Experiment Log

1st) splitting the data in test and train -

following is the code which solved my problem, i had to go through basic python codes and write it myself

In [ ]:
```
train_dir = os.listdir('/content/drive/MyDrive/IIITD/train/')
root_dir = '/content/drive/MyDrive/IIITD/train/'
for cls in train_dir:
  src = root_dir + cls
  allFileNames = os.listdir(src)
  print(allFileNames)
```

In [ ]:
```
train_dir = os.listdir('/content/drive/MyDrive/IIITD/data')
root_dir = '/content/drive/MyDrive/IIITD/'
val_ratio = 0.15
test_ratio = 0.05
for cls in train_dir:
# Creating partitions of the data after shuffeling
  src = root_dir+ 'data/' + cls # Folder to copy images from

  allFileNames = os.listdir(src)
  np.random.shuffle(allFileNames)
  train_FileNames, val_FileNames, test_FileNames = np.split(np.array(allFileNames),
                                                            [int(len(allFileNames)* (1 - (val_ratio + test_ratio))),
                                                            int(len(allFileNames)* (1 - test_ratio))])

  train_FileNames = [src+'/'+ name for name in train_FileNames.tolist()]
  val_FileNames = [src+'/' + name for name in val_FileNames.tolist()]
  test_FileNames = [src+'/' + name for name in test_FileNames.tolist()]

  print('Total images: ', len(allFileNames))
  print('Training: ', len(train_FileNames))
  print('Validation: ', len(val_FileNames))
  print('Testing: ', len(test_FileNames))

  # Copy-pasting images
  for name in train_FileNames:
      shutil.copy(name, root_dir +'train/' + cls)

  for name in val_FileNames:
      shutil.copy(name, root_dir +'valid/' + cls)

  for name in test_FileNames:
      shutil.copy(name, root_dir +'test/' + cls)
```

2nd) Converting Images to GrayScale

I first wrote a function to reads an image, convort it to gray scale and then save it back to the same directory

In [ ]:
```
train_dir = os.listdir('/content/drive/MyDrive/IIITD/data/')

root_dir = '/content/drive/MyDrive/IIITD/'
# os.makedirs(root_dir + '/train')
for cls in train_dir:
  src = root_dir+ 'data/' + cls
  allFileNames = os.listdir(src)
  # print(allFileNames)
  # print(src + '/'+ allFileNames)
  for img in allFileNames:
    IMG_LOC=src+'/'+img
    image = io.imread(IMG_LOC)
    gray_image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    cv.imwrite(IMG_LOC,gray_image)
    # print(gray_image)
    # print("-----")
    # print(img)
```

But when i was reading them with cv2, it was again coming as a RGB image, then i found that i can directly read them as gratscale, so i wrote the following code

In [ ]:
```
train_dir = os.listdir(r'E:\Internship\IIITD MIDAS\Task 3\Dataset')
root_dir = r'E:\Internship\IIITD MIDAS\Task 3\Dataset'
X = []
f=0
for cls in train_dir:
  src = root_dir +'\\'+ cls
  allFileNames = os.listdir(src)
  a = cls
  a = a[-2:]
  p = int(a)
  p
  # print(cls)
  # # print(src + '/'+ allFileNames)

  for img in allFileNames:
    # ytrain.append(p)
    IMG_LOC=src+'\\'+img
    # image = io.imread(IMG_LOC)
    image=cv.imread(IMG_LOC, cv.IMREAD_GRAYSCALE)
    image=cv.resize(image,(28,28))
    print(f)
    f=f+1
    # testing_data.append([np.array(img),img_num])
    X.append([np.array(image),np.array(p)])
  # print(X)
  # print(ytrain)
    # print(image)
```

3)Shape of the Images

My model was throwing error about the size of the input, then later i figured out that i had to resize the images

4)Loss function

I was using the "sparse_categorical_crossentropy" loass function which was giving some error, i finally solved my problem by using "categorical_crossentropy" loss

5) Saving the model

I was not able to save the model and re load the weights, but found the soulution by spening some time on google following is the code -

In [ ]:
```
checkpoint_path = "E:\Internship\IIITD MIDAS\Task 2.2-MNIST\checkpoints\cp-{epoch:04d}.ckpt"
cp_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_path,
    save_weights_only=True,
    save_freq=20,
    verbose=1)
```

6) Deciding the layers of Convolution Neural Network

7) Plotting Confusion matrix

finding quality metrics for a categorical Classification data was a little hard for me

Although i finally plotted the confusion matrix using the following code,which can be used to get the precision, recall and F2 score as well

In [ ]:
```
# Look at confusion matrix

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Predict the values from the validation dataset
Y_pred = model.predict(x_test)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(y_test,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(10))
```