Pattern Recognition and Machine Learning Course Project: CIFAR-10

Maniyar Suyash *(B19EE095),* Deep Ashokbhai Patel *(B19CSE028)* and Mohan Chhabaria *(B19EE096)*

Abstract

The paper depicts various machine learning algorithms and their performance on the CIFAR-10 dataset. The dataset includes 60,000 sample images of objects belonging to 10 different classes. The used classifiers are Support Vector Machines, K nearest neighbour, Convolutional Neural Network, Multilayer Perceptron and Random forest.

Index Terms

KNN, SVM, CNN, MLP, RF, CIFAR-10, classification, recognition classifier, labels, train, test, accuracy

I. INTRODUCTION

THIS paper talks about the implementation of natural image classification using a variety of classifiers on the CIFAR-10 dataset. The dataset contains 50000 train samples with 5000 images of each class. The test dataset contains 10000 randomly selected images from each class.

Natural Image Classification is one of the hot topics in computer vision and pattern recognition research. In recent years scene understanding and object classification has received intense attention as it is beneficial for many practical applications such as autonomous driving, robot vision and content-based image retrieval. Image classification came into existence for decreasing the gap between the computer vision and human vision by training the computer with the data. The image classification is achieved by differentiating the image into the prescribed category based on the content of the vision.

