1. Fisher's Linear Discriminant Function

Inorder to perform classification on dataset with two or more classes, Machine learning algorithms apply some kind of transformation to the input data by reducing the original input dimensions to a new one with smaller dimensions. Then the algorithm tries to classify the projected data by finding a linear separation. So, the Fisher's Linear Discriminant Function can be viewed as dimensionality reduction technique.
Fisher's idea is to maximize the function such that the separation between the projected class is large while also maintaining a small variance within each class which results in minimization of class overlap. Hence, Fisher's LDF projects the data to a smaller dimension and avoids class overlapping by maximizing the ratio between the between-class variance to the within-class variance.

1.1 Algorithm

- Find mean vectors for each class $\boldsymbol{\mu}_i$ and overall mean of data $\boldsymbol{\mu}$.

$$\boldsymbol{\mu}_i = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{x}_k^i \qquad \boldsymbol{\mu} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\mu}_i$$

- Find between class scatter matrix **B**

$$\boldsymbol{B} = \sum_{i=1}^{M}(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$$

- Find covariance matrix of each class $\sum_i$, and their sum **A.**

$$\boldsymbol{\Sigma}_i = \frac{1}{N}\sum_{k=1}^{N}(\boldsymbol{x}_k^i - \boldsymbol{\mu}_i)(\boldsymbol{x}_k^i - \boldsymbol{\mu}_i)^T \quad \boldsymbol{A} = \sum_{i=1}^{M} \boldsymbol{\Sigma}_i$$

- Compute eigenvectors and values of $\mathbf{A}^{-1}\mathbf{B}$
  - Since there are ten classes, the rank of **B** is 9, and we select dominant 9 eigenvalues($\lambda_1,..,\lambda_9$)
- Select the dominant eigenvectors ($\boldsymbol{h}_1,...,\boldsymbol{h}_9$) corresponding to the largest eigen-values as the LDF vectors.
- Using the nine Fisher LDF's, the ten classes are easy to separate defining a matrix **H.**
  - **H** = [$\boldsymbol{h}_1\ \boldsymbol{h}_2 \ldots\ldots \boldsymbol{h}_9$] is a d * 9 matrix, where d is a dimension of dataset.
- Transform the feature vectors $\boldsymbol{x}_k^i$ into the Fisher LDF space $\boldsymbol{f}_k^i = \boldsymbol{H}^T \boldsymbol{x}_k^i$
- Transform the class means and covariance matrices to the Fisher LDF space as
  - $\boldsymbol{m}_i = \boldsymbol{H}^T \boldsymbol{\mu}_i$ and $S_i = \boldsymbol{H}^T \sum_i \boldsymbol{H}$
- Classify data by assigning it to the class with the smallest Mahalanobis Distance
  - For each test cases k, find $d_i = (\boldsymbol{f}_k^i - \boldsymbol{m}_i)^T \boldsymbol{S}_i^{-1} (\boldsymbol{f}_k^i - \boldsymbol{m}_i)$ for all i classes

- Assign the label of the class for which d_i is the smallest

1.2 Dataset

For the experiment we used CIFAR-10 dataset. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. For our experiment, we took the first training batch as training set that contains 10000 images and only the first 1000 images from test batch as test sample.

1.3 Results

For evaluating the performance of the classifier, we observed the class prediction for both the training and test dataset and calculated the accuracy in both cases. Also we calculated the error rate for classification of each 10 classes on both test and training data as shown in Table I.The accuracy of the classifier when tested on training sample is 46% and on test data is 23%.

Data	**Error Rate(%)**										**Accuracy(%)**
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Train	58.4	31.93	71.41	61.22	70.87	55.81	44.66	45.35	63.21	31.09	46
Test	81.55	55.05	86.0	88.34	87.77	82.55	76.78	75.49	70.75	67.88	23

TableI. Error rate for each class and overall accuracy on test and train data

The confusion matrix of the classifier when evaluated on the test and training sample is shown in fig1 and fig2. We can see that most of the data points were misclassified with this algorithm.

