DELHI TECHNOLOGICAL UNIVERSITY

Established by Govt. of Delhi vide Act 6 of 2009

(FORMERLY DELHI COLLEGE OF ENGINEERING)

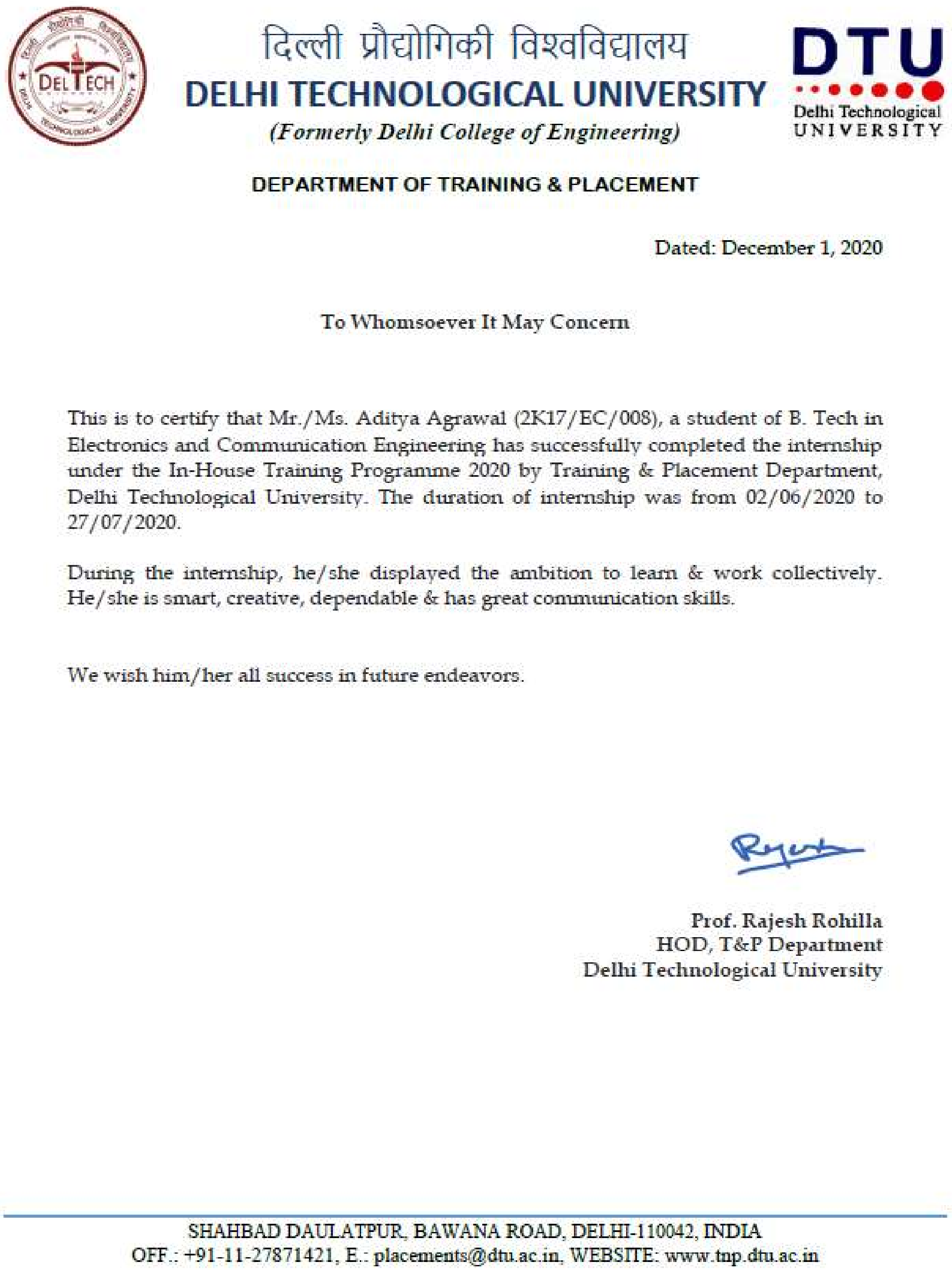
SHAHBAD DAULATPUR, MAIN BAWANA ROAD, DELHI - 110042

TRAINING DIARY

Name: - ADITYA AGRAWAL

Roll No: - 2K17/EC/008

Certificate of Training



दिल्ली प्रौद्योगिकी विश्वविद्यालय
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)

DTU Delhi Technological UNIVERSITY

DEPARTMENT OF TRAINING & PLACEMENT

Dated: December 1, 2020

To Whomsoever It May Concern

This is to certify that Mr./Ms. Aditya Agrawal (2K17/EC/008), a student of B. Tech in Electronics and Communication Engineering has successfully completed the internship under the In-House Training Programme 2020 by Training & Placement Department, Delhi Technological University. The duration of internship was from 02/06/2020 to 27/07/2020.

During the internship, he/she displayed the ambition to learn & work collectively. He/she is smart, creative, dependable & has great communication skills.

We wish him/her all success in future endeavors.

[illegible]

Prof. Rajesh Rohilla
HOD, T&P Department
Delhi Technological University

SHAHBAD DAULATPUR, BAWANA ROAD, DELHI-110042, INDIA
OFF.: +91-11-27871421, E.: placements@dtu.ac.in, WEBSITE: www.tnp.dtu.ac.in

Acknowledgement

I express my gratitude and special thanks to Prof. Rajesh Rohilla (Electronics and Communication Engineering) for being by mentor during my summer training and for his valuable support and guidance and providing all required resources for completion of my training even with the difficulties faced during the COVID-19 pandemic.

I am thankful to the Department of Electronics and Communication Engineering and the Department of Training and Placement for organising this training session. Also, I am thankful to all the people who have contributed to the realization of this summer training.

Lastly, I would like to thank my parents and my friends, who motivated me during these tough times of pandemic to pursue the training.

Aditya Agrawal

2K17/EC/008

Contents

Profile of Organisation

History

Delhi Technological University, (initially established with the name – Delhi Polytechnic) came into existence in the year 1941 to cater the needs of Indian industries for trained technical manpower with practical experience and sound theoretical knowledge. The institution was set up at historic Kashmere Gate campus as a follow up of the Wood and Abott Committee of 1938. It comprised of a multi-disciplinary and multi-level institution offering wide ranging programmes in engineering, technology, arts and sculpture, architecture, pharmacy and commerce. The national diploma awarded by the institution was recognized as equivalent to degree level for the purposes of employment. In 1952 the college was affiliated with University of Delhi and started formal Degree level Programmes.

Till 1962, the college was under the direct control of Ministry of Education, Government of India. In 1963, the administration of the college was handed over to Delhi Administration. Delhi College of Engineering was under the administrative control of Department of Technical Education & Training, Govt. of NCT of Delhi. For academic purposes, the college was affiliated to University of Delhi since 1952. From July 2009, the DCE has become Delhi Technological University vide Delhi act 6 of 2009.

Vision

To be a world class university through education, innovation and research for the service of humanity.

Mission

1. To establish centres of excellence in emerging areas of science, engineering, technology, management, and allied areas.
2. To foster an ecosystem for incubation, product development, transfer of technology and entrepreneurship.
3. To create environment of collaboration, experimentation, imagination and creativity.
4. To develop human potential with analytical abilities, ethics and integrity.
5. To provide environment friendly, reasonable and sustainable solutions for local & global needs.

Core Values

- Integrity: We do what we say through trust, transparency and honesty.
- Compassion: We foster civilized mindset for kindness, consideration and benevolence.
- Commitment: We dedicate ourselves completely to all our endeavours.
- Creativity: We nurture innovation imagination and ideation.
- Collaboration: We promote team work through togetherness, harmony and acknowledgment.
- Inclusion: We embrace diversity with respect, acceptance and affirmation.

Week Wise Report

Week 1 (Learning Basic Concepts of Machine Learning)

(02/06/2020 – 08/06/2020)

In my 1st week of the internship, I studied about the basic concepts of Machine Learning that I would need during my internship.

Machine learning is the science of getting computers to act without being explicitly programmed. A modern definition of Machine Learning is: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Example: playing checkers.

- E = the experience of playing many games of checkers
- T = the task of playing checkers.
- P = the probability that the program will win the next game.

In general, any machine learning problem can be assigned to one of two broad classifications:

- Supervised learning
- Unsupervised learning.

Supervised Learning

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output.

Supervised learning problems are categorized into "**regression**" and "**classification**" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

Example 1: Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output, so this is a regression problem.

We could turn this example into a classification problem by instead making our output about whether the house "sells for more or less than the asking price." Here we are classifying the houses based on price into two discrete categories.

Example 2:

1. Regression - Given a picture of a person, we have to predict their age on the basis of the given picture
2. Classification - Given a patient with a tumour, we have to predict whether the tumour is malignant or benign.

Unsupervised Learning

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data.

With unsupervised learning there is no feedback based on the prediction results.

Example:

- **Clustering**: Take a collection of 1,000,000 different genes and find a way to automatically group these genes into groups that are somehow similar or related by different variables, such as lifespan, location, roles, and so on.
- **Non-clustering**: The "Cocktail Party Algorithm", allows you to find structure in a chaotic environment. (i.e., identifying individual voices and music from a mesh of sounds at a cocktail party).

Model Representation

While defining a Machine Learning model we use the following notations: -

- $x^{(i)}$ denotes the “input” variables/features.
- $y^{(i)}$ denotes the “output” variables.
- $(x^{(i)}, y^{(i)})$ is called a training example.
- i=1 to m is called the training set.
- X denotes the space of input values.
- Y denotes the space of output values.

Formally describing the supervised learning problem, our goal is, given a training set, to learn a function h : X → Y so that h(x) is a “good” predictor for the corresponding value of y. For historical reasons, this function h is called a hypothesis.

When the target variable that we’re trying to predict is continuous, we call the learning problem a regression problem. When y can take on only a small number of discrete values, we call it a classification problem.

Cost Function

We can measure the accuracy of our hypothesis function by using a cost function. This takes an average difference of all the results of the hypothesis with inputs from x's and the actual output y's.

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

To make the machine learning model more efficient, our goal is to minimize the cost function.

Gradient Descent

Gradient Descent is an algorithm used to estimate the parameters in the hypothesis function. This algorithm is run until the cost function attains the minimum possible value.

The algorithm is:

Repeat until convergence:

$$\theta_j := \theta_j - \alpha \frac{\delta}{\delta\theta_j} J(\theta_0, \theta_1)$$

where,

j=0,1 is the feature index numbers

α is the learning rate.

Some applications of Machine Learning

1. Image Recognition
2. Speech Recognition
3. Traffic Prediction
4. Product Recommendations
5. Self-driving cars
6. Email Spam and Malware Filtering
7. Virtual Person Assistant
8. Online Fraud Detection
9. Stock market Trading
10. Medical Diagnosis
11. Automatic Language Translation

Week 2 (Installing Software, Libraries of Python)

(09/06/2020 – 15/06/2020)

In the 2nd week of the internship, I installed the software that was needed for the project work. I also studied about the various libraries of Python that are frequently used in machine learning projects.

