Project Report

San Jose State University



Project Report on

Traffic Sign Recognition

Subject: Data Mining(CMPE 255)

M.S in Software Engineering (Semester – 1)

(User Defined Project)

Submitted by:

Team Dragonites

Khushil Modi

(SJSU ID: 015923115)

Sarjak Patel

(SJSU ID:015945046)

Nevil Shah

(SJSU ID: 015964975)

Vishnu Yeruva

(SJSU ID: 015917551)

Guided by:

Prof. Vijay Eranti

Academic Year: 2021-22

**Acknowledgement**

We would like to express our deep sense of gratitude and sincere appreciation to our respected and esteemed Prof. Vijay Eranti,Assistant Professor,Computer Engineering Department, San Jose State University(SJSU) for his constant support and encouragement regarding the project work throughout this semester .

Last but not the least, we express our sincere thanks to all our classmates and friends, who either directly or indirectly, have helped us in some ways in our project.

Yours sincerely,

Khushil Modi

Sarjak Patel

Nevil Shah

Vishnu Yeruva

| **Chapter** | **Title** |
| --- | --- |
| 1 | Introduction |
| 1.1 Project Introduction |
| 1.2 Project Abstract |
| 1.3 Objective |
| 1.4 Approach |
| 2 | CNN Model |
| 2.1 Model Architecture |
| 2.2 Model Training |
| 2.3 Graphs |
| 2.3.1 Loss-Accuracy Graph |
| 3 | System development |
| 3.1 Hardware Requirements |
| 3.2 Software Requirements |
| 3.3 Tools Used |
| 4 | Implementation |
| 4.1 Dataset |
| 4.2 Detection & Classification |
| 4.2.1 Data Preprocessing |
| 4.2.2 Data Visualization |
| 4.2.3 Model Training |
| 4.2.4 Model Testing |
| 4.3 User Interface & Deployment |

**Chapter 1: Introduction**

**Chapter 1.1 Introduction**

Nowadays, more emphasis and research is given on the ability of a car to drive itself on a road. It means it has to make accurate decisions while driving on a road or a highway to avoid collisions. Hence, it is very important to keep the conditions on either side of the road in mind. And the most important characteristic of a driverless car is to detect and recognize various traffic signs on the road for maintaining the safety and security of the people inside and outside the vehicle.

The Roadway system has different characteristics whose principal aim is to regulate and control the flow of traffic smoothly and to ensure that drivers are bound by the rules so as to provide a safe environment to the nearby traffic to avoid road collisions.

**Chapter 1.2 Project Abstract**

Traffic sign Recognition is a software application whose aim is to recognize traffic signs as efficiently as possible. This report tries to solve a major problem of Road Safety which is to classify traffic signs. It also detects and filters the problem with the help of Convolution Neural Networks (CNNs) for image classification and delivers the accurate solution.

Traffic sign recognition software ensures that different types of traffic signs on both sides of the road are visible to the drivers while driving the car. This software can recognize different types of traffic signs like “Stop”,”School Ahead”,”Speed Limit”,”No parking” and many more.

We have designed and developed an application for traffic sign recognition which will play a pivotal role in driverless cars.After providing an image as an input, it recognizes the traffic sign image and gives the description of it based on the model trained and tested using the signs dataset . The model is trained using CNN and Flask framework for Front-End development.

**Chapter 1.3 Objective**

The main goal of this project is to design and develop a web-based application which classifies the type of traffic sign for the image given as input. We are intending a system which is able to recognize the traffic signs on either side of the road to give applicable support to the user or the machine in the driverless cars.

**Chapter 1.4 Approach**

The approach consists of assembling a model with the help of Convolution Neural Networks(CNNs) for image classification by extracting different traffic signs from an image.

