Project Report

Course: DS5220

Course Name: Supervised Machine Learning

Name: Dinesh Sai Pappuru

NUID: 002784485

Introduction:

In this project I have used 2 methods to train CIFAR-10 dataset.

1. SVM with kernel
2. DNN without Convolutional layer

SVM (Support Vector Machines):

SVM is a powerful supervised algorithm that works best on smaller datasets but on complex ones. Support Vector Machine, abbreviated as SVM can be used for both regression and classification tasks, but generally, they work best in classification problems. They were very famous around the time they were created, during the 1990s, and keep on being the go-to method for a high-performing algorithm with a little tuning.

It is a supervised machine learning problem where we try to find a hyperplane that best separates the two classes.

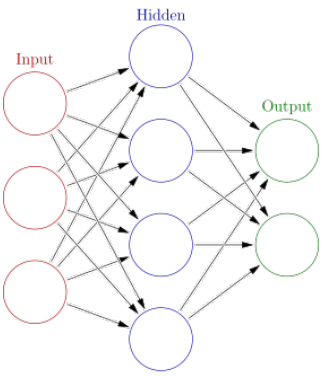
Linear SVM

When the data is perfectly linearly separable only then we can use Linear SVM. Perfectly linearly separable means that the data points can be classified into 2 classes by using a single straight line (if 2D).

DNN

Deep neural networks (DNN) is a class of machine learning algorithms similar to the artificial neural network and aims to mimic the information processing of the brain. DNN shave more than one hidden layer (l) situated between the input and output. Each layer contains a given number of units (neurons) that apply a certain functional transformation to the input.

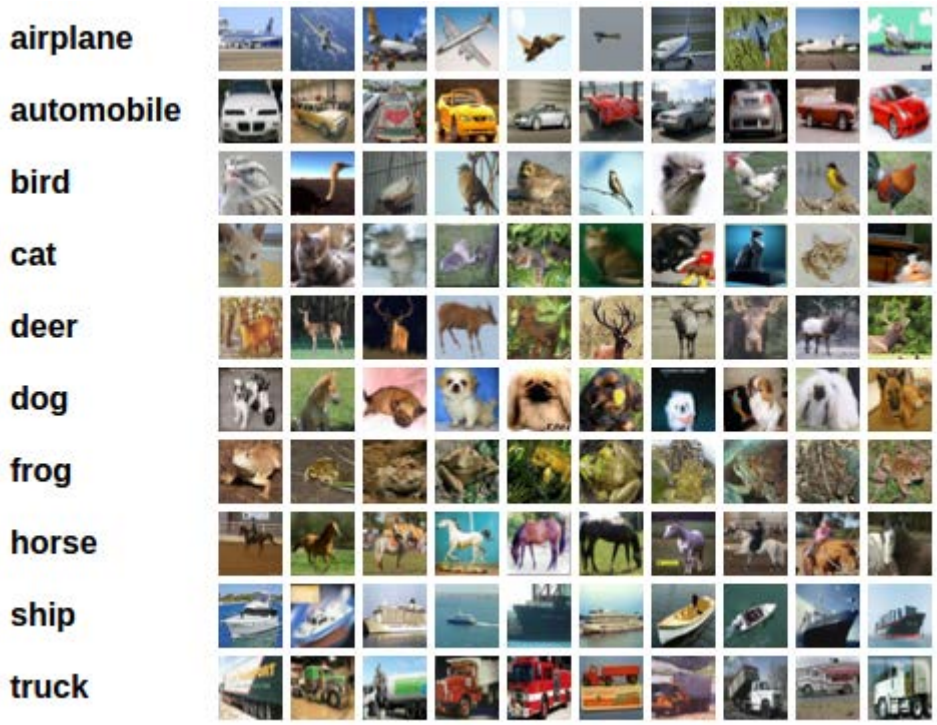
To fairly compare the various models in this work, only feed-forward layers were used. Despite the considerable interest, DNNs have gained in regression applications, few have dealt with the uncertainty in the prediction. This could be due to the complexity required to perform such an analysis.



Dataset:

CIFAR-10 is a dataset of 50,000 32x32 coloured training images and 10,000 test images, labelled with 10 categories.