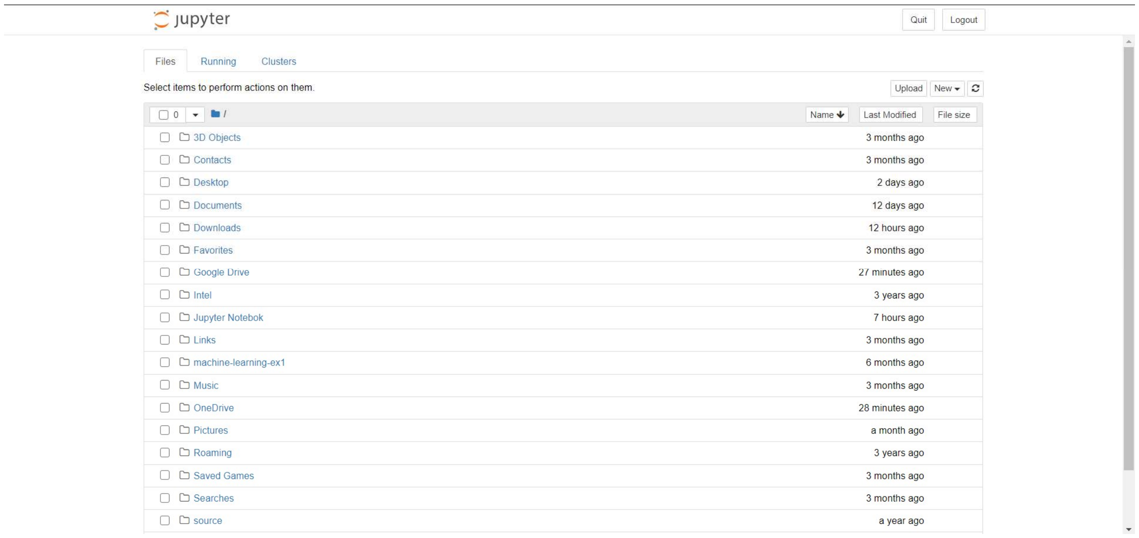
Jupyter Notebook

The software I used for programming during the internship is Jupyter Notebook. The language of programming was Python 3.7

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

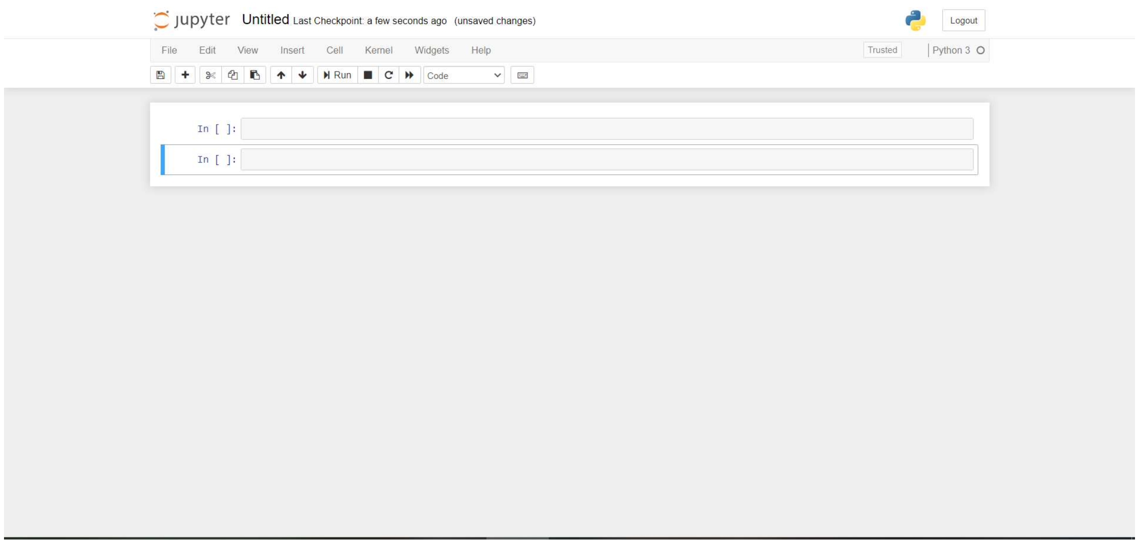
Advantages of using Jupyter Notebook are: -

1. Language of choice: Jupyter supports over 40 programming languages, including Python, R, Julia, and Scala.
2. Share notebooks: Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.
3. Interactive output: Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.
4. Big data integration: Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, TensorFlow.



Select items to perform actions on them.

Name	Last Modified	File size
3D Objects	3 months ago	
Contacts	3 months ago	
Desktop	2 days ago	
Documents	12 days ago	
Downloads	12 hours ago	
Favorites	3 months ago	
Google Drive	27 minutes ago	
Intel	3 years ago	
Jupyter Notebok	7 hours ago	
Links	3 months ago	
machine-learning-ex1	6 months ago	
Music	3 months ago	
OneDrive	28 minutes ago	
Pictures	a month ago	
Roaming	3 years ago	
Saved Games	3 months ago	
Searches	3 months ago	
source	a year ago	


jupyter Untitled Last Checkpoint: a few seconds ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [ ]:
In [ ]:

Libraries of Python

Implementing ML and AI algorithms require a well-structured & well-tested environment to empower developers to come up with the best quality coding solutions. To reduce development time, there are countless Python libraries for machine learning. Python library or framework is a pre-written program that is ready to use on common coding tasks. Some of the most used libraries are: -

1. Keras

Keras is a leading open-source Python library written for constructing neural networks and machine learning projects. It offers almost all standalone modules including optimizers, neural layers, activation functions, initialization schemes, cost functions, and regularization schemes. It makes it easy to add new modules just like adding new functions and classes

Keras makes it simple for machine learning beginners to design and develop a neural network. Keras Python also deals with convolution neural networks. It includes algorithms for normalization, optimizer, and activation layers. Instead of being an end-to-end Python machine learning library, Keras functions as a user-friendly, extensible interface that enhances modularity & total expressiveness.

2. Scikit-learn

Scikit-learn is a prominent open-source Python machine learning library with a broad range of clustering, regression, and classification algorithms. DBSCAN, gradient boosting, random forests, vector machines, and k-means are a few examples. It can interoperate with numeric and scientific libraries of Python like NumPy and SciPy. It is a commercially usable artificial intelligence library. This Python library supports both supervised as well as unsupervised ML.

3. NumPy

NumPy or Numerical Python is linear algebra developed in Python. Almost all Python machine-learning packages like Matplotlib, SciPy, Scikit-learn, etc rely on this library to a reasonable extent. It comes with functions for dealing with complex mathematical operations like linear algebra, Fourier transformation, random number and features that work with matrices and n-arrays in Python. NumPy Python package also performs scientific computations. It is widely used in handling sound waves, images, and other binary functions.

4. Pandas

Python Pandas is an open-source library that offers a wide range of tools for data manipulation & analysis. With this library, you can read data from a broad range of sources like CSV, SQL databases, JSON files, and Excel. It is useful for data as well as analysing basic trends & patterns.

It enables you to manage complex data operation with just one or two commands. Python Pandas comes with several inbuilt methods for combining data and grouping & filtering time-series functionality. Overall, Pandas is not just limited to handle data-related tasks; it serves as the best starting point to create more focused and powerful data tools.

5. OpenCV

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. OpenCV-Python makes use of NumPy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from NumPy arrays. This also makes it easier to integrate with other libraries that use NumPy such as SciPy and Matplotlib.

6. matplotlib.pyplot

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes.

7. TensorFlow

TensorFlow is an end-to-end python machine learning library for performing high-end numerical computations. TensorFlow can handle deep neural networks for image recognition, handwritten digit classification, recurrent neural networks, NLP (Natural Language Processing), word embedding and PDE (Partial Differential Equation). TensorFlow Python ensures excellent architecture support to allow easy computation deployments across a wide range of platforms, including desktops, servers, and mobile devices.

Abstraction is the major benefit of TensorFlow Python towards machine learning and AI projects. This feature allows the developers to focus on comprehensive logic of the app instead of dealing with the mundane details of implementing algorithms. With this library, python developers can now effortlessly leverage AI and ML to create unique responsive applications, which responds to user inputs like facial or voice expression.

Week 3 (Colour Detection Project)

(16/06/2020 – 22/06/2020)

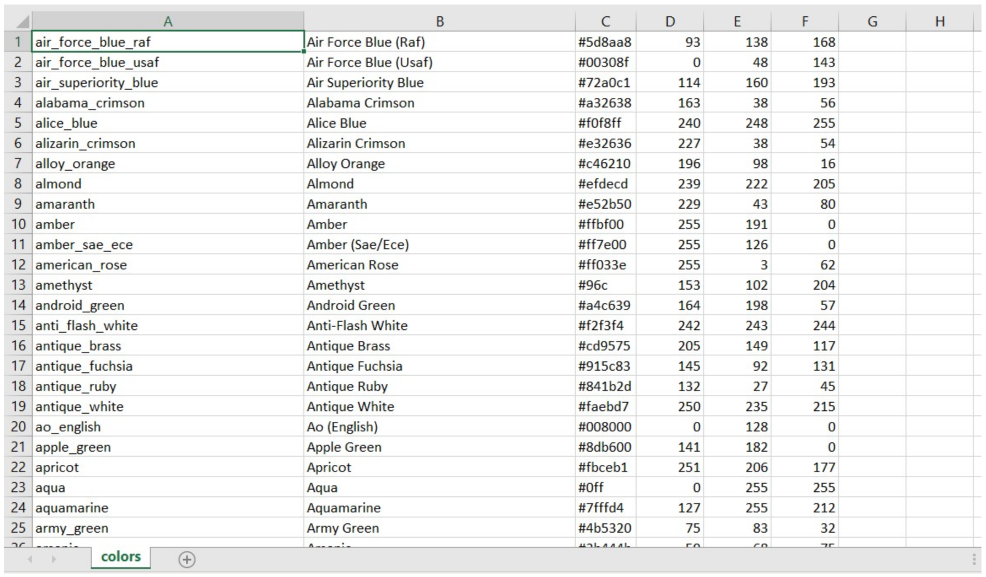
In the 3rd week of my internship, I was assigned to make a colour detection project using Python. It was a small python project, intended to get me familiar with some basic libraries of Python. The project utilizes the following libraries: -

- cv2 (OpenCV) (version 4.4.0.46)
- NumPy (version 1.16.2)
- Pandas (version 0.24.2)

The aim of this project is to build an application through which you can automatically get the name of the colour in an image by clicking on them.

Dataset (colors.csv)

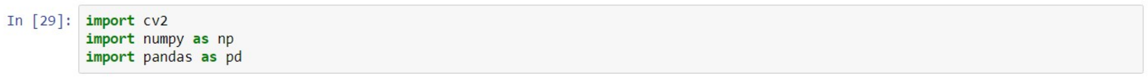
The dataset used for this project is a csv (comma separated values) file that includes 865 colour names mapped with their hex values and RGB values. The following screenshot shows how the values are stored in the csv file.



	A	B	C	D	E	F	G	H
1	air_force_blue_raf	Air Force Blue (Raf)	#5d8aa8	93	138	168		
2	air_force_blue_usaf	Air Force Blue (Usaf)	#00308f	0	48	143		
3	air_superiority_blue	Air Superiority Blue	#72a0c1	114	160	193		
4	alabama_crimson	Alabama Crimson	#a32638	163	38	56		
5	alice_blue	Alice Blue	#f0f8ff	240	248	255		
6	alizarin_crimson	Alizarin Crimson	#e32636	227	38	54		
7	alloy_orange	Alloy Orange	#c46210	196	98	16		
8	almond	Almond	#efdecd	239	222	205		
9	amaranth	Amaranth	#e52b50	229	43	80		
10	amber	Amber	#ffbf00	255	191	0		
11	amber_sae_ece	Amber (Sae/Ece)	#ff7e00	255	126	0		
12	american_rose	American Rose	#ff033e	255	3	62		
13	amethyst	Amethyst	#96c	153	102	204		
14	android_green	Android Green	#a4c639	164	198	57		
15	anti_flash_white	Anti-Flash White	#f2f3f4	242	243	244		
16	antique_brass	Antique Brass	#cd9575	205	149	117		
17	antique_fuchsia	Antique Fuchsia	#915c83	145	92	131		
18	antique_ruby	Antique Ruby	#841b2d	132	27	45		
19	antique_white	Antique White	#faebd7	250	235	215		
20	ao_english	Ao (English)	#008000	0	128	0		
21	apple_green	Apple Green	#8db600	141	182	0		
22	apricot	Apricot	#fbceb1	251	206	177		
23	aqua	Aqua	#0ff	0	255	255		
24	aquamarine	Aquamarine	#7fffd4	127	255	212		
25	army_green	Army Green	#4b5320	75	83	32		

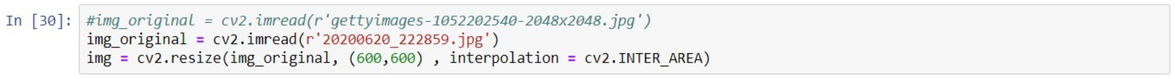
Making of the project

1. Importing the necessary libaries.



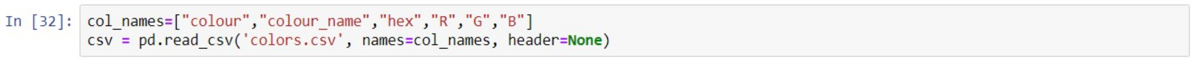
```
In [29]: import cv2
         import numpy as np
         import pandas as pd
```