II. IMPLEMENTATION

A. Data Visualisation

1) Here are some sample images with their class labels from the training set

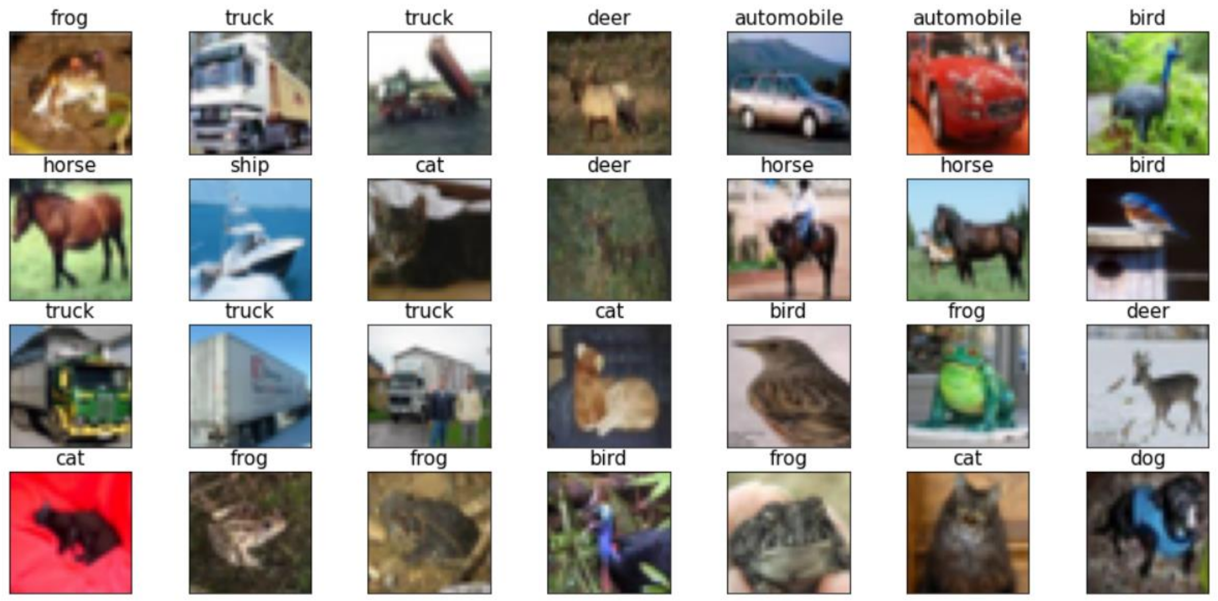


Fig. 1: CIFAR data set

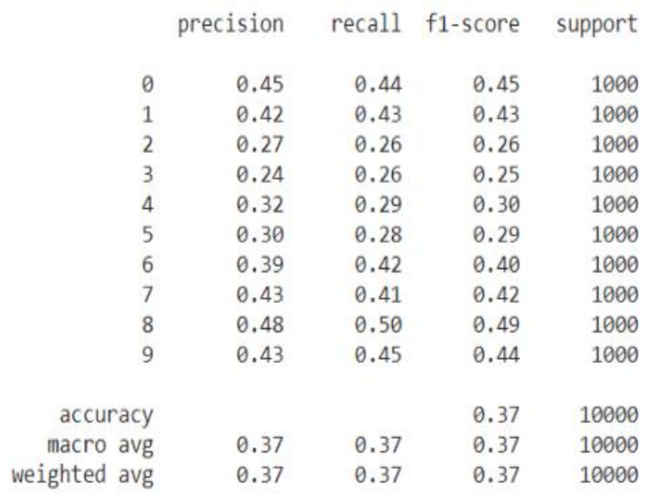
B. Support Vector Machines

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. The goal of the SVM algorithm is to create the best line or decision boundary (hyperplane) that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category for unknown data points. It uses support vectors that are data points closest to the margin or lie on the margins of the hyperplane.

To start with we used a standard scaler to scale the data so that the distribution is a normal distribution for all features. It subtracts the mean from all the feature values and then is divided by the variance. This helps us bring the data into a limited range
Now, as the number of attributes or pixel values is very high, that is 3072 so we used Linear Discriminant Analysis (LDA) which is a feature reduction technique. The number of components used is 9, it is selected because of the following equation:

No of components = min (L-1, no of features), where L = no of classes (here 10)

Next, we need to tune the hyperparameters for our SVC models, so we used grid search CV, which uses an iterative method to give us the best parameters for our model. The final or best hyperparameters came out to be [Kernel: ‘rbf’ and C: 1]. The trained classifier gave test accuracy of 37.37% and train accuracy of 54.6%. The classification report and the confusion matrix can be seen in figure 2 and figure 3 respectively. The confusion matrix shows that class ‘ship’ has been predicted most accurately compared to any other class.

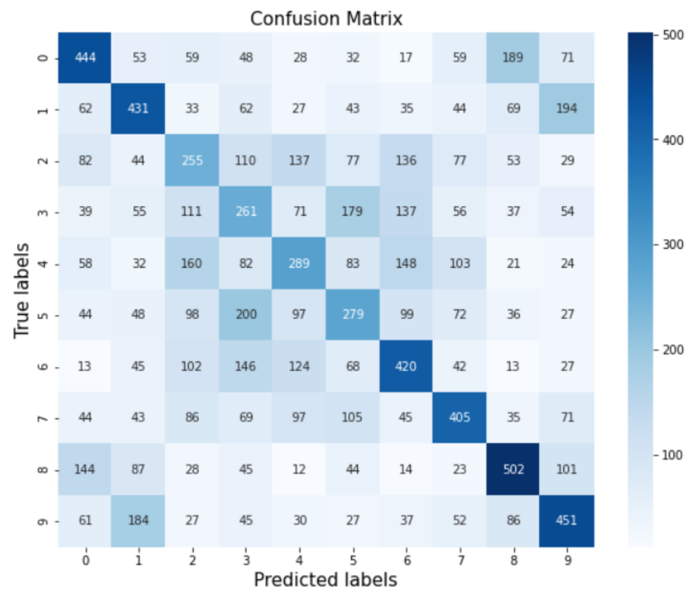


```
              precision    recall  f1-score   support

           0       0.45      0.44      0.45      1000
           1       0.42      0.43      0.43      1000
           2       0.27      0.26      0.26      1000
           3       0.24      0.26      0.25      1000
           4       0.32      0.29      0.30      1000
           5       0.30      0.28      0.29      1000
           6       0.39      0.42      0.40      1000
           7       0.43      0.41      0.42      1000
           8       0.48      0.50      0.49      1000
           9       0.43      0.45      0.44      1000

    accuracy                           0.37     10000
   macro avg       0.37      0.37      0.37     10000
weighted avg       0.37      0.37      0.37     10000
```

