Handwritten Alphabet Recognition

Objective

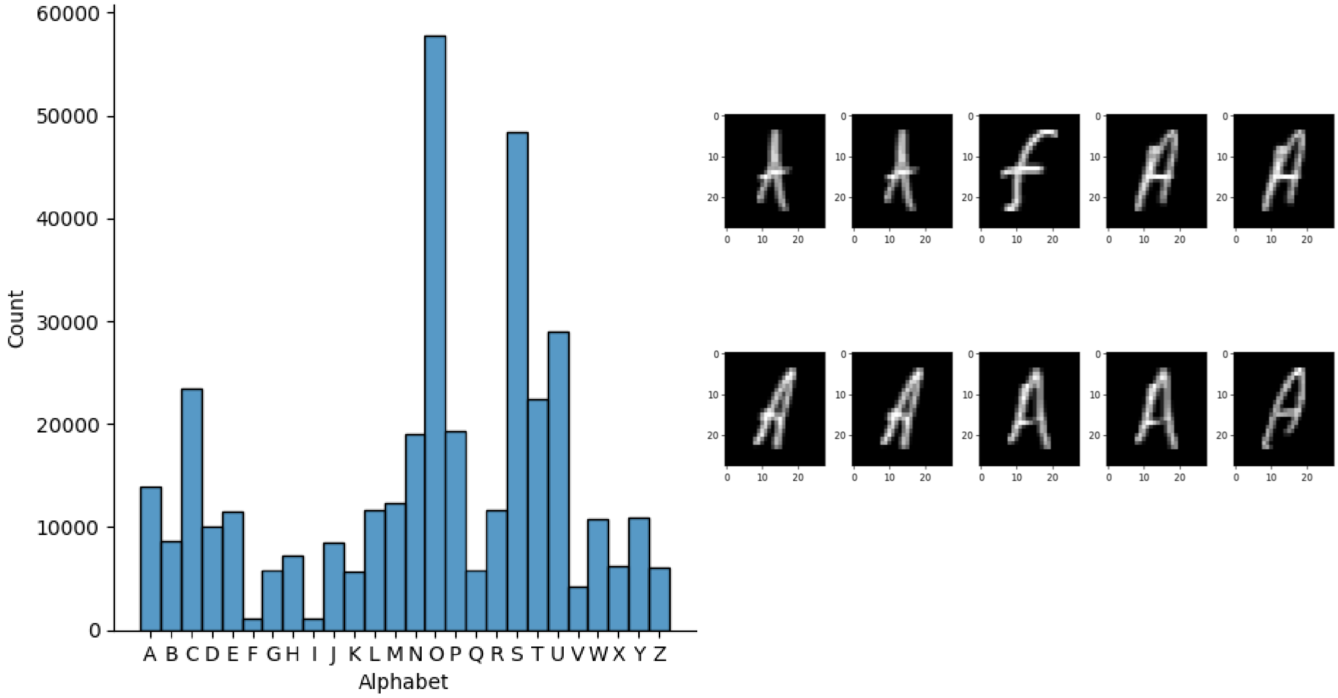
To build a CNN model which can detect hand written alphabet.

Data Description

The dataset used for this project is a CSV file named ‘A_Z Handwritten Data.csv’ that contain the information of 372450 gray scale image of alphabets in 28x28 pixel each in which first column is the alphabet encoded in integer from 0 to 25.

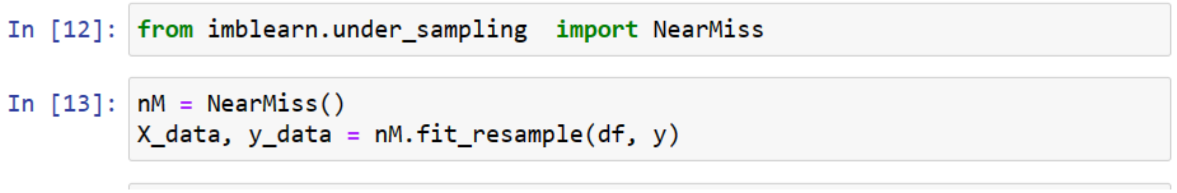
Dataset source: https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format

Data Distribution & its sample



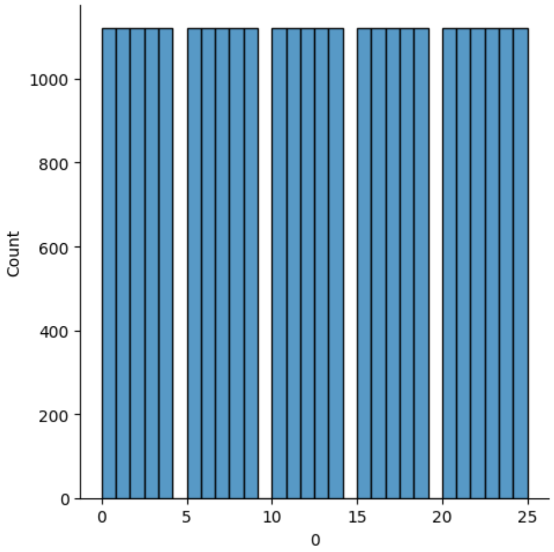
Data Preprocessing

- To deal with imbalance of data, Near-Miss algorithm were used to balance the dataset