2. Reading the image using the imread function of opencv library.



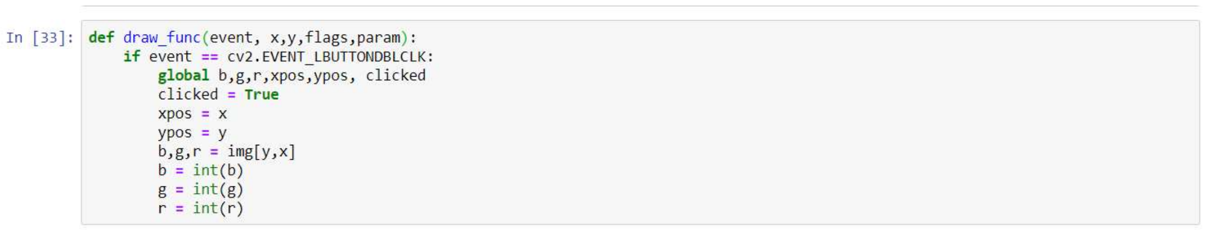
```
In [30]: #img_original = cv2.imread(r'gettyimages-1052202540-2048x2048.jpg')
         img_original = cv2.imread(r'20200620_222859.jpg')
         img = cv2.resize(img_original, (600,600) , interpolation = cv2.INTER_AREA)
```

3. Reading the csv file with read_csv function of pandas. This loads the CSV file into pandas DataFrame. I also assigned a name to each column for easy accessing.



```
In [32]: col_names=["colour","colour_name","hex","R","G","B"]
         csv = pd.read_csv('colors.csv', names=col_names, header=None)
```

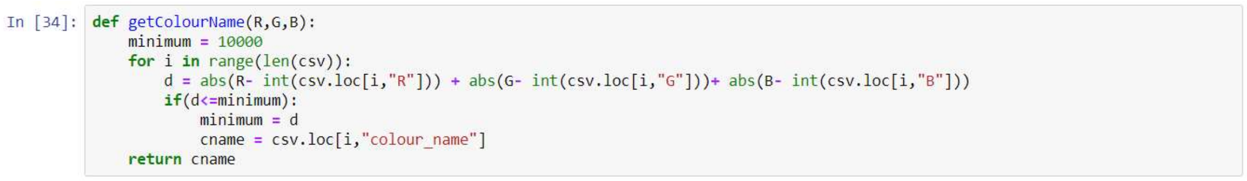
4. I created a draw_func to get x,y coordinates of mouse double click. It will calculate the rgb values of the pixel which we double click.



```
In [33]: def draw_func(event, x,y,flags,param):
             if event == cv2.EVENT_LBUTTONDBLCLK:
                 global b,g,r,xpos,ypos, clicked
                 clicked = True
                 xpos = x
                 ypos = y
                 b,g,r = img[y,x]
                 b = int(b)
                 g = int(g)
                 r = int(r)
```

5. I defined a function called getColourName which returns the colour name using the RGB values. It does so by calculating the minimum distance from all colours and returns the colour with the minimum distance. The distance is calculated by: -

d= abs(Red - ith RedColour) + abs(Green - ith GreenColour) + abs(Blue - ith BlueColour)



```
In [34]: def getColourName(R,G,B):
             minimum = 10000
             for i in range(len(csv)):
                 d = abs(R- int(csv.loc[i,"R"])) + abs(G- int(csv.loc[i,"G"]))+ abs(B- int(csv.loc[i,"B"]))
                 if(d<=minimum):
                     minimum = d
                     cname = csv.loc[i,"colour_name"]
             return cname
```

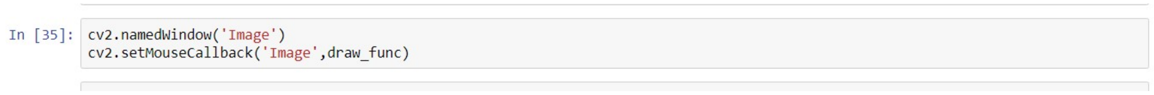
Week 4 (Results of Colour Detection Project)

(23/06/2020 – 29/06/2020)

In the 4th week of my internship, I continued making the colour detection project and was able to get accurate results.

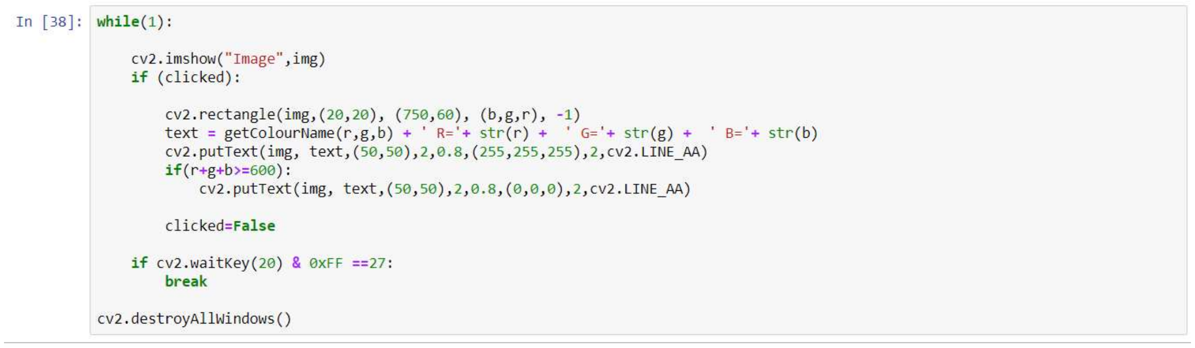
Making of the project

6. I first created a window named 'image' using namedWindow function of cv2. Then I set a callback function which calls the draw_func() whenever a mouse function happens. draw_func() was set so that it only works if mouse is double clicked.



```
In [35]: cv2.namedWindow('Image')
         cv2.setMouseCallback('Image',draw_func)
```

7. In the final block of code I used the cvs.imshow() function to draw the image on the window. When the user double clicks the window, we draw a rectangle using cv2.rectangle() function, get the colour name using getColourName() function and draw the name on the window using cv2.putText() function. For very light colours we will display text in black colour. The loop can be broken using the 'esc' key. cv2.destroyAllWindows() closes any window that was opened by Jupyter.



```
In [38]: while(1):

             cv2.imshow("Image",img)
             if (clicked):

                 cv2.rectangle(img,(20,20), (750,60), (b,g,r), -1)
                 text = getColourName(r,g,b) + ' R='+ str(r) +  ' G='+ str(g) +  ' B='+ str(b)
                 cv2.putText(img, text,(50,50),2,0.8,(255,255,255),2,cv2.LINE_AA)
                 if(r+g+b>=600):
                     cv2.putText(img, text,(50,50),2,0.8,(0,0,0),2,cv2.LINE_AA)

                 clicked=False

             if cv2.waitKey(20) & 0xFF ==27:
                 break

         cv2.destroyAllWindows()
```

Results

If we double click on the image window, we get to know the name of the pixel colour. I ran the program on three different images and got accurate results.

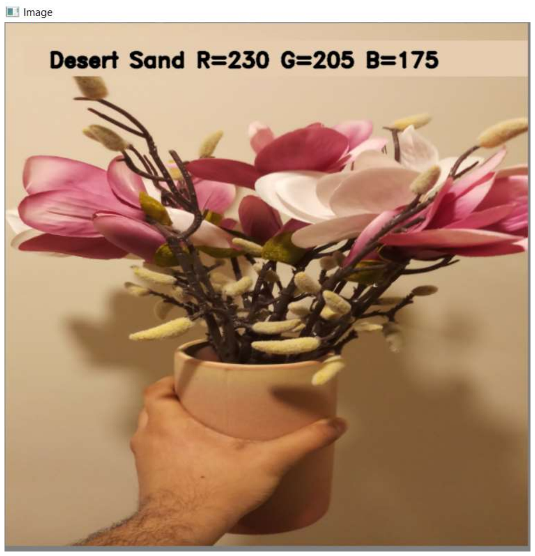


Image
Desert Sand R=230 G=205 B=175

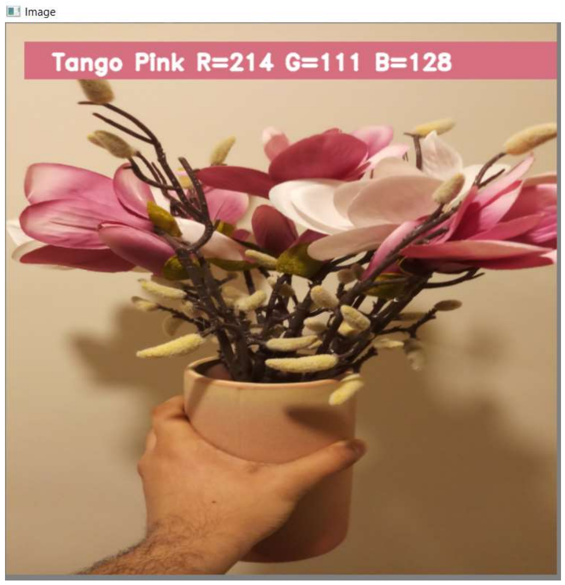


Image
Tango Pink R=214 G=111 B=128

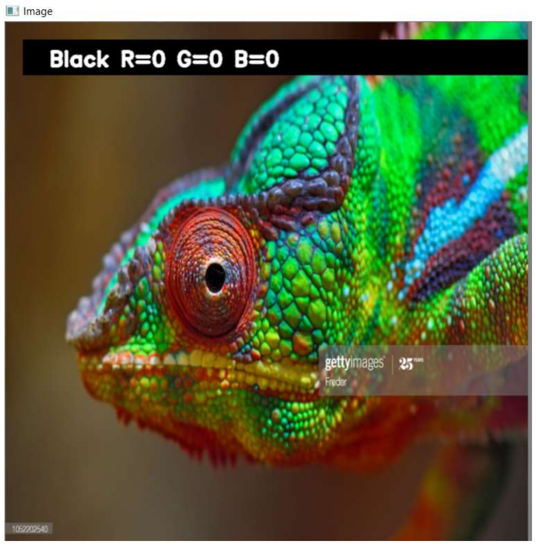


Image
Black R=0 G=0 B=0

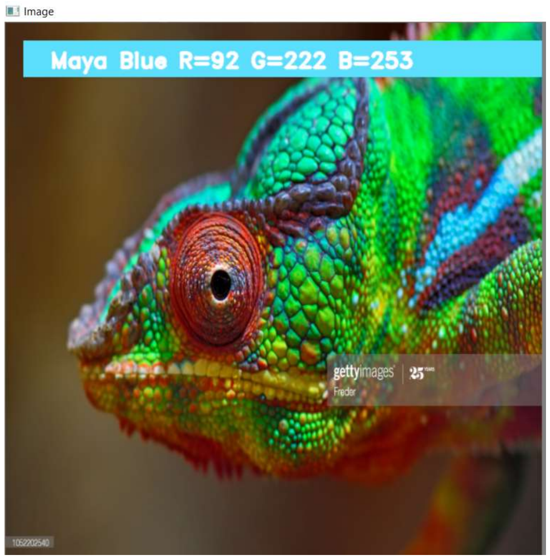


Image
Maya Blue R=92 G=222 B=253

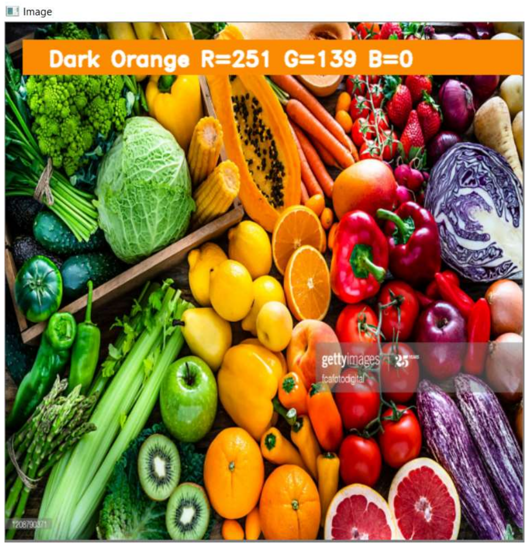


Image
Dark Orange R=251 G=139 B=0

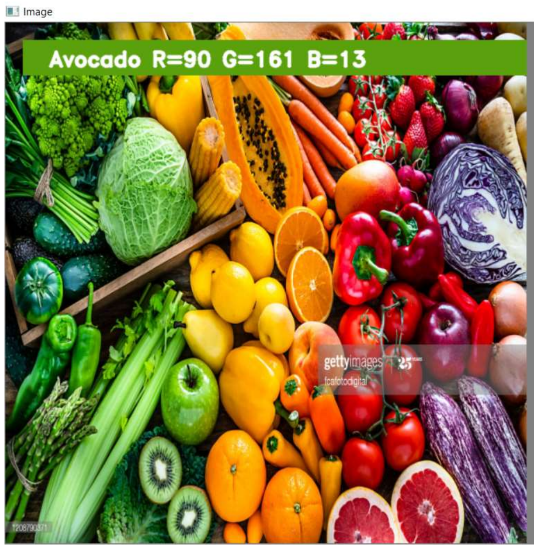


Image
Avocado R=90 G=161 B=13

Week 5 (Handwritten Character Recognition Project)