**Chapter 2 CNN Model**

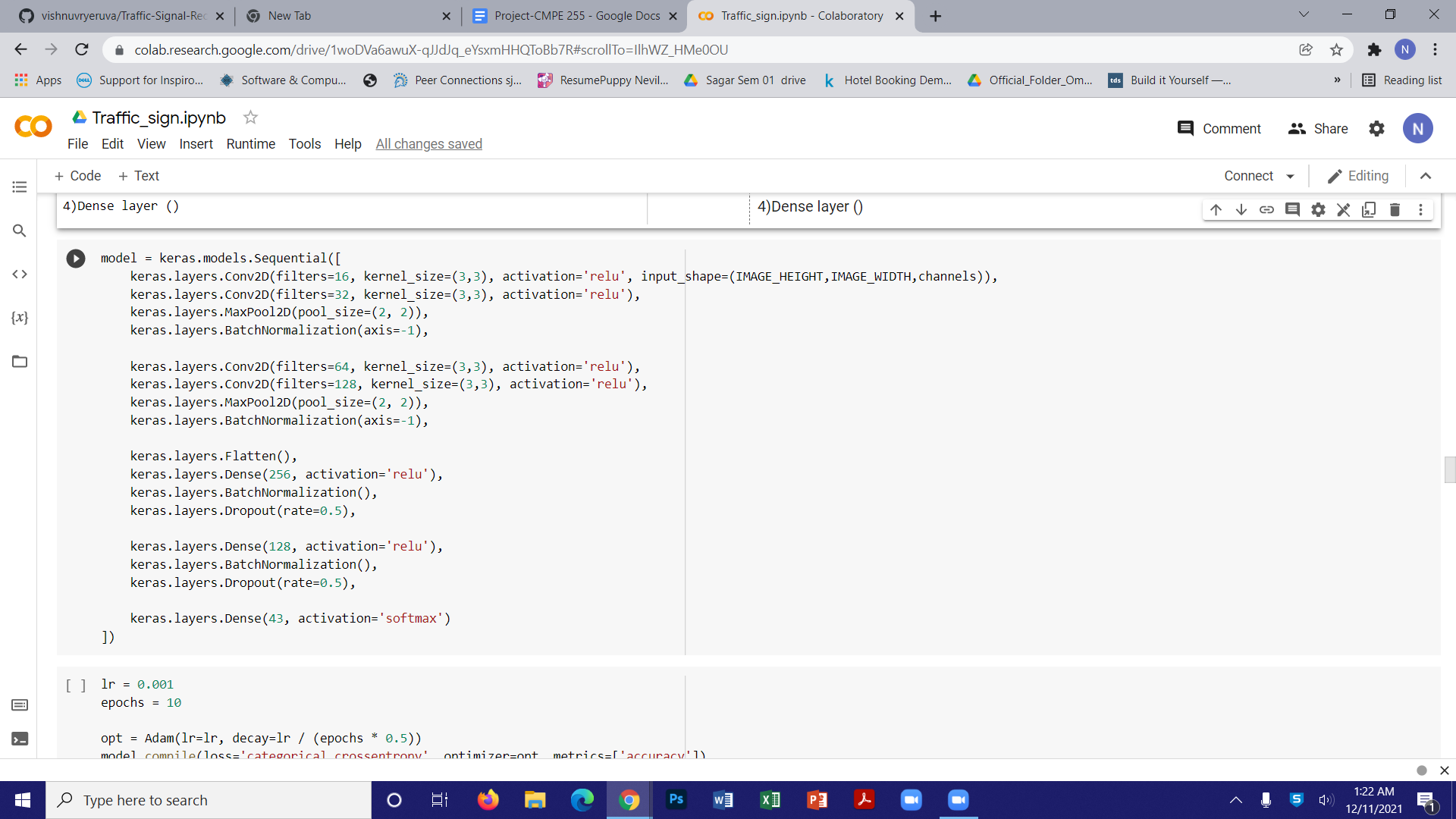
**Chapter 2.1 Model Architecture**

The model is built using Convolution Neural Networks for traffic-sign recognition from images due to its high accuracy. CNNs builds a hierarchical model whose working is like a network of neurons and gives an output as a layer where all the neurons are connected with each other and thereafter, the output is given on the image given as input.

The structure of CNN-built model is as follows :-

* 2D Convolution layer(Output filters = 32, “ReLU” activation function)
* MaxPool 2D layer
* Dropout layer(to prevent overfitting)
* 2D Convolution layer(Output filters = 64, “ReLU” activation function)
* MaxPool 2D layer
* Dropout layer(Probability = 25%)
* Flatten layer (to convert in single dimension)
* Dense Layer(256 nodes, ReLU activation function)
* Dropout layer(Probability = 25%)
* Dense Layer(256 nodes, ReLU activation function)