Sample appearance of dataset:



The categories names which I used to label them are as follows:

[plane, car, bird, cat, deer, dog, frog, horse, ship, truck]

Sample images of datasets after loading into the code:

The CIFAR-10 training dataset was imported from the keras library and had a shape of (49000, 32, 32, 3).The 32 x 32 x 3 color images were then flattened into a vector, reshaping the training

dataset to have the shape (49000, 3072). Hence, it is important to note that only 49 thousand images were used when working on the SVM.

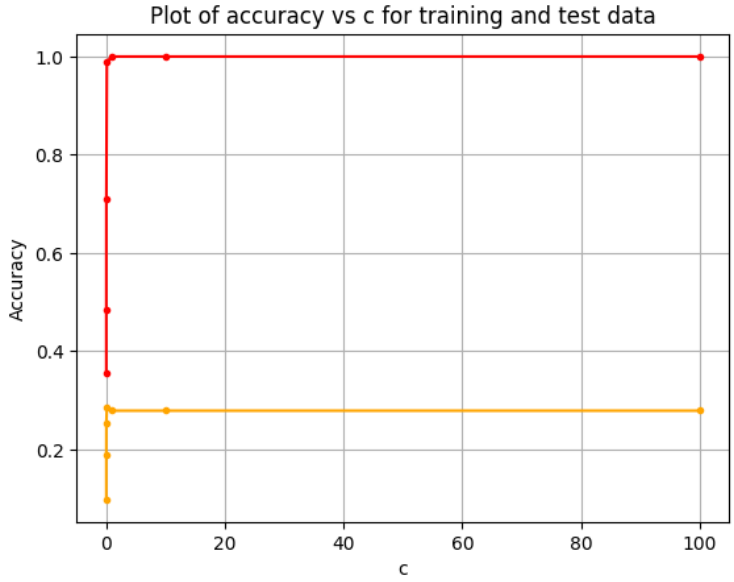
Implementation and analysis of Linear SVM:

While training SVM linear model, parameter 'C' plays a crucial role in determining the accuracy of the model.

The 'C' parameter controls how much you want to punish your model for each misclassified point for a given curve:

Large Values of C	Small Values of c
Large effect of noisy points.	Low effect of noisy points.
A plane with very few misclassifications will be given precedence.	Planes that separate the points well will be found, even if there are some misclassifications

While using multiple values of C(0.0001,0.001,0.01,0.1,1,10,100) for training the data I found the following curve as represented in this plot.

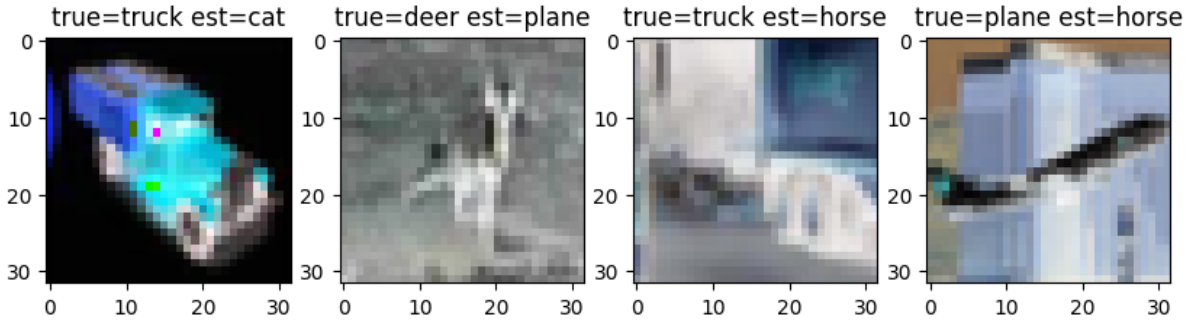


After checking this and deciding, the ideal 'C' parameter found was C=0.1. This is not the case for all the data. It depends on type of the dataset and its size of dataset.

Kernel	C	Training Accuracy (%)	Test Accuracy (%)
Linear	0.0001	35.5	9.8
	0.001	48.5	18.8
	0.01	70.9	25.3
	0.1	98.9	28.6
	1	1	27.9
	10	1	27.9
	100	1	27.9

Using the scikit learns inbuilt models to SVM linear kernel and using SVC (Support Vector Classifier) I found the accuracy of the trained model is around 28%.

With this amount of accuracy not all the prediction will be true. Following are some of the predictions which did not turn out to be correct.



