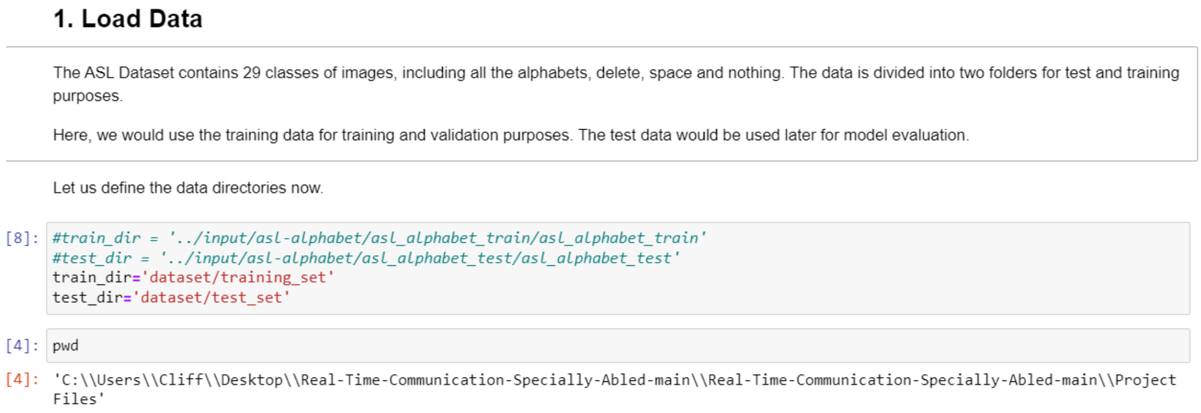
Import ImageDataGenerator Library And Configure It

Import ImageDataGenerator and create an instance for which include shearing, rescale, zooming, etc to make the model robust with different types of images.



1. Load Data

The ASL Dataset contains 29 classes of images, including all the alphabets, delete, space and nothing. The data is divided into two folders for test and training purposes.

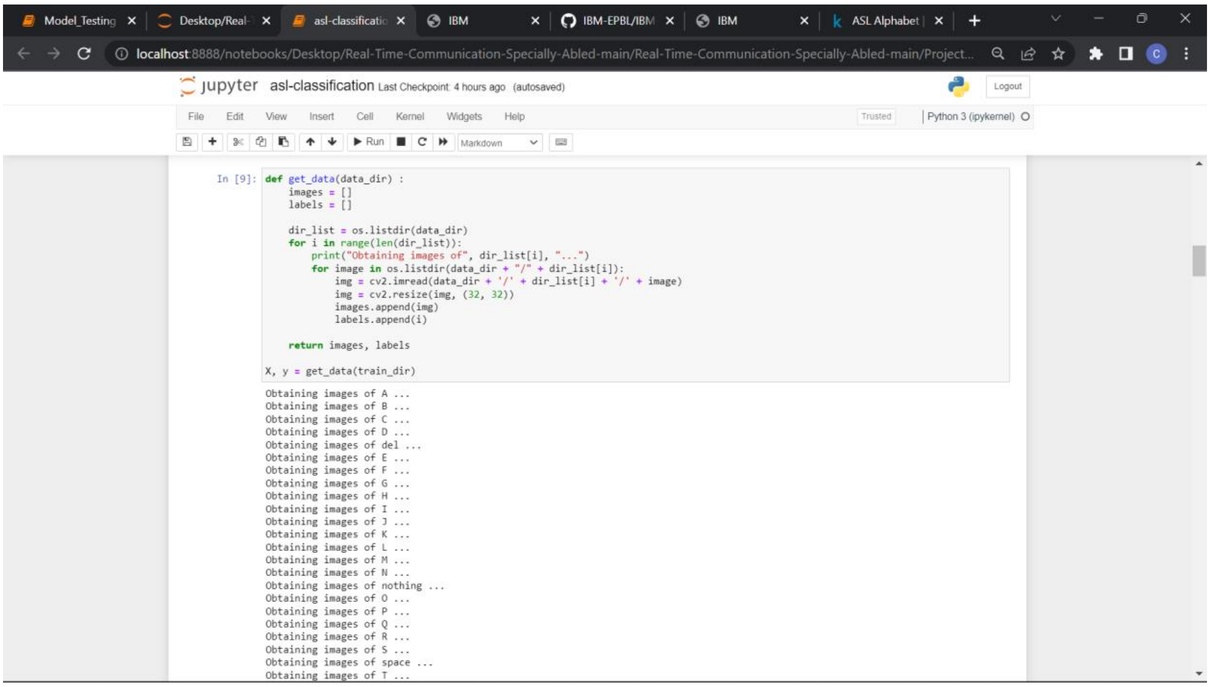
Here, we would use the training data for training and validation purposes. The test data would be used later for model evaluation.

