Arvind Maurya – AIMLCEP-Batch03

Assignment report for Q3:

Location of Assignment in Github:

https://github.com/arvind-maurya/MachineLearning/blob/master/AIMLCEP_ARVIND_MAURYA_ASSIGNMENT/AIMLCEP_ARVIND_MAURYA_Q3.ipynb

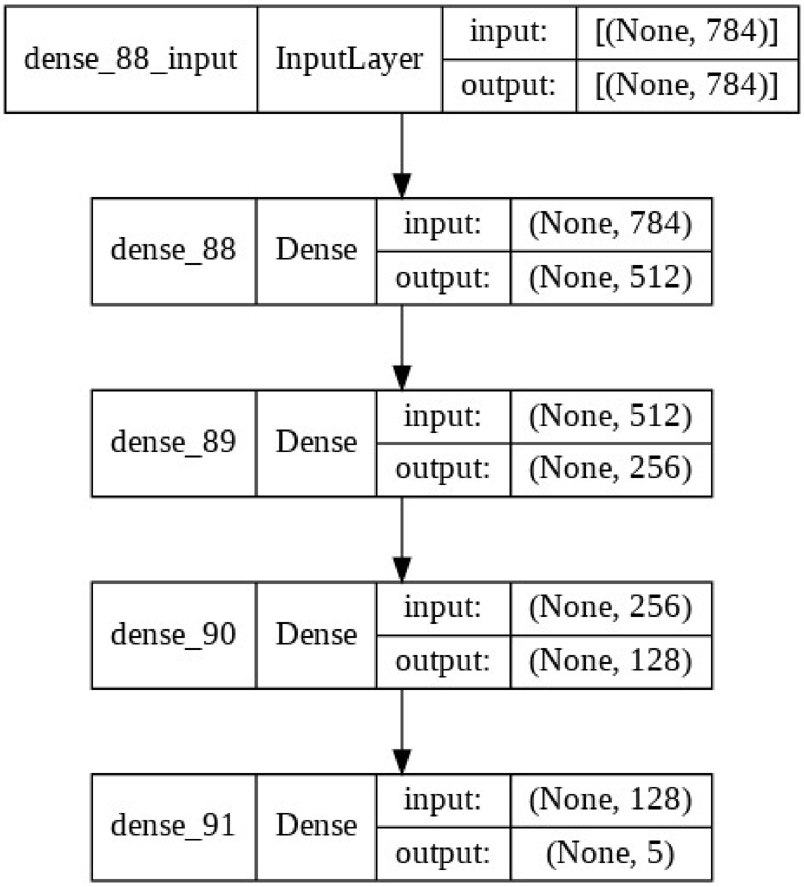
===

3. (b) [C, R]: Use the train partition to train a multi-layer perceptron (MLP) with an input layer, 3 hidden layers and an output layer. You are free to choose the number of neurons and their activation functions in the hidden layers. Use a soft-max at the output layer and a cross-entropy loss function to perform classification. Describe the MLP architecture you have used. Using the MLP model, report the accuracy, precision, recall, F1 score for the train set and test set.

Following is the MLP layer which is defined.

```
#Create NN layers now
model = keras.Sequential()
model.add(layers.Dense(512,input_shape=input_shape,activation='relu')) #i/p Layer + hidden layer1
model.add(layers.Dense(256,activation='relu')) # hidden layer2
model.add(layers.Dense(128,activation='relu')) # hidden layer2
model.add(layers.Dense(num_classes,activation='softmax')) #output layer
```