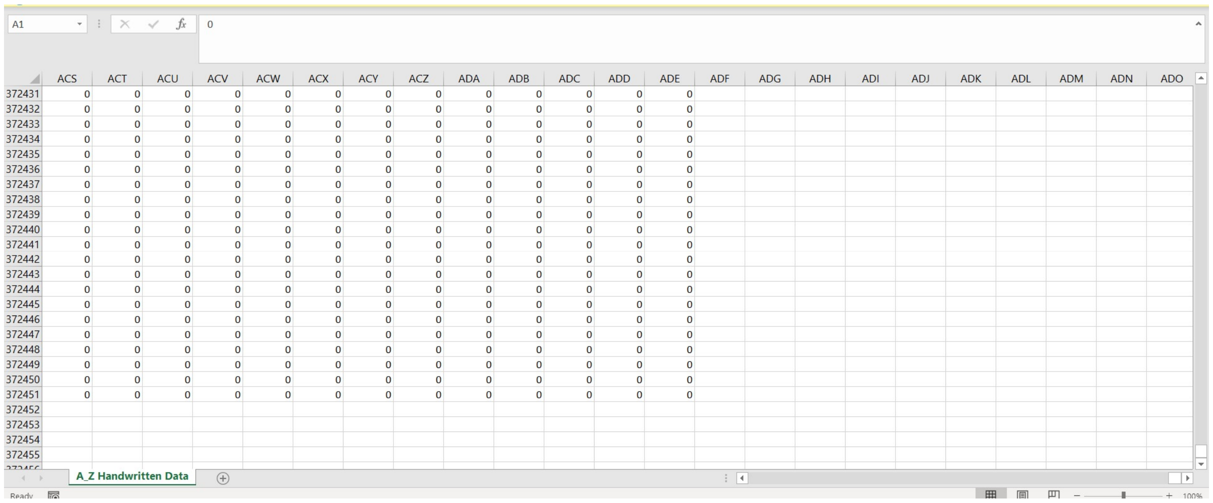
(30/06/2020 – 06/07/2020)

In the 5th week of my internship, I received my final task to create a Handwritten Character Recognition project. The aim of the project is to recognize characters from images of handwritten characters (English alphabets from A-Z). I used neural networks as the model in this project to achieve my aim. The following libraries were used in this project: -

- cv2 (OpenCV) (version 4.4.0.46)
- NumPy (version 1.16.2)
- Pandas (version 0.24.2)
- scikit-learn (version 0.20.3)
- matplotlib (version 3.0.3)
- keras (version 2.1.0)
- tqdm (version 4.31.1)
- TensorFlow (Keras uses TensorFlow in backend and for some image pre-processing) (version 1.13.1)

Dataset (A_Z Handwritten Data.csv)

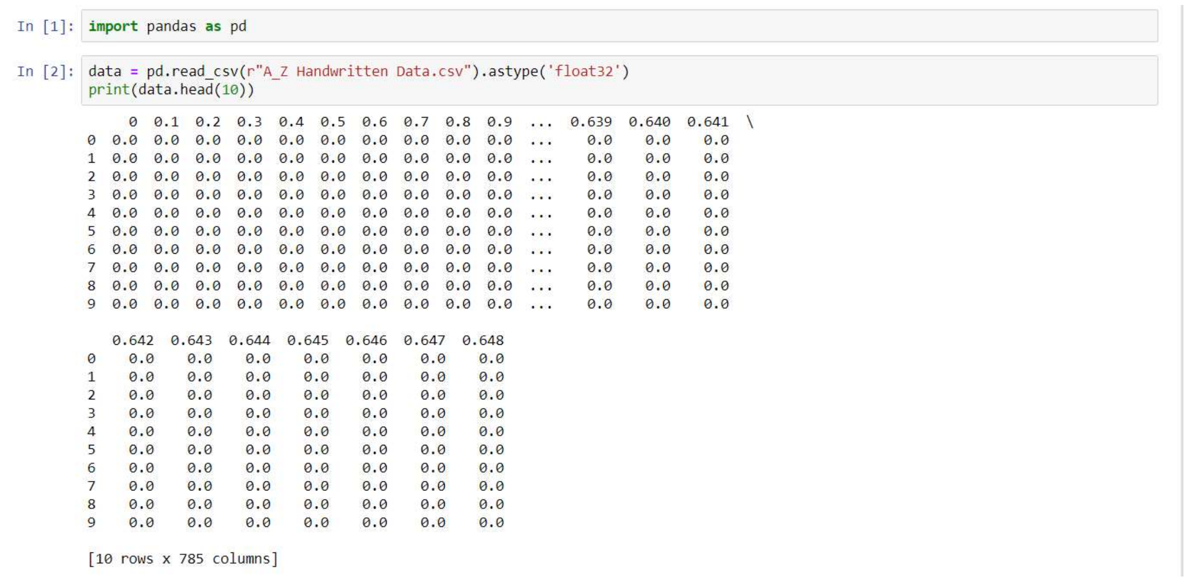
The dataset used for this project is contains 372450 images of capitalized handwritten English alphabets in the form of a CSV file. The dataset contains 26 folders (A-Z) containing handwritten images in size 28x28 pixels, each alphabet in the image is centre fitted to 20x20 pixels box. Each image is stored as Gray-level. When viewed in excel, it has 785 columns (for 28x28 pixels) and 372451 rows (for 372450 images).



	ACS	ACT	ACU	ACV	ACW	ACX	ACY	ACZ	ADA	ADB	ADC	ADD	ADE	ADF	ADG	ADH	ADI	ADJ	ADK	ADL	ADM	ADN	ADO
372431	0	0	0	0	0	0	0	0	0	0	0	0	0										
372432	0	0	0	0	0	0	0	0	0	0	0	0	0										
372433	0	0	0	0	0	0	0	0	0	0	0	0	0										
372434	0	0	0	0	0	0	0	0	0	0	0	0	0										
372435	0	0	0	0	0	0	0	0	0	0	0	0	0										
372436	0	0	0	0	0	0	0	0	0	0	0	0	0										
372437	0	0	0	0	0	0	0	0	0	0	0	0	0										
372438	0	0	0	0	0	0	0	0	0	0	0	0	0										
372439	0	0	0	0	0	0	0	0	0	0	0	0	0										
372440	0	0	0	0	0	0	0	0	0	0	0	0	0										
372441	0	0	0	0	0	0	0	0	0	0	0	0	0										
372442	0	0	0	0	0	0	0	0	0	0	0	0	0										
372443	0	0	0	0	0	0	0	0	0	0	0	0	0										
372444	0	0	0	0	0	0	0	0	0	0	0	0	0										
372445	0	0	0	0	0	0	0	0	0	0	0	0	0										
372446	0	0	0	0	0	0	0	0	0	0	0	0	0										
372447	0	0	0	0	0	0	0	0	0	0	0	0	0										
372448	0	0	0	0	0	0	0	0	0	0	0	0	0										
372449	0	0	0	0	0	0	0	0	0	0	0	0	0										
372450	0	0	0	0	0	0	0	0	0	0	0	0	0										
372451	0	0	0	0	0	0	0	0	0	0	0	0	0										
372452																							
372453																							
372454																							
372455																							

Reading the dataset

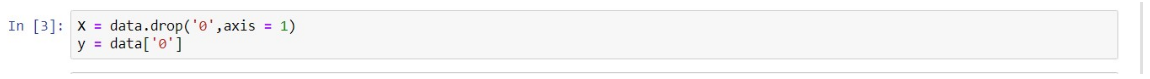
We import the pandas libraries to read the csv dataset. We read the csv dataset using the pd.read_csv() function and save it as a pandas dataframe. The astype() function converts the data into float type. We also print the first 10 images of the dataset using data.head(10) function.



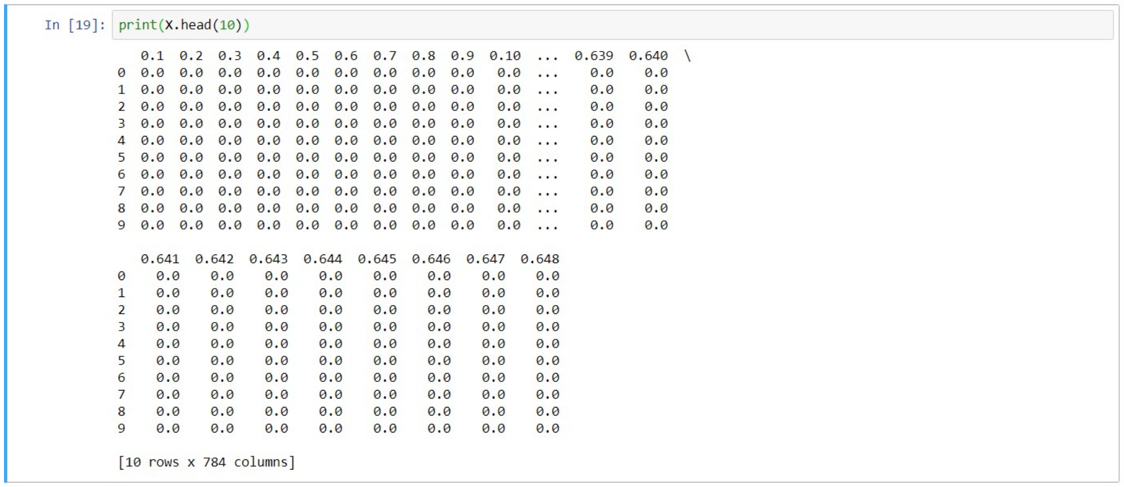
```
In [1]: import pandas as pd

In [2]: data = pd.read_csv(r"A_Z Handwritten Data.csv").astype('float32')
        print(data.head(10))

          0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  ...  0.639  0.640  0.641  \
        0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
        1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
        2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
        3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
        4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
        5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
        6  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
        7  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
        8  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
        9  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0

           0.642  0.643  0.644  0.645  0.646  0.647  0.648
        0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
        1    0.0    0.0    0.0    0.0    0.0    0.0    0.0
        2    0.0    0.0    0.0    0.0    0.0    0.0    0.0
        3    0.0    0.0    0.0    0.0    0.0    0.0    0.0
        4    0.0    0.0    0.0    0.0    0.0    0.0    0.0
        5    0.0    0.0    0.0    0.0    0.0    0.0    0.0
        6    0.0    0.0    0.0    0.0    0.0    0.0    0.0
        7    0.0    0.0    0.0    0.0    0.0    0.0    0.0
        8    0.0    0.0    0.0    0.0    0.0    0.0    0.0
        9    0.0    0.0    0.0    0.0    0.0    0.0    0.0

        [10 rows x 785 columns]
```