Let us define the data directories now.

```
[8]: #train_dir = '../input/asl-alphabet/asl_alphabet_train/asl_alphabet_train'
     #test_dir = '../input/asl-alphabet/asl_alphabet_test/asl_alphabet_test'
     train_dir='dataset/training_set'
     test_dir='dataset/test_set'
```

```
[4]: pwd
```

```
[4]: 'C:\\Users\\Cliff\\Desktop\\Real-Time-Communication-Specially-Abled-main\\Real-Time-Communication-Specially-Abled-main\\Project
     Files'
```



Model_Testing
Desktop/Real-
asl-classificatio
IBM
IBM-EPBL/IBM
IBM
ASL Alphabet |
localhost:8888/notebooks/Desktop/Real-Time-Communication-Specially-Abled-main/Real-Time-Communication-Specially-Abled-main/Project...
jupyter
asl-classification Last Checkpoint: 4 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Markdown

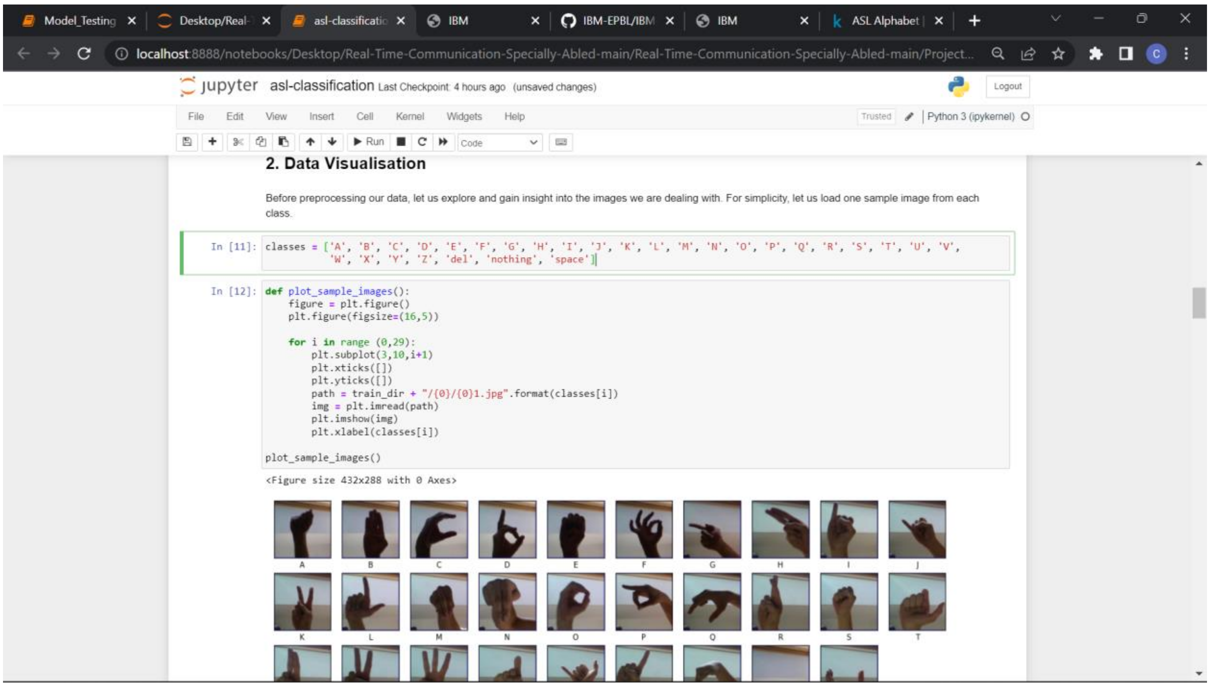
```
In [9]: def get_data(data_dir) :
            images = []
            labels = []

            dir_list = os.listdir(data_dir)
            for i in range(len(dir_list)):
                print("Obtaining images of", dir_list[i], "...")
                for image in os.listdir(data_dir + "/" + dir_list[i]):
                    img = cv2.imread(data_dir + '/' + dir_list[i] + '/' + image)
                    img = cv2.resize(img, (32, 32))
                    images.append(img)
                    labels.append(i)

            return images, labels

        X, y = get_data(train_dir)
```

```
Obtaining images of A ...
Obtaining images of B ...
Obtaining images of C ...
Obtaining images of D ...
Obtaining images of del ...
Obtaining images of E ...
Obtaining images of F ...
Obtaining images of G ...
Obtaining images of H ...
Obtaining images of I ...
Obtaining images of J ...
Obtaining images of K ...
Obtaining images of L ...
Obtaining images of M ...
Obtaining images of N ...
Obtaining images of nothing ...
Obtaining images of O ...
Obtaining images of P ...
Obtaining images of Q ...
Obtaining images of R ...
Obtaining images of S ...
Obtaining images of space ...
Obtaining images of T ...
```