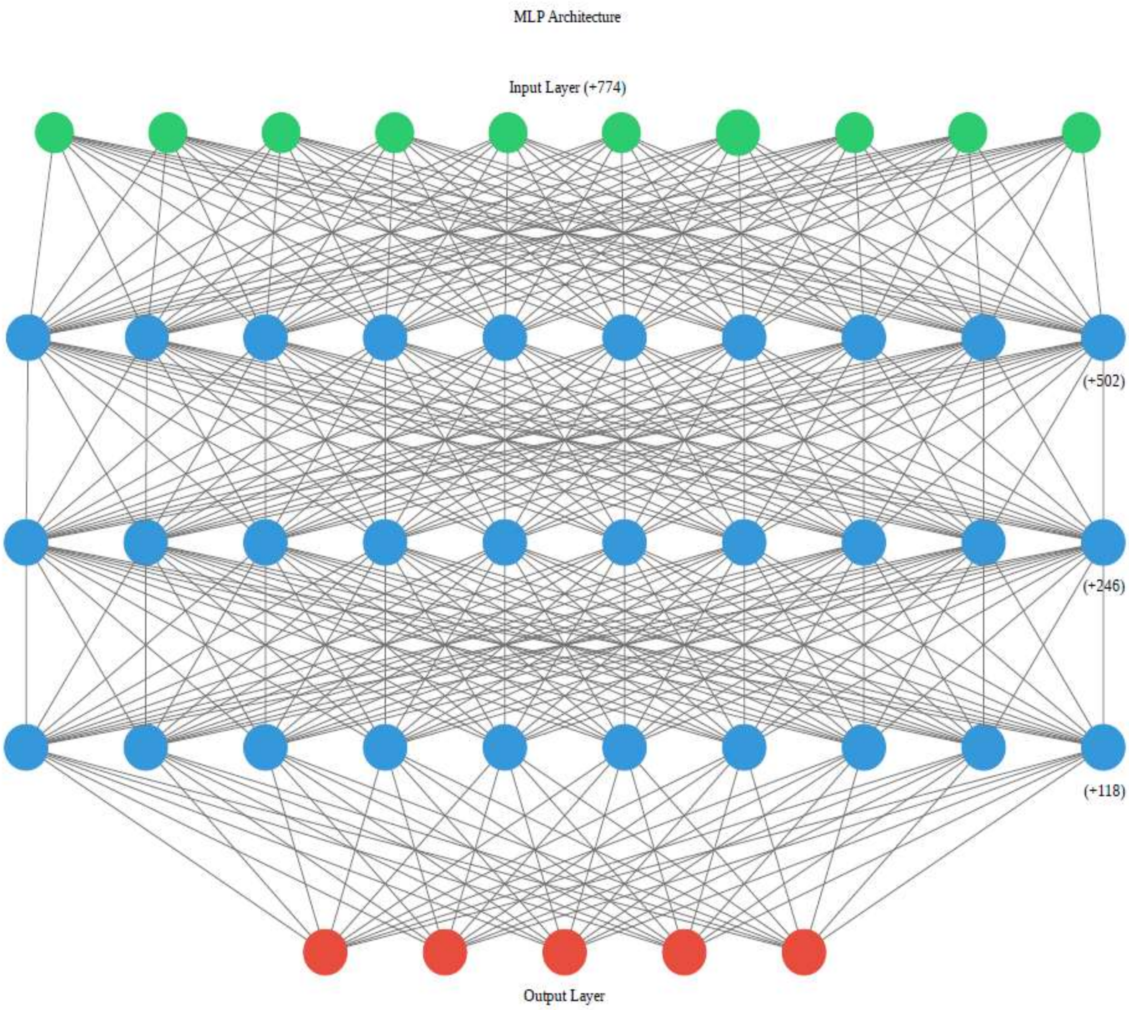
Below is the architecture definition



Model Summary:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               401920