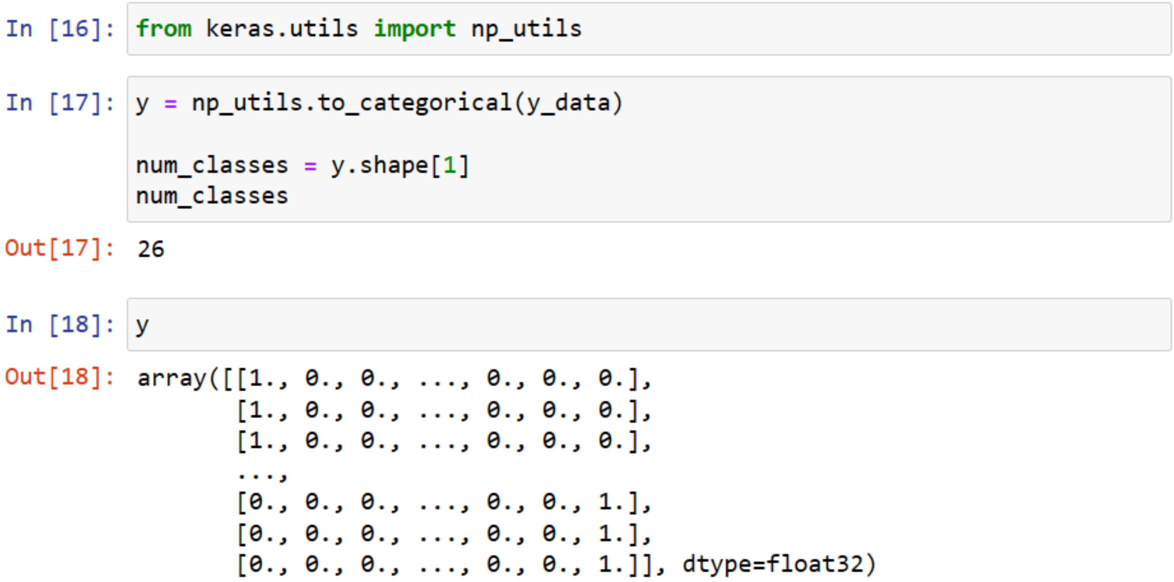


```
In [12]: from imblearn.under_sampling  import NearMiss

In [13]: nM = NearMiss()
         X_data, y_data = nM.fit_resample(df, y)
```

- Dataset after resampling

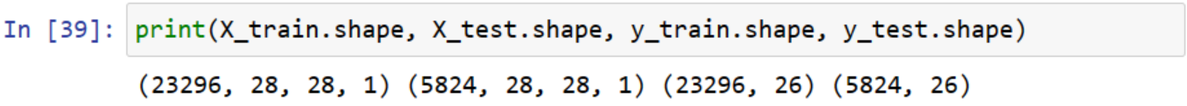


- Output were encoded into nominal order using kereas.utils



```
In [16]: from keras.utils import np_utils

In [17]: y = np_utils.to_categorical(y_data)

         num_classes = y.shape[1]
         num_classes

Out[17]: 26

In [18]: y

Out[18]: array([[1., 0., 0., ..., 0., 0., 0.],
                [1., 0., 0., ..., 0., 0., 0.],
                [1., 0., 0., ..., 0., 0., 0.],
                ...,
                [0., 0., 0., ..., 0., 0., 1.],
                [0., 0., 0., ..., 0., 0., 1.],
                [0., 0., 0., ..., 0., 0., 1.]], dtype=float32)
```