Fig. 2. Classification Report for SVM



	0	1	2	3	4	5	6	7	8	9
0	444	53	59	48	28	32	17	59	189	71
1	62	431	33	62	27	43	35	44	69	194
2	82	44	255	110	137	77	136	77	53	29
3	39	55	111	261	71	179	137	56	37	54
4	58	32	160	82	289	83	148	103	21	24
5	44	48	98	200	97	279	99	72	36	27
6	13	45	102	146	124	68	420	42	13	27
7	44	43	86	69	97	105	45	405	35	71
8	144	87	28	45	12	44	14	23	502	101
9	61	184	27	45	30	27	37	52	86	451

Fig. 3. Confusion Matrix for SVM

C. K- Nearest Neighbour

KNN is also a supervised machine learning technique that can be used for classification as well as regression problems. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).
To select the value of K that works best for our data was our first task, so for that we used the KNN algorithm several times with different values of K for a subset of our training set and chose the K that gives the maximum accuracy for our test data. The list of values used for this is [**5, 8, 10, 20, 30, 50, 100, 150**]. Fig. 4. shows the outcomes of this cross validation.

Also, as the previous method for selecting K is used on a smaller dataset that is 5000 samples, we need to check if it works for a larger dataset too. So, for this we used some values of k including 10 and performed cross validation by subsequently increasing the number of data points after each iteration. From the plot we can see that K = 10 works best for our data and thus, we selected k as 10. Next was to fit the model with these tuned hyperparameters. The test accuracy for this classifier came out to be 35.37% and the other results can be seen from the classification report and heatmap (Fig. 5.) that detail the predictions.

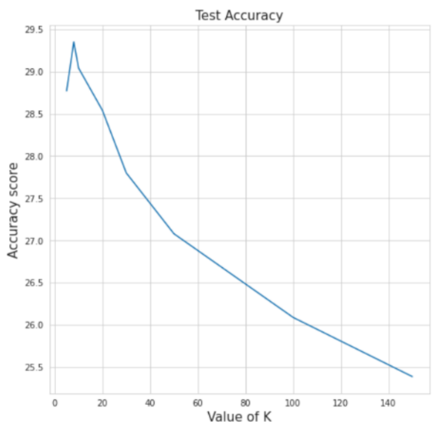
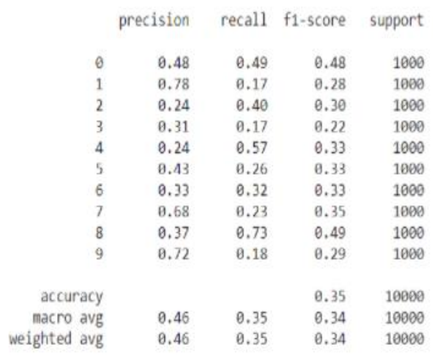


Fig. 4. Test Accuracy for various values of K


```
              precision    recall  f1-score   support

           0       0.48      0.49      0.48      1000
           1       0.78      0.17      0.28      1000
           2       0.24      0.40      0.30      1000
           3       0.31      0.17      0.22      1000
           4       0.24      0.57      0.33      1000
           5       0.43      0.26      0.33      1000
           6       0.33      0.32      0.33      1000
           7       0.68      0.23      0.35      1000
           8       0.37      0.73      0.49      1000
           9       0.72      0.18      0.29      1000

    accuracy                           0.35     10000
   macro avg       0.46      0.35      0.34     10000
weighted avg       0.46      0.35      0.34     10000
```

