Machine Learning Lab
Assignment 3

Sonu Kumar Mahanty
Roll - 001811001038
Semester - 7th
B.E. I.T. 4th Year

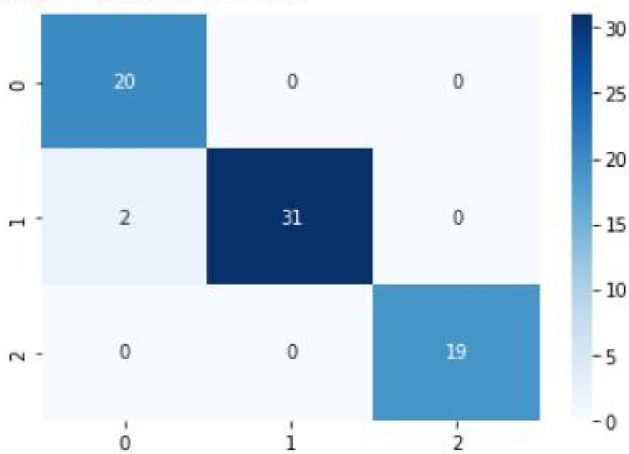
ENTIRE ASSIGNMENT LINK (GOOGLE COLLAB + COMPARISON TABLE):
https://drive.google.com/folderview?id=16CmJS0N2cjgD3-vtxowXJziNXKQ2lKpm

PART 1

1) Wine Dataset

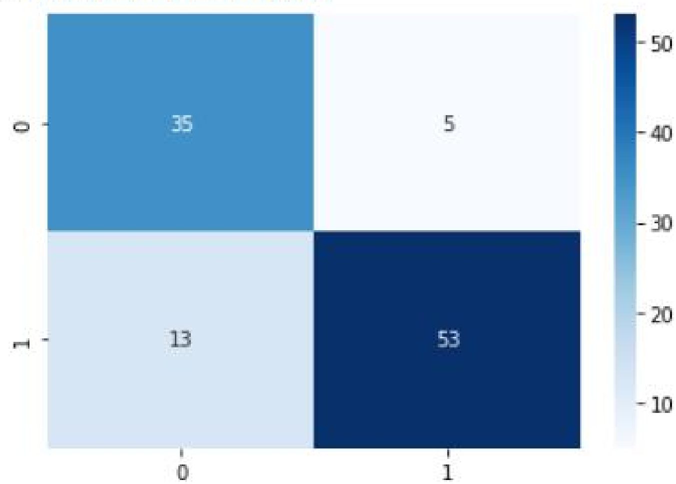
1.1) GaussianHMM Without Tuning

```
Confusion Matrix:
[[20  0  0]
 [ 2 31  0]
 [ 0  0 19]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.91      1.00      0.95        20
           2       1.00      0.94      0.97        33
           3       1.00      1.00      1.00        19

    accuracy                           0.97        72
   macro avg       0.97      0.98      0.97        72
weighted avg       0.97      0.97      0.97        72

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9722222222222222
```



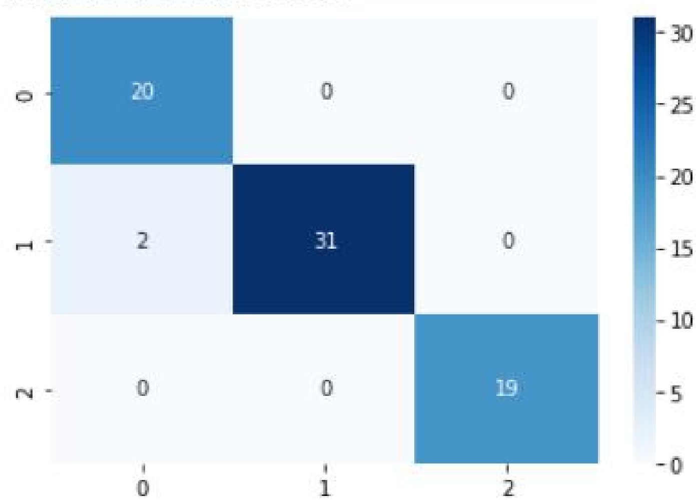
1.2) GaussianHMM With Tuning

```
Confusion Matrix:
[[35  5]
 [13 53]]
---------------------------------------------------
---------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.73      0.88      0.80        40
           g       0.91      0.80      0.85        66

    accuracy                           0.83       106
   macro avg       0.82      0.84      0.83       106
weighted avg       0.84      0.83      0.83       106

---------------------------------------------------
---------------------------------------------------
Accuracy:
0.8301886792452831
```



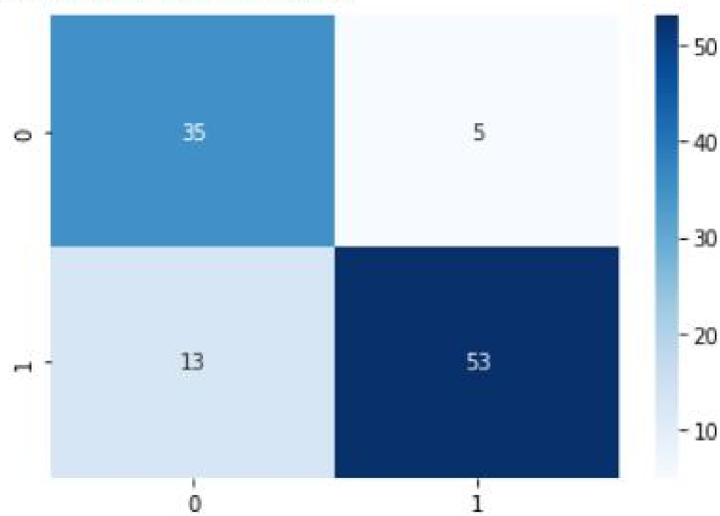
1.3) GMMHMM Without Tuning

```
Confusion Matrix:
[[20  0  0]
 [ 2 31  0]
 [ 0  0 19]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.91      1.00      0.95        20
           2       1.00      0.94      0.97        33
           3       1.00      1.00      1.00        19

    accuracy                           0.97        72
   macro avg       0.97      0.98      0.97        72
weighted avg       0.97      0.97      0.97        72

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9722222222222222
```



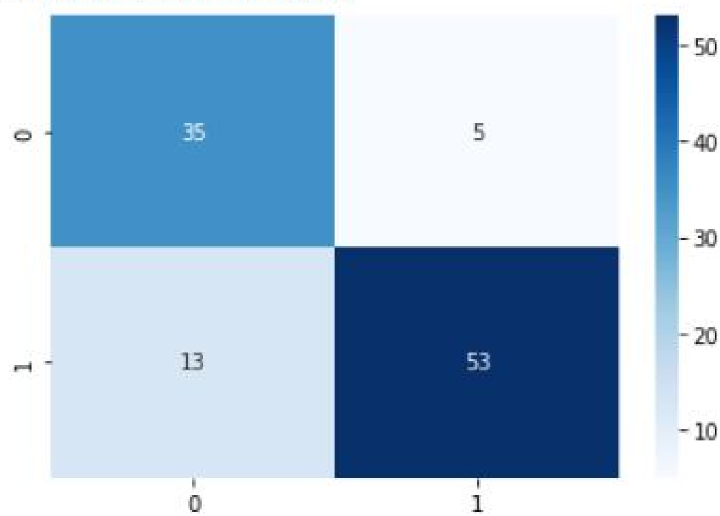
1.4) GMMHMM With Tuning

```
Confusion Matrix:
[[35  5]
 [13 53]]
---------------------------------------------------
---------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.73      0.88      0.80        40
           g       0.91      0.80      0.85        66

    accuracy                           0.83       106
   macro avg       0.82      0.84      0.83       106
weighted avg       0.84      0.83      0.83       106

---------------------------------------------------
---------------------------------------------------
Accuracy:
0.8301886792452831
```



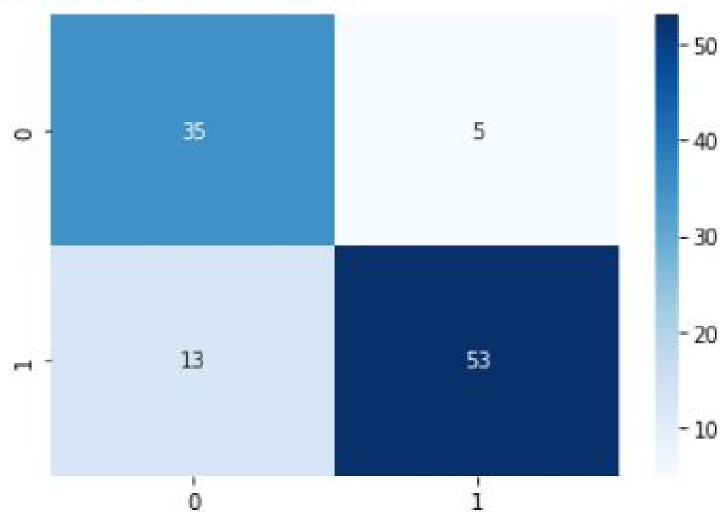
1.5) MultinomialHMM Without Tuning

```
Confusion Matrix:
[[35  5]
 [13 53]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.73      0.88      0.80        40
           g       0.91      0.80      0.85        66

    accuracy                           0.83       106
   macro avg       0.82      0.84      0.83       106
weighted avg       0.84      0.83      0.83       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8301886792452831
```



1.6) MultinomialHMM Without Tuning

```
Confusion Matrix:
[[35  5]
 [13 53]]
-------------------------------------------------
-------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.73      0.88      0.80        40
           g       0.91      0.80      0.85        66

    accuracy                           0.83       106
   macro avg       0.82      0.84      0.83       106
weighted avg       0.84      0.83      0.83       106

-------------------------------------------------
-------------------------------------------------
Accuracy:
0.8301886792452831
```

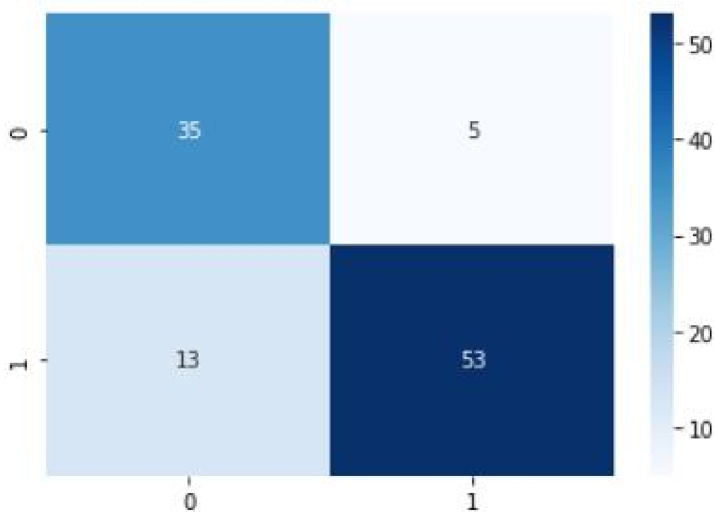


The maximum accuracy was achieved when the Train-Test split ratio was 70:30, which was achieved by using the Gaussian Model. The maximum range of accuracies was achieved by the Gaussian Model, followed by the GMMHMM model, which is followed by the MultinomialHMM model.

2) Ionosphere Dataset

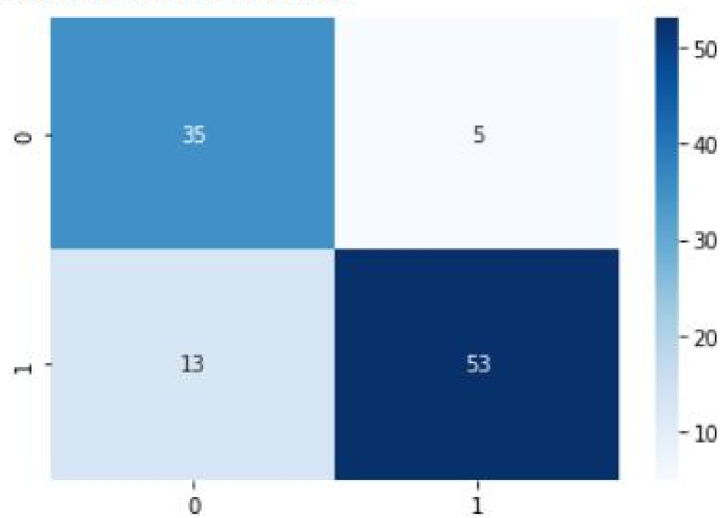
2.1) GaussianHMM Without Tuning

```
Confusion Matrix:
[[35  5]
 [13 53]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.73      0.88      0.80        40
           g       0.91      0.80      0.85        66

    accuracy                           0.83       106
   macro avg       0.82      0.84      0.83       106
weighted avg       0.84      0.83      0.83       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8301886792452831
```



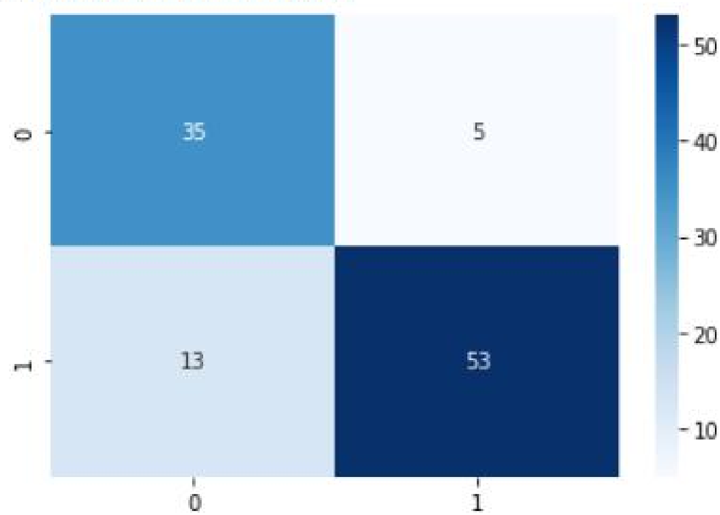
2.2) GaussianHMM With Tuning

```
Confusion Matrix:
[[35  5]
 [13 53]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.73      0.88      0.80        40
           g       0.91      0.80      0.85        66

    accuracy                           0.83       106
   macro avg       0.82      0.84      0.83       106
weighted avg       0.84      0.83      0.83       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8301886792452831
```



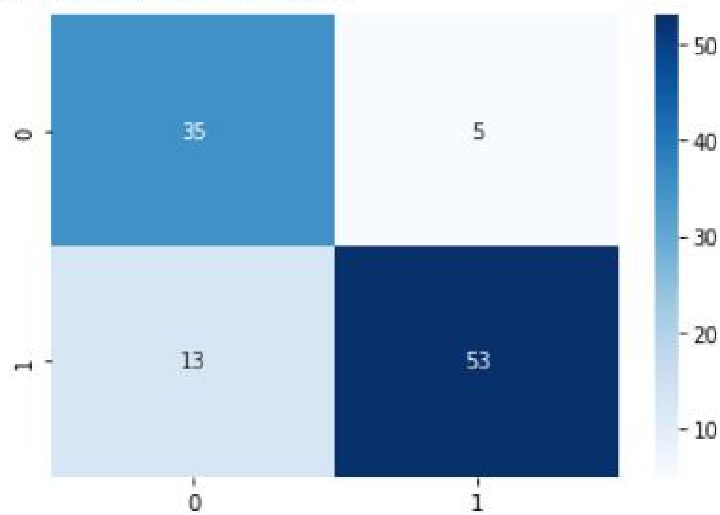
2.3) GMMHMM Without Tuning

```
Confusion Matrix:
[[35  5]
 [13 53]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.73      0.88      0.80        40
           g       0.91      0.80      0.85        66

    accuracy                           0.83       106
   macro avg       0.82      0.84      0.83       106
weighted avg       0.84      0.83      0.83       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8301886792452831
```



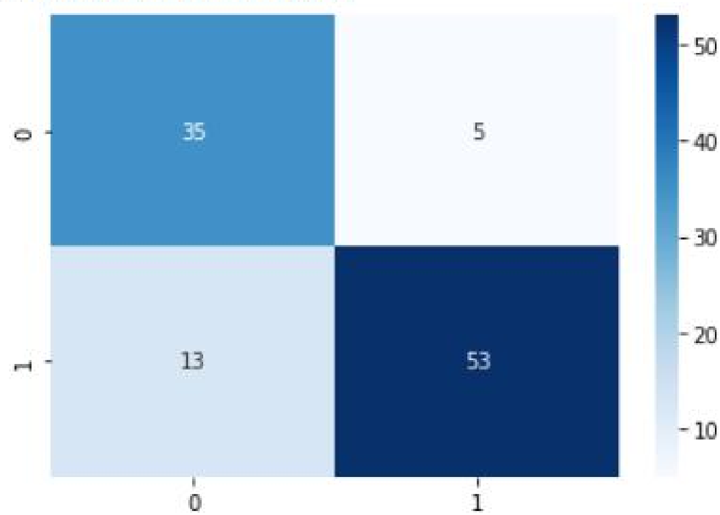
2.4) GMMHMM With Tuning

```
Confusion Matrix:
[[35  5]
 [13 53]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.73      0.88      0.80        40
           g       0.91      0.80      0.85        66

    accuracy                           0.83       106
   macro avg       0.82      0.84      0.83       106
weighted avg       0.84      0.83      0.83       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8301886792452831
```