Model_Testing
Desktop/Real-
asl-classificatio
IBM
IBM-EPBL/IBM
IBM
ASL Alphabet |
localhost:8888/notebooks/Desktop/Real-Time-Communication-Specially-Abled-main/Real-Time-Communication-Specially-Abled-main/Project...
jupyter asl-classification Last Checkpoint: 4 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T

2. Data Visualisation

Before preprocessing our data, let us explore and gain insight into the images we are dealing with. For simplicity, let us load one sample image from each class.

```
In [11]: classes = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
                    'W', 'X', 'Y', 'Z', 'del', 'nothing', 'space']
```

```
In [12]: def plot_sample_images():
             figure = plt.figure()
             plt.figure(figsize=(16,5))

             for i in range (0,29):
                 plt.subplot(3,10,i+1)
                 plt.xticks([])
                 plt.yticks([])
                 path = train_dir + "/{0}/{0}1.jpg".format(classes[i])
                 img = plt.imread(path)
                 plt.imshow(img)
                 plt.xlabel(classes[i])

         plot_sample_images()
```

```
<Figure size 432x288 with 0 Axes>
```

3. Data Preprocessing

Before feeding the data to our model, we convert into numpy arrays and normalise the values by dividing the image pixel values by 255.

Then, we divide our training data into training and testing sets to be used by the model.

```
In [13]: def preprocess_data(X, y):
             np_X = np.array(X)
             normalised_X = np_X.astype('float32')/255.0

             label_encoded_y = utils.to_categorical(y)

             x_train, x_test, y_train, y_test = train_test_split(normalised_X, label_encoded_y, test_size = 0.1)

             return x_train, x_test, y_train, y_test

         x_train, x_test, y_train, y_test = preprocess_data(X, y)
```

Let us confirm the size of training and testing data.

```
In [14]: print("Training data:", x_train.shape)
         print("Test data:", x_test.shape)
```

```
Training data: (78300, 32, 32, 3)
Test data: (8700, 32, 32, 3)
```