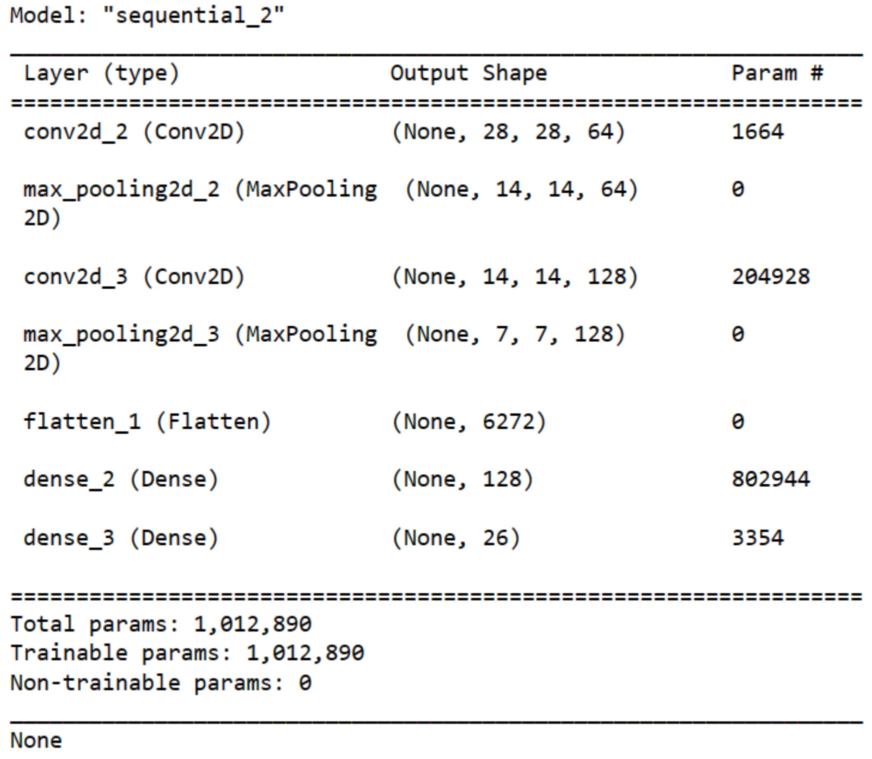
- Dataset were also normalized by dividing them with 255
- At last, dataset were split into train and test data with test size of 20% using sklearn.model_selection having final shape of dataset as follow



```
In [39]: print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

         (23296, 28, 28, 1) (5824, 28, 28, 1) (23296, 26) (5824, 26)
```

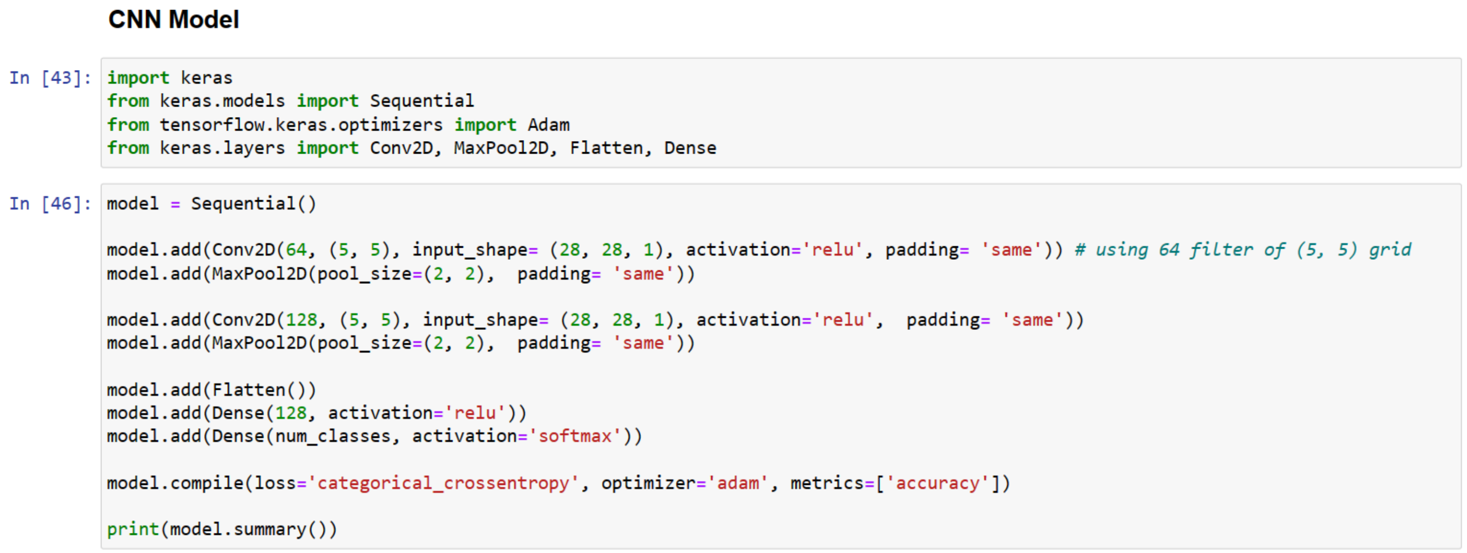
Model Building

- CNN sequential model was built with following Architecture.