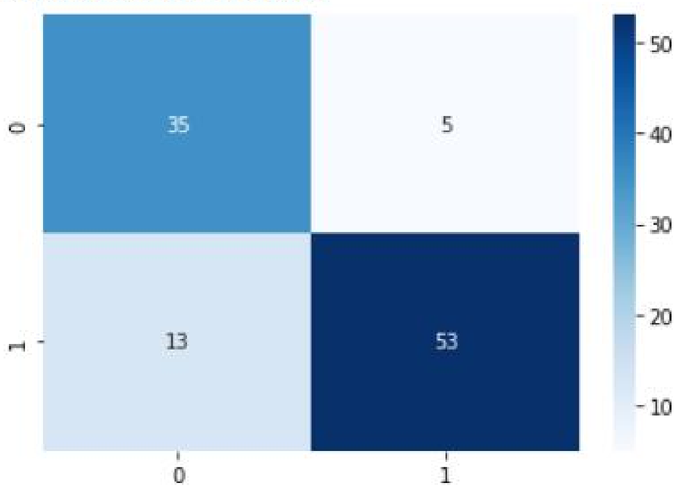
2.5) MultinomialHMM Without Tuning

```
Confusion Matrix:
[[35  5]
 [13 53]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.73      0.88      0.80        40
           g       0.91      0.80      0.85        66

    accuracy                           0.83       106
   macro avg       0.82      0.84      0.83       106
weighted avg       0.84      0.83      0.83       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8301886792452831
```



2.6) MultinomialHMM Without Tuning

```
Confusion Matrix:
[[35  5]
 [13 53]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.73      0.88      0.80        40
           g       0.91      0.80      0.85        66

    accuracy                           0.83       106
   macro avg       0.82      0.84      0.83       106
weighted avg       0.84      0.83      0.83       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8301886792452831
```

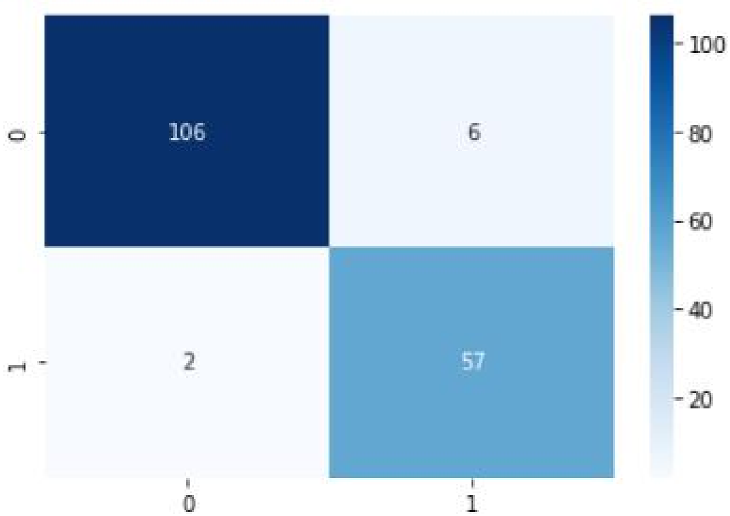


The maximum accuracy was achieved when the Train-Test split ratio was 70:30, which was achieved by using the Gaussian Model. The maximum range of accuracies was achieved by the Gaussian Model, followed by the GMMHMM model, which is followed by the MultinomialHMM model.

3) Breast Cancer Dataset

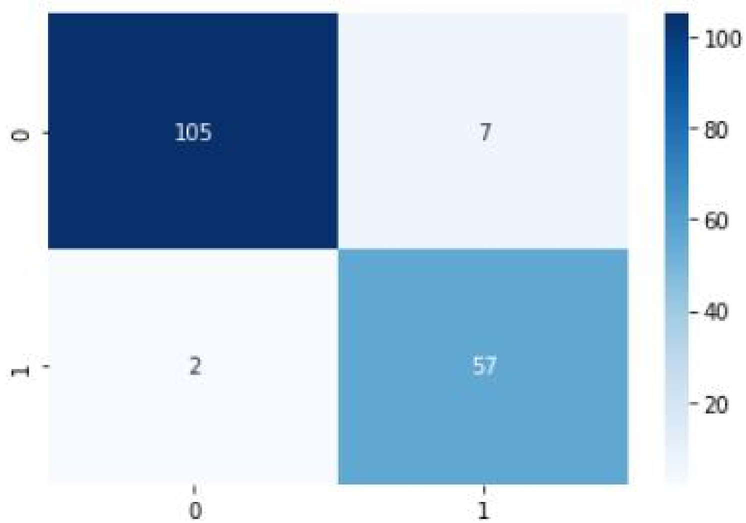
3.1) GaussianHMM Without Tuning

```
Confusion Matrix:
[[106   6]
 [  2  57]]
-----------------------------------------------
-----------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.98      0.95      0.96       112
           M       0.90      0.97      0.93        59

    accuracy                           0.95       171
   macro avg       0.94      0.96      0.95       171
weighted avg       0.96      0.95      0.95       171

-----------------------------------------------
-----------------------------------------------
Accuracy:
0.9532163742690059
```



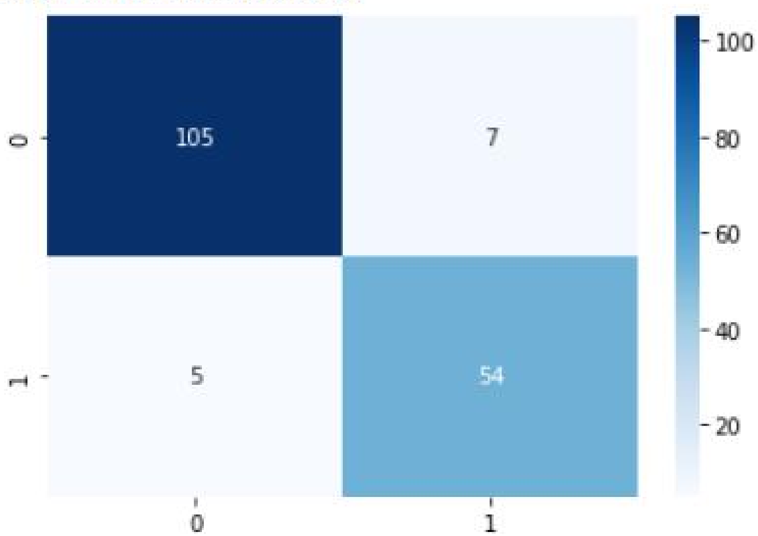
3.2) GaussianHMM With Tuning

```
Confusion Matrix:
[[105   7]
 [  2  57]]
------------------------------------------------
------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.98      0.94      0.96       112
           M       0.89      0.97      0.93        59

    accuracy                           0.95       171
   macro avg       0.94      0.95      0.94       171
weighted avg       0.95      0.95      0.95       171

------------------------------------------------
------------------------------------------------
Accuracy:
0.9473684210526315
```



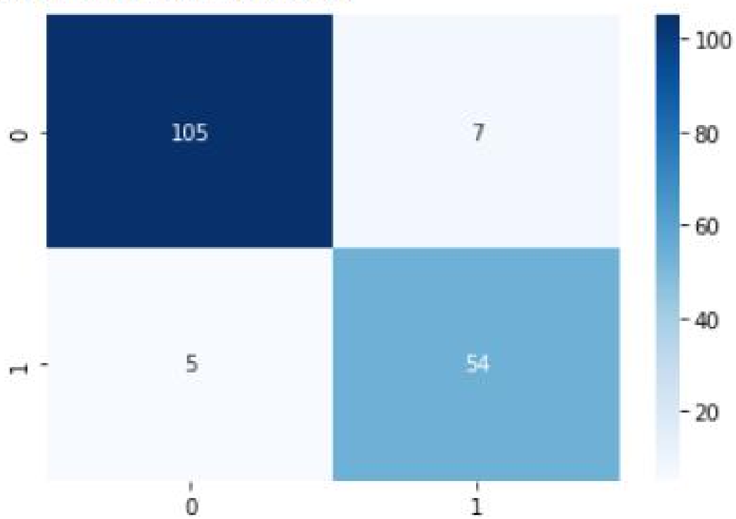
3.3) GMMHMM Without Tuning

```
Confusion Matrix:
[[105   7]
 [  5  54]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.95      0.94      0.95       112
           M       0.89      0.92      0.90        59

    accuracy                           0.93       171
   macro avg       0.92      0.93      0.92       171
weighted avg       0.93      0.93      0.93       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9298245614035088
```