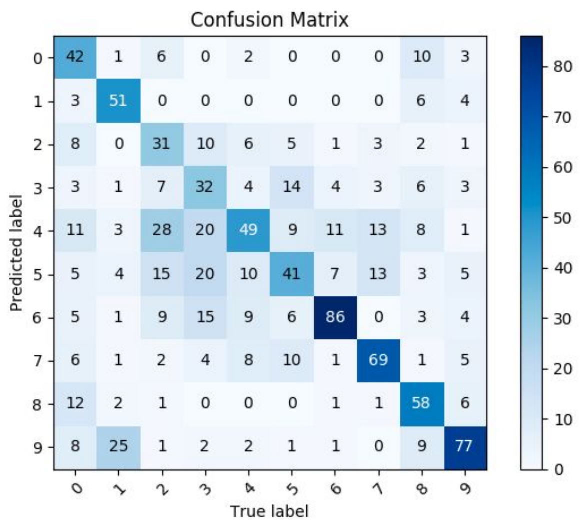


Fig1.Confusion matrix when evaluated on test data

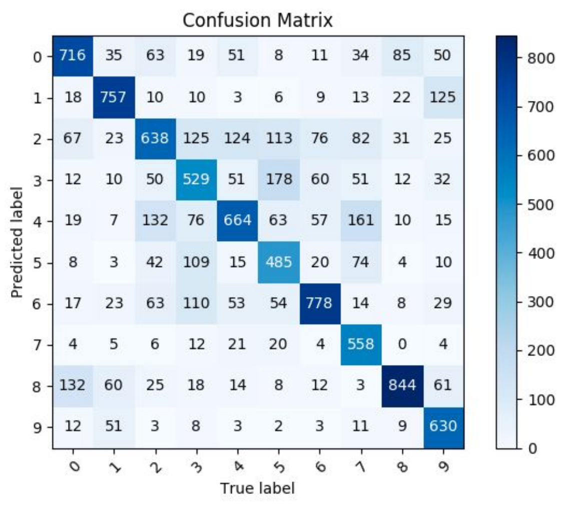


Fig2.Confusion matrix when evaluated on test data

2. Convolution Neural Network (CNN)

CNN is a class of deep neural network that have proven very effective in areas such as image recognition and classification. CNN uses mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. It consists of an input layer and a series of convolutional layers called hidden layers. The activation function is generally a RELU layer which is followed by additional convolutons such as pooling layers and fully connected layers. So,There are four main operations in the CNN: Convolution, ReLU, Pooling, Fully connected layer.

2.1 Convolution

The primary purpose of Convolution is to extract features from the input image. Each convolution layer in a network have the following attributes:

- A tensor input of (number of images * image width * image height * image depth) shape
- Convolution kernel whose height and width are hyper parameters and depth equals to that of the image. Kernel slides over the input image to produce a feature map.

2.2 RELU

ReLU is applied for every pixel after every convolution operation which replaces all negative pixel values in the feature map by zero to introduce non-linearity in CNN. Other nonlinear functions like sigmoid can also be used but ReLU has better performance in most situations.

2.2 Pooling

Poling reduces the dimension of the data but retains the most important information. Pooling are of different types: Max, Average, Sum etc. In max pooling, we take the largest element from the feature map within the specified window. Pooling reduces the number of parameters and computations in the network, therefore, controlling overfitting.

