Modern Assignment 1: Object recognition using Convolutional neural networks

Navya Battula

Background:

The basic ideology behind this project is to use a preprocessed image dataset like CIFAR 10 to build a convolutional neural network such that it efficiently performs object recognition tasks with better accuracies. A convolution neural network is a set of layers composed of convolution layers and pooling layers along with a flattening layer and a fully connected layer at the end. The key idea of this project is to use these layers and the related parameters appropriately to get satisfactory accuracies on training and test sets. My initial attempts towards this project include building the CNN model with random parameter values for number of filters, number of layers, stride, number of neurons in the fully connected layer, etc. However these parameters were later fine tuned for the purpose of fitting the model accurately (without overfitting).

Dataset:

As described above the dataset used in this project is CIFAR 10 which consists of 60000 color images of size 32x32 size divided into 10 categories which include planes, cars, cars, dogs, ships, etc. The dataset will be divided into 50000 training images and 10000 test images. We import the dataset from keras.datasets and normalize its pixel values.

Model definition:

The model used in this project makes use of 6 convolution layers with filters 48, 96 and 192 with filter size 3x3 and 3 max pooling layers with filter size 2x2 and stride 2x2 along with padding. Two dense layers were used as fully connected layers with 512 and 256 neurons each with ReLU activation function and an output layer with 10 neurons with softmax activation function. Initially the model has no optimizations added to it and only consists of the regular layers as described above. The training accuracy turned out to be 99.34% and the test accuracy turned out to be 74.35%. The detailed architecture of the initial model without optimizations is shown below.

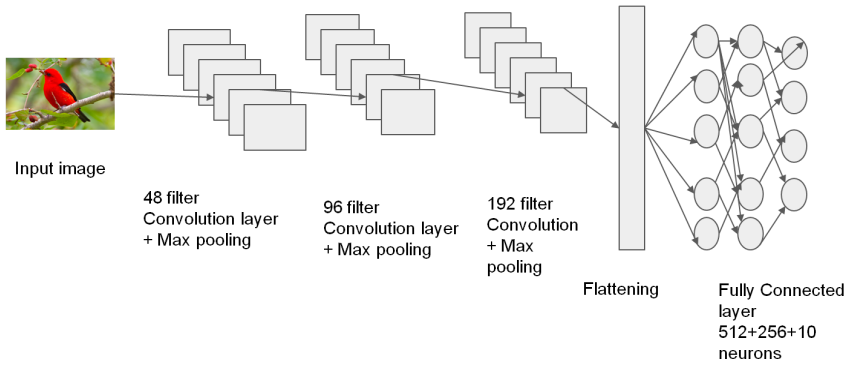


Figure 1: Rough architecture diagram of the model used in this project.

Now in order to improve the test accuracy, the following changes have been included into the model:

1. Introduce dropout in between layers to reduce overfitting in the networks.
2. Using Batch Normalization inorder to stabilize the process of learning.

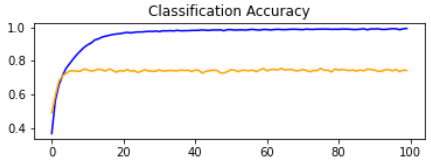
After adding these regularizations to the model, the model performance saw some improvement in terms of training accuracy. The accuracy on the test dataset turned out to be 83.78%.

Compiling the model:

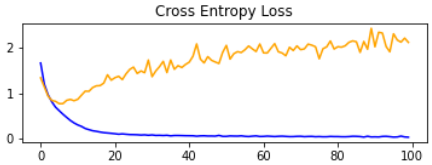
The model makes use of Adam optimizer instead of gradient descent because it is relatively faster. In the initial attempts without regularizations, the batch size was taken to be 64. When the regularizations were applied, the batch size was changed to 128. The model ran for 100 epochs in both cases and took 10-15 minutes to train (using the metal GPU in apple). The learning rate is set to be 0.001.

Model evaluation:

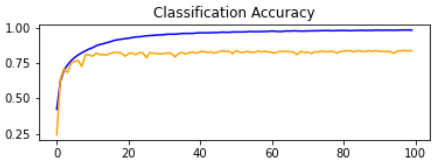
If we observe the graph1 and graph2 below, it will be apparent that the model before regularization has a lot of overfitting as the training and testing accuracies and losses seem to have large differences in them. However after the regularizations, if we observe graph3 and graph4, we could see that the differences in train and test accuracies and losses reduced a lot. This gives us an insight on how without regularizations like dropout and batch normalization, models tend to overfit. Other regularizations like data augmentation and weight decay can also be applied to the model however were not applied in this project due to time constraint.



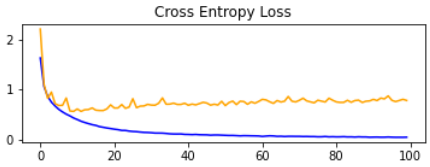
Graph1: Accuracy curve of basic model without any regularizations. The blue color represents training accuracy and the red line represents validation accuracy.



Graph2: Loss curve for basic model without optimizations. The blue color represents training accuracy and the red line represents validation accuracy.



Graph3: Accuracy curve for model after optimizations. The blue color represents training accuracy and the red line represents validation accuracy.



Graph4: Loss curve for model after optimizations. The blue color represents training loss and the red line represents validation loss.

Environment details:

This project was implemented in jupyter notebook in anaconda environment. The model makes use of high level machine learning frameworks and libraries like tensorflow and keras. The tensorflow version used in this project is 2.8.0 and that of keras is 2.8.0. Other libraries used in this project are matplotlib for the purpose of visualization (version - 3.5.1) and numpy for pixel based operations (version - 1.21.2). Hardware support for this project includes metal GPU by Apple.

Efforts towards the project:

The basic contribution towards this project is finding the appropriate values of parameters such as number of filters, number of layers, stride size, filter size, dropout values, learning rate, number of neurons in dense layers, batch size, etc. This was achieved by brute force approach by randomly putting together various values. Another major contribution towards this work is applying regularizations for increasing the model performance on test set. The basic objective of the project was to achieve better and accurate object recognition out of the model in work. Therefore an important effort would be to mix and match various values for parameters and various regularization techniques until a best accuracy is seen. Upon a little bit of research, it was found a few Vit-H/14 and CaiT-M-36 U 224 give excellent accuracies with the CIFAR 10 dataset. But we limit the scope of this project only to CNN. Also there are various techniques available in the wild like data augmentation, weight decay, etc which could be added along with the present regularization techniques in our model to boost the model performance. But due to time constraints we limit the work in this project only to these two regularizations.