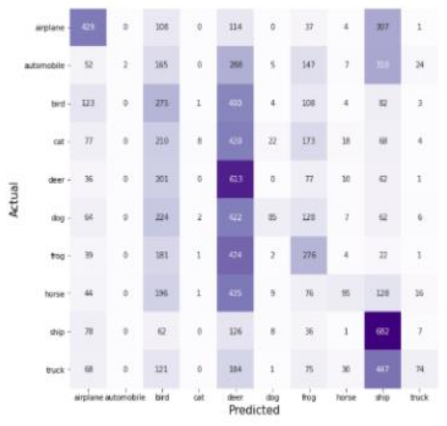
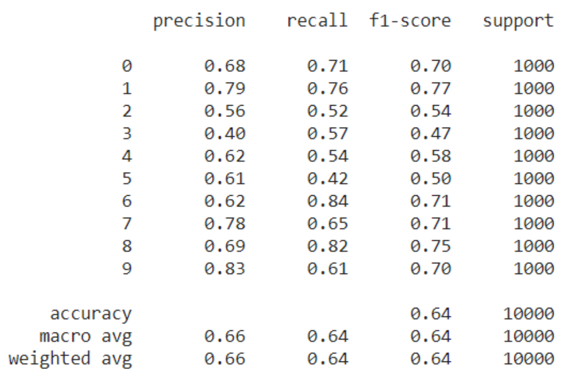


Fig. 5 Classification report (a) and Confusion Matrix for KNN

D. Convolutional Neural Network

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to the various objects in the image which are differentiable from each other. A CNN is built with many layers and we have used various layers in our sequential model, which takes these layers and adds them in a sequence to make a network.
We built 2 different networks one was for RGB images and the other for grayscale images.

RGB images: To begin with we divided the arrays (image arrays) by 255 as the pixel value ranges from 0-255, this enables us to get data in a common range of 0-1. The initial layers added to the model were Conv2D and Maxpool, Conv2D extracts the best features and reduces the dimensions and the Maxpool layers tries to reduce the computational power required to process the data. They were followed by a sequence of flatten and dense layers and finally our trained model predicted with an accuracy of nearly 56%. As the above process is not very efficient for prediction, to improve we added more layers which included a pair of Conv2D and Maxpool layers, followed by the same series of flatten and dense layers. This model provided better results and the accuracy came out to be 64%. The classification report and the performance is shown in Fig. 6 and Fig. 7 respectively.


```
              precision    recall  f1-score   support

           0       0.68      0.71      0.70      1000
           1       0.79      0.76      0.77      1000
           2       0.56      0.52      0.54      1000
           3       0.40      0.57      0.47      1000
           4       0.62      0.54      0.58      1000
           5       0.61      0.42      0.50      1000
           6       0.62      0.84      0.71      1000
           7       0.78      0.65      0.71      1000
           8       0.69      0.82      0.75      1000
           9       0.83      0.61      0.70      1000

    accuracy                           0.64     10000
   macro avg       0.66      0.64      0.64     10000
weighted avg       0.66      0.64      0.64     10000
```

Fig.6 Classification Report for CNN (RGB images)