Splitting the dataset read into images and labels

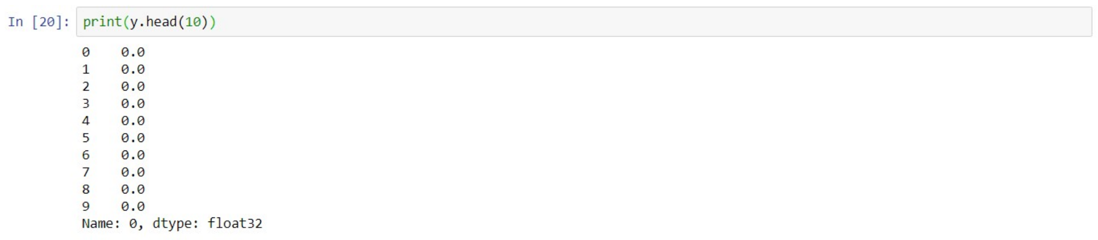
The first column of the dataset contains the labels of the images. We drop the labels columns from the dataframe using data.drop() function so that X only contains the images. The dropped column is stored in y as the labels of the images.



```
In [3]: X = data.drop('0',axis = 1)
        y = data['0']
```



```
In [19]: print(X.head(10))

           0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  0.10  ...  0.639  0.640  \
         0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0  ...    0.0    0.0
         1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0  ...    0.0    0.0
         2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0  ...    0.0    0.0
         3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0  ...    0.0    0.0
         4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0  ...    0.0    0.0
         5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0  ...    0.0    0.0
         6  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0  ...    0.0    0.0
         7  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0  ...    0.0    0.0
         8  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0  ...    0.0    0.0
         9  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0  ...    0.0    0.0

            0.641  0.642  0.643  0.644  0.645  0.646  0.647  0.648
         0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
         1    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
         2    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
         3    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
         4    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
         5    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
         6    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
         7    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
         8    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
         9    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0

         [10 rows x 784 columns]
```



```
In [20]: print(y.head(10))

         0    0.0
         1    0.0
         2    0.0
         3    0.0
         4    0.0
         5    0.0
         6    0.0
         7    0.0
         8    0.0
         9    0.0
         Name: 0, dtype: float32
```

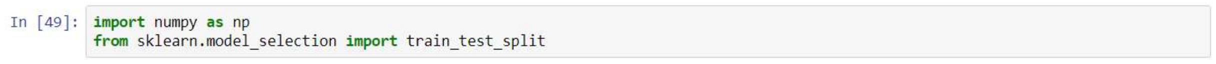
Week 6 (Reshaping the data)

(07/07/2020 – 13/07/2020)

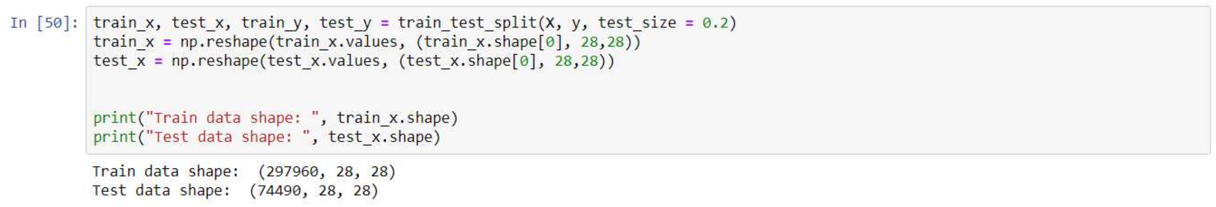
In the 6th week of my internship, I continued making the handwritten character recognition project. In this week, I reshaped the data so that it can be displayed as images and can be used in our CNN model. I also displayed random alphabets from the dataset.

Reshaping the data so that it can be displayed as an image

We split the data into train and test sets using train_test_split() function of scikit-learn. 80% data is used in train set and 20% data is used in test set. We also reshape the train & test image data so that they can be displayed as an image, as initially in the CSV file they were present as 784 columns of pixel data. So we convert it to 28×28 pixels using reshape() function of NumPy.



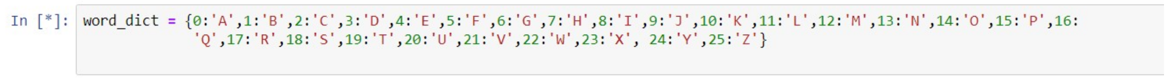
```
In [49]: import numpy as np
         from sklearn.model_selection import train_test_split
```



```
In [50]: train_x, test_x, train_y, test_y = train_test_split(X, y, test_size = 0.2)
         train_x = np.reshape(train_x.values, (train_x.shape[0], 28,28))
         test_x = np.reshape(test_x.values, (test_x.shape[0], 28,28))


         print("Train data shape: ", train_x.shape)
         print("Test data shape: ", test_x.shape)

         Train data shape:  (297960, 28, 28)
         Test data shape:  (74490, 28, 28)
```

Dictionary for getting characters from index values

All the labels in y are present in the form of floating point values. In the future we will convert them to integer values. So we create a dictionary word_dict to map the integer values with the characters.



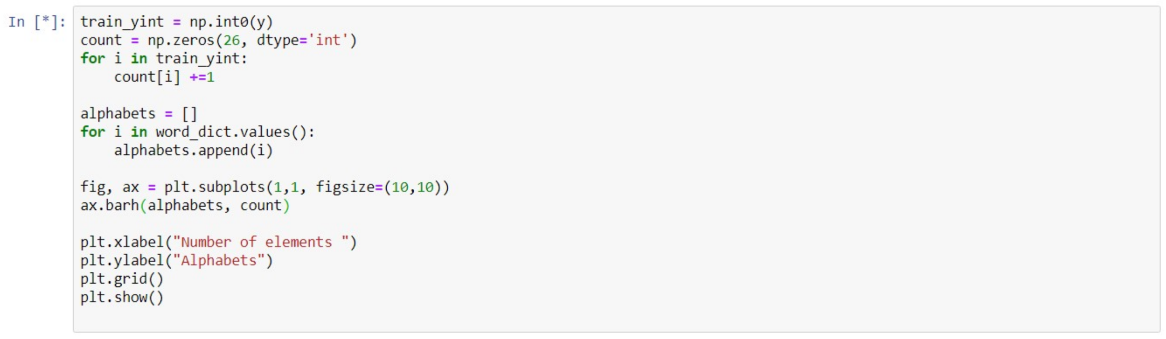
```
In [*]: word_dict = {0:'A',1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'J',10:'K',11:'L',12:'M',13:'N',14:'O',15:'P',16:
                     'Q',17:'R',18:'S',19:'T',20:'U',21:'V',22:'W',23:'X', 24:'Y',25:'Z'}
```

Plotting the number of alphabets in the dataset

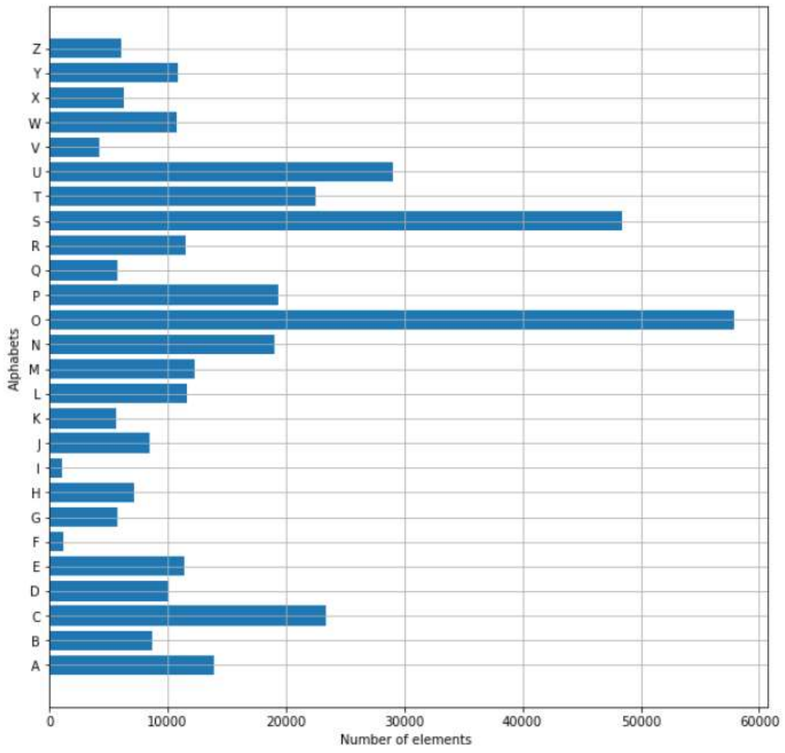
We plot the distribution of alphabets in our dataset using matplotlib.pyplot. Firstly we convert the y labels into integer values and append into the count list according to the label. This count list has the number of images present in the dataset belonging to each alphabet.

Next we create a list name 'alphabets' which contains all the characters using the values() function of the dictionary.

Next, using the count & alphabets lists we draw the horizontal bar plot using subplots() function of matplotlib.pyplot.



```
In [*]: train_yint = np.int0(y)
        count = np.zeros(26, dtype='int')
        for i in train_yint:
            count[i] +=1

        alphabets = []
        for i in word_dict.values():
            alphabets.append(i)

        fig, ax = plt.subplots(1,1, figsize=(10,10))
        ax.barh(alphabets, count)

        plt.xlabel("Number of elements ")
        plt.ylabel("Alphabets")
        plt.grid()
        plt.show()
```



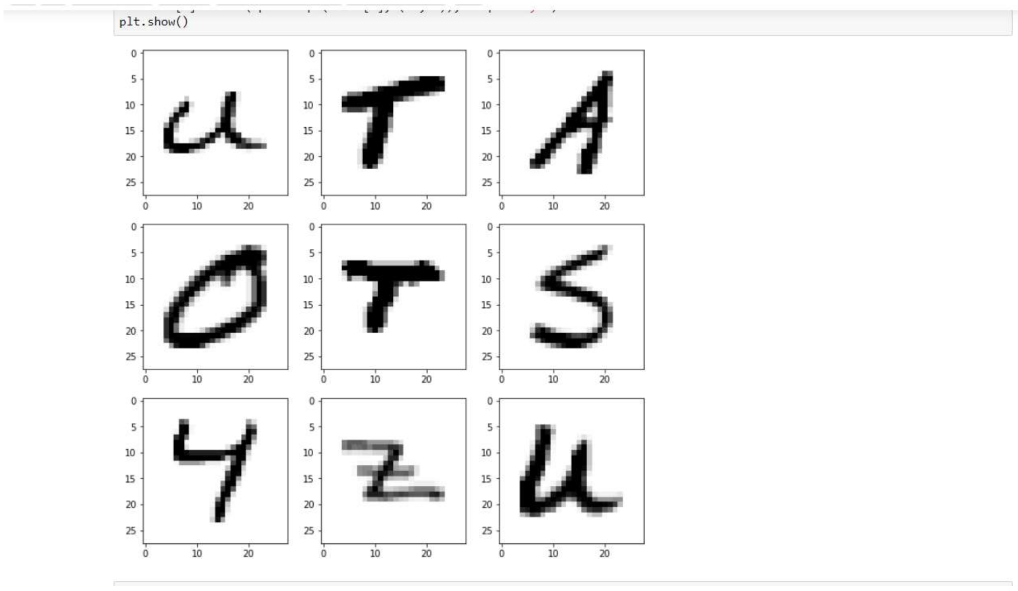
Shuffling the data and Displaying

We shuffle some of the images of the train set. The shuffling is done using the shuffle() function of sklearn.utils so that we can display some random images.

Next we create 9 plots in 3×3 shape using subplots() function & display the grayscale images of 9 alphabets using imshow() function.

```
In [*]: from sklearn.utils import shuffle
```

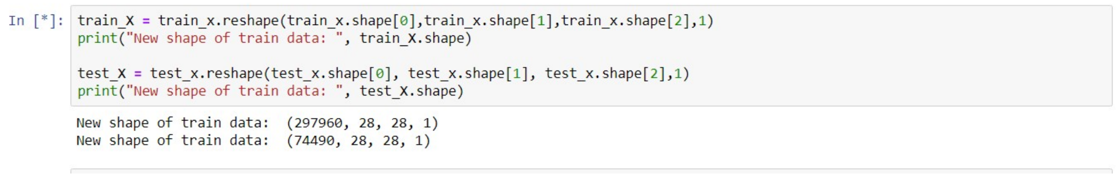
```
In [*]: shuff = shuffle(train_x[:100])

        fig, ax = plt.subplots(3,3, figsize = (10,10))
        axes = ax.flatten()

        for i in range(9):
            axes[i].imshow(np.reshape(shuff[i], (28,28)), cmap="Greys")
        plt.show()
```



```
plt.show()
```

Reshaping the training & test dataset so that it can be put in the model

We reshape the train & test image datasets using the reshape() function so that it is suitable for use in our CNN model that will be implemented for this project. We do this because CNN model requires that the input image data have only one column.



```
In [*]: train_X = train_x.reshape(train_x.shape[0],train_x.shape[1],train_x.shape[2],1)
        print("New shape of train data: ", train_X.shape)

        test_X = test_x.reshape(test_x.shape[0], test_x.shape[1], test_x.shape[2],1)
        print("New shape of train data: ", test_X.shape)

        New shape of train data:  (297960, 28, 28, 1)
        New shape of train data:  (74490, 28, 28, 1)
```

Week 7 (Creating CNN Model)

(14/07/2020 – 20/07/2020)

In the 7th week of my internship, I continued making the handwritten character recognition project. In this week, I created a Convolutional Neural Network (CNN) model that will be trained over a dataset containing images of alphabets and used in predicting characters from images of handwritten characters.

