A Report on the experiments performed on CIFAR-10 dataset

Shrutimoy Das
2018701011

November 10, 2018

Abstract

In this project, classifiers have been trained using a variety of standard machine learning techniques : Multi-Layer Perceptron, Linear SVM, Logistic Regression and, Random Forests, with tuning of various hyper-parameters. These classifiers were then tested on the test sample: first on the raw test data and then on reduced dimensions of the test data. The loss plot, accuracy plot and accuracy scores for different models have been included in the report.

1 Introduction

The CIFAR-10 Object Recognition problem involves accurately classifying images of airplane, car, bird, cat, deer, dog, frog, horse, ship, and truck into their respective classes. Each image is represented as a 32 x 32 RGB image in png format. The dataset consists of 60,000 images, of which 50,000 images are for training and the remaining 10,000 are for testing the learned models.

2 Procedure

2.1 Preprocessing

2.1.1 Training set-Validation Set split

Over the 50,000 images that have been taken for training, a 80 : 20 split has been performed that results in 40,000 images for training and 10,000 images for validation.

2.1.2 Grayscaling

Each pixel of an image has been considered as a feature. Thus the dimensionality of each image is very high. In the case of CIFAR-10 data, where each 32 x 32 image is in RGB, there are 3072 features for each image. To reduce the number of features, each image has been converted to grayscale. This results in each image being converted to a 32 x 32 image, which contains 1024 pixels, i.e., 1024 features.

2.1.3 Dimensionality Reduction

Each 32 x 32 image consists of 1024 features. However many of the pixels(features) are common among the images. Thus dimensionality of the features of each image has been further reduced in a way so as to retain the maximum variance in information. This reduction has been done using two popular techniques :

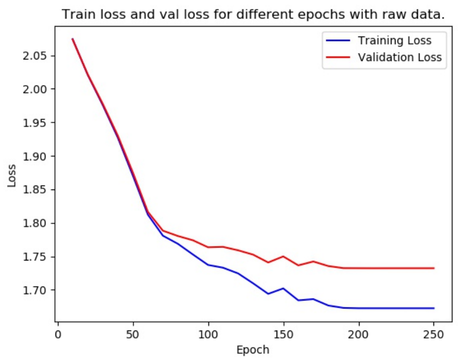
- Principal Components Analysis (PCA) : 250 principal components of each image have been taken for representation as 95% of the variance is maintained by these principal components.
- Linear Discriminant Analysis (LDA) : Since the given problem is a 10 class problem, 9 linear discriminants have been chosen for representation. The analysis has been performed on the 250 principal components obtained above. In this case, 83% of the variance has been retained.

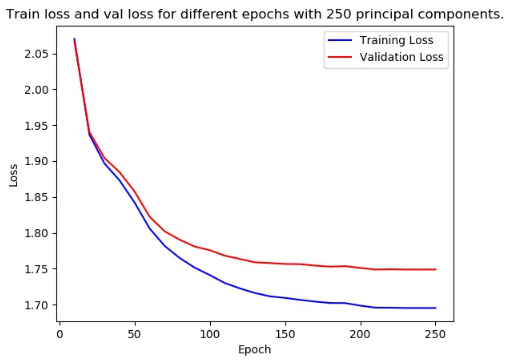
2.2 Training and Validation

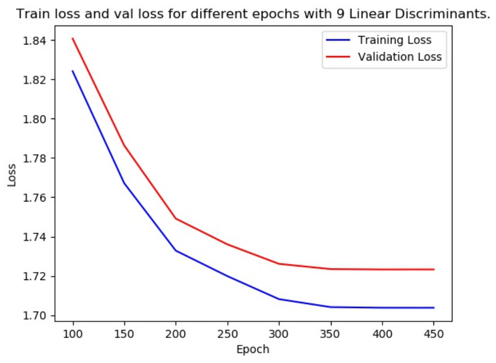
Four standard machine learning techniques have been used for training the classifiers for the given task : Multi-Layer Perceptron, Linear SVM, Logistic Regression and Random Forest. All these algorithms have been implemented by using the *scikit-learn* library of Python. Hyperparameter tuning has been performed for each model to prevent overfitting. The implementation of the various techniques has been shown below.

