# CAP 5415- Assignment 4

## Computer Vision

**Abraham Jose**

**Question1:**
Train and test a 10-class Fisher Linear Discriminant Function classifier using the Mahalanobis distance as the metric for classification.

The performance while using fisher was inferior when compared to CNN model
Dataset per class, testing and training for both case is follows from 1 to 10:

| | | |
|---|---|---|
| Class-label **:airplane** | **train-1005** | **test-100** |
| Class-label **:automobile** | **train-974** | **test-100** |
| Class-label **:bird** | **train-1032** | **test-100** |
| Class-label **:cat** | **train-1016** | **test-100** |
| Class-label **:deer** | **train-999** | **test-100** |
| Class-label **:dog** | **train-937** | **test-100** |
| Class-label **:frog** | **train-1030** | **test-100** |
| Class-label **:horse** | **train-1001** | **test-100** |
| Class-label **:ship** | **train-1025** | **test-100** |
| Class-label **:truck** | **train-981** | **test-100** |

Confusion Matrix is as follows:
( predicted in column and actual in rows)

**[[ 0. 0. 9. 2. 9. 3. 65. 0. 12. 0.]**
**[ 0. 0. 5. 8. 2. 9. 72. 0. 4. 0.]**
**[ 0. 0. 12. 4. 7. 7. 65. 0. 5. 0.]**
**[ 0. 0. 9. 2. 8. 13. 67. 0. 1. 0.]**
**[ 0. 0. 6. 2. 7. 12. 71. 0. 2. 0.]**
**[ 0. 0. 8. 5. 5. 10. 71. 0. 1. 0.]**
**[ 0. 0. 11. 4. 5. 11. 69. 0. 0. 0.]**
**[ 0. 0. 16. 3. 11. 9. 54. 0. 7. 0.]**
**[ 0. 0. 9. 0. 9. 10. 51. 0. 21. 0.]**
**[ 0. 0. 8. 7. 5. 8. 57. 0. 15. 0.]]**

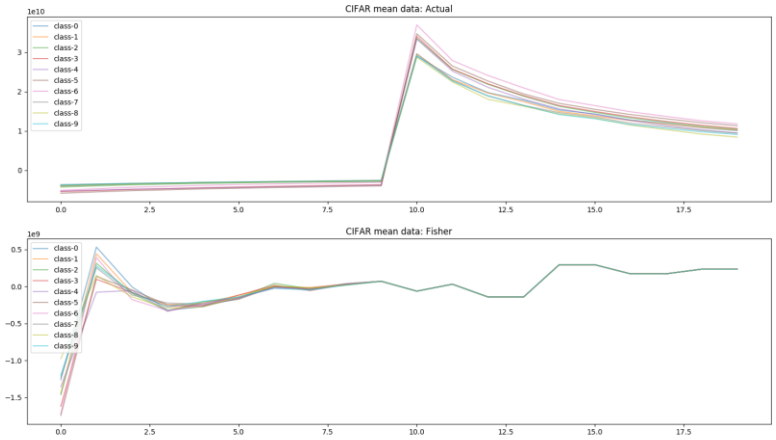
Accuracy: **12%** (While using 3 dominant features, **13%** accuracy achieved)
Class-wise error rates are as follows:

**Class-airplane:100.0%**
**Class-automobile:100.0%**
**Class-bird:88.0%**
**Class-cat:98.0%**
**Class-deer:93.0%**
**Class-dog:90.0%**
**Class-frog:31.0%**
**Class-horse:100.0%**
**Class-ship:79.0%**
**Class-truck:100.0%**

To reproduce the result, execute:

***python Harris_Fisher_Classifier.py***

The mean vectors in Fisher space and actual are as follows. I have used a 20-dimensional hyperspace for the dataset conversion.



**Question2:**
Train and test the CNN as per the architecture:

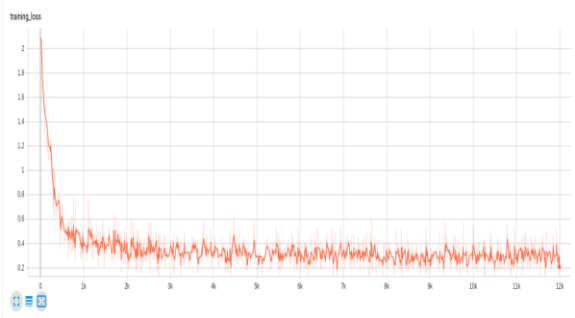
The model is created in pytorch and model summary is as follows.
For zero center normalization of a given batch I have used batch norm with trainable parameter.

**Model Summary:**

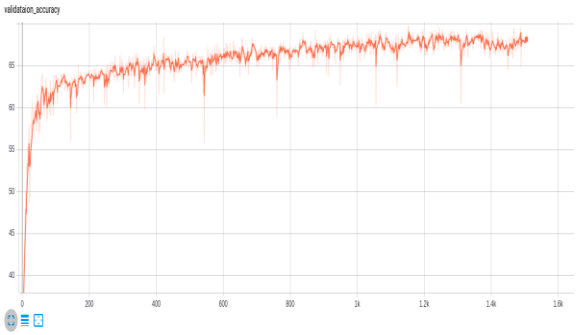
```
----------------------------------------------------------------------------------------------------
      Layer (type)               Output Shape         Param #
====================================================================
    BatchNorm2d-              [-1, 3, 32, 32]               6
        Conv2d-2             [-1, 32, 32, 32]           2,432
        Conv2d-3             [-1, 32, 12, 12]          25,632
        Conv2d-4             [-1, 64, 2, 2]            51,264
        Linear-5                   [-1, 64]             4,160
        Linear-6                   [-1, 10]               650
====================================================================
Total params: 84,144  | Trainable params: 84,144 | Non-trainable params: 0
----------------------------------------------------------------------------------------------------
Input size (MB): 0.01 | Forward/backward pass size (MB): 0.31
Params size (MB): 0.32 | Estimated Total Size (MB): 0.64
----------------------------------------------------------------------------------------------------
```

Training Log: **../data/training_log.log**

**Training loss:**




**Validation accuracy:**



Accuracy obtained = **70.2%**

Confusion Matrix is as follows:
( predicted in column and actual in rows)

**[[69. 5. 3. 1. 1. 1. 1. 1. 11. 7.]**
**[ 0. 93. 0. 0. 2. 0. 0. 1. 1. 3.]**
**[ 9. 1. 50. 15. 7. 11. 4. 0. 2. 1.]**
**[ 7. 0. 9. 53. 5. 11. 7. 6. 0. 2.]**
**[ 5. 1. 2. 4. 69. 2. 3. 8. 2. 4.]**
**[ 4. 1. 10. 20. 4. 54. 4. 2. 0. 1.]**
**[ 1. 1. 6. 6. 6. 3. 73. 2. 0. 2.]**
**[ 0. 1. 0. 6. 10. 4. 1. 75. 0. 3.]**
**[ 4. 7. 3. 0. 1. 0. 3. 0. 81. 1.]**
**[ 1. 3. 0. 2. 0. 1. 0. 2. 6. 85.]]**

Error rates class- wise:

**Class-airplane:31.0%**
**Class-automobile:7.0%**
**Class-bird:50.0%**
**Class-cat:47.0%**
**Class-deer:31.0%**
**Class-dog:46.0%**
**Class-frog:27.0%**
**Class-horse:25.0%**
**Class-ship:19.0%**
**Class-truck:15.0%**

To reproduce the result execute:

***python CNN_Classifier.py --test_model = True***

For tensorboard visulaization:

***tensorboard –logdir=../data/model/runs***

From the given tabulation, it is clear that the CNN model has superior performance when compared to Harris feature detector using Fisher analysis to classify the detected features. This is attributed to the feature extraction using feature mask while using CNNs.