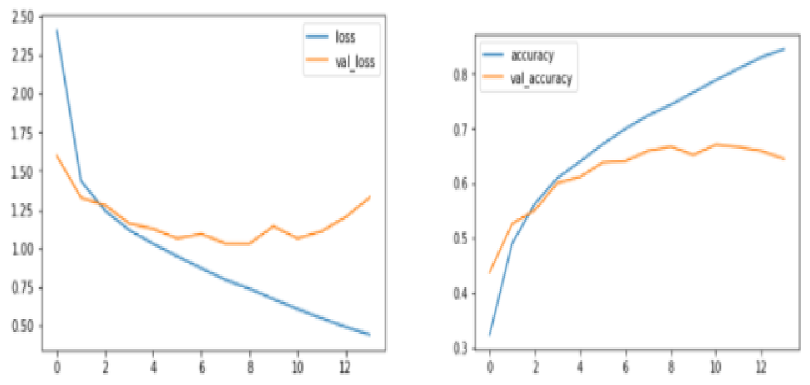
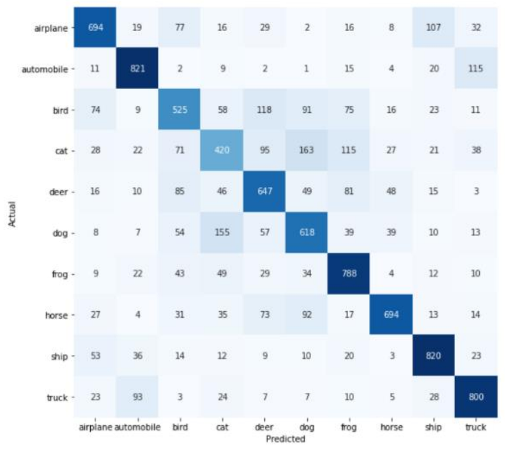


Fig. 7. Loss and accuracy with number of epochs for CNN

GrayScale Images: Firstly, we converted the set of coloured images into grayscale, and followed the same approach for preprocessing that is dividing by 255 so to normalise the data. The model here varied a bit, we used 6 Conv2d layers and 2 Maxpool layers because the grayscale images need more computations for classification. This was followed by a series of flatten and dense layers. The model was fed with the training data and 20 epochs with batch size 0f 32. The accuracy improved a bit and increased to 68%. The confusion matrix is shown in Fig. 8. Fig. 9. shows some predictions for the same.



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	694	19	77	16	29	2	16	8	107	32
automobile	11	821	2	9	2	1	15	4	20	115
bird	74	9	525	58	118	91	75	16	23	11
cat	28	22	71	420	95	163	115	27	21	38
deer	16	10	85	46	647	49	81	48	15	3
dog	8	7	54	155	57	618	39	39	10	13
frog	9	22	43	49	29	34	788	4	12	10
horse	27	4	31	35	73	92	17	694	13	14
ship	53	36	14	12	9	10	20	3	820	23
truck	23	93	3	24	7	7	10	5	28	800

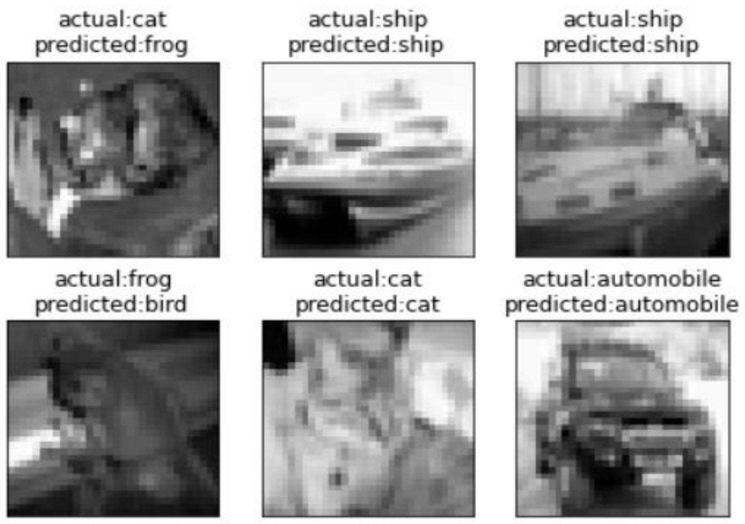


Fig. 8. Confusion matrix for CNN (Grayscale)

Fig. 9. Predicted and Actual Labels

E. Multilayer Perceptron

Multilayer Perceptron is a feed forward Artificial Neural Network. In the implementation we have used more than One hidden layer hence It is a Deep Artificial Neural Network. Table 2 represents the architecture of the classifier.