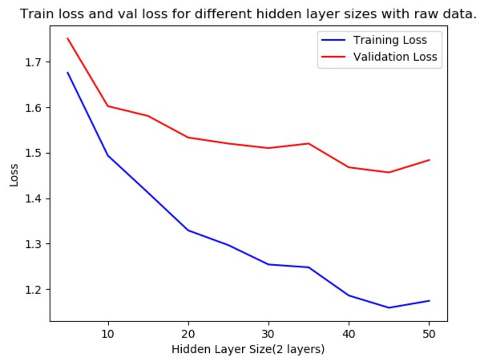
2.2.1 Multilayer Perceptron(MLP)

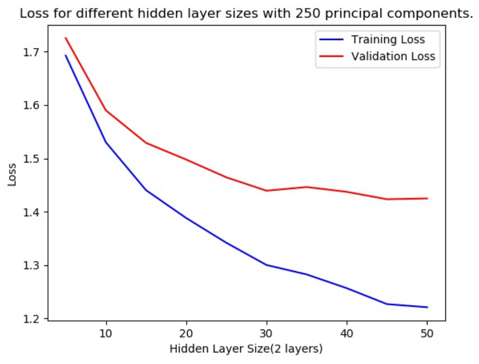
The Multi Layer Neural Network has been implemented with the parameters set as : *Learning Rate* = 0.01*(adaptive)*, *batch size* = 200, *activation* = *Logistic*, *solver* = *SGD*. A network with two hidden layers of equal sizes has been considered. Hyperparameter tuning has been performed on two parameters : *epoch* and *hidden layer size*. The loss obtained by varying the parameters have been shown in the plots below.(for each representation - raw data, PCA, LDA). It can be seen that as the values for the parameters increase, the training loss decrease and the validation loss increase beyond a certain point. This increase in validation loss is due to *overfitting* of the model to the training data. It can be controlled by taking those values of the parameters for which validation loss is minimum.











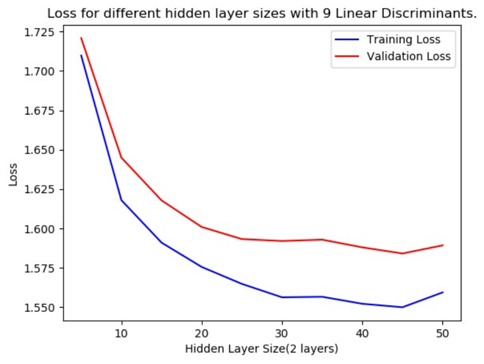


Figure 1: Loss during hyperparameter tuning: the first three plots show loss for different epochs; the last three plots show loss for different hidden layer sizes.

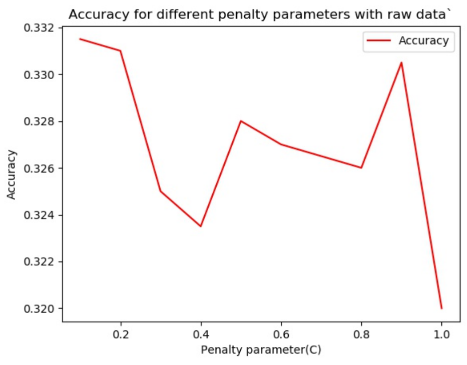
After hyperparameter tuning the following values for the parameters were obtained to reduce overfitting.

HYPERPARAMETER VALUES			
HYPERPARAMETER	RAW DATA	PCA	LDA
Epoch	210	210	400
Hidden Layer Size	(45,45)	(45,45)	(45,45)

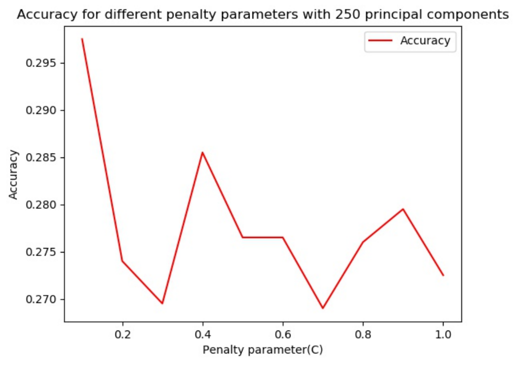
These hyperparameter values have been used for testing the classifier on the testing data. The accuracy obtained and F1 scores have been shown later.