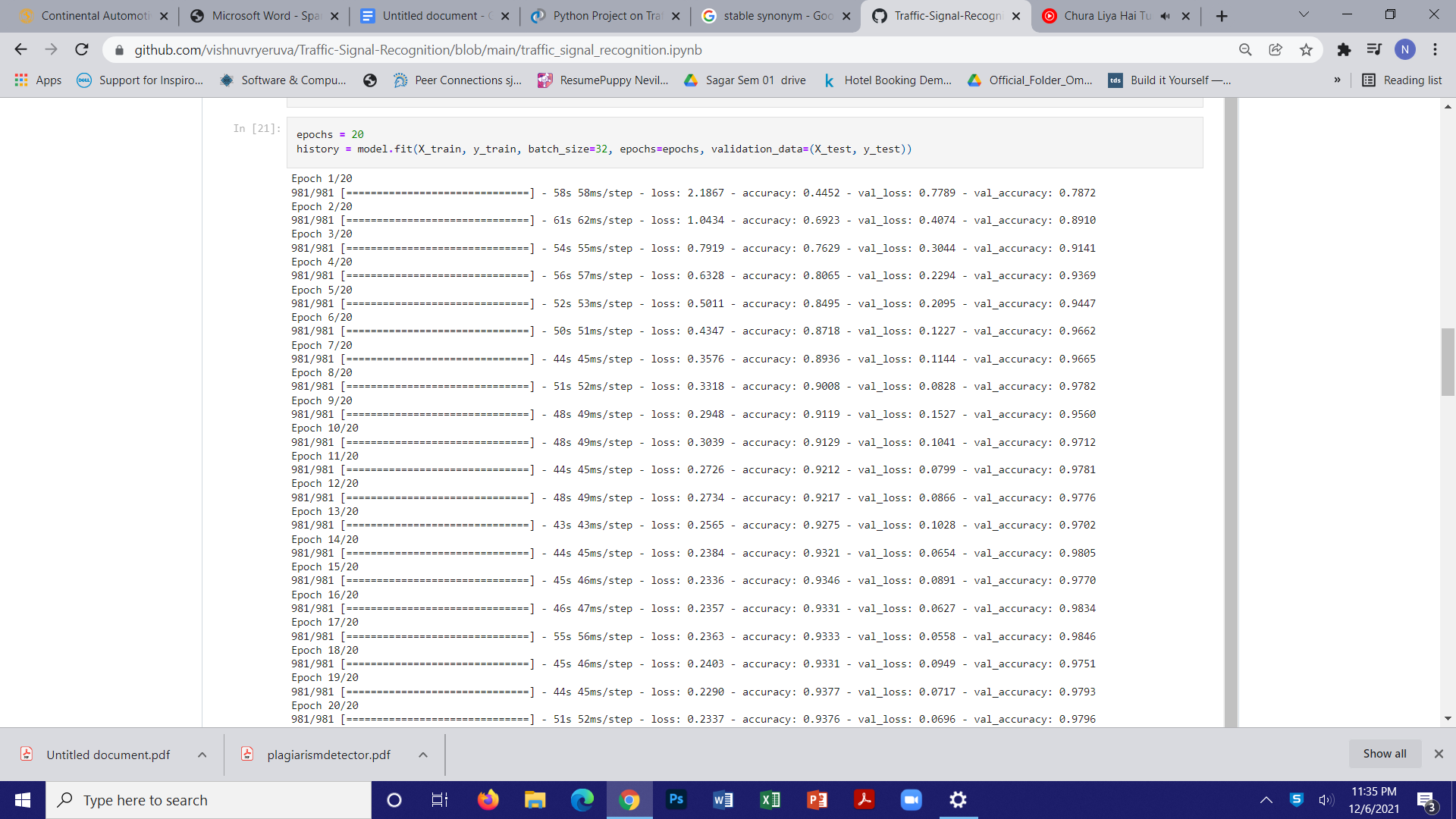
The above mentioned layers were added while training the model to classify the traffic signs as output. It uses “Adam’s” optimizer for training this model . Also, as we have multiple classes for categorization, we have used “Categorical Crossentropy” as the loss function.



**Chapter 2.2 Model Training**

After constructing the architecture and compilation of the model, the next phase is Model training. The model gave better results with Batch Size 64. Also, it was found

that the accuracy remained steady after training the model with 20 Epochs. In addition to that, the accuracy scored is approximately 94%.



**Chapter 2.3 Graphs:**

**Chapter 2.3.1 Loss/Accuracy Graph**

**Chapter 3 System Development**

**Chapter 3.1 Hardware Requirements**

* RAM: 8GB or more (16 GB Recommended)
* CPU: I7 processor recommended
* Storage:256 Gigabytes SSD
* GPU: Nvidia NVS 310, GT, GTS, RTS (Recommended)

**Chapter 3.2 Software Requirements**

* Tensorflow
* OpenCV
* Access to Cloud platforms like Google Cloud platform, AWS Sagemaker.
* IDEs like Spyder, Jupyter Notebook, Google Colab
* Python v3.0.0 and above

**Chapter 3.3 Tools Used**

| Sr. No. | Tool Name | Description |
| --- | --- | --- |
| 1 | Anaconda Navigator | Used to provide environment space for python/Jupyter lab |
| 2 | Python 3 / Jupyter notebook | Used for Training the data and as language to work and for pre-required libraries to fetch and use |
| 3 | Flask | For front-end implementation |
| 4 | Mutiple Libraries | Numpy, pandas, tensorflow, sklearn,matplotlib,keras, openCV2 |
| 5 | Pycharm | To provide IDE features and run the code with problem solving environment |
| 6 | Heroku | For software deployment |

**Chapter 4 Implementation**

**Chapter 4.1 Dataset**

German Traffic Sign Recognition Benchmark(GTSRB) dataset

For traffic sign recognition, we looked for dataset which contains images for traffic signs.This dataset consists of multi-class, single image classification images. Furthermore, this dataset consists of more than 51k images and 43 classes.