Implementation and analysis of DNN without Convolutional layer:

Implementing the DNN was carried out using torch library. As we know that DNN consists of various layers, here in this I used 4 layers for the model. The Input layer consists of size 32x32x3 as that is the size of our images. Later in the hidden layers we used 100 neurons each and at last the output consists of 10 neurons to exactly classify 10 classes. This is implemented using nn.ModuleList() and nn.Linear() from pytorch library.

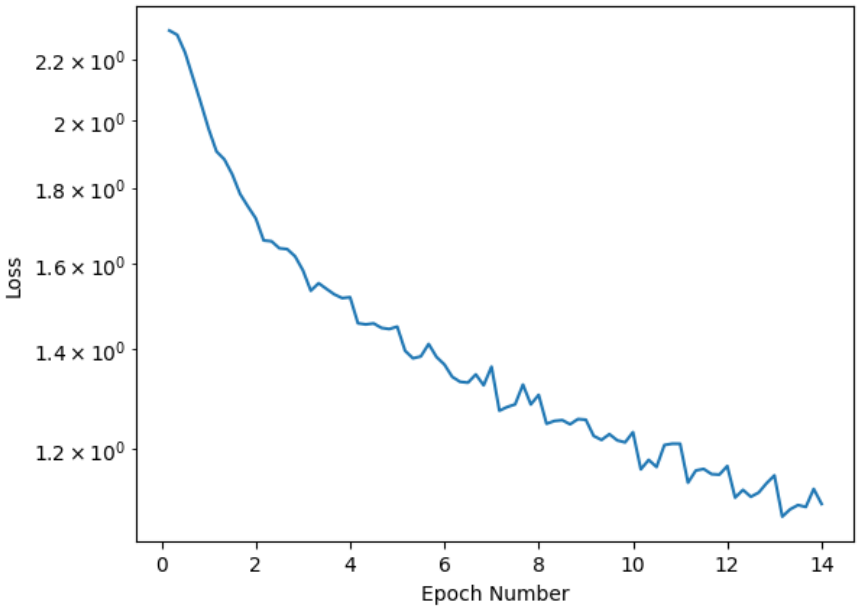
Further the optimizer used is this model is stochastic gradient descent.

Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable). It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data). Especially in high-dimensional optimization problems this reduces the very high computational burden, achieving faster iterations in exchange for a lower convergence rate.

And the loss criterion is set as cross entropy loss. This criterion computes the cross entropy loss between input logits and target.

It is useful when training a classification problem with *C* classes. If provided, the optional argument weight should be a 1D *Tensor* assigning weight to each of the classes.

By varying the learning rate and number of epochs performed on the model we can reduce our training loss, it is shown as follows:



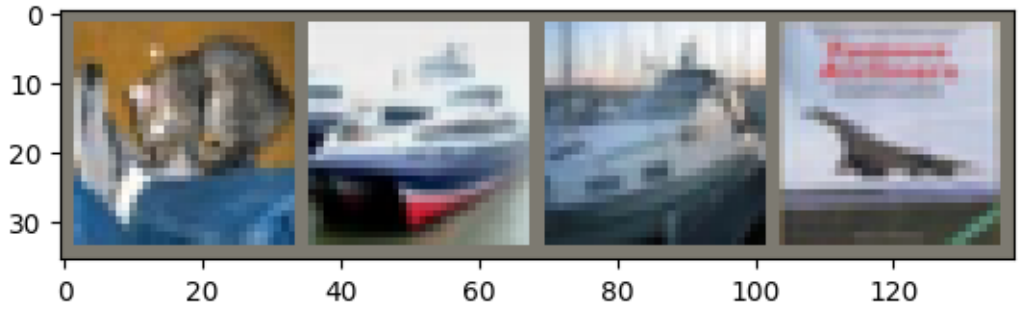
The number of epochs used here are as 14 and learning rate is set as the optimal 0.001.

By using different epochs I got the following accuracy:

Layers	Hidden Units	Epochs	Accuracy (%)
4	100	5	48
		10	52
		15	53

After implementing the model, the accuracy turn out to be around 52%.

Some of the test and predicted outputs are as follows:



Test Input: cat ship ship plane

Predicted: cat truck plane plane

Connclusion:

After comparing both the models we can clearly see that the model which was implemented using DNN (Deep Neural Netowrk) has a higher accuracy when compared with Linear Kernel SVM model. But the amount of time and resources used for DNN is comparatively higher than the DNN so it all depends on the users requirements where if they can afford time and resources they can use DNN or else they can proceed with SVM with Linear Kernel.