Question 4 Image Classification (60 points) – Competition

In this exercise, your task is to create a Convolutional Neural Network (CNN) based image classifier for the CIFAR-10 dataset of images.

To solve the image classification problem using CIFAR-10 dataset, I went through online resources and followed the architecture from Stanford CS231n: Convolutional Neural Networks for Visual Recognition, lecture notes.

Network Architecture:

Three main types of layers are used to build ConvNet architectures: **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer**. These layers are stacked to form a full ConvNet architecture. The sequence is as follows:

[INPUT - CONV - RELU - POOL - FC]

For example : if the input size is 32x32x3, the following steps are performed to apply ConvNet.

- INPUT holds raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R, G, B.
- CONV layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This results in volume such as [32x32x3] as we decided to use 3 filters.
- RELU layer applies an elementwise activation function, such as the max (0,x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x3]).
- POOL layer performs a down sampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- FC (i.e. fully-connected) layer computes the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

85% testing accuracy is achieved with 150 epoch. Some of the other details of this architecture are summarized in the table.

Loss function	Optimization	Parameters
Cross-entropy	Stochastic Gradient Descent	Learning rate=0.001, weight_decay= 1E-5, momentum=0.9

To deal with overfitting problem, I have used dropout, L2 regularization, and batch normalization.

Dropout randomly drop units (along with their connections) from the neural network during training. The reduction in number of parameters in each step of training has effect of regularization.

L2 regularization allows to apply penalties on layer parameters during optimization. During gradient descent parameter update, the L2 regularization is applied using weight_decay parameter in SGD.

BatchNormalization normalizes the activation of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

Architecture: