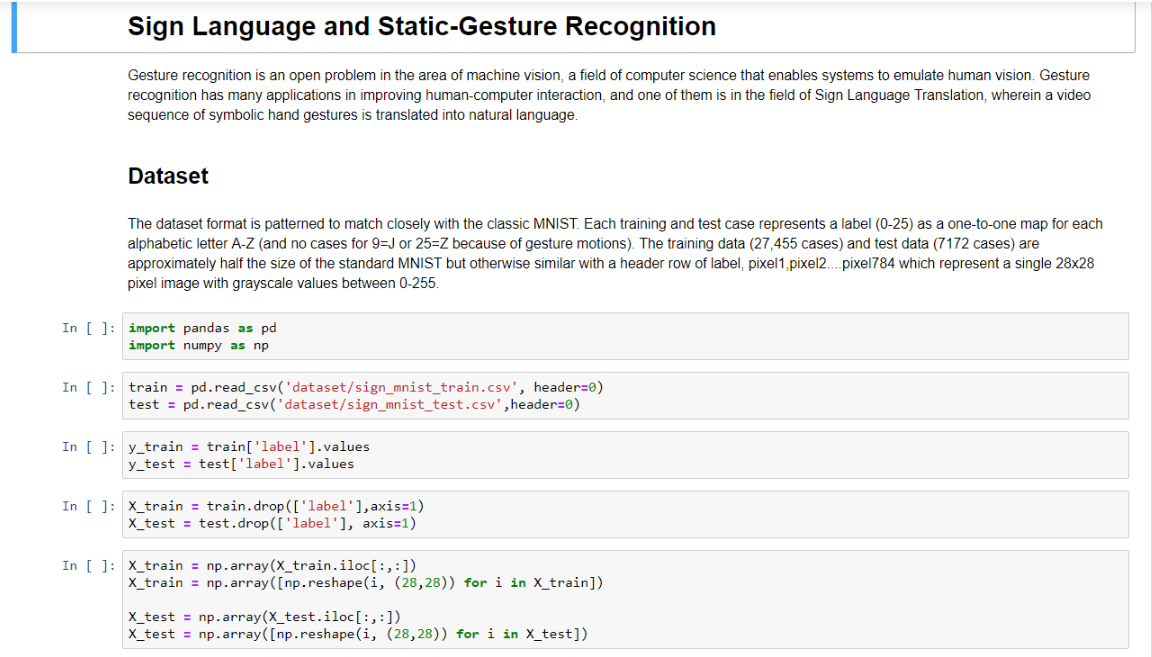


Keras model layering on the dataset



**Code in Jupyter Notebook:**

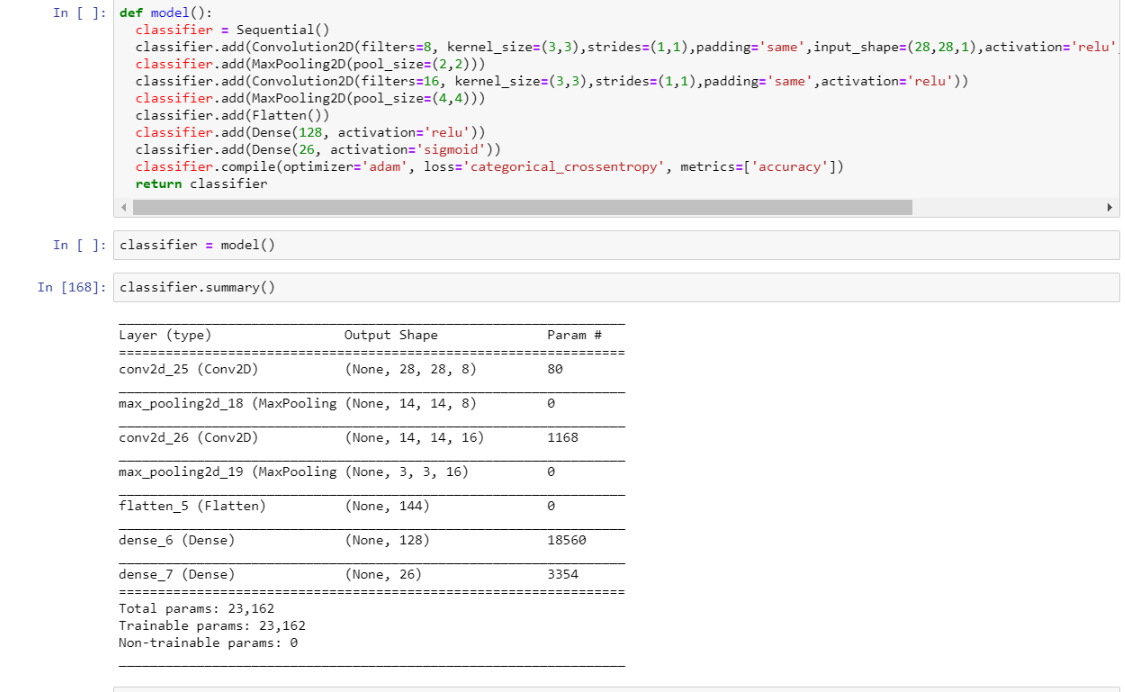
Importing modules, reading the dataset and assigning them in lists.



Reshape the dataset to 28x28 pixel and import modules.



Implementing the CNN model by seven layers and get the model summary in tabular format.



Fitting the model with the dataset and then checking the accuracy of the model with the test dataset.



**Chapter 6**

**Conclusion and Results**

*“The advance of technology is based on making it fit in so that you don't really even notice it, so it’s part of everyday life.”*

-Bill Gates

We will conclude the project by saying that technology can make people tasks and numerous operations easier.

People can utilize technology to benefit their life for performing operations which are complex in nature.

We in this project, achieved various features which are included in the project which are:

* Can detect any kind of gesture which is provided in the dataset.
* Eliminates the background so can be operated in a place where there is no much movement in the background.

***Results:***

*Training Set Accuracy: 96.06 %*

*Test Set Accuracy: 87.77%*

**Chapter 7**

**References**

<https://www.hindawi.com/journals/tswj/2014/267872/>

<https://www.geeksforgeeks.org/project-idea-dynamic-hand-gesture-recognition-using-neural-network/>

<https://github.com/EvilPort2/SimpleGestureRecognition/blob/master/README_cnn.md>

<https://medium.com/@blackbird98/sign-language-and-static-gesture-recognition-eaafd6ab00a5>

<https://ieeexplore.ieee.org/document/6179213>

<https://ieeexplore.ieee.org/document/8368821>

<https://ieeexplore.ieee.org/document/7544882>