2.2.2 Soft Margin Linear SVM

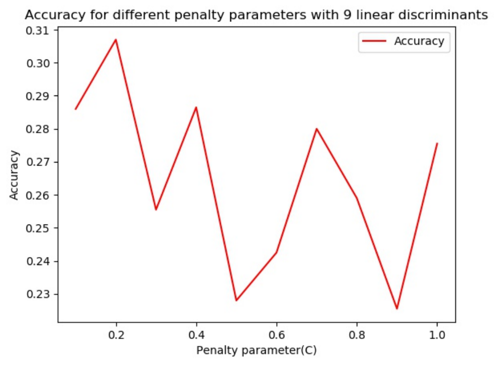
The Soft Margin Linear SVM has been implemented with the parameters set as : *Iterations* = 500, *dual* = True, *penalty* = *L2*, *Loss* =*Hinge*, *Intercept Scaling* = 1. Hyperparameter tuning has been performed on the penalty parameter,C. The accuracy plots obtained by varying the penalty parameter have been shown below. As the search for the penalty parameter for the SVM has been done in a small set $(0, 1]$, overfitting is not observed. Finally a hyperparameter value is chosen for which accuracy is maximum.



□□



□□



□□

Figure 2: Accuracy during hyperparameter tuning: the three plots show accuracy obtained for different values of C for raw data,PCA and LDA respectively

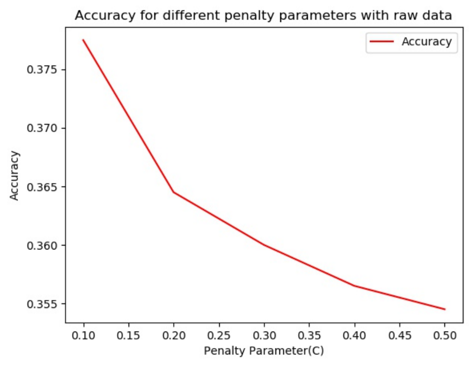
After hyperparameter tuning the following values for the penalty parameter were obtained to reduce overfitting.

HYPERPARAMETER VALUES			
HYPERPARAMETER	RAW DATA	PCA	LDA
Penalty Parameter,C	0.1	0.1	0.2

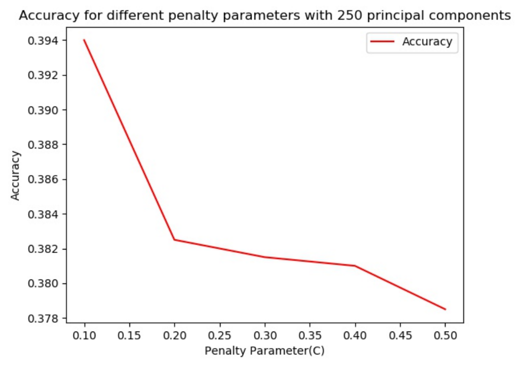
These hyperparameter values have been used for testing the classifier on the testing data. The accuracy obtained and F1 scores have been shown later.

2.2.3 Logistic Regression

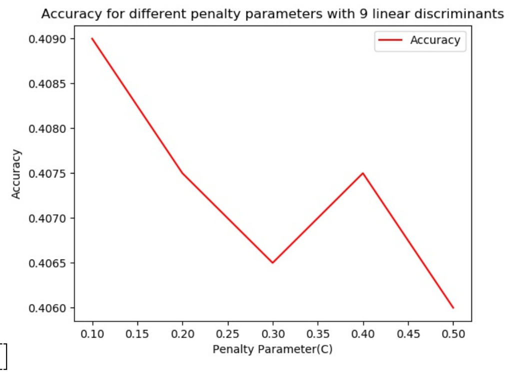
Logistic Regression has been implemented with the following parameters :*Penalty = L2*, *solver =Newton CG*,*maxiter* = 500. Hyperparameter tuning has been performed on the penalty parameter,C. The accuracy plots obtained by varying the penalty parameter have been shown below. As the search for the penalty parameter for Logistic Regression has been done in a small set $(0, 1]$, overfitting is not observed. Finally a hyperparameter value is chosen for which accuracy is maximum.