Layer	Number of neurons	Activation Function
Input Layer	3072	Relu
Hidden Layer 1	3000	Relu
Hidden Layer 2	1000	Relu
Output Layer	10	Sigmoid

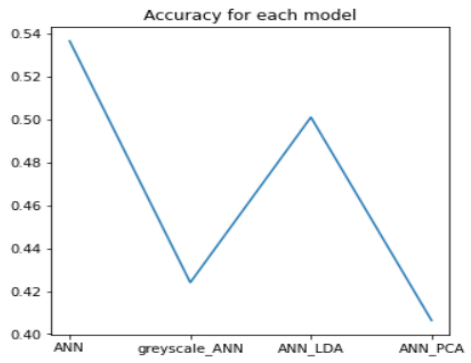
Table 2. Architecture of Implemented ANN

Layer	Number of neurons	Activation function
Input Layer	1024	Relu
Hidden layer1	1000	Relu
Hidden Layer2	300	Relu
Output Layer	10	Sigmoid

Table 3. Architecture of Implemented ANN (grayscale images)

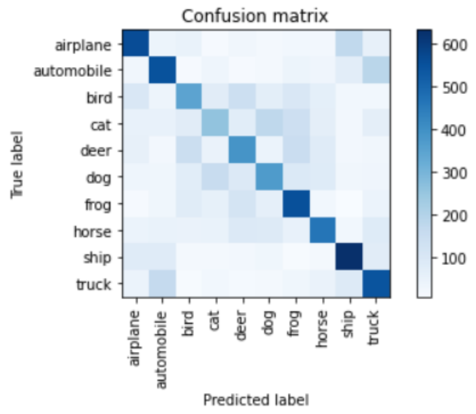
A dropout layer was used between Hidden layer 1 and 2 with dropout rate=0.1, optimizer used was Stochastic Gradient Descent. The model was trained with 50000 images and was tested on 10000 images and we got an accuracy of 0.5365. Later the data was converted to grayscale images and it was trained on a slightly different architecture because now there we only 32*32 =1024 input neurons. The architecture used to train the grayscale images is shown in Table 3.
The model was trained with 50000 grayscale images and was tested on 10000 grey images and we got an accuracy of 42.4%.
Linear Discriminant Analysis was performed on the CIFAR10 dataset and after the dimensionality reduction we arrived at 9 features. The accuracy on the test set was found to be 50.68%. After using the simple ANN architecture with feature selection techniques, we get the results shown in Fig. 10.

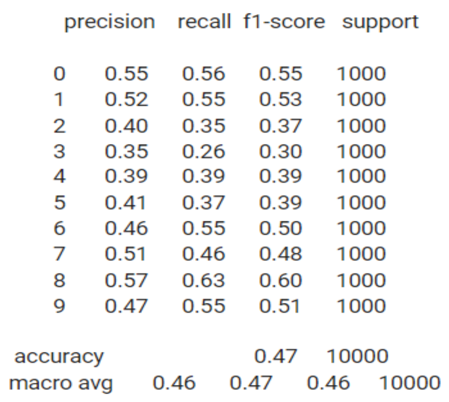
Fig.10.Accuracy Of ANN variants



F. Random Forest

Random Forest or Random Decision Forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the model of the classes or mean average prediction of the individual tree. They have a capacity for processing huge amounts of data with high training speeds. The important parameters of random forests are the depth of the tree and the number of trees. By increasing the depth of the tree increases performance, although this also increases the memory required to store the trees during experiments. In our experiments, our random forests showed the best classification performance in terms of accuracy and computational time with a maximum tree depth of 70, and number of trees set to 100. The test accuracy of the model we got was 47% and Fig. 11. shows the classification report and the confusion matrix for the same.





```
              precision    recall  f1-score   support

           0       0.55      0.56      0.55      1000
           1       0.52      0.55      0.53      1000
           2       0.40      0.35      0.37      1000
           3       0.35      0.26      0.30      1000
           4       0.39      0.39      0.39      1000
           5       0.41      0.37      0.39      1000
           6       0.46      0.55      0.50      1000
           7       0.51      0.46      0.48      1000
           8       0.57      0.63      0.60      1000
           9       0.47      0.55      0.51      1000

    accuracy                           0.47     10000
   macro avg       0.46      0.47      0.46     10000
```