2.4 Fully connected layer

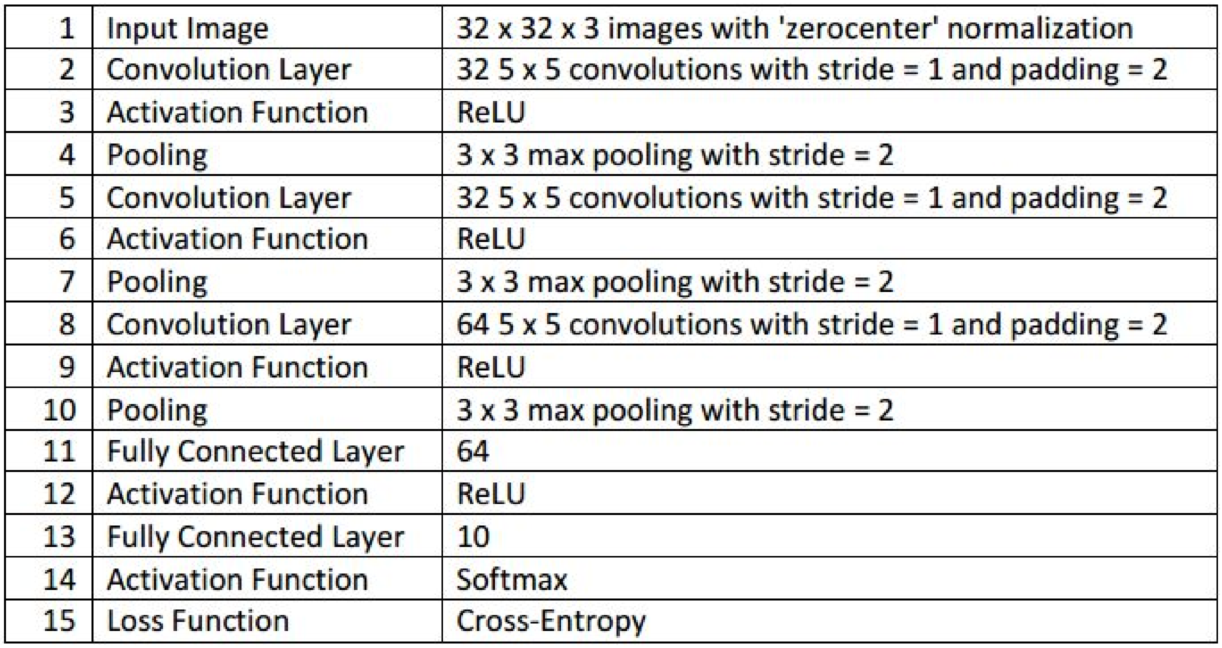
The fully connected layer is a traditional multi layer perceptron where every neuron in one layer is connected to every neuron in another layer. It uses a softmax activation function in the output layer to classify the images.

2.5 Full Network

The entire network can be seen to be composed of various convolutional and pooling layers. Input image is passed to the first convolutional layers. The convolved output is obtained after ReLU operation as an activation map. Each filter applied in the convolution layer gives a different feature during training the network. Pooling layers are also added to reduce the number of parameters. The output layer is the fully connected layer in which the inputs from the previous layer is flattened to transform the output into the number of classes as specified in the network. The loss function is also evaluated at the output layer and the error is propagated to update the weights and bias values. Each training cycle has forward and backward pass.

2.6 Implementation

We have used Keras library written in python which can run on top of other machine learning libraries like TensorFlow. We have designed a model using "add" function in Keras with three convolution layers followed by max pooling and two fully connected layers. We have passed the parameters such as filter size, activation, stride, padding in the add function as specified in figure III. And the we compiled the model using "compile" function of Keras. Finally, we train the model using "fit" function and evaluate the model in terms of accuracy. For fitting the model, we need to specify number of training epochs and batch size.The test dataset was used as a validation dataset and evaluated at the end of each training epoch.The architecture of our CNN is model is shown in tableII:



1	Input Image	32 x 32 x 3 images with 'zerocenter' normalization
2	Convolution Layer	32 5 x 5 convolutions with stride = 1 and padding = 2
3	Activation Function	ReLU
4	Pooling	3 x 3 max pooling with stride = 2
5	Convolution Layer	32 5 x 5 convolutions with stride = 1 and padding = 2
6	Activation Function	ReLU
7	Pooling	3 x 3 max pooling with stride = 2
8	Convolution Layer	64 5 x 5 convolutions with stride = 1 and padding = 2
9	Activation Function	ReLU
10	Pooling	3 x 3 max pooling with stride = 2
11	Fully Connected Layer	64
12	Activation Function	ReLU
13	Fully Connected Layer	10
14	Activation Function	Softmax
15	Loss Function	Cross-Entropy