▯▯



▯▯



▯▯

Figure 3: Accuracy during hyperparameter tuning: the three plots show accuracy obtained for different values of C for raw data,PCA and LDA respectively

After hyperparameter tuning the following values for the penalty parameter were obtained to reduce overfitting.

HYPERPARAMETER VALUES			
HYPERPARAMETER	RAW DATA	PCA	LDA
Penalty Parameter,C	0.1	0.1	0.1

These hyperparameter values have been used for testing the classifier on the testing data. The accuracy obtained and F1 scores have been shown later.

2.2.4 Random Forest

Random Forest has been implemented with the default parameters set in *scikit-learn*. Hyperparameter tuning has been done on *n estimators*, which is the number of decision trees and *depth*, which is the depth of each tree(stump).The accuracy plots for different values of the hyperparameters are shown below.

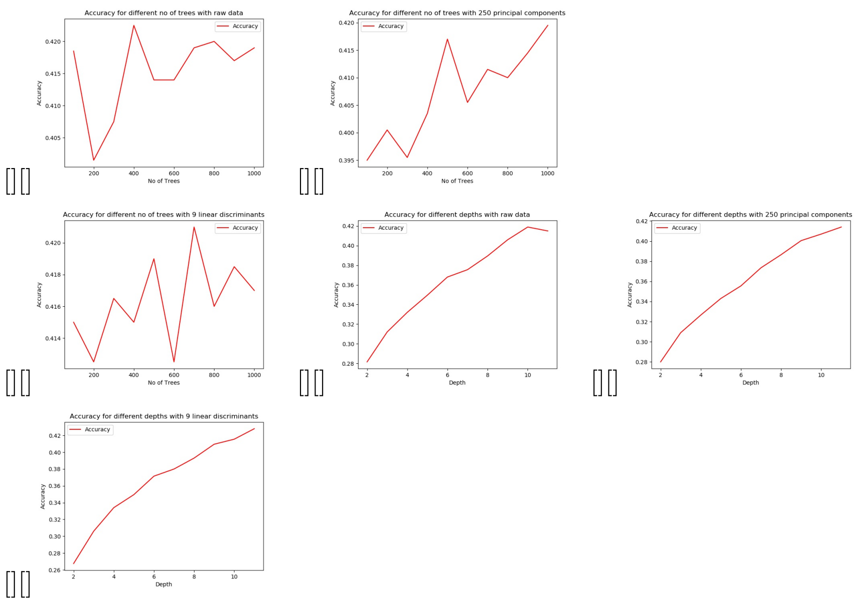


Figure 4: Accuracies during hyperparameter tuning: the first three plots show accuracies for different no of trees; the last three plots show accuracies for different depths of each tree.

After hyperparameter tuning the following values for the parameters were obtained.

HYPERPARAMETER VALUES			
HYPERPARAMETER	RAW DATA	PCA	LDA
No of Trees	400	1000	700
Depth	10	11	11

These hyperparameter values have been used for testing the classifier on the testing data. The accuracy obtained and F1 scores have been shown later.

2.3 Results

The classifiers were tested on the testing set of 10,000 samples. The accuracies and F1 Scores obtained(micro averaging) are shown below.

RESULTS			
CLASSIFIER	FEATURES	ACCURACY	F1-SCORE
MLP	raw data	0.4964	0.4964
MLP	pca	0.5049	0.5049
MLP	lda	0.4347	0.4347
Linear SVM	raw data	0.328	0.328
Linear SVM	pca	0.2915	0.2915
Linear SVM	lda	0.3015	0.3015
Logistic Regression	raw data	0.3775	0.3775
Logistic Regression	pca	0.394	0.394
Logistic Regression	lda	0.409	0.409
Random Forest	raw data	0.425	0.425
Random Forest	pca	0.412	0.412
Random Forest	lda	0.4245	0.4245