Fig 11. Confusion matrix and Classification report for Random Forest

III. CONCLUSIONS

1. Comparing all the classification algorithms used we can state that CNN has performed the best, also using grayscale images benefits a bit, but this small improvement can be compensated by increase in computation cost. The plot in Fig. 12. compares the accuracy of every classification described in this paper.
2. Data Augmentation can be used to improve accuracy of CNN as it can increase the number of samples. It obviously requires us to change the architecture of the network and find an efficient one but still is an improvement. Here are some examples of samples generated by augmentation shown in Fig. 13.

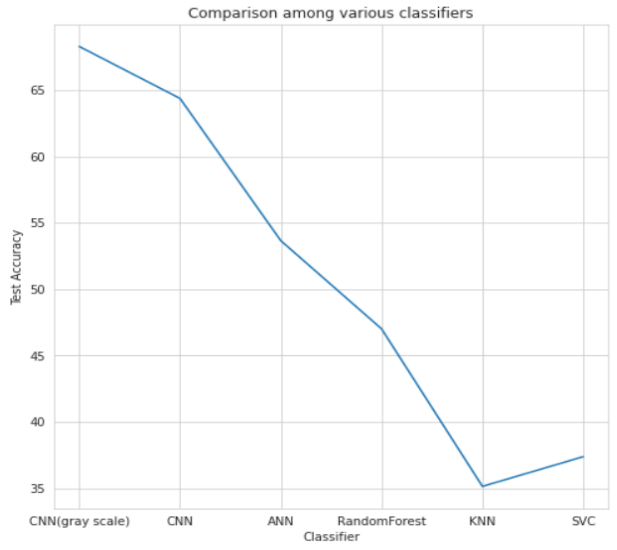


Fig. 12. Comparison among used classifiers

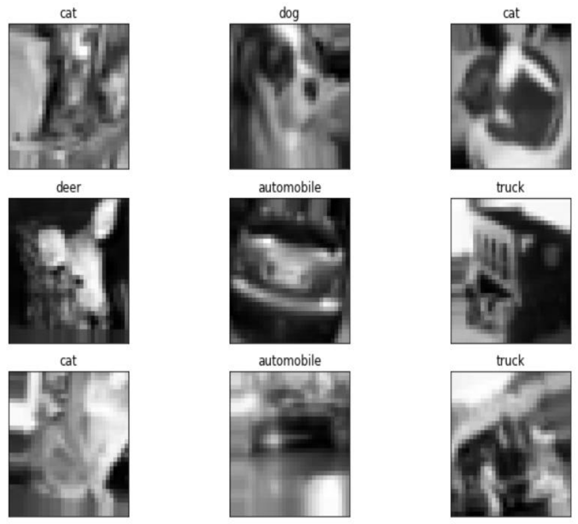


Fig. 13. Images created by Data Augmentation

3. CNN has proved to be better than ANN as it does not take dimensional information into account but CNN focuses on dimensional information and thus, gives better results

Contributions

Maniyar Suyash – Implemented MLP with analysis using feature selection techniques (LDA/PCA)

Deep Ashokbhai Patel – Implemented Random Forest

Mohan Chhabaria – Implemented SVM, KNN and CNN

References

1. Sklearn documentation
2. Keras documentation
3. Tensorflow documentation
4. https://github.com/moritzhambach/Image-Augmentation-in-Keras-CIFAR-10
5. https://www.tensorflow.org/tutorials/images/cnn