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 5)                 645

=================================================================
Total params: 566,789
Trainable params: 566,789
Non-trainable params: 0
```

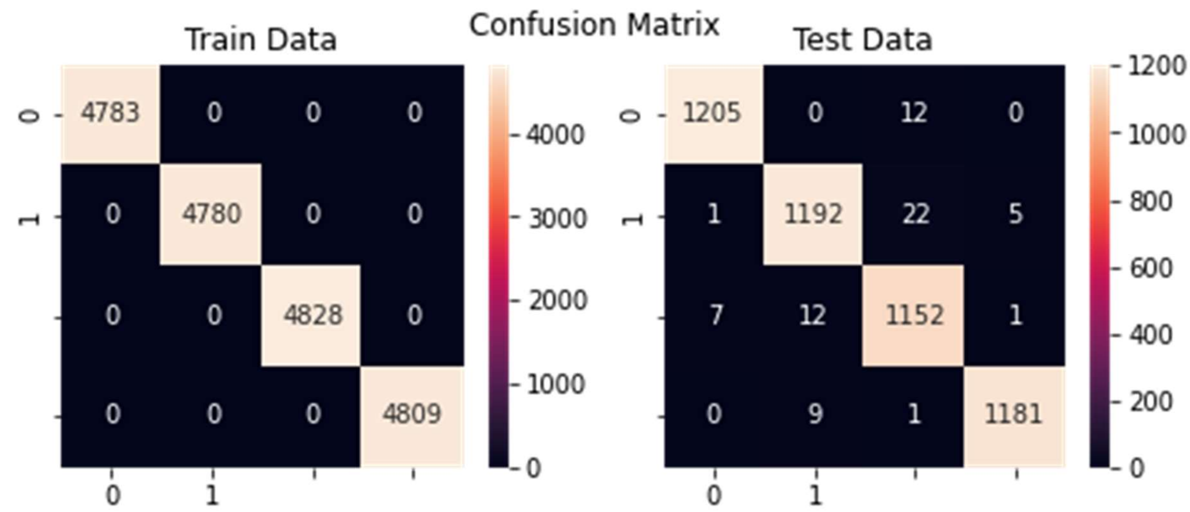
Below is the visual representation for MLP architecture used



Green – Input Layer
Blue – Hidden Layer
Red – Output Layer.
Below is the accuracy, precision, recall, F1 score for the train set and test set got from MLP Classifier

```
=======================================================
Train Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00      4783
           2       1.00      1.00      1.00      4780
           3       1.00      1.00      1.00      4828
           4       1.00      1.00      1.00      4809

    accuracy                           1.00     19200
   macro avg       1.00      1.00      1.00     19200
weighted avg       1.00      1.00      1.00     19200

=======================================================
Test Classification Report:
              precision    recall  f1-score   support

           1       0.99      0.99      0.99      1217
```

```
           2       0.98      0.98      0.98      1220
           3       0.97      0.98      0.98      1172
           4       0.99      0.99      0.99      1191

    accuracy                           0.99      4800
   macro avg       0.99      0.99      0.99      4800
weighted avg       0.99      0.99      0.99      4800

=======================================================
Train confusion matrix:
 [[4783    0    0    0]
 [   0 4780    0    0]
 [   0    0 4828    0]
 [   0    0    0 4809]]

