pip install tensorflow

import matplotlib.pyplot as plt

import cv2

import numpy as np

from keras.models import Sequential

from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout

from tensorflow.keras.optimizers import SGD, Adam

from keras.callbacks import ReduceLROnPlateau, EarlyStopping

from tensorflow.keras.utils import to\_categorical

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.utils import shuffle

df = pd.read\_csv('A\_Z Handwritten Data.csv')

df.head(10)

X = df.drop('0',axis = 1)

y = df['0']

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)

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'}

y\_int = np.int0(y)

count = np.zeros(26, dtype='int')

for i in y\_int:

    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()

shuff = shuffle(train\_x[:100])

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

axes = ax.flatten()

for i in range(9):

    \_, shu = cv2.threshold(shuff[i], 30, 200, cv2.THRESH\_BINARY)

    axes[i].imshow(np.reshape(shuff[i], (28,28)), cmap="Greys")

plt.show()

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)

train\_yOHE = to\_categorical(train\_y, num\_classes = 26, dtype='int')

print("New shape of train labels: ", train\_yOHE.shape)

test\_yOHE = to\_categorical(test\_y, num\_classes = 26, dtype='int')

print("New shape of test labels: ", test\_yOHE.shape)

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"))

model.compile(optimizer = Adam(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(train\_X, train\_yOHE, epochs=1,  validation\_data = (test\_X,test\_yOHE))

model.summary()

model.save(r'model\_hand.h5')

print("The validation accuracy is :", history.history['val\_accuracy'])

print("The training accuracy is :", history.history['accuracy'])

print("The validation loss is :", history.history['val\_loss'])

print("The training loss is :", history.history['loss'])

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()