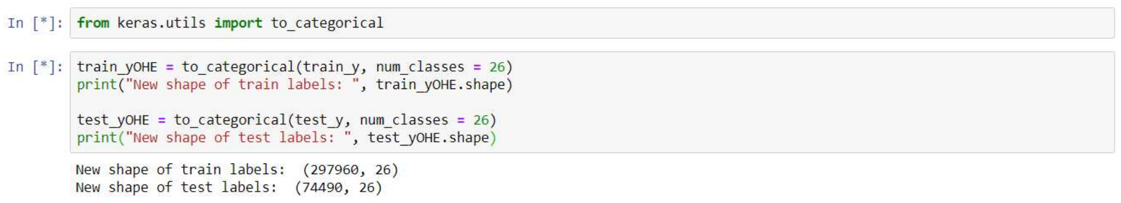
Convolutional Neural Networks are used to extract the features of the images using several layers of filters. It extracts the feature of an image and converts it into lower dimension without losing its characteristics.

There are five different layers in CNN

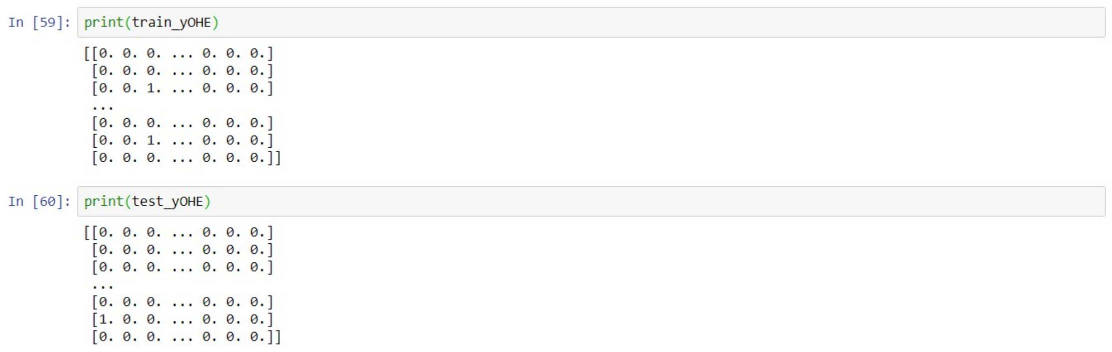
- Input layer: - Contains image data. Data needs to be reshaped into a single column.
- Convo layer: - features of the image are get extracted within this layer
- Pooling layer: - used to reduce the spatial volume of input image after convolution
- Dense/Fully connected (FC) layer: - It is used to classify images between different category by training.
- Output layer: - Output layer contains the labels in the form of a one-hot encoded vector.

Converting the labels to categorical values

We convert the single float values in the y label dataframe to categorical values using the to_categorical() function of keras.utils. This is done because the CNN model takes input of labels & generates the output as a vector of probabilities.



```
In [*]: from keras.utils import to_categorical

In [*]: train_yOHE = to_categorical(train_y, num_classes = 26)
        print("New shape of train labels: ", train_yOHE.shape)

        test_yOHE = to_categorical(test_y, num_classes = 26)
        print("New shape of test labels: ", test_yOHE.shape)

        New shape of train labels:  (297960, 26)
        New shape of test labels:  (74490, 26)
```



```
In [59]: print(train_yOHE)

         [[0. 0. 0. ... 0. 0. 0.]
          [0. 0. 0. ... 0. 0. 0.]
          [0. 0. 1. ... 0. 0. 0.]
          ...
          [0. 0. 0. ... 0. 0. 0.]
          [0. 0. 1. ... 0. 0. 0.]
          [0. 0. 0. ... 0. 0. 0.]]

In [60]: print(test_yOHE)

         [[0. 0. 0. ... 0. 0. 0.]
          [0. 0. 0. ... 0. 0. 0.]
          [0. 0. 0. ... 0. 0. 0.]
          ...
          [0. 0. 0. ... 0. 0. 0.]
          [1. 0. 0. ... 0. 0. 0.]
          [0. 0. 0. ... 0. 0. 0.]]
```

Creating a CNN Model

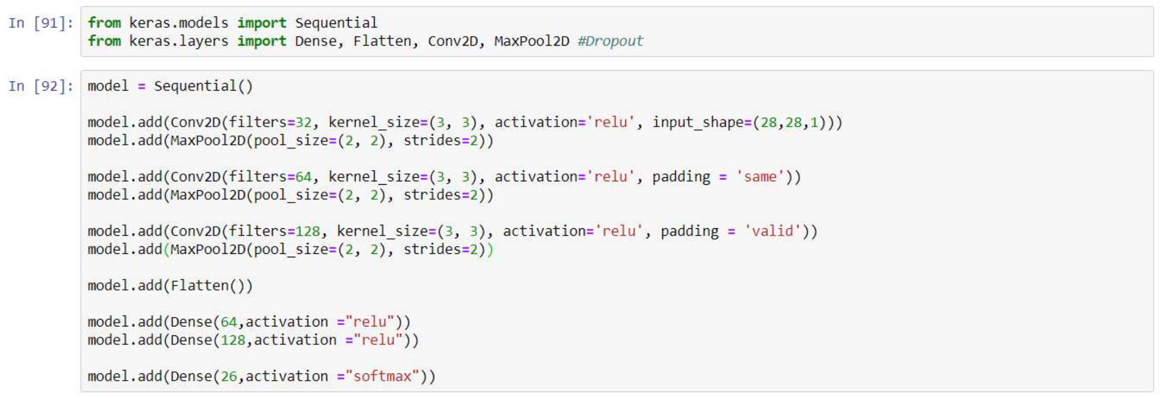
In the next step I created the CNN model. Input data fed to first Convo layer, output of that Convo layer acts as input for next Convo layer and so on. We will add up Convo layers followed by maxpool layers. The output of the maxpool layers and convolution layers are flattened into a vector of single dimension and are given as an input to the Dense layer (The fully connected network) to predict the correct labels

The Sequential() function is used to group a linear stack of layers into a tf.keras.Model. The add() function adds a layer instance on top of the layer stack.

We used the following functions from keras.layers: -

- Dense(): - Implements the dense/FC layer.
- Flatten(): - Flattens the input into a vector of single dimension
- Conv2D(): - This layer creates a convolution kernel that extracts the features of the images.
- MaxPool2D(): - This reduces the spatial volume of input image after convolution

This is the CNN model that I designed for training the model over the training dataset.



```
In [91]: from keras.models import Sequential
         from keras.layers import Dense, Flatten, Conv2D, MaxPool2D #Dropout

In [92]: model = Sequential()

         model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(28,28,1)))
         model.add(MaxPool2D(pool_size=(2, 2), strides=2))

         model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'))
         model.add(MaxPool2D(pool_size=(2, 2), strides=2))

         model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'valid'))
         model.add(MaxPool2D(pool_size=(2, 2), strides=2))

         model.add(Flatten())

         model.add(Dense(64,activation ="relu"))
         model.add(Dense(128,activation ="relu"))

         model.add(Dense(26,activation ="softmax"))
```

Compiling & Fitting Model

We used the Adam optimizers from keras.optimizers. Adam stands for Adaptive Moment Estimation. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first order and second-order moments. In addition to storing an exponentially decaying average of past squared gradients, Adam also keeps an exponentially decaying average of past gradients, like momentum.

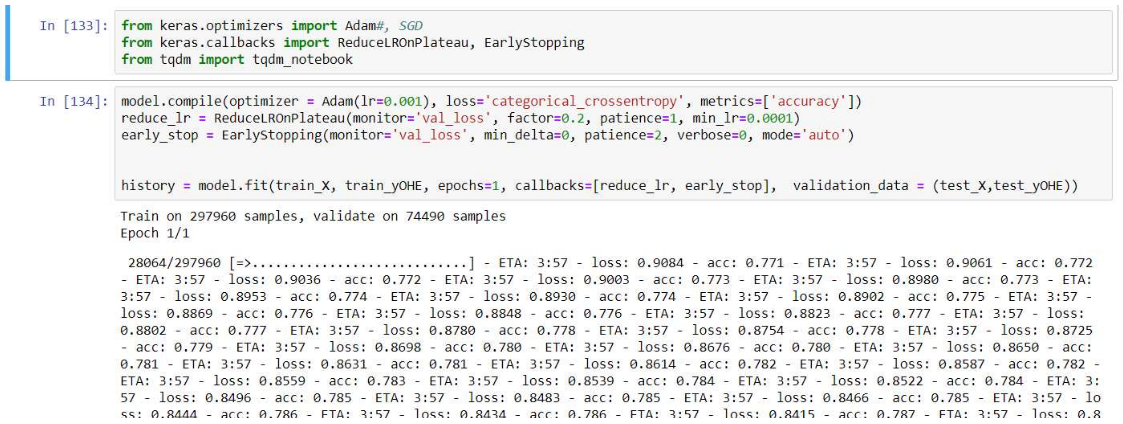
We use the following two callback (object that can perform actions at various stages of training) functions: -

- ReduceLROnPlateau: -Reduce learning rate when a metric has stopped improving.
- EarlyStopping: - Stop training when a monitored metric has stopped improving.

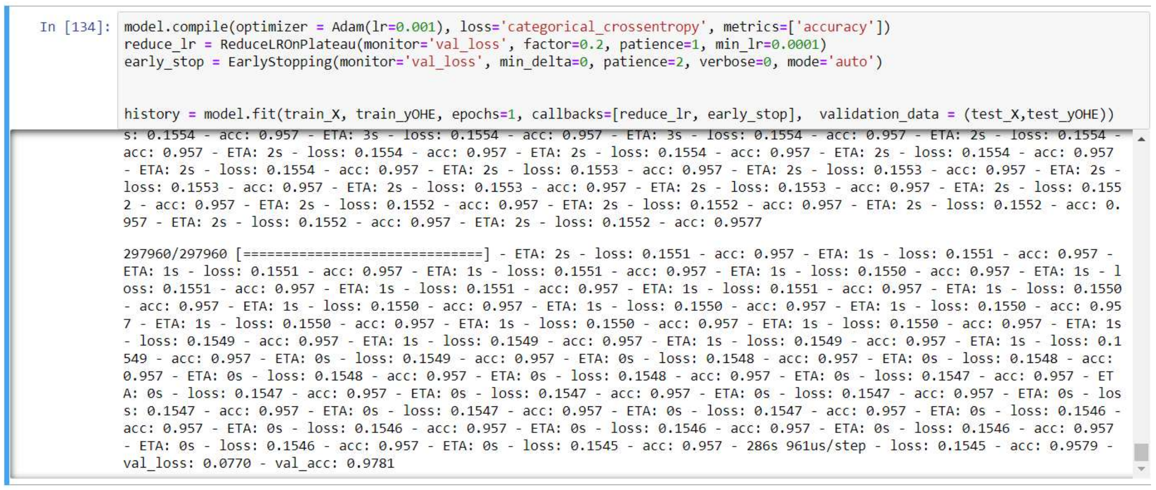
We use tqdm_notebook to add progress bars to the result of training the model.

First, we initialize all parameters and compile our model with Adam optimizer. While compiling the model we have defined the optimizing function & the loss function to be used for fitting. Then we setup the two callbacks to be used. After compiling our model, we train our model using the fit() method.

It takes approximately 4 minutes to train the model.



