Mujtaba Shahid Faizi

BSCS-5A

#131818

Lab 7 of Computer Vision

**CODE:**

**from** \_\_future\_\_ **import** print\_function  
**import** keras  
**from** keras.datasets **import** mnist  
**from** keras.models **import** Sequential  
**from** keras.layers **import** Dense, Dropout, Flatten  
**from** keras.layers **import** Conv2D, MaxPooling2D  
**from** keras **import** backend **as** K  
**import** numpy **as** np  
**from** scipy **import** interp  
**import** matplotlib.pyplot **as** plt  
**from** itertools **import** cycle  
**from** sklearn.metrics **import** roc\_curve, auc  
  
*#---------------------------TRAINING MODEL----------------------------*batch\_size = 128  
num\_classes = 10  
epochs = 10  
  
*# input image dimensions*img\_rows, img\_cols = 28, 28  
  
*# the data, split between train and test sets*(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()  
  
**if** K.image\_data\_format() == **'channels\_first'**:  
 x\_train = x\_train.reshape(x\_train.shape[0], 1, img\_rows, img\_cols)  
 x\_test = x\_test.reshape(x\_test.shape[0], 1, img\_rows, img\_cols)  
 input\_shape = (1, img\_rows, img\_cols)  
**else**:  
 x\_train = x\_train.reshape(x\_train.shape[0], img\_rows, img\_cols, 1)  
 x\_test = x\_test.reshape(x\_test.shape[0], img\_rows, img\_cols, 1)  
 input\_shape = (img\_rows, img\_cols, 1)  
  
x\_train = x\_train.astype(**'float32'**)  
x\_test = x\_test.astype(**'float32'**)  
x\_train /= 255  
x\_test /= 255  
print(**'x\_train shape:'**, x\_train.shape)  
print(x\_train.shape[0], **'train samples'**)  
print(x\_test.shape[0], **'test samples'**)  
  
*# convert class vectors to binary class matrices*y\_train = keras.utils.to\_categorical(y\_train, num\_classes)  
y\_test = keras.utils.to\_categorical(y\_test, num\_classes)  
  
model = Sequential()  
model.add(Conv2D(16, kernel\_size=(2, 2),border\_mode=**'same'**,  
 activation=**'relu'**,  
 input\_shape=input\_shape))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Conv2D(36, (3, 3),border\_mode=**'same'**, activation=**'relu'**))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Dropout(0.25))  
model.add(Flatten())  
model.add(Dense(128, activation=**'relu'**))  
model.add(Dropout(0.5))  
model.add(Dense(num\_classes, activation=**'softmax'**))  
  
model.compile(loss=keras.losses.categorical\_crossentropy,  
 optimizer=keras.optimizers.Adadelta(),  
 metrics=[**'accuracy'**])  
print(model.summary())  
model.fit(x\_train, y\_train,  
 batch\_size=batch\_size,  
 epochs=epochs,  
 verbose=1,  
 validation\_data=(x\_test, y\_test))  
score = model.evaluate(x\_test, y\_test, verbose=0)  
y\_score = model.predict(x\_test)  
print(**'Test loss:'**, score[0])  
print(**'Test accuracy:'**, score[1])  
  
  
*#---------------------PLOTTING ROC CURVE FOR EACH CLASS------------------------  
# Plot linewidth.*lw = 2  
  
*# Compute ROC curve and ROC area for each class*fpr = dict()  
tpr = dict()  
roc\_auc = dict()  
**for** i **in** range(num\_classes):  
 fpr[i], tpr[i], \_ = roc\_curve(y\_test[:, i], y\_score[:, i])  
 roc\_auc[i] = auc(fpr[i], tpr[i])  
  
*# Compute micro-average ROC curve and ROC area*fpr[**"micro"**], tpr[**"micro"**], \_ = roc\_curve(y\_test.ravel(), y\_score.ravel())  
roc\_auc[**"micro"**] = auc(fpr[**"micro"**], tpr[**"micro"**])  
  
*# Compute macro-average ROC curve and ROC area  
  
# First aggregate all false positive rates*all\_fpr = np.unique(np.concatenate([fpr[i] **for** i **in** range(num\_classes)]))  
  
*# Then interpolate all ROC curves at this points*mean\_tpr = np.zeros\_like(all\_fpr)  
**for** i **in** range(num\_classes):  
 mean\_tpr += interp(all\_fpr, fpr[i], tpr[i])  
  