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_2 (Conv2D)           (None, 28, 28, 64)        1664      
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 14, 14, 64)       0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 14, 14, 128)       204928    
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0         
 2D)                                                             
                                                                 
 flatten_1 (Flatten)         (None, 6272)              0         
                                                                 
 dense_2 (Dense)             (None, 128)               802944    
                                                                 
 dense_3 (Dense)             (None, 26)                3354      
                                                                 
=================================================================
Total params: 1,012,890
Trainable params: 1,012,890
Non-trainable params: 0
_________________________________________________________________
None
```

- Code of the architecture



CNN Model

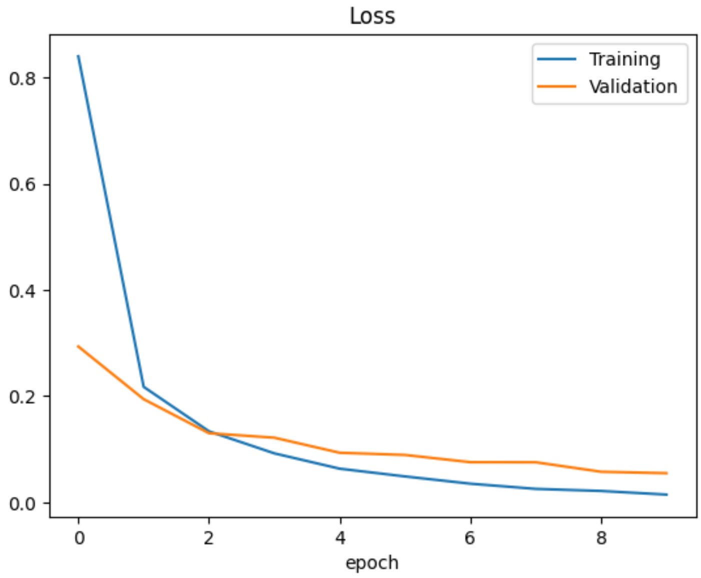
```
In [43]: import keras
         from keras.models import Sequential
         from tensorflow.keras.optimizers import Adam
         from keras.layers import Conv2D, MaxPool2D, Flatten, Dense
```

```
In [46]: model = Sequential()

         model.add(Conv2D(64, (5, 5), input_shape= (28, 28, 1), activation='relu', padding= 'same')) # using 64 filter of (5, 5) grid
         model.add(MaxPool2D(pool_size=(2, 2),  padding= 'same'))

         model.add(Conv2D(128, (5, 5), input_shape= (28, 28, 1), activation='relu',  padding= 'same'))
         model.add(MaxPool2D(pool_size=(2, 2),  padding= 'same'))

         model.add(Flatten())
         model.add(Dense(128, activation='relu'))
         model.add(Dense(num_classes, activation='softmax'))

         model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

         print(model.summary())
```

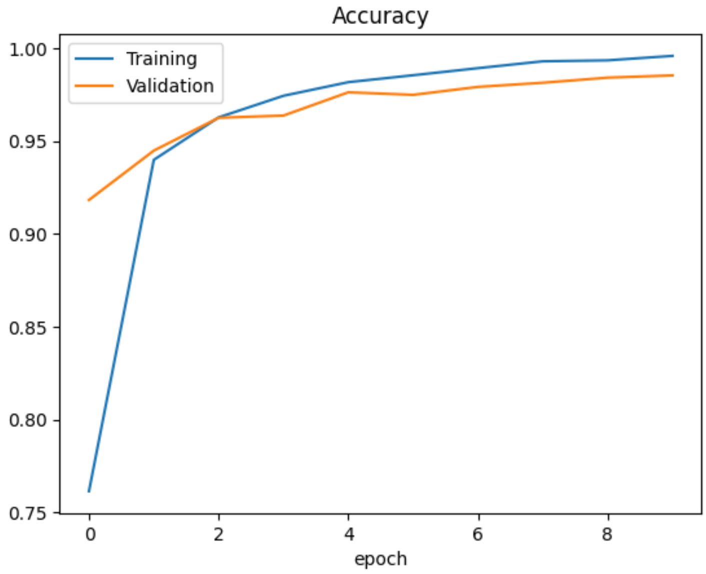
Results

- Accuracy of the CNN model on test data: 98.54%
- Accuracy of the CNN model on train data: 99.59%
- Accuracy of the CNN model on overall data: 99.53%

- Plot of training loss and validation loss



- Plot of training accuracy and validation accuracy



Conclusion

The CNN model for hand written alphabet recognition was build using keras library which gives the accuracy of 99%.

Team Details

Group 8

Name	Branch	Semester/Year
PINTU RAJ	Information Technology	5th Sem/3rd year
AYUSH KUMAR	Information Technology	5th Sem/3rd year
GAUTAM KUMAR PANDEY	Information Technology	5th Sem/3rd year