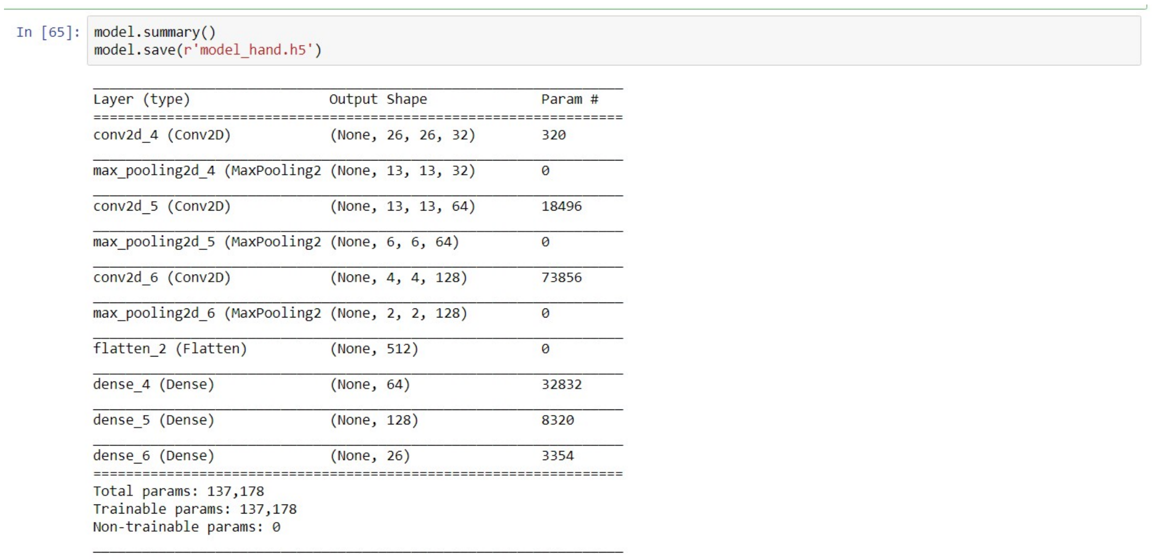
```
In [133]: from keras.optimizers import Adam#, SGD
          from keras.callbacks import ReduceLROnPlateau, EarlyStopping
          from tqdm import tqdm_notebook

In [134]: model.compile(optimizer = Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
          reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.0001)
          early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0, mode='auto')


          history = model.fit(train_X, train_yOHE, epochs=1, callbacks=[reduce_lr, early_stop],  validation_data = (test_X,test_yOHE))

          Train on 297960 samples, validate on 74490 samples
          Epoch 1/1

           28064/297960 [=>...........................] - ETA: 3:57 - loss: 0.9084 - acc: 0.771 - ETA: 3:57 - loss: 0.9061 - acc: 0.772
          - ETA: 3:57 - loss: 0.9036 - acc: 0.772 - ETA: 3:57 - loss: 0.9003 - acc: 0.773 - ETA: 3:57 - loss: 0.8980 - acc: 0.773 - ETA:
          3:57 - loss: 0.8953 - acc: 0.774 - ETA: 3:57 - loss: 0.8930 - acc: 0.774 - ETA: 3:57 - loss: 0.8902 - acc: 0.775 - ETA: 3:57 -
          loss: 0.8869 - acc: 0.776 - ETA: 3:57 - loss: 0.8848 - acc: 0.776 - ETA: 3:57 - loss: 0.8823 - acc: 0.777 - ETA: 3:57 - loss:
          0.8802 - acc: 0.777 - ETA: 3:57 - loss: 0.8780 - acc: 0.778 - ETA: 3:57 - loss: 0.8754 - acc: 0.778 - ETA: 3:57 - loss: 0.8725
          - acc: 0.779 - ETA: 3:57 - loss: 0.8698 - acc: 0.780 - ETA: 3:57 - loss: 0.8676 - acc: 0.780 - ETA: 3:57 - loss: 0.8650 - acc:
          0.781 - ETA: 3:57 - loss: 0.8631 - acc: 0.781 - ETA: 3:57 - loss: 0.8614 - acc: 0.782 - ETA: 3:57 - loss: 0.8587 - acc: 0.782 -
          ETA: 3:57 - loss: 0.8559 - acc: 0.783 - ETA: 3:57 - loss: 0.8539 - acc: 0.784 - ETA: 3:57 - loss: 0.8522 - acc: 0.784 - ETA: 3:
          57 - loss: 0.8496 - acc: 0.785 - ETA: 3:57 - loss: 0.8483 - acc: 0.785 - ETA: 3:57 - loss: 0.8466 - acc: 0.785 - ETA: 3:57 - lo
          ss: 0.8444 - acc: 0.786 - ETA: 3:57 - loss: 0.8434 - acc: 0.786 - ETA: 3:57 - loss: 0.8415 - acc: 0.787 - ETA: 3:57 - loss: 0.8
```



```
In [134]: model.compile(optimizer = Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
          reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.0001)
          early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0, mode='auto')


          history = model.fit(train_X, train_yOHE, epochs=1, callbacks=[reduce_lr, early_stop],  validation_data = (test_X,test_yOHE))
          s: 0.1554 - acc: 0.957 - ETA: 3s - loss: 0.1554 - acc: 0.957 - ETA: 3s - loss: 0.1554 - acc: 0.957 - ETA: 2s - loss: 0.1554 -
          acc: 0.957 - ETA: 2s - loss: 0.1554 - acc: 0.957 - ETA: 2s - loss: 0.1554 - acc: 0.957 - ETA: 2s - loss: 0.1554 - acc: 0.957
          - ETA: 2s - loss: 0.1554 - acc: 0.957 - ETA: 2s - loss: 0.1553 - acc: 0.957 - ETA: 2s - loss: 0.1553 - acc: 0.957 - ETA: 2s -
          loss: 0.1553 - acc: 0.957 - ETA: 2s - loss: 0.1553 - acc: 0.957 - ETA: 2s - loss: 0.1553 - acc: 0.957 - ETA: 2s - loss: 0.155
          2 - acc: 0.957 - ETA: 2s - loss: 0.1552 - acc: 0.957 - ETA: 2s - loss: 0.1552 - acc: 0.957 - ETA: 2s - loss: 0.1552 - acc: 0.
          957 - ETA: 2s - loss: 0.1552 - acc: 0.957 - ETA: 2s - loss: 0.1552 - acc: 0.9577

          297960/297960 [==============================] - ETA: 2s - loss: 0.1551 - acc: 0.957 - ETA: 1s - loss: 0.1551 - acc: 0.957 -
          ETA: 1s - loss: 0.1551 - acc: 0.957 - ETA: 1s - loss: 0.1551 - acc: 0.957 - ETA: 1s - loss: 0.1550 - acc: 0.957 - ETA: 1s - l
          oss: 0.1551 - acc: 0.957 - ETA: 1s - loss: 0.1551 - acc: 0.957 - ETA: 1s - loss: 0.1551 - acc: 0.957 - ETA: 1s - loss: 0.1550
          - acc: 0.957 - ETA: 1s - loss: 0.1550 - acc: 0.957 - ETA: 1s - loss: 0.1550 - acc: 0.957 - ETA: 1s - loss: 0.1550 - acc: 0.95
          7 - ETA: 1s - loss: 0.1550 - acc: 0.957 - ETA: 1s - loss: 0.1550 - acc: 0.957 - ETA: 1s - loss: 0.1550 - acc: 0.957 - ETA: 1s
          - loss: 0.1549 - acc: 0.957 - ETA: 1s - loss: 0.1549 - acc: 0.957 - ETA: 1s - loss: 0.1549 - acc: 0.957 - ETA: 1s - loss: 0.1
          549 - acc: 0.957 - ETA: 0s - loss: 0.1549 - acc: 0.957 - ETA: 0s - loss: 0.1548 - acc: 0.957 - ETA: 0s - loss: 0.1548 - acc:
          0.957 - ETA: 0s - loss: 0.1548 - acc: 0.957 - ETA: 0s - loss: 0.1548 - acc: 0.957 - ETA: 0s - loss: 0.1547 - acc: 0.957 - ET
          A: 0s - loss: 0.1547 - acc: 0.957 - ETA: 0s - loss: 0.1547 - acc: 0.957 - ETA: 0s - loss: 0.1547 - acc: 0.957 - ETA: 0s - los
          s: 0.1547 - acc: 0.957 - ETA: 0s - loss: 0.1547 - acc: 0.957 - ETA: 0s - loss: 0.1547 - acc: 0.957 - ETA: 0s - loss: 0.1546 -
          acc: 0.957 - ETA: 0s - loss: 0.1546 - acc: 0.957 - ETA: 0s - loss: 0.1546 - acc: 0.957 - ETA: 0s - loss: 0.1546 - acc: 0.957
          - ETA: 0s - loss: 0.1546 - acc: 0.957 - ETA: 0s - loss: 0.1545 - acc: 0.957 - 286s 961us/step - loss: 0.1545 - acc: 0.9579 -
          val_loss: 0.0770 - val_acc: 0.9781
```

Getting a model Summary and saving the model

We use model.summary() to see all parameters and shapes in each layers in our models. We also save the model as 'model_hand.h5' using model.save() function.



```
In [65]: model.summary()
         model.save(r'model_hand.h5')

         _________________________________________________________________
         Layer (type)                 Output Shape              Param #
         =================================================================
         conv2d_4 (Conv2D)            (None, 26, 26, 32)        320
         _________________________________________________________________
         max_pooling2d_4 (MaxPooling2 (None, 13, 13, 32)        0
         _________________________________________________________________
         conv2d_5 (Conv2D)            (None, 13, 13, 64)        18496
         _________________________________________________________________
         max_pooling2d_5 (MaxPooling2 (None, 6, 6, 64)          0
         _________________________________________________________________
         conv2d_6 (Conv2D)            (None, 4, 4, 128)         73856
         _________________________________________________________________
         max_pooling2d_6 (MaxPooling2 (None, 2, 2, 128)         0
         _________________________________________________________________
         flatten_2 (Flatten)          (None, 512)               0
         _________________________________________________________________
         dense_4 (Dense)              (None, 64)                32832
         _________________________________________________________________
         dense_5 (Dense)              (None, 128)               8320
         _________________________________________________________________
         dense_6 (Dense)              (None, 26)                3354
         =================================================================
         Total params: 137,178
         Trainable params: 137,178
         Non-trainable params: 0
         _________________________________________________________________
```

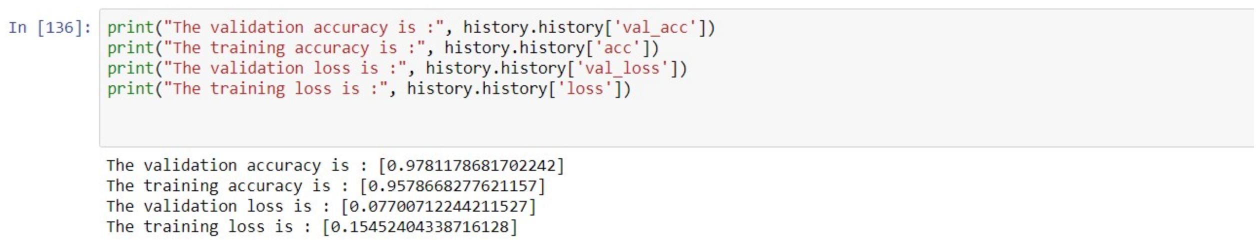
Week 8 (Results of Handwritten Character Recognition)

(21/07/2020 – 27/07/2020)

In the 8th and final week of my internship, I finalized the handwritten character recognition project and obtain accurate results.

Getting Accuracies & Losses for the Train & Validation set

We print out the training & validation accuracies along with the training & validation losses for character recognition.

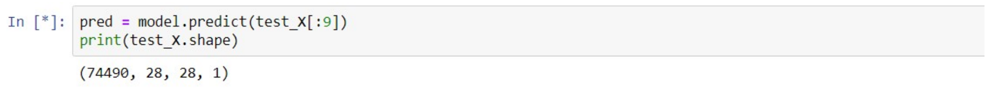


```
In [136]: print("The validation accuracy is :", history.history['val_acc'])
          print("The training accuracy is :", history.history['acc'])
          print("The validation loss is :", history.history['val_loss'])
          print("The training loss is :", history.history['loss'])
```

```
The validation accuracy is : [0.9781178681702242]
The training accuracy is : [0.9578668277621157]
The validation loss is : [0.07700712244211527]
The training loss is : [0.15452404338716128]
```

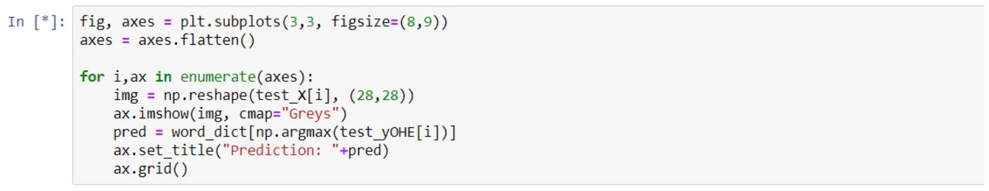
Doing Some Predictions on Test Data

We run our model on some images from the test data & display their predicted labels. We create 9 subplots of (3,3) shape & visualize some of the test dataset alphabets along with their predictions, that are made using the model.predict() function for text recognition.