*# Finally average it and compute AUC*mean\_tpr /= num\_classes  
  
fpr[**"macro"**] = all\_fpr  
tpr[**"macro"**] = mean\_tpr  
roc\_auc[**"macro"**] = auc(fpr[**"macro"**], tpr[**"macro"**])  
  
*# Zoom in view of the upper left corner.*plt.figure(1)  
plt.xlim(0, 0.2)  
plt.ylim(0.8, 1)  
plt.plot(fpr[**"micro"**], tpr[**"micro"**],  
 label=**'micro-average ROC curve (area = {0:0.2f})'  
 ''**.format(roc\_auc[**"micro"**]),  
 color=**'deeppink'**, linestyle=**':'**, linewidth=4)  
  
plt.plot(fpr[**"macro"**], tpr[**"macro"**],  
 label=**'macro-average ROC curve (area = {0:0.2f})'  
 ''**.format(roc\_auc[**"macro"**]),  
 color=**'navy'**, linestyle=**':'**, linewidth=4)  
  
colors = cycle([**'aqua'**, **'darkorange'**, **'cornflowerblue'**, **'grey'**,**'blue'**,**'red'**,**'orange'**,**'purple'**,**'green'**,**'yellow'**])  
**for** i, color **in** zip(range(num\_classes), colors):  
 plt.plot(fpr[i], tpr[i], color=color, lw=lw,  
 label=**'ROC curve of class {0} (area = {1:0.2f})'  
 ''**.format(i, roc\_auc[i]))  
  
plt.plot([0, 1], [0, 1], **'k--'**, lw=lw)  
plt.xlabel(**'False Positive Rate'**)  
plt.ylabel(**'True Positive Rate'**)  
plt.title(**'Some extension of Receiver operating characteristic to multi-class'**)  
plt.legend(loc=**"lower right"**)  
plt.show()

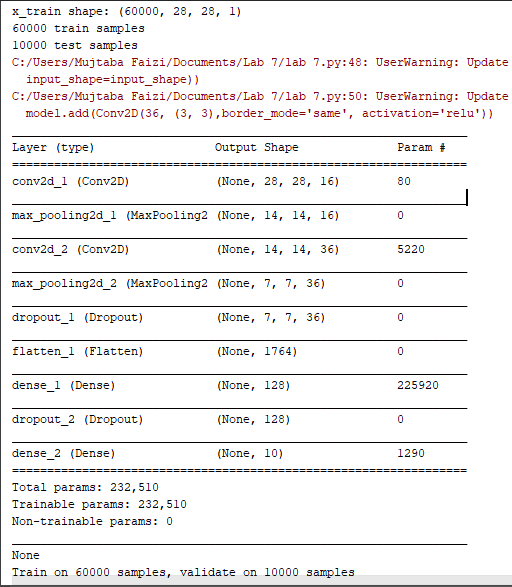
**SCREENSHOT:**

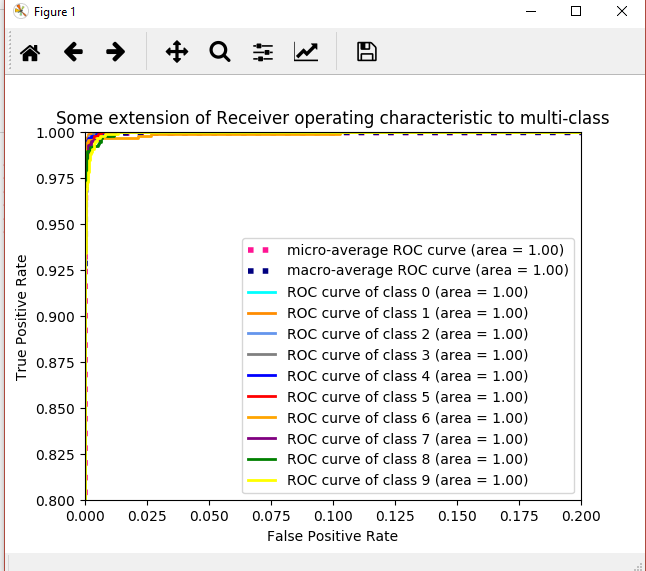
Validation set accuracy = 98%

Test set accuracy = 99%



**CNN STRUCTURE :**





Where,

True positive rate = TP / (TP + FN)

True negative rate = 1 – TN / (TN + FP)