Table II: CNN architecture

2.7. Results

For the CIFAR-10 dataset with 10 classes (0 – 9), we took 10000 images as training sample and 1000 images as test sample. For evaluating the performance of the classifier, we observed the class prediction for both the training and test dataset and calculated the accuracy in both cases.
The learning rate for the experiment was taken 0.001 and momentum of 0.9 and the batch size of 64.
The learning behavior of the model in terms of loss and accuracy during training and the estimation of the model performance is shown in the fig3. The blue lines indicate model performance on the training dataset and orange lines indicate performance on the test dataset for different number of epochs. The accuracy on test dataset increases until epochs is 20 and beyond this it remains constant. So if we train the model beyond epochs =20, the accuracy on training set might increase but this gives overfitting problem. Similarly the loss on the test set remains constant after 20 epochs. So the CNN has the best result when epochs is 20.

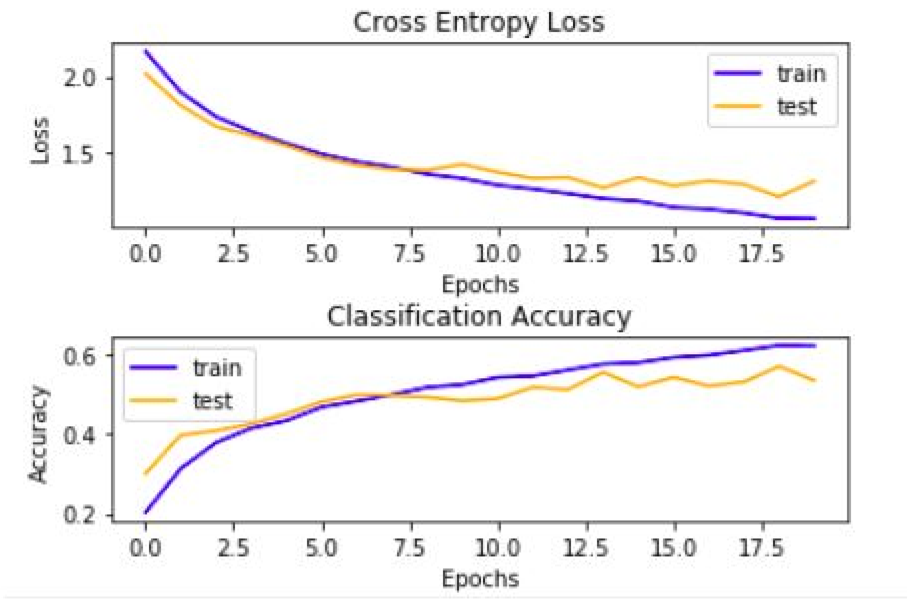


Fig3. accuracy and loss of the CNN model vs epochs

2.7.1. Accuracy

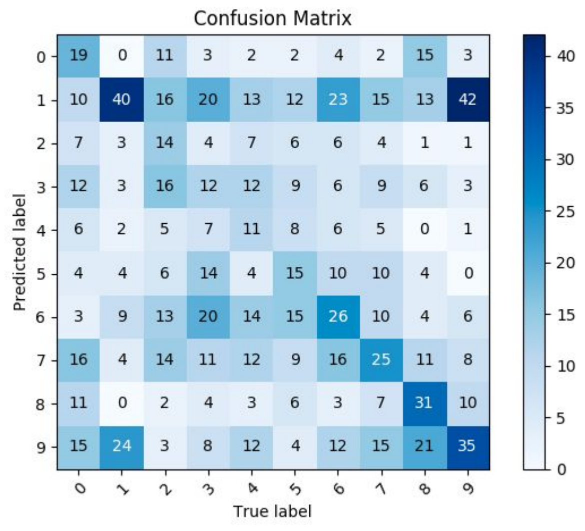
Also we calculated the error rate for classification of each 10 class on both test and training data as shown in Table III.The accuracy of the classifier when tested on training sample is 65.99% and on test data is 54.7%. In case of training data, class '5' has the highest misclassification error and class '8' has the least misclassification error. Similarly, in the case of test data, class '3' has the highest misclassification error and class '6' has the least misclassification error.