Link for the dataset:- <https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign>

**Chapter 4.2 Detection & Classification**

There are mainly 4 phases of this stage:

1)Data Preprocessing

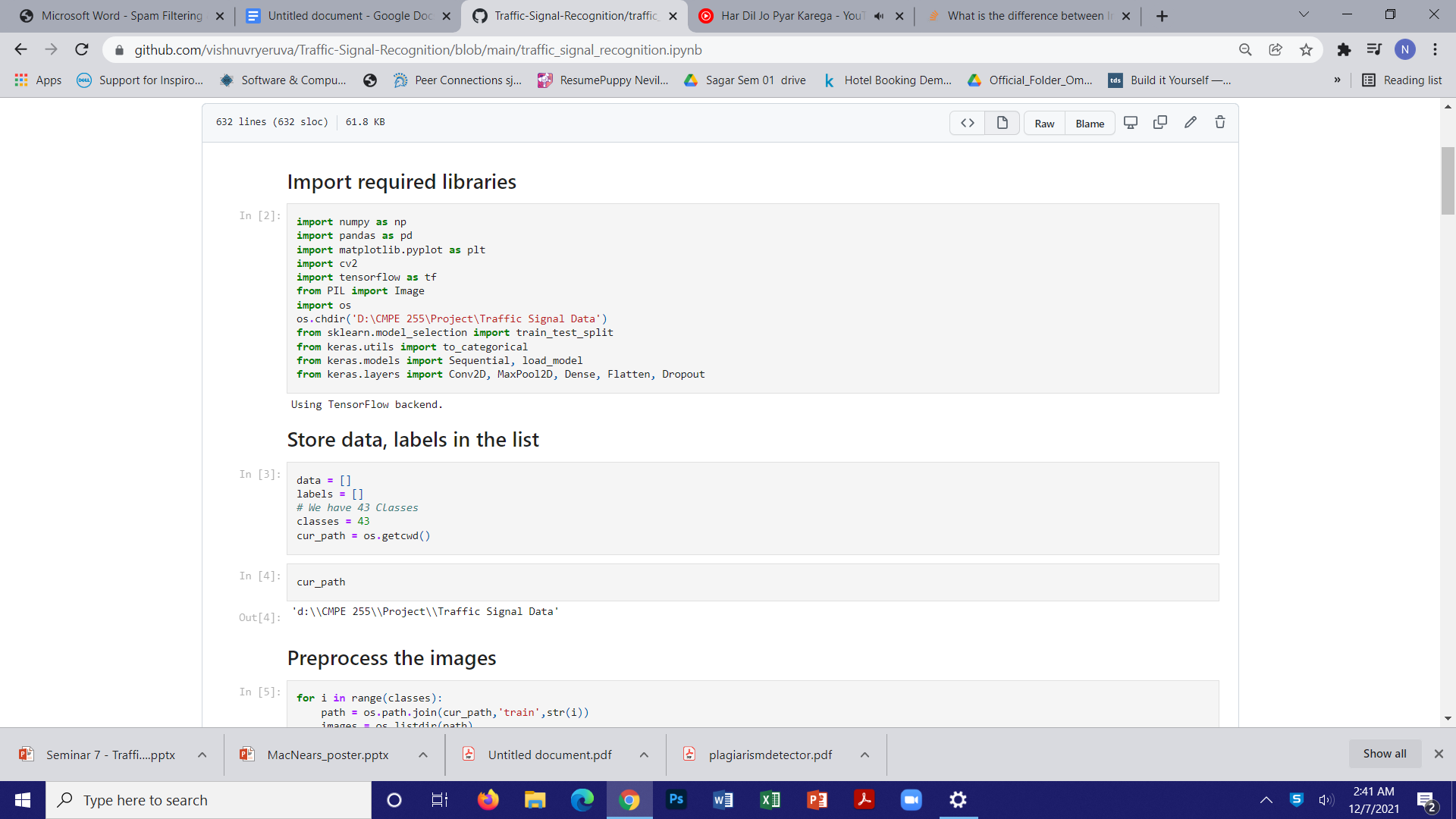
2)Data Visualization

3)Model Training

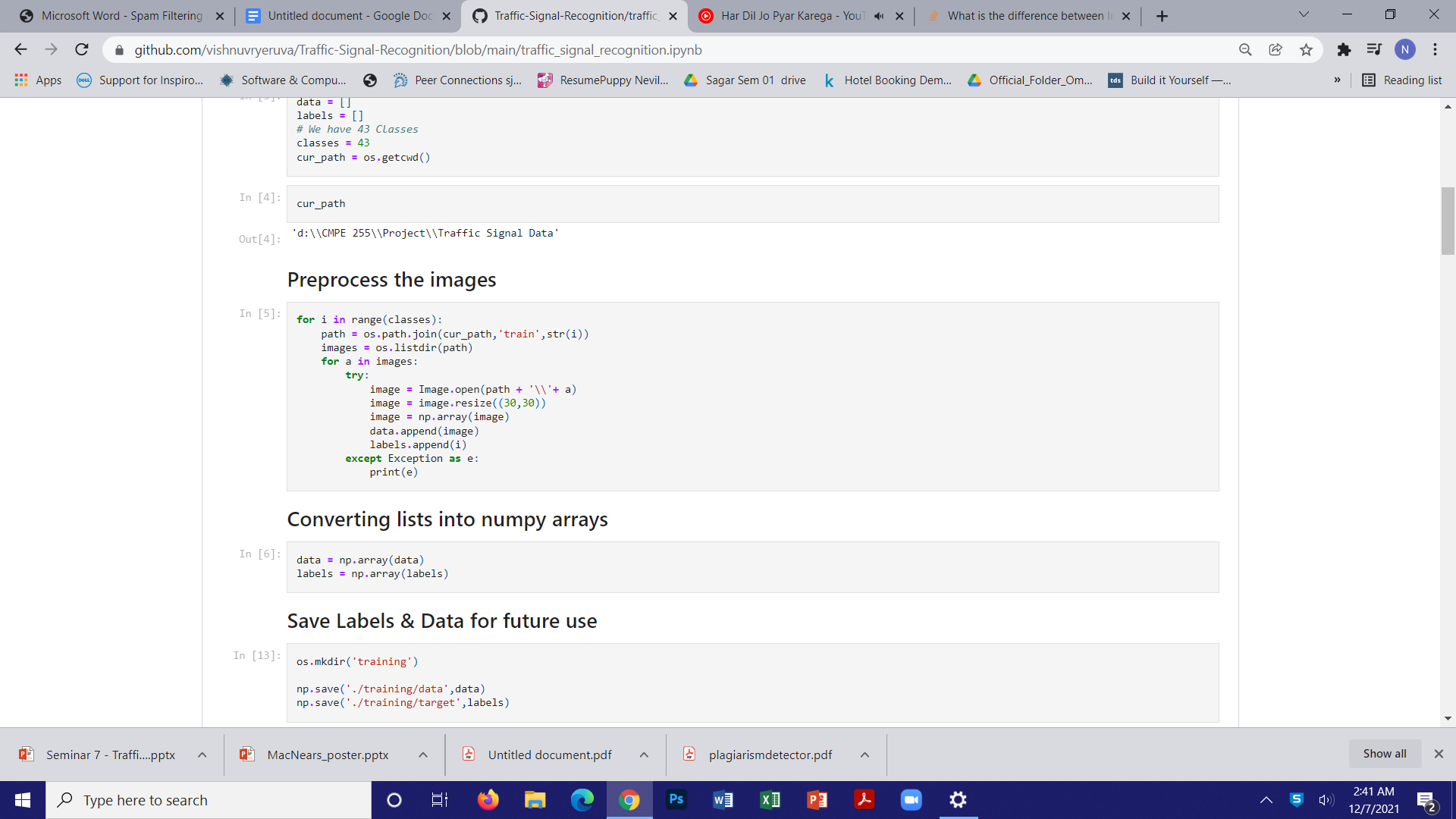
4)Model Testing

**Chapter 4.2.1 Data Preprocessing**

* This is the first stage of the project. First and foremost,the necessary libraries are imported and the image dataset is loaded for further preprocessing.

