Practical-4

Objective :

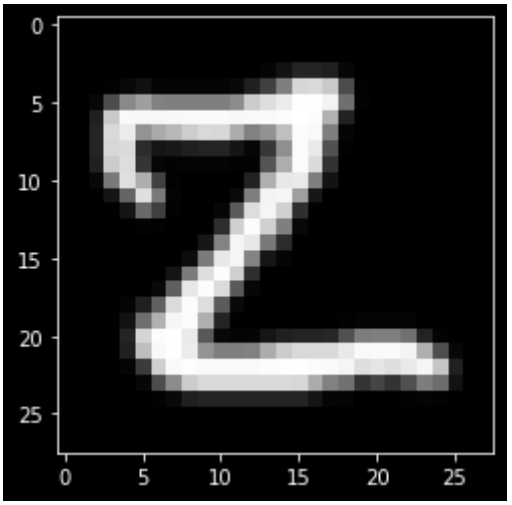
Design a character recognition system using neural network.

Description:

Design and build a convolutional neural network for recognizing characters which are trained using handwritten character set published in EMNIST dataset.

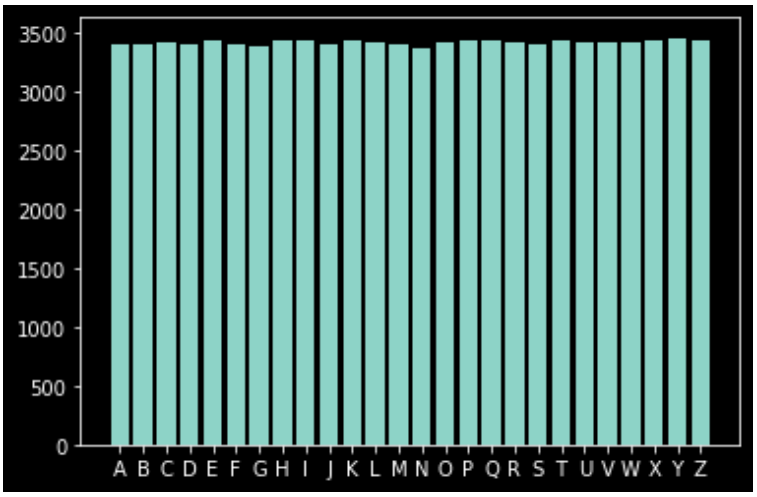
Dataset:

1. Name: EMNIST Letters dataset.
2. Source: https://www.kaggle.com/crawford/emnist
3. Info:
    - The EMNIST Letters dataset merges a balanced set of the uppercase and lowercase letters into a single 26-class.
    - This samples in dataset were collected form high-school students and the census employees.
    - Data is collected from 145,600 samples and is balanced into 26 classes.
    - This dataset contains 88800 train and test images each of size 28x28.
4. A sample form data set

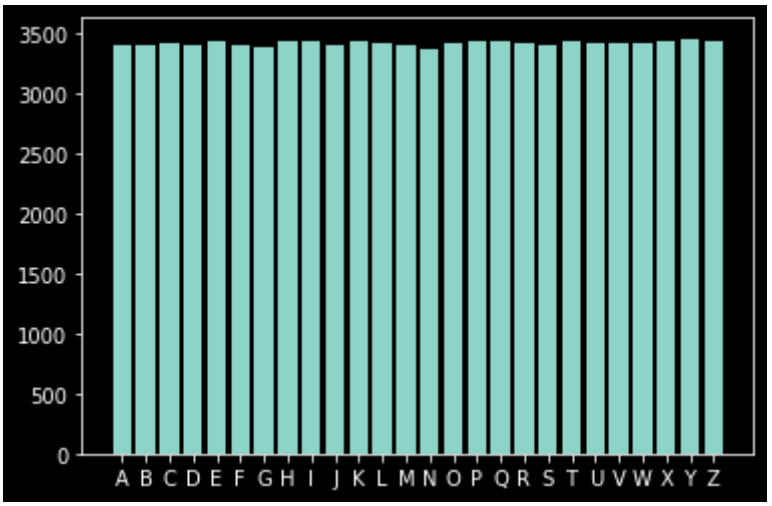


5. Training data and testing data are disjoint.

6. Data distribution in training data.



7. Data distribution in testing data.



Procedure:

1. Read testing and training data.
2. Since data(x) and class are in same file, slice them into x and y
3. Load labels meta and make dictionary to map it with actual label
4. Preprocessing
    - Apply black and white filter
    - Since the images in dataset are in -90deg orientation, rotate it by 90 d
    - Normalize image by using (0,1) technique
    - Reshape image from 28x28 to 28x28x1 to explicitly mention dimension
    - Convert y to categorical
5. Build CNN
    - The CNN Architecture is as following
        - ★ Convolution layer (64 filter of 3x3 size, relu).
        - ★ Convolution layer (32 filter of 3x3 size, relu).
        - ★ Maxpooling.

        - ★ Convolution layer (32 filter of 3x3 size, relu).
        - ★ Convolution layer (16 filter of 3x3 size, relu) .
        - ★ Flatten generated output for dense layer.
        - ★ Dense layer (128 neurons ,relu).
        - ★ Dense layer for classification (26, softmax) .
    - Optimizer: adam
    - Loss: categorical_crossentropy
    - Batch size: 128
    - Epochs: 10
    - Validation split: 20%

6. Use testing data to calculate accuracy.

Implementation:

```
# 1 loading data

train_data = pd.read_csv('data/emnist-letters-train.csv',header=None)
test_data = pd.read_csv('data/emnist-letters-test.csv',header=None)

# 2 split

x = np.array(train_data.iloc[:,1:].values)
y = np.array(train_data.iloc[:,0].values)
y = y-1

# 3 load labels

label_data = pd.read_csv("data/emnist-balancedmapping.txt",delimiter = ' ',header=None)
labels_dict = dict()
for i in range(label_data.shape[0]):
    labels_dict[i] = label_data.loc[i][1]
labels_dict

# 4 preprocessing

 def rotate(image):
      image = image.reshape(28, 28)
     image = np.fliplr(image)
    image = np.rot90(image)
   return image

 x = np.apply_along_axis(rotate, 1, x)
 x = x/255.0
 x = x.reshape(-1,28,28,1)
 y = to_categorical(y, 26)
```

```
# 5 cnn

model=Sequential()
model.add(Conv2D(64,(3,3),input_shape=(28,28,1),activation='relu'))
model.add(Conv2D(32,(3,3),activation='relu')) model.add(MaxPool2D())
model.add(Conv2D(32,(3,3),activation='relu')) model.add(Conv2D(16,(3,3),activation='relu'))
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(26,activation='softmax'))
model.summary()
model.compile(optimizer='adam',metrics=['accuracy'],loss='categorical_crossentrop y')
model_cnn = model.fit(x,y,batch_size=128,epochs=10,validation_split=0.2)

# 6 Testing

x_test = np.array(train_data.iloc[:,1:].values)
y_test = np.array(train_data.iloc[:,0].values)
y_test = y_test-1
x_test = np.apply_along_axis(rotate, 1, x_test)
x_test = x_test/255.0
x_test = x_test.reshape(-1,28,28,1)
preditctions = cnn.predict(x_test)
result = np.argmax(preditctions,axis=1)
```

Output:

Training metrices:

- loss:0.0944
- accuracy: 0.9620
- validation loss: 0.2707
- validation accuracy: 0.9278

Testing accuracy: 0.9584