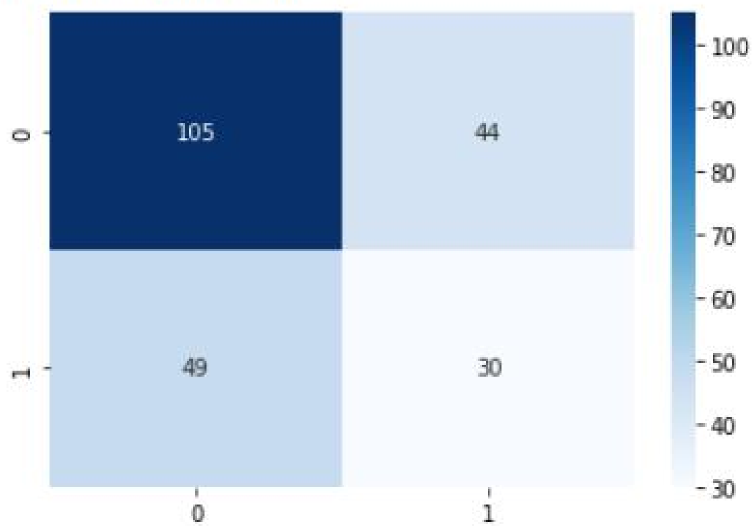
3.4) GMMHMM With Tuning

```
Confusion Matrix:
[[105   7]
 [  5  54]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.95      0.94      0.95       112
           M       0.89      0.92      0.90        59

    accuracy                           0.93       171
   macro avg       0.92      0.93      0.92       171
weighted avg       0.93      0.93      0.93       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9298245614035088
```



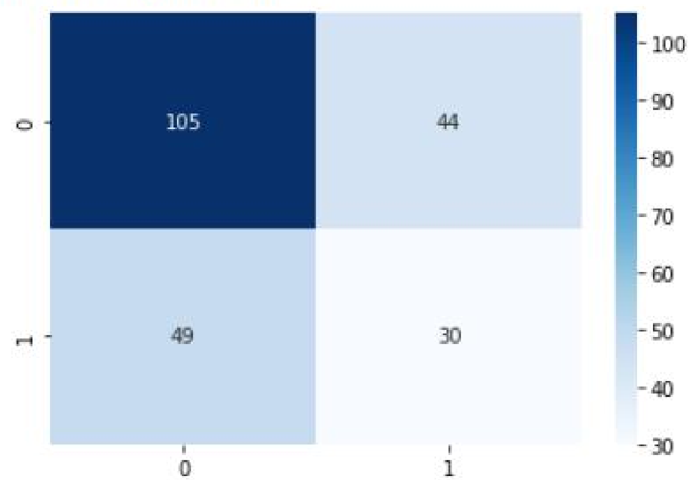
3.5) MultinomialHMM Without Tuning

```
Confusion Matrix:
[[105  44]
 [ 49  30]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.68      0.70      0.69       149
           M       0.41      0.38      0.39        79

    accuracy                           0.59       228
   macro avg       0.54      0.54      0.54       228
weighted avg       0.59      0.59      0.59       228

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.5921052631578947
```



3.6) MultinomialHMM Without Tuning

```
Confusion Matrix:
[[105  44]
 [ 49  30]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.68      0.70      0.69       149
           M       0.41      0.38      0.39        79

    accuracy                           0.59       228
   macro avg       0.54      0.54      0.54       228
weighted avg       0.59      0.59      0.59       228

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.5921052631578947
```



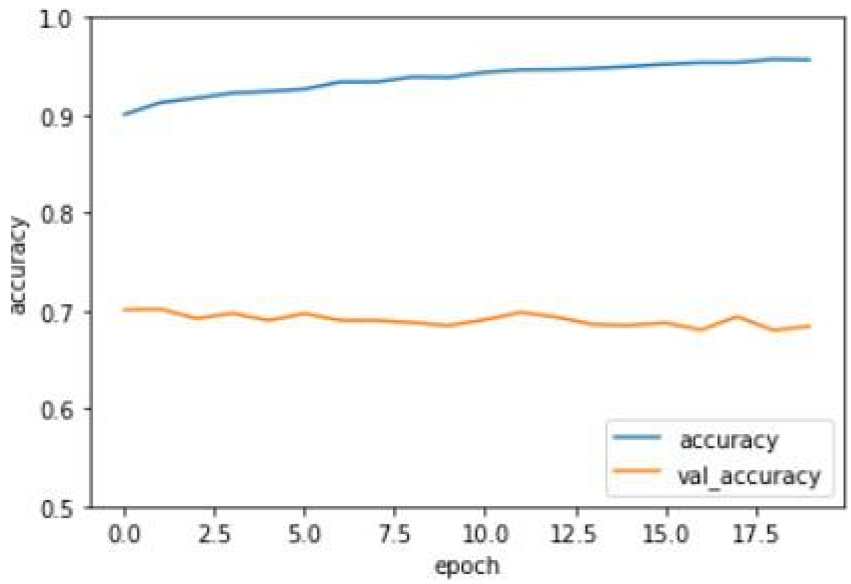
The maximum accuracy was achieved when the Train-Test split ratio was 70:30, which was achieved by using the Gaussian Model. The maximum range of accuracies was achieved by the Gaussian Model, followed by the GMMHMM model, which is followed by the MultinomialHMM model.

PART 2

1) CIFAR-10

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #

conv2d 6 (Conv2D)            (None, 30, 30, 32)        896
__________
max_pooling2d_4 (MaxPooling2 (None, 15, 15, 32)        0

conv2d 7 (Conv2D)            (None, 13, 13, 64)        18496
__________
max_pooling2d_5 (MaxPooling2 (None, 6, 6, 64)          0

conv2d 8 (Conv2D)            (None, 4, 4, 64)          36928
_________
flatten (Flatten)            (None, 1024)              0
______
dense (Dense)                (None, 64)                65600
______
dense 1 (Dense)              (None, 10)                650

Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

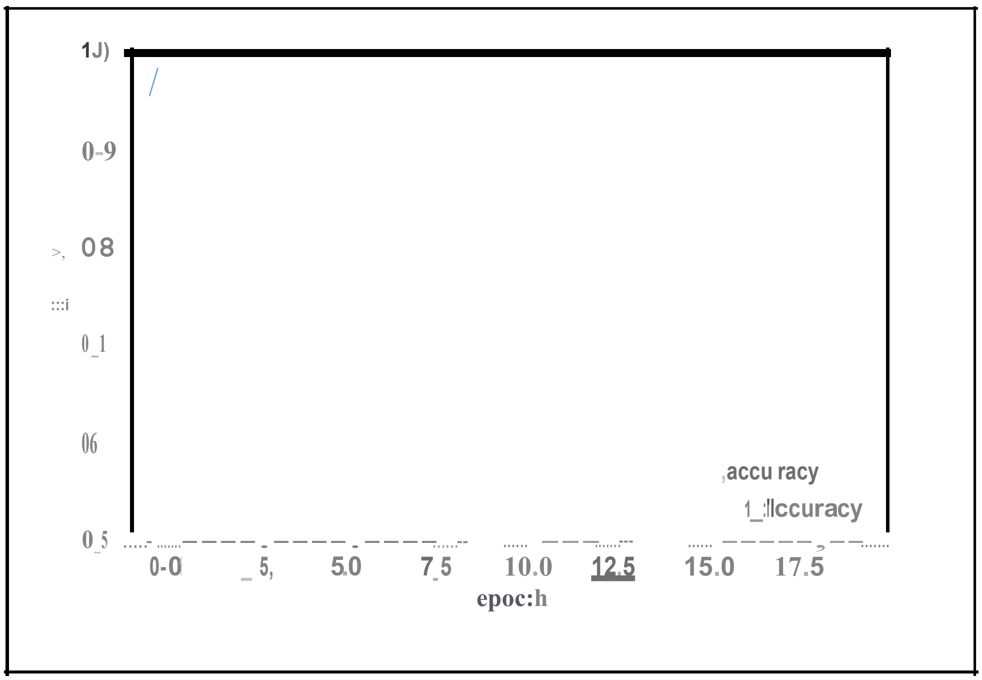
```
cpocn # /L<J
1563/1563 [==============================]-- 69s 44ms/step  loss: 0.1561 · accuracy: 0.9437  val loss: 1.8716  val_accuracy: 0.6910
Epoch 12/20
1563/1563 [==============================]---69s 44ms/step  loss: 0.1498 - accuracy: 0.9463  val loss: 1.9775  val_accuracy: 0.6986
Epoch 13/20
1563/1563 [==============================]-- 69s 44ms/step  loss: 0.1524 · accuracy: 0.9466  val loss: 2.0503  val_accuracy: 0.6936
Epoch 14/20
1563/1563 [==============================]-- 6-9s 44ms/step  loss: 0.1490 · accuracy: 0.9477  val loss: 2.0715  val_accuracy: 0.6861
Epoch 15/20
1563/1563 [==============================]---69s 44ms/step  loss: 0.1429 - accuracy: 0.9497  val loss: 2.1616  val_accuracy: 0.6852
Epoch 16/20
1563/1563 [==============================]-- 69s 44ms/step  loss: 0.1356 · accuracy: 0.9520  val loss: 2.2363  val_accuracy: 0.6877
Epoch 17/20
1563/1563 [==============================]-- 6-9s 44ms/step  loss: 0.1327 · accuracy: 0.9537  val loss: 2.2239  val_accuracy: 0.6806
Epoch 18/20
1563/1563 [==============================]---69s 44ms/step  loss: 0.1348 - accuracy: 0.9538  val loss: 2.2102  val_accuracy: 0.6943
Epoch 19/20
1563/1563 [==============================]-- 69s 44ms/step  loss: 0.1250 · accuracy: 0.9571  val loss: 2.3900  val_accuracy: 0.6802
Epoch 20/20
1563/1563 [==============================]---6-9s 44ms/step  loss: 0.1245 - accuracy: 0.9565  val loss: 2.3173  val_accuracy: 0.6842
```



2) MNIST

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_3 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_6 (Dense)              (None, 64)                36928
_________________________________________________________________
dense_7 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

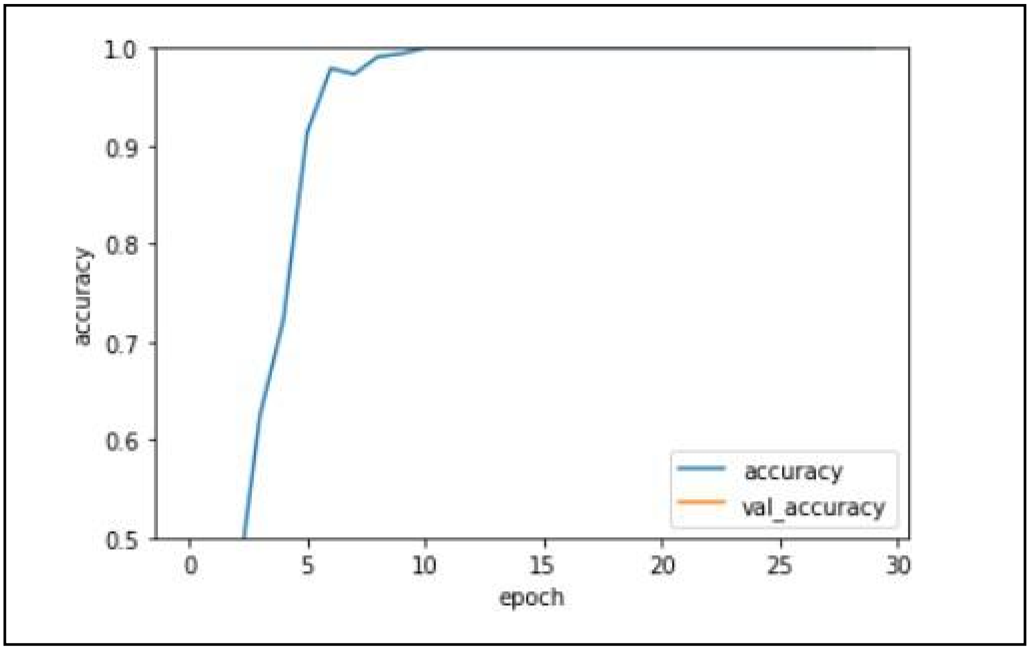
```
1875/1875 [------------------------------] - 57s 30ms/step - loss: 0.0079 - accuracy: 0.9973 - val_loss: 0.0319 - val_accuracy: 0.9913
Epoch 12/20
1875/1875 [------------------------------] - 57s 31ms/step - loss: 0.0084 - accuracy: 0.9973 - val_loss: 0.0329 - val_accuracy: 0.9921
Epoch 13/20
1875/1875 [------------------------------] - 57s 31ms/step - loss: 0.0067 - accuracy: 0.9980 - val_loss: 0.0343 - val_accuracy: 0.9918
Epoch 14/20
1875/1875 [------------------------------] - 58s 31ms/step - loss: 0.0078 - accuracy: 0.9973 - val_loss: 0.0390 - val_accuracy: 0.9908
Epoch 15/20
1875/1875 [------------------------------] - 58s 31ms/step - loss: 0.0048 - accuracy: 0.9985 - val_loss: 0.0374 - val_accuracy: 0.9930
Epoch 16/20
1875/1875 [------------------------------] - 58s 31ms/step - loss: 0.0069 - accuracy: 0.9980 - val_loss: 0.0336 - val_accuracy: 0.9923
Epoch 17/20
1875/1875 [------------------------------] - 57s 31ms/step - loss: 0.0049 - accuracy: 0.9985 - val_loss: 0.0430 - val_accuracy: 0.9916
Epoch 18/20
1875/1875 [------------------------------] - 57s 31ms/step - loss: 0.0053 - accuracy: 0.9982 - val_loss: 0.0397 - val_accuracy: 0.9915
Epoch 19/20
1875/1875 [------------------------------] - 58s 31ms/step - loss: 0.0048 - accuracy: 0.9986 - val_loss: 0.0540 - val_accuracy: 0.9903
Epoch 20/20
1875/1875 [------------------------------] - 58s 31ms/step - loss: 0.0043 - accuracy: 0.9987 - val_loss: 0.0419 - val_accuracy: 0.9919
```