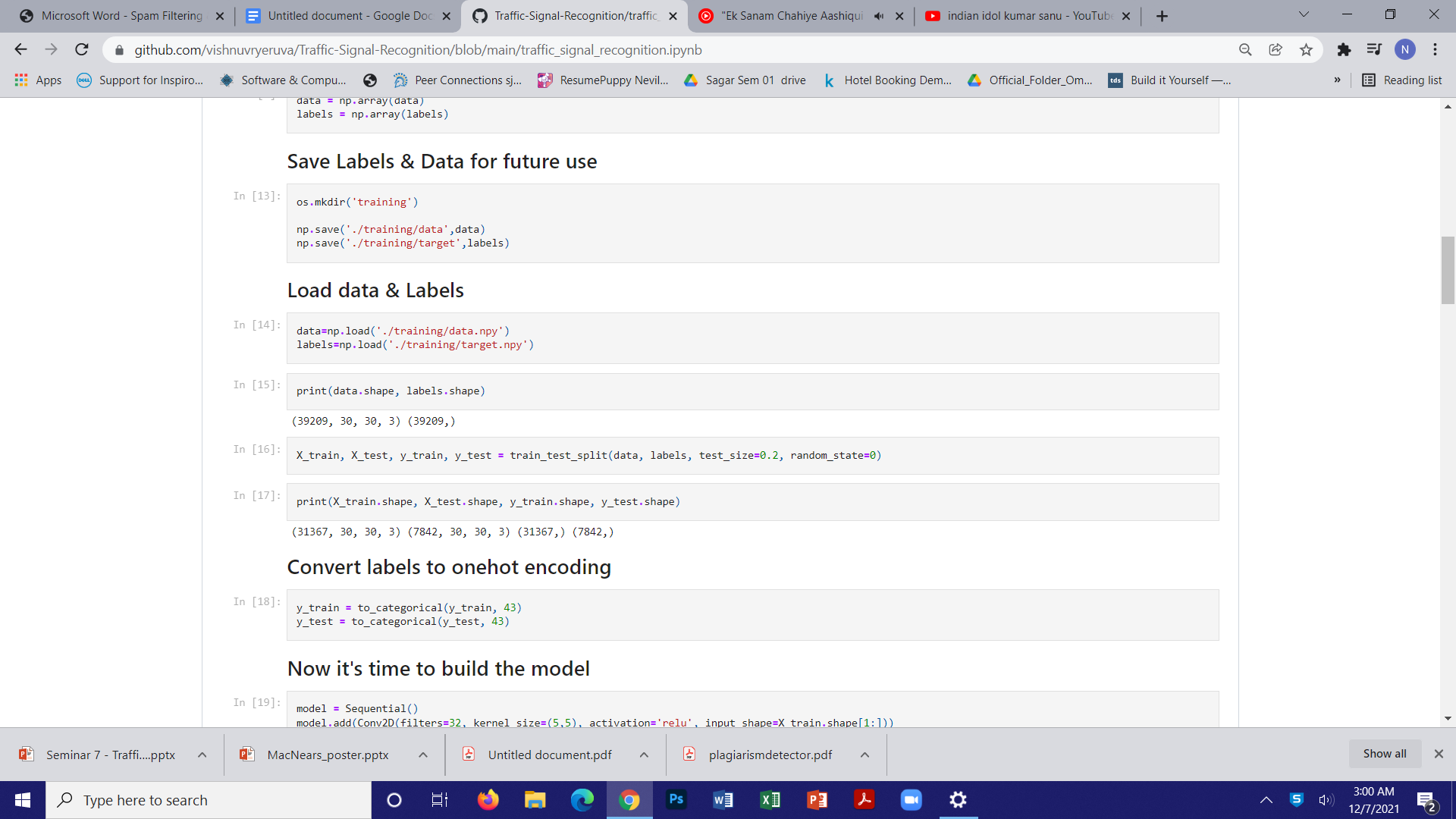


* After loading the dataset,the images are resized to 30x30 image
* The images are thereafter converted into numpy array.