Data	**Error Rate(%)**										**Accuracy(%)**
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Train	28.75	22.79	38.17	47.93	33.53	48.23	24.46	44.25	17.65	35.77	65.99
Test	41.74	37.07	69.0	78.64	60.0	62.79	16.07	36.27	19.81	39.44	54.7

TableIII. Error rate for each class and overall accuracy on test and train data

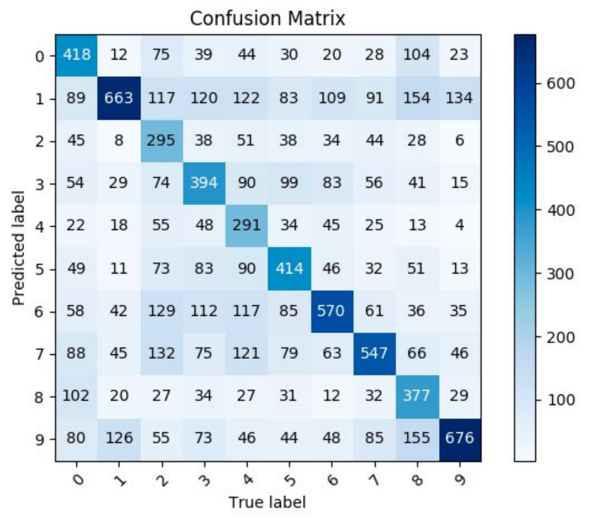
2.7.2. Confusion Matrix

The confusion matrix of the CNN when evaluated on the test and training sample is shown in the figure4 and figure5. We can see that most of the data points were correctly classified.



Predicted \ True	0	1	2	3	4	5	6	7	8	9
0	19	0	11	3	2	2	4	2	15	3
1	10	40	16	20	13	12	23	15	13	42
2	7	3	14	4	7	6	6	4	1	1
3	12	3	16	12	12	9	6	9	6	3
4	6	2	5	7	11	8	6	5	0	1
5	4	4	6	14	4	15	10	10	4	0
6	3	9	13	20	14	15	26	10	4	6
7	16	4	14	11	12	9	16	25	11	8
8	11	0	2	4	3	6	3	7	31	10
9	15	24	3	8	12	4	12	15	21	35

Fig4. Confusion matrix when evaluated on test data



Predicted \ True	0	1	2	3	4	5	6	7	8	9
0	418	12	75	39	44	30	20	28	104	23
1	89	663	117	120	122	83	109	91	154	134
2	45	8	295	38	51	38	34	44	28	6
3	54	29	74	394	90	99	83	56	41	15
4	22	18	55	48	291	34	45	25	13	4
5	49	11	73	83	90	414	46	32	51	13
6	58	42	129	112	117	85	570	61	36	35
7	88	45	132	75	121	79	63	547	66	46
8	102	20	27	34	27	31	12	32	377	29
9	80	126	55	73	46	44	48	85	155	676

Fig5. Confusion matrix when evaluated on train data

Instructions to run the code

1. Install the following required dependencies
```
$ sudo apt-get install python3-pip
$ pip3 install numpy
$ pip3 install matplotlib
$ pip3 install keras
$ pip3 install tensorflow
```
2. Unzip the file “4805322.zip”
```
$ sudo apt-get install unzip
$ unzip 4805322.zip
```

3. From inside the code directory, run the python files
```
$ python3 CNN.py
$ python3 Fisher_LDF.py
```

References

[1] https://sthalles.github.io/fisher-linear-discriminant/
[2]https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
[3]https://ermlab.com/en/blog/nlp/cifar-10-classification-using-keras-tutorial/

[4]https://en.wikipedia.org/wiki/Convolutional_neural_network