3) SAVEE

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 155, 318, 32)      320
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 77, 159, 32)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 75, 157, 64)       18496
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 37, 78, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 35, 76, 64)        36928
_________________________________________________________________
flatten_3 (Flatten)          (None, 170240)            0
_________________________________________________________________
dense_6 (Dense)              (None, 64)                10895424
_________________________________________________________________
dense_7 (Dense)              (None, 10)                650
=================================================================
Total params: 10,951,818
Trainable params: 10,951,818
Non-trainable params: 0
_________________________________________________________________
```

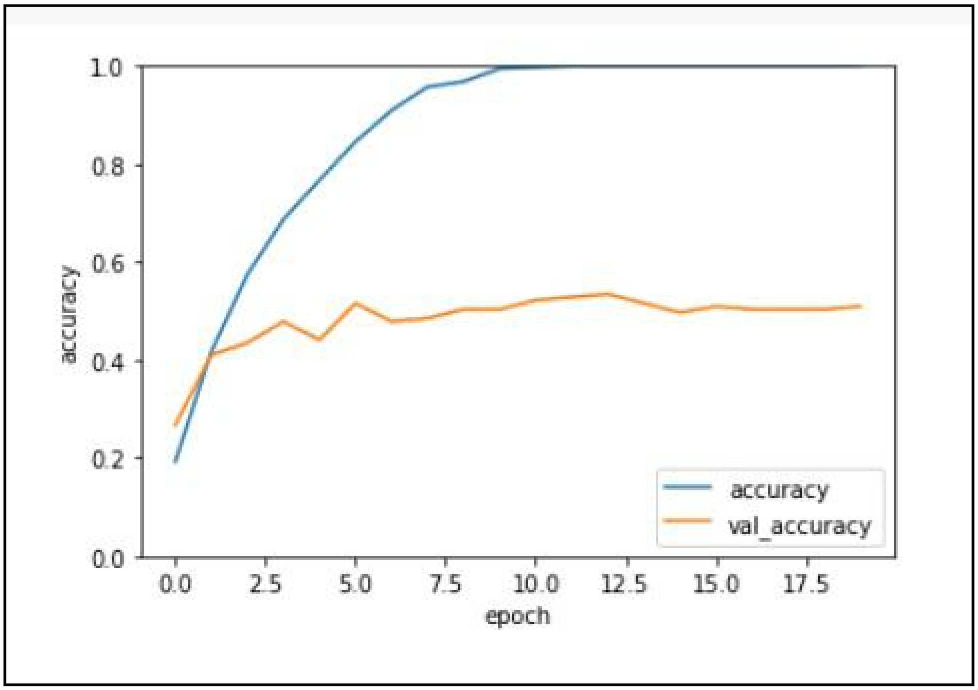
```
11/11 [==============================] - 27s 2s/step - loss: 2.2025e-05 - accuracy: 1.0000 - val_loss: 7.0087 - val_accuracy: 0.3056
Epoch 25/30
11/11 [==============================] - 27s 2s/step - loss: 1.9328e-05 - accuracy: 1.0000 - val_loss: 7.0391 - val_accuracy: 0.2986
Epoch 26/30
11/11 [==============================] - 27s 2s/step - loss: 1.7196e-05 - accuracy: 1.0000 - val_loss: 7.0967 - val_accuracy: 0.2986
Epoch 27/30
11/11 [==============================] - 27s 2s/step - loss: 1.5431e-05 - accuracy: 1.0000 - val_loss: 7.1239 - val_accuracy: 0.3056
Epoch 28/30
11/11 [==============================] - 27s 2s/step - loss: 1.3852e-05 - accuracy: 1.0000 - val_loss: 7.1493 - val_accuracy: 0.2986
Epoch 29/30
11/11 [==============================] - 27s 2s/step - loss: 1.2641e-05 - accuracy: 1.0000 - val_loss: 7.2041 - val_accuracy: 0.2986
Epoch 30/30
11/11 [==============================] - 27s 2s/step - loss: 1.1668e-05 - accuracy: 1.0000 - val_loss: 7.2112 - val_accuracy: 0.2986
```



4) EmoDB

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 155, 318, 32)      320
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 77, 159, 32)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 75, 157, 64)       18496
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 37, 78, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 35, 76, 64)        36928
_________________________________________________________________
flatten_4 (Flatten)          (None, 170240)            0
_________________________________________________________________
dense_8 (Dense)              (None, 64)                10895424
_________________________________________________________________
dense_9 (Dense)              (None, 10)                650
=================================================================
Total params: 10,951,818
Trainable params: 10,951,818
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 14/20
12/12 [==============================] - 30s 2s/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 3.9037 - val_accuracy: 0.5155
Epoch 15/20
12/12 [==============================] - 30s 2s/step - loss: 7.0827e-04 - accuracy: 1.0000 - val_loss: 4.0446 - val_accuracy: 0.4969
Epoch 16/20
12/12 [==============================] - 30s 2s/step - loss: 4.9740e-04 - accuracy: 1.0000 - val_loss: 4.1150 - val_accuracy: 0.5093
Epoch 17/20
12/12 [==============================] - 30s 3s/step - loss: 3.8747e-04 - accuracy: 1.0000 - val_loss: 4.1542 - val_accuracy: 0.5031
Epoch 18/20
12/12 [==============================] - 30s 2s/step - loss: 3.0542e-04 - accuracy: 1.0000 - val_loss: 4.2023 - val_accuracy: 0.5031
Epoch 19/20
12/12 [==============================] - 31s 3s/step - loss: 2.5256e-04 - accuracy: 1.0000 - val_loss: 4.2239 - val_accuracy: 0.5031
Epoch 20/20
12/12 [==============================] - 30s 2s/step - loss: 2.1154e-04 - accuracy: 1.0000 - val_loss: 4.2753 - val_accuracy: 0.5093
```



It was observed that the more layers we add the higher accuracy we can achieve. At the same time, if we keep on adding more layers, the final accuracy will saturate. Also, the number of convolution and the pooling layers play an important role in training the model.

PART 3

1) VGG-16

1.1) CIFAR-10

```
Epoch 1/5
63/63 [==============================] - 97s 855ms/step - loss: nan - accuracy: 0.0985
Epoch 2/5
63/63 [==============================] - 47s 743ms/step - loss: nan - accuracy: 0.1010
Epoch 3/5
63/63 [==============================] - 47s 745ms/step - loss: nan - accuracy: 0.1010
Epoch 4/5
63/63 [==============================] - 47s 744ms/step - loss: nan - accuracy: 0.1010
Epoch 5/5
63/63 [==============================] - 47s 744ms/step - loss: nan - accuracy: 0.1010

[ ] model.evaluate(X_test_resized, y_test)

63/63 [==============================] - 14s 225ms/step - loss: nan - accuracy: 0.0980
[nan, 0.09799999743700027]
```

1.2) MNIST

```
Epoch 1/5
63/63 [==============================] - 71s 877ms/step - loss: 3.6193 - accuracy: 0.0885
Epoch 2/5
63/63 [==============================] - 48s 764ms/step - loss: 2.5311 - accuracy: 0.0990
Epoch 3/5
63/63 [==============================] - 48s 763ms/step - loss: 2.5246 - accuracy: 0.1105
Epoch 4/5
63/63 [==============================] - 48s 763ms/step - loss: 2.5036 - accuracy: 0.1040
Epoch 5/5
63/63 [==============================] - 48s 763ms/step - loss: 2.4806 - accuracy: 0.0965

model.evaluate(X_test_new, y_test)

63/63 [==============================] - 15s 233ms/step - loss: 2.6352 - accuracy: 0.1095
[2.6351511478424072, 0.10949999839067459]
```

1.3) SAVEE

```
8/8 [==============================] - 6s 708ms/step - loss: nan - accuracy: 0.1208
Epoch 45/50
8/8 [==============================] - 6s 706ms/step - loss: nan - accuracy: 0.1208
Epoch 46/50
8/8 [==============================] - 6s 705ms/step - loss: nan - accuracy: 0.1208
Epoch 47/50
8/8 [==============================] - 6s 706ms/step - loss: nan - accuracy: 0.1208
Epoch 48/50
8/8 [==============================] - 6s 709ms/step - loss: nan - accuracy: 0.1208
Epoch 49/50
8/8 [==============================] - 6s 706ms/step - loss: nan - accuracy: 0.1208
Epoch 50/50
8/8 [==============================] - 6s 706ms/step - loss: nan - accuracy: 0.1208
```

```
model.evaluate(X_test_resized, y_test)
```

```
8/8 [==============================] - 2s 215ms/step - loss: nan - accuracy: 0.1292
[nan, 0.12916666269302368]
```

1.4) EmoDB

```
9/9 [==============================] - 6s 711ms/step - loss: nan - accuracy: 0.2247
Epoch 14/20
9/9 [==============================] - 6s 712ms/step - loss: nan - accuracy: 0.2247
Epoch 15/20
9/9 [==============================] - 6s 713ms/step - loss: nan - accuracy: 0.2247
Epoch 16/20
9/9 [==============================] - 6s 712ms/step - loss: nan - accuracy: 0.2247
Epoch 17/20
9/9 [==============================] - 6s 712ms/step - loss: nan - accuracy: 0.2247
Epoch 18/20
9/9 [==============================] - 6s 711ms/step - loss: nan - accuracy: 0.2247
Epoch 19/20
9/9 [==============================] - 6s 712ms/step - loss: nan - accuracy: 0.2247
Epoch 20/20
9/9 [==============================] - 6s 712ms/step - loss: nan - accuracy: 0.2247
```

```
model.evaluate(X_test_resized, y_test)
```

```
9/9 [==============================] - 6s 718ms/step - loss: nan - accuracy: 0.2500
[nan, 0.25]
```

The entire model can be broken down into 5 blocks, where each block contains 3 convolution and 1 max-pooling layers.

Looking at the complexity of the model and the limitations of google colab, I have reduced the input size for the model,i.e., i have taken 2000 training data points and 2000 testing data points.