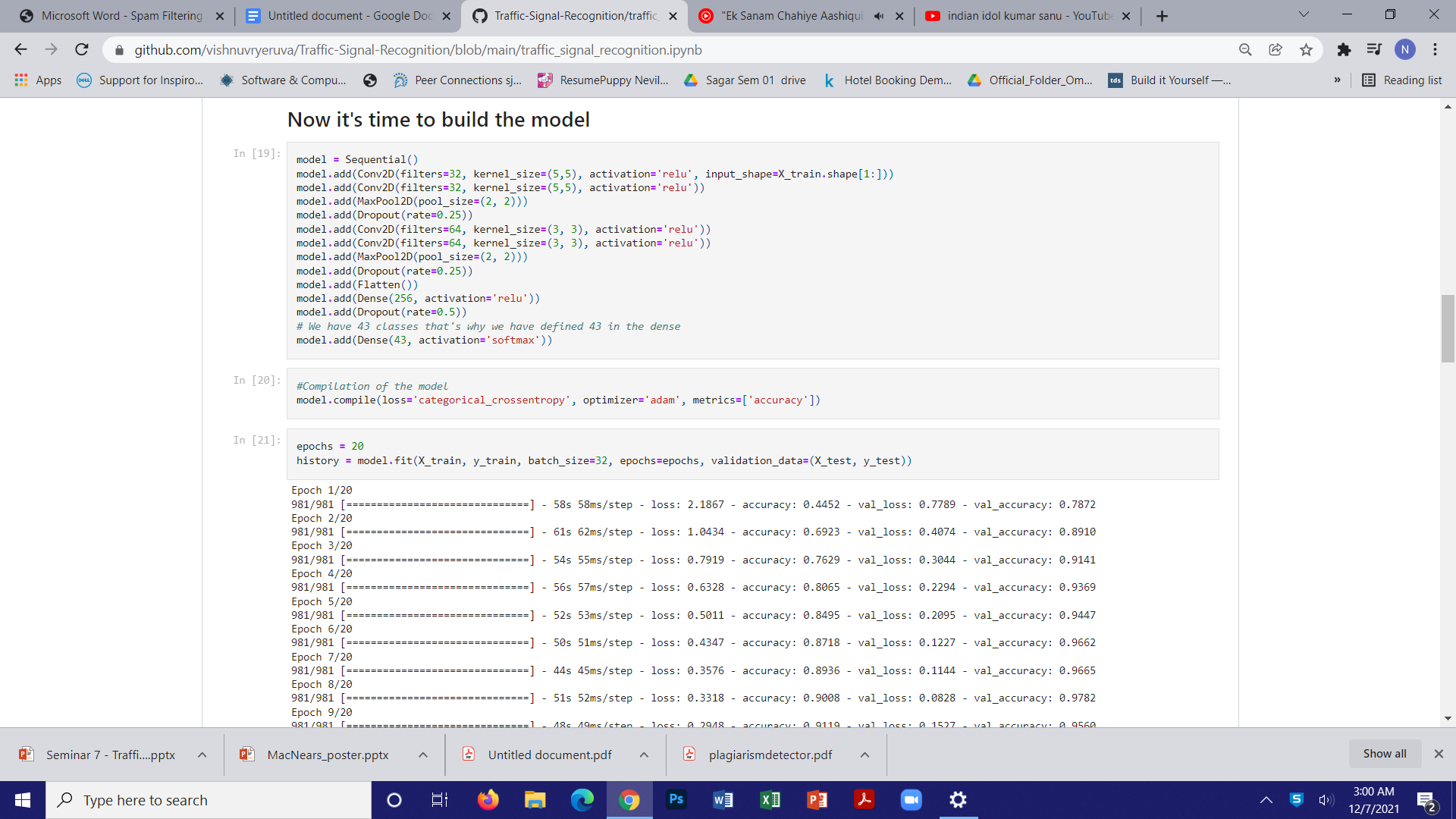


**Chapter 4.2.2 Model Training**

* After Data Preprocessing, the next stage is Model Training.
* The Dataset is, hereby, divided into training and testing datasets with number of records in the ratio 4:1.



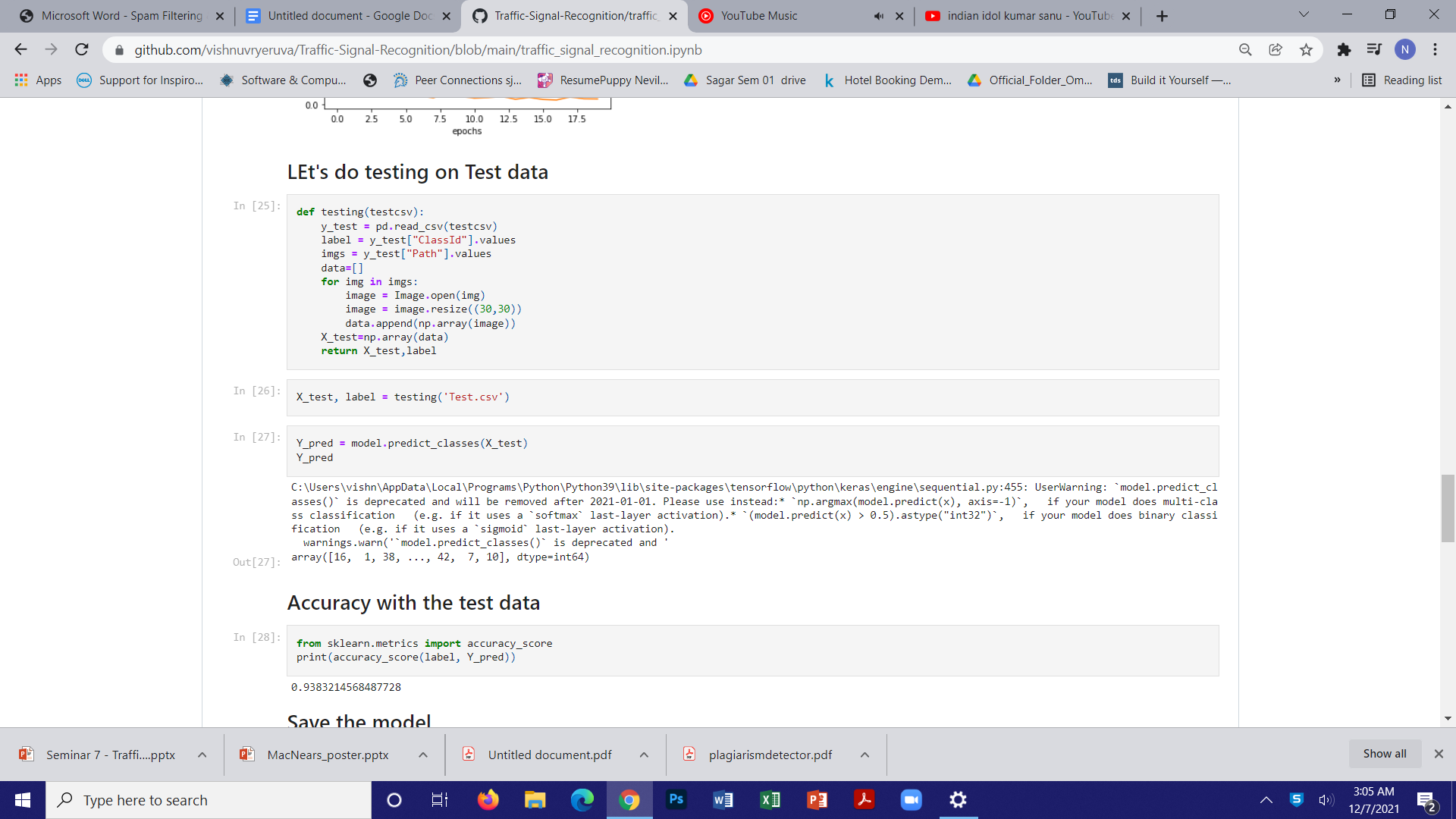
* Then a sequential model is defined and the different types of layers like 2D convolution, dense layer, MaxPool layer etc. are added and in this way, the architecture of the model is defined.
* Thereafter, the model is compiled where ”Adam’s optimizer” is used and the loss function defined here is categorical cross entropy.



