**Multi Layer Perceptron**

**Introduction**

The MultiLayer perceptron is a class of feedforward artificial neural network that has one or more layer of computational nodes. The perceptron computes a single output from multiple real-valued inputs by forming a linear combination according to its input weights and then possibly putting the output through some nonlinear activation function.

**Data**

We have used Cifar-10 dataset which consists of 60000 32\*32 colour images in 10 classes(mutually exclusive), of 6000 images per class. Test images : 10000 images ;Train images : 50000 images.

**Code**

We have 3 trials of building our MLP, each one improvised for better performance. We state our findings as below :

Trial 1 :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trial # | batch\_size | Epochs | Perceptron count | optimizer | Layer added? Specify | Stats Summary | Suggestion | Change in Stats |
| 1 | 150 | 25 | 1072 | Rmsprop() | No | Train loss: 1.4589578762817383  **Train accuracy: 0.50072**  Test loss: 1.619868154525757  Test accuracy: 0.4551 | Changing the Epoch(60) count increases the accuracy | Train loss: 1.305778570022583  **Train accuracy: 0.53574**  Test loss: 1.6725452089309691  Test accuracy: 0.4437 |
| 2 | 125 | 32 | 1024 | Rmsprop() | Yes  model.add(Dense(1024, activation='relu'))  model.add(Dropout(0.35))  model.add(Dense(1024, activation='relu')) | Train loss: 1.4482624677276612  **Train accuracy: 0.50138**  Test loss: 1.5178697803497314  Test accuracy: 0.4723 | Adding more layers and higher dropout percentage(helped against over fitting), and also epochs further tuned to 35 | Train loss: 1.3808243402862548  **Train accuracy: 0.52742**  Test loss: 1.46340176486969  Test accuracy: 0.4868 |

We modelled a total of 4 models based on Multilayer perceptron.

**Case-wise Study**

**Case 1** :

Our first case is when we took a higher perceptron count and we modelled on a single layer with no additional hidden layer. We plotted a graph of training accuracy Vs. validation accuracy to test if our model was overfitting. In this case, we did not see our data over fitting, but we intended to reach a better accuracy. So we adjusted the epoch count to get to conclude that training accuracy improvised from **50% to 53 %**

**Case 2** :

In the second case, we reduced the perceptron count and we added an additional layer, which caused overfitting as per the training accuracy and validation accuracy graph. Hence, we added a dropout of 35%, and we achieved a better accuracy by retuning the epoch count to 35 from 32. The overall training accuracy improvised from **50.13% to 52.74%**

**Final Conclusion** :

We understand that achieving higher accuracy in a dataset of images is difficult. Hence, from the base code of 48% by multiple iterations, and observing the graph, we have been able to impute the data to achieve higher accuracy in cases of comparision.

**Overall highest training accuracy : 53%**

Since our dataset contains images, it will be trained for a higher accuracy using Convolutional Neural Network.

In our section 2 of our assignment, we have used **CNN** to train our model and achieved an overall accuracy of **about 84 %**.

**Reference** :

<https://github.com/vrakesh/CIFAR-10-Classifier>

<https://github.com/aidiary/keras-examples/blob/master/mlp/cifar10.py>