2) ResNet-50

2.1) CIFAR-10

```
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.2/resnet5
102858752/102853048 [==============================] - 1s 0us/step
102866944/102853048 [==============================] - 1s 0us/step
Epoch 1/5
63/63 [==============================] - 81s 703ms/step - loss: 2.9229 - accuracy: 0.0975
Epoch 2/5
63/63 [==============================] - 42s 673ms/step - loss: 2.4506 - accuracy: 0.1040
Epoch 3/5
63/63 [==============================] - 42s 673ms/step - loss: 2.2995 - accuracy: 0.1755
Epoch 4/5
63/63 [==============================] - 42s 672ms/step - loss: 2.1401 - accuracy: 0.2325
Epoch 5/5
63/63 [==============================] - 42s 672ms/step - loss: 2.0272 - accuracy: 0.2715
```

```
model.evaluate(X_test_resized, y_test)
```

```
63/63 [==============================] - 15s 217ms/step - loss: 16.8393 - accuracy: 0.0000e+00
[16.839269638061523, 0.0]
```

2.2) MNIST

```
Epoch 6/10
63/63 [==============================] - 44s 705ms/step - loss: 0.0647 - accuracy: 0.9825
Epoch 7/10
63/63 [==============================] - 44s 705ms/step - loss: 0.0655 - accuracy: 0.9815
Epoch 8/10
63/63 [==============================] - 44s 705ms/step - loss: 0.0429 - accuracy: 0.9890
Epoch 9/10
63/63 [==============================] - 44s 705ms/step - loss: 0.0272 - accuracy: 0.9950
Epoch 10/10
63/63 [==============================] - 44s 704ms/step - loss: 0.0121 - accuracy: 0.9985
```

```
model.evaluate(X_test_new, y_test)
```

```
63/63 [==============================] - 14s 227ms/step - loss: 13.1569 - accuracy: 0.0000e+00
[13.156914710998535, 0.0]
```

2.3) SAVEE

```
8/8 [==============================] - 5s 669ms/step - loss: 1.0197 - accuracy: 0.6833
Epoch 5/10
8/8 [==============================] - 5s 667ms/step - loss: 0.5054 - accuracy: 0.9167
Epoch 6/10
8/8 [==============================] - 5s 669ms/step - loss: 0.2390 - accuracy: 0.9833
Epoch 7/10
8/8 [==============================] - 5s 671ms/step - loss: 0.0966 - accuracy: 1.0000
Epoch 8/10
8/8 [==============================] - 5s 668ms/step - loss: 0.0691 - accuracy: 1.0000
Epoch 9/10
8/8 [==============================] - 5s 669ms/step - loss: 0.0550 - accuracy: 1.0000
Epoch 10/10
8/8 [==============================] - 5s 666ms/step - loss: 0.0281 - accuracy: 1.0000
```

```
model.evaluate(X_test_resized, y_test)
```

```
8/8 [==============================] - 3s 215ms/step - loss: 8.7594 - accuracy: 0.0000e+00
[8.759380340576172, 0.0]
```

2.4) EmoDB

```
Epoch 3/10
9/9 [==============================] - 6s 663ms/step - loss: 1.1062 - accuracy: 0.6367
Epoch 4/10
9/9 [==============================] - 6s 661ms/step - loss: 0.6534 - accuracy: 0.7678
Epoch 5/10
9/9 [==============================] - 6s 662ms/step - loss: 0.3835 - accuracy: 0.8914
Epoch 6/10
9/9 [==============================] - 6s 662ms/step - loss: 0.3716 - accuracy: 0.8689
Epoch 7/10
9/9 [==============================] - 6s 662ms/step - loss: 0.2297 - accuracy: 0.9213
Epoch 8/10
9/9 [==============================] - 6s 661ms/step - loss: 0.1096 - accuracy: 0.9775
Epoch 9/10
9/9 [==============================] - 6s 664ms/step - loss: 0.1170 - accuracy: 0.9850
Epoch 10/10
9/9 [==============================] - 6s 659ms/step - loss: 0.2414 - accuracy: 0.9251

model.evaluate(X_test_resized, y_test)

9/9 [==============================] - 4s 304ms/step - loss: 7.2902 - accuracy: 0.0000e+00
[7.290168285369873, 0.0]
```

Looking at the complexity of the model and the limitations of google colab, I have reduced the input size for the model,i.e., I have taken 2000 training data points and 2000 testing data points.

3) Recurrent Neural Networks (RNN)

3.1) CIFAR-10

```
Epoch 3/10
200/200 [==============================] - 111s 557ms/step - loss: 2.0085 - accuracy: 0.2645
Epoch 4/10
200/200 [==============================] - 112s 558ms/step - loss: 1.9649 - accuracy: 0.2771
Epoch 5/10
200/200 [==============================] - 111s 557ms/step - loss: 1.9583 - accuracy: 0.2816
Epoch 6/10
200/200 [==============================] - 111s 557ms/step - loss: 1.9388 - accuracy: 0.2896
Epoch 7/10
200/200 [==============================] - 111s 557ms/step - loss: 1.9371 - accuracy: 0.2899
Epoch 8/10
200/200 [==============================] - 111s 556ms/step - loss: 1.9254 - accuracy: 0.2989
Epoch 9/10
200/200 [==============================] - 111s 557ms/step - loss: 1.9188 - accuracy: 0.2966
Epoch 10/10
200/200 [==============================] - 111s 556ms/step - loss: 1.9341 - accuracy: 0.2930
```

```
model.evaluate(test_images, test_labels)
```

```
157/157 [==============================] - 38s 225ms/step - loss: 1.9601 - accuracy: 0.2912
[1.9600898027420044, 0.29120001196861267]
```

3.2) MNIST

```
print('Test Accuracy of the model on the 10000 test images: {} %'.format(100 * correct / total))
```

```
Test Accuracy of the model on the 10000 test images: 97.77 %
```

3.3) SAVEE

```
[ ] model.evaluate(X_test, y_test)
```

```
6/6 [==============================] - 2s 57ms/step - loss: 1.4087 - accuracy: 0.4323
[1.408677577972412, 0.4322916567325592]
```

3.4) EmoDB

```
8/8 [==============================] - 1s 130ms/step - loss: 0.2239 - accuracy: 0.9144
Epoch 50/50
8/8 [==============================] - 1s 127ms/step - loss: 0.2858 - accuracy: 0.8984
```

```
[ ] model.evaluate(X_test, y_test)
```

```
6/6 [==============================] - 2s 54ms/step - loss: 1.7593 - accuracy: 0.5590
[1.7592660188674927, 0.5590062141418457]
```

Looking at the complexity of the model and the limitations of google colab, I have reduced the input size for the model,i.e., I have taken 2000 training data points and 2000 testing data points.

4) AlexNet

4.1) CIFAR-10

```
Epoch 1/5
16/16 [==============================] - 61s 4s/step - loss: 2.6788 - accuracy: 0.1240
Epoch 2/5
16/16 [==============================] - 59s 4s/step - loss: 2.6332 - accuracy: 0.0960
Epoch 3/5
16/16 [==============================] - 60s 4s/step - loss: 2.5360 - accuracy: 0.1220
Epoch 4/5
16/16 [==============================] - 59s 4s/step - loss: 2.4222 - accuracy: 0.1400
Epoch 5/5
16/16 [==============================] - 60s 4s/step - loss: 2.3684 - accuracy: 0.1400
```

+ Code + Text

```
[ ] model.evaluate(X_test_resized, y_test)

7/7 [==============================] - 6s 753ms/step - loss: 2.3611 - accuracy: 0.0750
[2.3610661029815674, 0.07500000298023224]
```

4.2) MNIST

```
Epoch 1/5
63/63 [==============================] - 474s 8s/step - loss: 2.0734 - accuracy: 0.2610
Epoch 2/5
63/63 [==============================] - 476s 8s/step - loss: 1.7821 - accuracy: 0.3680
Epoch 3/5
63/63 [==============================] - 468s 7s/step - loss: 1.6773 - accuracy: 0.4395
Epoch 4/5
63/63 [==============================] - 469s 7s/step - loss: 1.5820 - accuracy: 0.4810
Epoch 5/5
63/63 [==============================] - 472s 7s/step - loss: 1.5318 - accuracy: 0.5040
```

```
[ ] model.evaluate(X_test_new, y_test)
```

```
63/63 [==============================] - 107s 2s/step - loss: 2.3417 - accuracy: 0.1170
[2.3417277336120605, 0.11699999868869781]
```