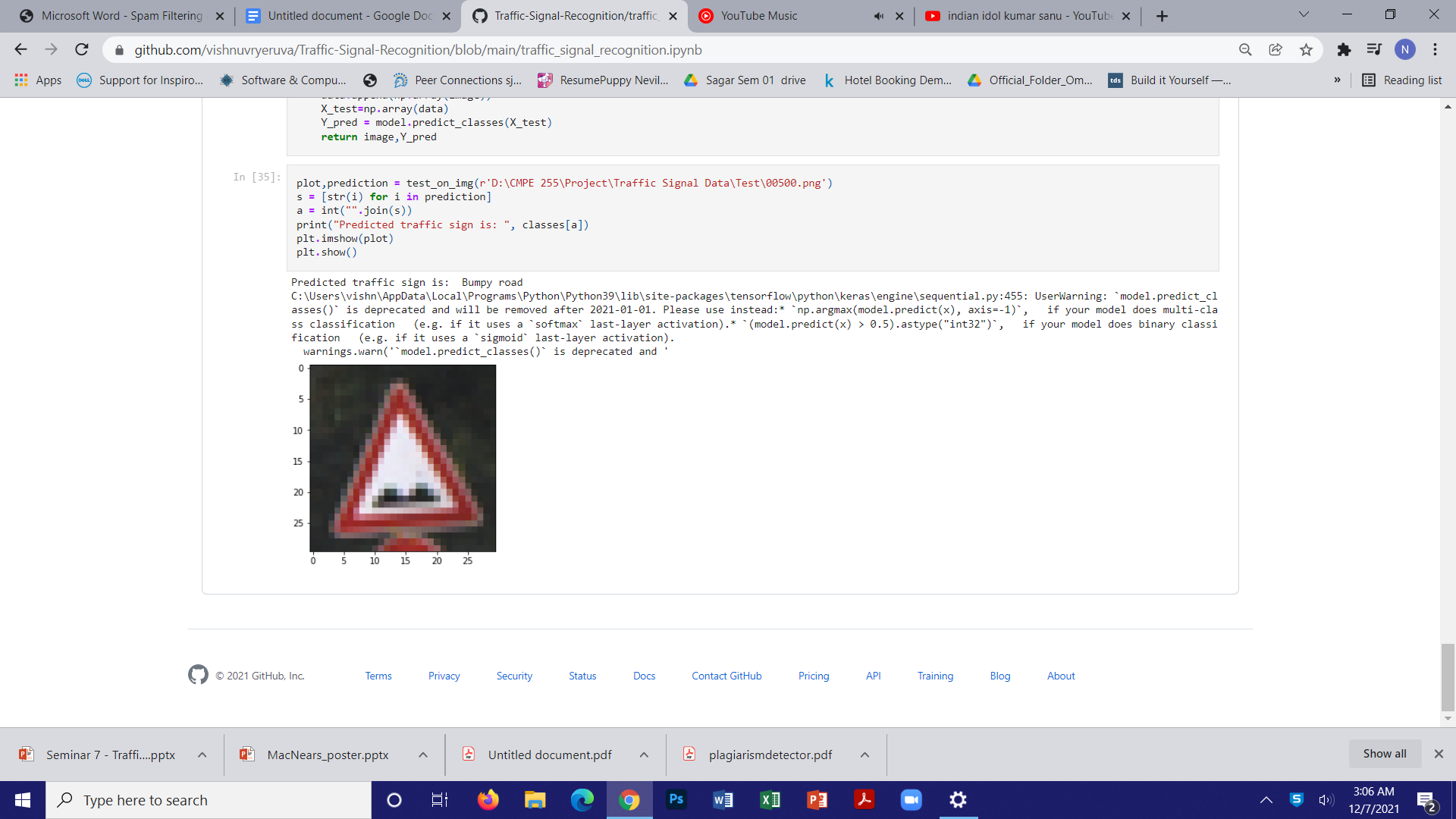
* Afterwards, the model is fitted and trained over the traffic sign dataset using Epochs = 20 and accuracy obtained is about 94%.

**Chapter 4.2.3 Model Testing**

* After the model training using CNNs is completed, the accuracy of the model is tested over the testing data (remaining 20% of the dataset)



* When we observe the above figure closely, the accuracy of the model over the test dataset is also close to 94%.Now , we have given an image to classify which traffic sign it is.



* This is the end of the testing phase.

**Chapter 4.3 User Interface & Deployment**

* For the user interface, we have used the Flask framework where users have to upload an image and the software will classify the type of traffic sign with the help of CNN model.
* With the aim of the software to operate entirely over cloud and accessible over the internet, we have deployed this software over Heroku which acts as a platform as a service(PaaS) .