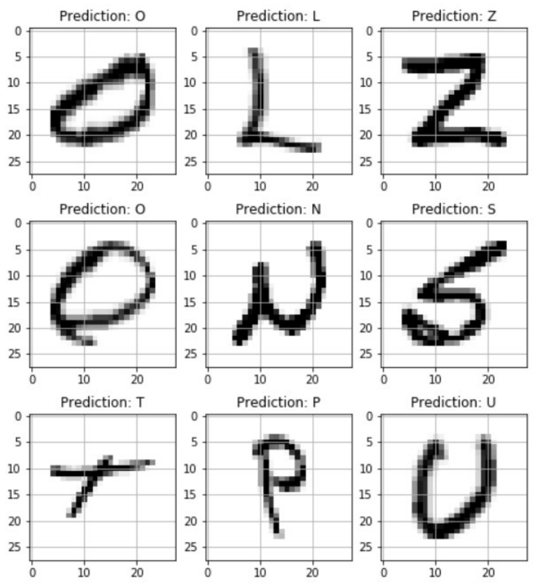


```
In [*]: pred = model.predict(test_X[:9])
        print(test_X.shape)
```

```
(74490, 28, 28, 1)
```



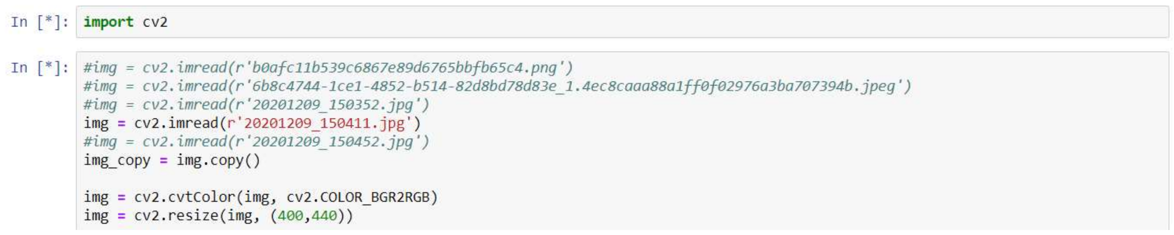
```
In [*]: fig, axes = plt.subplots(3,3, figsize=(8,9))
        axes = axes.flatten()

        for i,ax in enumerate(axes):
            img = np.reshape(test_X[i], (28,28))
            ax.imshow(img, cmap="Greys")
            pred = word_dict[np.argmax(test_yOHE[i])]
            ax.set_title("Prediction: "+pred)
            ax.grid()
```



Doing Prediction on External Image

Here we read an external image using cv2.imread() and made a copy of it that is to go through some processing to be fed to the model for the prediction.

The img read is then converted from BGR representation (as OpenCV reads the image in BGR format) to RGB for displaying the image, & is resized to our required dimensions that we want to display the image in.

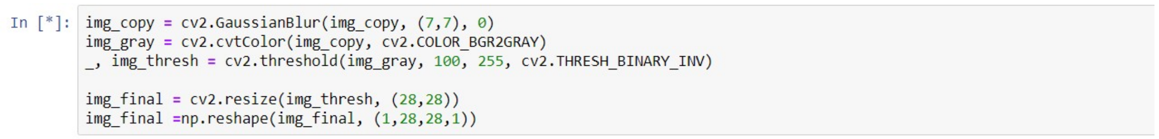


```
In [*]: import cv2

In [*]: #img = cv2.imread(r'b0afc11b539c6867e89d6765bbfb65c4.png')
        #img = cv2.imread(r'6b8c4744-1ce1-4852-b514-82d8bd78d83e_1.4ec8caaa88a1ff0f02976a3ba707394b.jpeg')
        #img = cv2.imread(r'20201209_150352.jpg')
        img = cv2.imread(r'20201209_150411.jpg')
        #img = cv2.imread(r'20201209_150452.jpg')
        img_copy = img.copy()

        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (400,440))
```

Next, we do some processing on the copied image (img_copy).

We convert the image from BGR to grayscale and apply thresholding to it. We do not need to apply a threshold we could use the grayscale to predict, but we do it to keep the image smooth without any sort of hazy grey colours in the image that could lead to wrong predictions.

The image is to be then resized using cv2.resize() function into the dimensions that the model takes as input, along with reshaping the image using np.reshape() so that it can be used as model input.

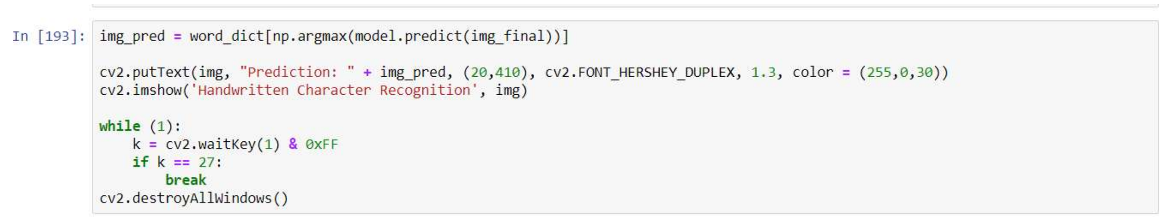


```
In [*]: img_copy = cv2.GaussianBlur(img_copy, (7,7), 0)
        img_gray = cv2.cvtColor(img_copy, cv2.COLOR_BGR2GRAY)
        _, img_thresh = cv2.threshold(img_gray, 100, 255, cv2.THRESH_BINARY_INV)

        img_final = cv2.resize(img_thresh, (28,28))
        img_final =np.reshape(img_final, (1,28,28,1))
```

Now we make a prediction using the processed image & use the np.argmax() function to get the index of the class with the highest predicted probability. Using this we get to know the exact character through the word_dict dictionary.

This predicted character is then displayed on the frame.

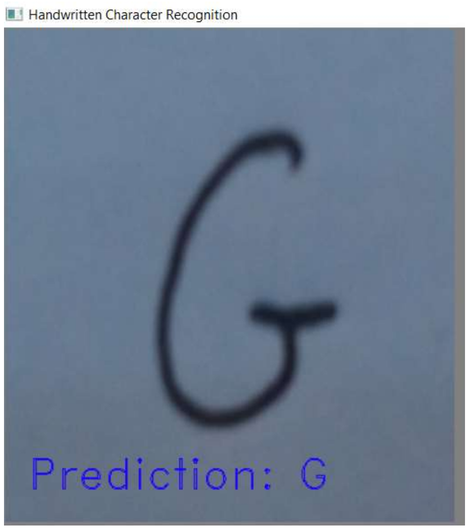
We also sett up a waitKey in a while loop that will be stuck in loop until Esc is pressed, & when it gets out of loop using cv2.destroyAllWindows() we destroy any active windows created to stop displaying the frame.



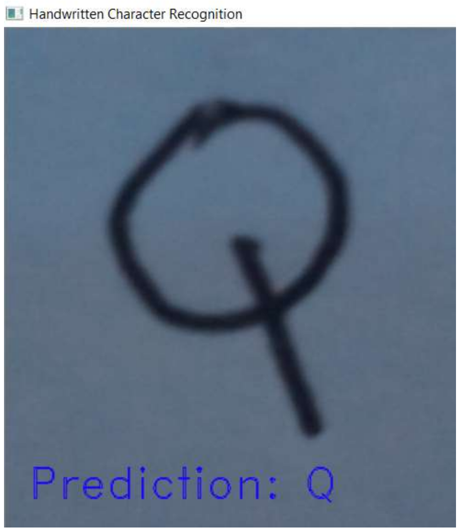
```
In [193]: img_pred = word_dict[np.argmax(model.predict(img_final))]

          cv2.putText(img, "Prediction: " + img_pred, (20,410), cv2.FONT_HERSHEY_DUPLEX, 1.3, color = (255,0,30))
          cv2.imshow('Handwritten Character Recognition', img)

          while (1):
              k = cv2.waitKey(1) & 0xFF
              if k == 27:
                  break
          cv2.destroyAllWindows()
```

Results

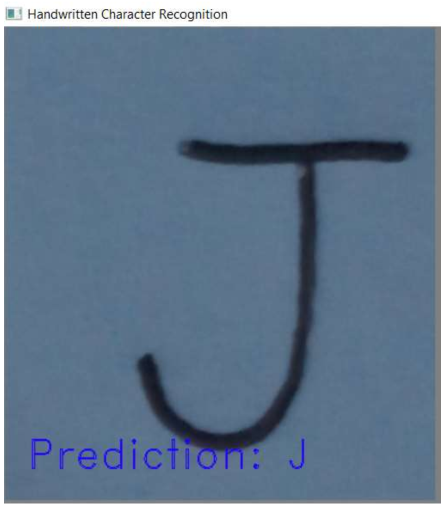
I ran the model on 6 external images of handwritten characters and the results were accurate.



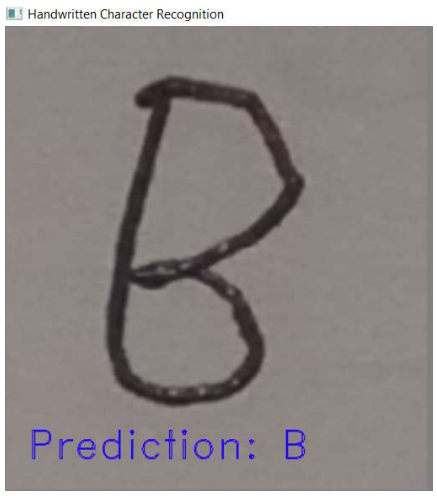
Prediction: G



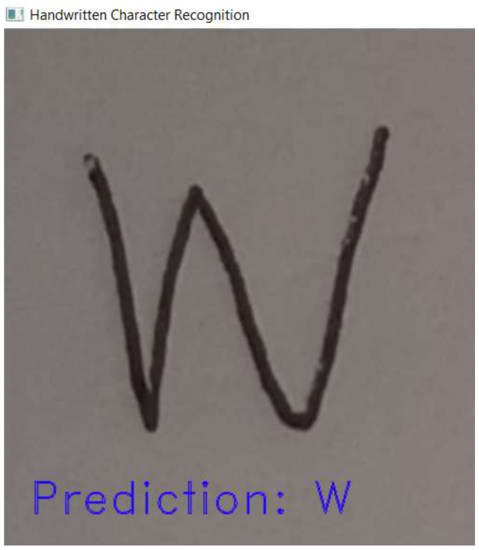
Prediction: Q



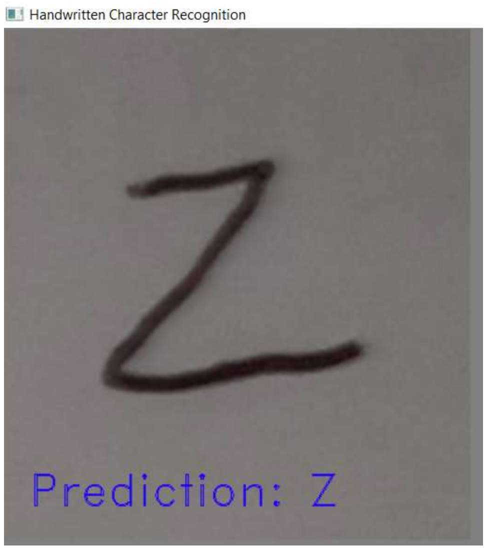
Prediction: J



Prediction: B



Prediction: W



Prediction: Z

Learning and Achievements

This internship was a great opportunity to learn about the basic concepts of Machine Learning and develop my skills for using various algorithms, methods and libraries used frequently in the domain of Machine Learning.

I was able to develop two projects during this internship. A Colour Detection project and A Handwritten Character Recognition project. I have learnt how to use various Python libraries and how to implement convolutional neural networks in the Python language.

I have gained confidence that I will be able to complete more complex Machine Learning Projects in the future using the knowledge I have gained during the internship.