# CNN Code:

|  |
| --- |
| import numpy as np  import pandas as pd  import keras  from keras.models import Sequential  from keras.layers import Conv2D, Dense, MaxPool2D, Dropout, Flatten, AveragePooling2D  from keras.optimizers import Adam  from keras.preprocessing.image import ImageDataGenerator  import os  from keras.utils.np\_utils import to\_categorical  import cv2  import time  import tensorflow as tf  !unzip '/content/Reduced MNIST Data.zip'  def create\_dataset(img\_folder):    img\_data\_array=[]  class\_name=[]    for dir1 in os.listdir(img\_folder):  for file in os.listdir(os.path.join(img\_folder, dir1)):  image\_path= os.path.join(img\_folder, dir1, file)  image= cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)  image=np.array(image)  image = image.astype('double')  image /= 255  img\_data\_array.append(image)  class\_name.append(dir1)  return img\_data\_array, class\_name  activation\_type = 'relu'  filters\_tried = 6  model = Sequential()  model.add(Conv2D(filters=filters\_tried, kernel\_size=(5,5), padding='same', strides=(1, 1), activation=activation\_type, input\_shape=(28, 28, 1)))  model.add(AveragePooling2D(pool\_size=(2, 2), strides=(2, 2)))  model.add(Conv2D(filters=16, kernel\_size=(5,5), strides=(1, 1), padding='valid', activation=activation\_type))  model.add(AveragePooling2D(pool\_size=(2, 2), strides=(2, 2)))  model.add(Flatten())  model.add(Dense(120, activation=activation\_type))  model.add(Dense(84, activation=activation\_type))  model.add(Dense(10, activation='softmax'))  model.build()  model.summary()  filters\_tried = 8  activation\_type = 'sigmoid'  model2 = Sequential()  model2.add(Conv2D(filters=filters\_tried, kernel\_size=(5,5), padding='same', strides=(1, 1), activation=activation\_type, input\_shape=(28, 28, 1)))  model2.add(AveragePooling2D(pool\_size=(2, 2), strides=(2, 2)))  model2.add(Conv2D(filters=16, kernel\_size=(5,5), strides=(1, 1), padding='valid', activation=activation\_type))  model2.add(AveragePooling2D(pool\_size=(2, 2), strides=(2, 2)))  model2.add(Flatten())  model2.add(Dense(120, activation=activation\_type))  model2.add(Dense(84, activation=activation\_type))  model2.add(Dense(10, activation='softmax'))  model2.build()  model2.summary()  adam = Adam(lr=0.01)  model.compile(optimizer='adam',loss=tf.keras.losses.categorical\_crossentropy,metrics=['accuracy'])  model2.compile(optimizer='adam',loss=tf.keras.losses.categorical\_crossentropy,metrics=['accuracy'])  # model.fit(X\_train ,Y\_train, batch\_size=128, steps\_per\_epoch=len(X\_train)/100, epochs=30)  **# Training data set part**  img\_data, class\_name = create\_dataset (r'/content/Reduced MNIST Data/Reduced Trainging data')  target\_dict={k: v for v, k in enumerate(np.unique(class\_name))}  target\_val= [target\_dict[class\_name[i]] for i in range(len(class\_name))]  X\_train = np.array(img\_data).astype('double')  Y\_train = np.array(list(map(int,target\_val)))  nsamples, nx, ny = X\_train.shape  X\_train = X\_train.reshape((nsamples,nx\*ny))  X\_train = X\_train.reshape(X\_train.shape[0], 28, 28, 1)  Y\_train = to\_categorical(Y\_train)  callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)  t0= time.process\_time ()  model.fit(x= X\_train, y=Y\_train, epochs=25, batch\_size=128, callbacks=[callback])  t1\_svm\_linear = time.process\_time () - t0  print("Time elapsed for Training:: ", t1\_svm\_linear, " sec")  t0= time.process\_time ()  model2.fit(x= X\_train, y=Y\_train, epochs=40, batch\_size=128, callbacks=[callback])  t1\_svm\_linear = time.process\_time () - t0  print("Time elapsed for Training:: ", t1\_svm\_linear, " sec")  **## Test dataset part**  img\_data\_test, class\_name\_test = create\_dataset(r'/content/Reduced MNIST Data/Reduced Testing data')  target\_dict\_test={k: v for v, k in enumerate(np.unique(class\_name\_test))}  target\_val\_test= [target\_dict\_test[class\_name\_test[i]] for i in range(len(class\_name\_test))]  X\_test = np.array(img\_data\_test)  y\_test = np.array(list(map(int, target\_val\_test)))  nsamples, nx, ny = X\_test.shape  X\_d2\_test = X\_test.reshape((nsamples,nx\*ny))  X\_test = X\_d2\_test.reshape(X\_d2\_test.shape[0], 28, 28, 1)  Y\_test = to\_categorical(y\_test)  t0= time.process\_time ()  score = model.evaluate(X\_test, Y\_test, batch\_size=32)  t1\_svm\_linear = time.process\_time () - t0  print("Time elapsed for Testing:: ", t1\_svm\_linear, " sec")  t0= time.process\_time ()  score = model2.evaluate(X\_test, Y\_test, batch\_size=32)  t1\_svm\_linear = time.process\_time () - t0  print("Time elapsed for Testing:: ", t1\_svm\_linear, " sec") |