4.3) SAVEE

```
Epoch 4/10
8/8 [==============================] - 56s 7s/step - loss: 2.4215 - accuracy: 0.1583
Epoch 5/10
8/8 [==============================] - 56s 7s/step - loss: 2.2042 - accuracy: 0.2333
Epoch 6/10
8/8 [==============================] - 57s 7s/step - loss: 2.2080 - accuracy: 0.2042
Epoch 7/10
8/8 [==============================] - 56s 7s/step - loss: 2.1114 - accuracy: 0.2792
Epoch 8/10
8/8 [==============================] - 57s 7s/step - loss: 2.1120 - accuracy: 0.2542
Epoch 9/10
8/8 [==============================] - 56s 7s/step - loss: 2.0292 - accuracy: 0.2583
Epoch 10/10
8/8 [==============================] - 57s 7s/step - loss: 2.1150 - accuracy: 0.2417
```

```
model.evaluate(X_test_resized, y_test)
```

```
8/8 [==============================] - 13s 2s/step - loss: 2.2758 - accuracy: 0.2375
[2.275780200958252, 0.23749999701976776]
```

4.4) EmoDB

```
11/11 [==============================] - 75s 7s/step - loss: 2.2610 - accuracy: 0.1838
Epoch 6/10
11/11 [==============================] - 75s 7s/step - loss: 2.1741 - accuracy: 0.2025
Epoch 7/10
11/11 [==============================] - 77s 7s/step - loss: 2.0738 - accuracy: 0.2336
Epoch 8/10
11/11 [==============================] - 76s 7s/step - loss: 2.0492 - accuracy: 0.2679
Epoch 9/10
11/11 [==============================] - 76s 7s/step - loss: 2.0359 - accuracy: 0.2679
Epoch 10/10
11/11 [==============================] - 76s 7s/step - loss: 1.9726 - accuracy: 0.3115
```

```
model.evaluate(X_test_resized, y_test)
```

```
7/7 [==============================] - 12s 2s/step - loss: 2.1748 - accuracy: 0.2336
[2.1748006343841553, 0.23364485800266266]
```

Looking at the complexity of the model and the limitations of google colab, I have reduced the input size for the model,i.e., I have taken 2000 training data points and 2000 testing data points.

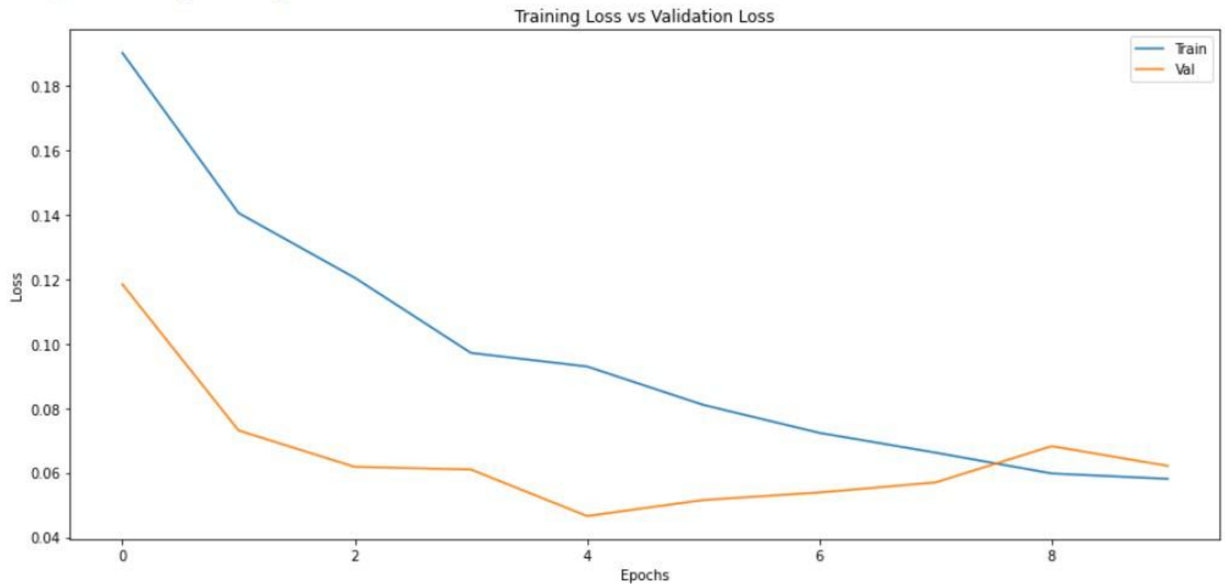
5) GoogLeNet

5.1) CIFAR-10

```
output_2_loss: 2.0650 - val_output_accuracy: 0.2305 - val_auxilliary_output_1_accuracy: 0.2400 - val_auxilliary_output_2_accuracy: 0.2240

output_2_loss: 2.0244 - val_output_accuracy: 0.2470 - val_auxilliary_output_1_accuracy: 0.2630 - val_auxilliary_output_2_accuracy: 0.2585

output_2_loss: 2.0076 - val_output_accuracy: 0.2355 - val_auxilliary_output_1_accuracy: 0.2735 - val_auxilliary_output_2_accuracy: 0.2660
```

5.2) MNIST

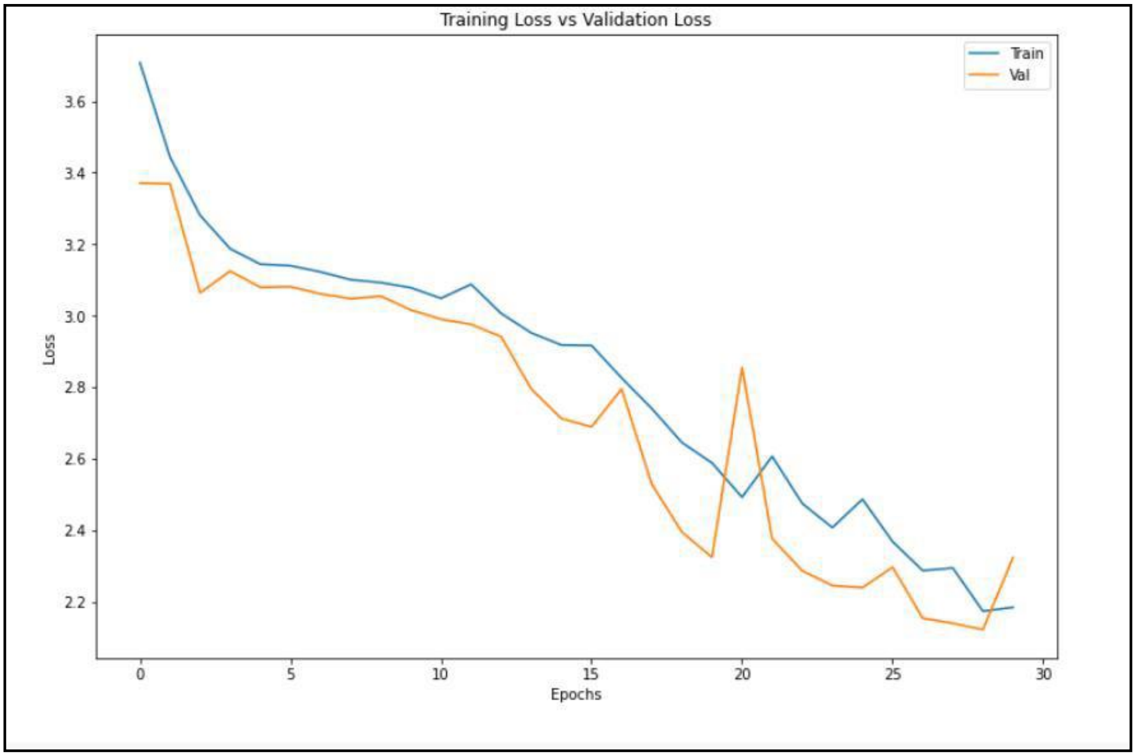
```
<matplotlib.legend.Legend at 0x7feaad89cf50>
```




```
model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 24s 6
[0.05425573140382767,
 0.03818700090050697,
 0.02569129876792431,
 0.027871087193489075,
 0.9873999953269958,
 0.9926000237464905,
 0.9902999997138977]
```

5.3) SAVEE



0 5 10

```
model.evaluate(X_test, y_test)

6/6 [==============================]
[2.324364423751831,
 1.458065390586853,
 1.4511628150939941,
 1.4365006685256958,
 0.3541666567325592,
 0.3333333432674408,
 0.3802083432674408]
```

5.4) EmoDB


```
model.evaluate(X_test, y_test)

7/7  [==============================   ]
[2.8008506298065186,
 1.7843737602233887,
 1.6508854627609253,
 1.7373706102371216,
 0.30841121077537537,
 0.38785046339035034,
 0.3644859790802002]
```

Looking at the complexity of the model and the limitations of google colab, I have reduced the input size for the model,i.e., I have taken 2000 training data points and 2000 testing data points.