Test confusion matrix:
 [[1205    0   12    0]
 [   1 1192   22    5]
 [   7   12 1152    1]
 [   0    9    1 1181]]
```

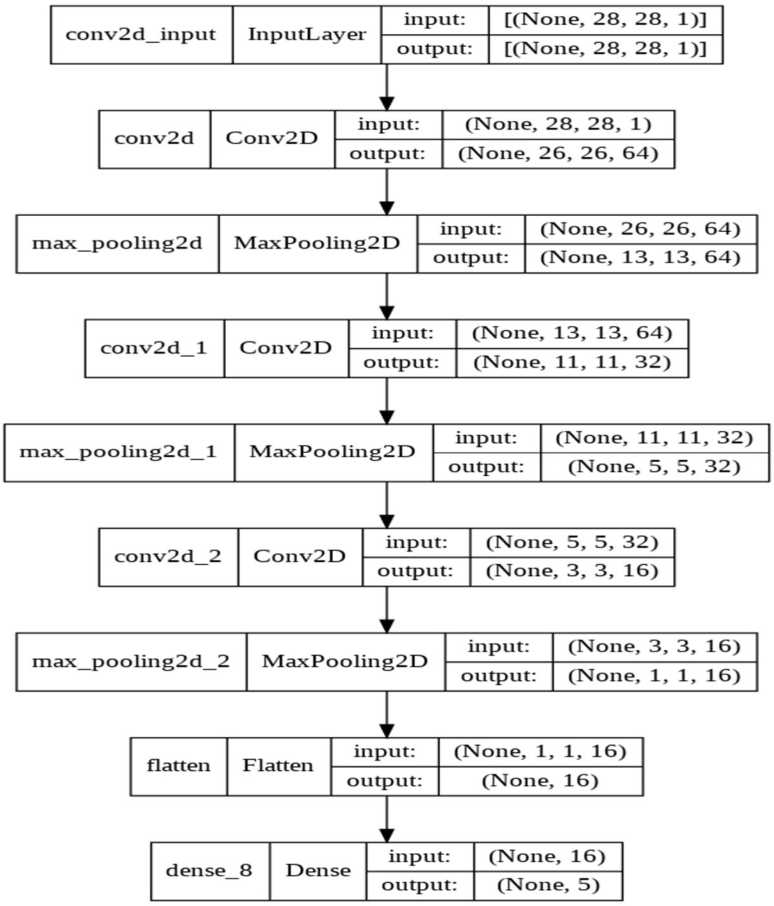


3.(c): [C, R] Now consider that each data point represented in vector form can be represented as a pxp matrix for a suitable p. Using this transformed train data, build a convolution neural network (CNN) with 3 convolutions cum max-pool blocks (where max-pool follows a convolution operation) followed by a fully connected layer and an output layer. You are free to choose the kernel size, stride and padding in each convolution operation. Also use a max-pool layer of appropriate grid size in each layer. Use a soft-max at the output layer and a cross-entropy loss function to perform classification. Describe the CNN architecture you have used. Using the CNN model, report the accuracy, precision, recall, F1 score for the train set and test set.

```
#create model
model_cnn = keras.Sequential()
#add model layers
model_cnn.add(layers.Conv2D(64,kernel_size=3, activation='relu', input_
shape=(28,28,1)))
model_cnn.add(layers.MaxPooling2D(pool_size=2))
model_cnn.add(layers.Conv2D(32, kernel_size=3, activation='relu'))
```

```
model_cnn.add(layers.MaxPooling2D(pool_size=2))
model_cnn.add(layers.Conv2D(16, kernel_size=3, activation='relu'))
model_cnn.add(layers.MaxPooling2D(pool_size=2))
model_cnn.add(layers.Flatten())
model_cnn.add(layers.Dense(5, activation='softmax'))
```

CNN Model Used:

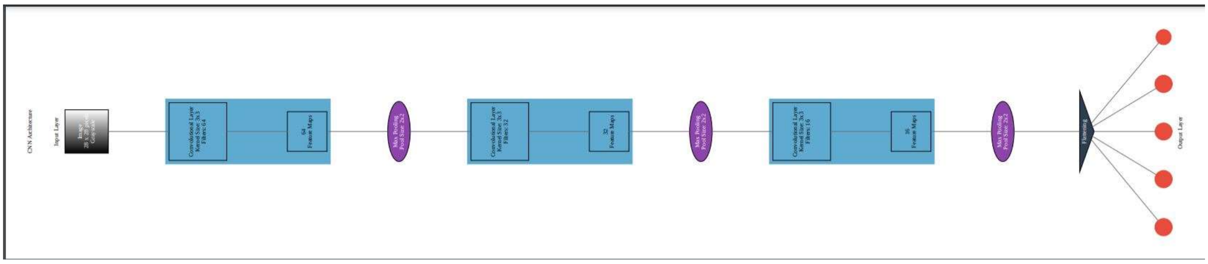


Model Summary:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 32)        18464

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 32)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 16)          4624

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 16)         0
 2D)
```

```
 flatten (Flatten)            (None, 16)                0

 dense_8 (Dense)              (None, 5)                 85

=================================================================
Total params: 23,813
Trainable params: 23,813
Non-trainable params: 0
```

CNN Architecture Used:



Below is the accuracy, precision, recall, F1 score for the train set and test set got from CNN.

```
=====================================================
Train Classification Report:
              precision    recall  f1-score   support

           1       0.99      0.99      0.99      4795
           2       0.99      0.97      0.98      4788
           3       0.97      0.99      0.98      4772
           4       1.00      0.98      0.99      4845

    accuracy                           0.99     19200
   macro avg       0.99      0.99      0.99     19200
weighted avg       0.99      0.99      0.99     19200

=====================================================
Test Classification Report:
              precision    recall  f1-score   support

           1       0.99      0.99      0.99      1205
```

```
           2       0.98      0.98      0.98      1212
           3       0.97      0.99      0.98      1228
           4       0.99      0.98      0.99      1155

    accuracy                           0.98      4800
   macro avg       0.98      0.98      0.98      4800
weighted avg       0.98      0.98      0.98      4800

======================================================
Train confusion matrix:
 [[4770    0   25    0]
 [  10 4666   92   20]
 [  35   11 4725    1]
 [   3   60   17 4765]]
Test confusion matrix:
 [[1193    0   12    0]
 [   4 1185   18    5]
 [  10    2 1215    1]
 [   4   17    2 1132]]
```

(d) [R] Compare and contrast the performance obtained from MLP and CNN.

Comparing Train dataset classification report:

```
======================================================
Train Classification Report for MLP:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00      4783
           2       1.00      1.00      1.00      4780
           3       1.00      1.00      1.00      4828
           4       1.00      1.00      1.00      4809

    accuracy                           1.00     19200
   macro avg       1.00      1.00      1.00     19200
weighted avg       1.00      1.00      1.00     19200

======================================================
Train Classification Report for CNN:
              precision    recall  f1-score   support
```

	precision	recall	f1-score	support
1	0.99	0.99	0.99	4795
2	0.99	0.97	0.98	4788
3	0.97	0.99	0.98	4772
4	1.00	0.98	0.99	4845
accuracy			**0.99**	**19200**
macro avg	0.99	0.99	0.99	19200
weighted avg	0.99	0.99	0.99	19200

Inference:

1. If you see the train dataset accuracy from MLP has come out to be 1 which indicates that MLP is working perfectly for this dataset.
2. Accuracy obtain from CNN model also is 0.99. We can say that CNN is also working perfectly.
3. All the class has been predicted very efficiently by both the model.

Comparing Test dataset classification report:

==

Test Classification Report for MLP:

	precision	recall	f1-score	support
1	0.99	0.99	0.99	1217
2	0.98	0.98	0.98	1220
3	0.97	0.98	0.98	1172
4	0.99	0.99	0.99	1191
accuracy			**0.99**	**4800**
macro avg	0.99	0.99	0.99	4800
weighted avg	0.99	0.99	0.99	4800

==

Test Classification Report for CNN:

	precision	recall	f1-score	support
1	0.99	0.99	0.99	1205
2	0.98	0.98	0.98	1212
3	0.97	0.99	0.98	1228

```
           4       0.99      0.98      0.99      1155

    accuracy                           0.98      4800
   macro avg       0.98      0.98      0.98      4800
weighted avg       0.98      0.98      0.98      4800

======================================================
```

Inference:

4. If you see the test dataset accuracy from MLP has come out to be 0.99 which indicates that MLP is working perfectly for test dataset.
5. Accuracy obtain from CNN model also is 0.98. We can say that CNN is also working perfectly.
6. All the class has been predicted very efficiently by both the model.

So, CNN and MLP both can be used as classification techniques for this data set.