AISHWARYA HASTAK – FALL 2023

Lab 3 – Training AlexNET model on CIFAR10 Dataset

UNDERSTANDING OF THE PROBLEM

The problem is a multi-class image classification problem, where we have to identify the object in the images from the CIFAR dataset from a set of 1000 possible categories. However, we are using the CIFAR10 dataset, which has only 10 classes.

To do this job, we will build a deep neural net model following the AlexNET architecture, with 5 convolution networks,pooling layers, 3 fully connected layers and dropout layers.

THOUGHT BEHIND THE CODE

There is four main parts to the code:

Loading the Dataset: The images are of size 32*32 pixels. We transform all images to a common size of 227*227 to be able to implement the AlexNET architecture. We load the dataset available in the PyTorch Datasets class and define the training and testing dataloader.

Defining the model: We define a class for a custom model AlexNet that inherits nn.Module, whichallows for automatic parameter initialization and tracking and gradient computation. We define the layers in the initialization method and define the forward pass. We will modify the last Fully connected layer to have 10 output classes, instead of 1000 as in the original architecture, since we are using the CIFAR10 dataset.

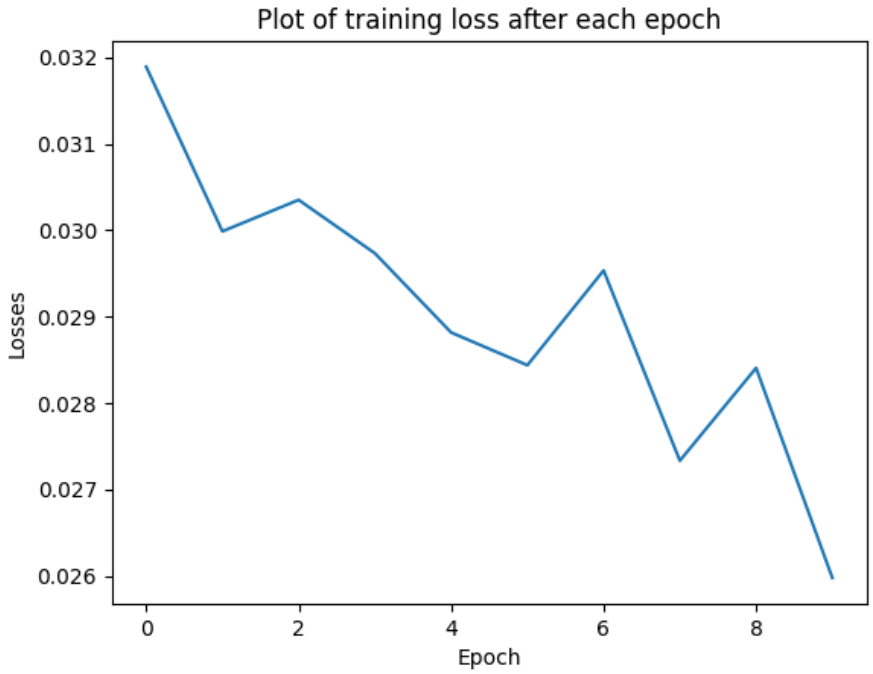
Training: We set the model in training mode using model.train(). We iterate over each batch in the training dataloader for n number of epochs, setting the gradients to zero in the beginning of each batch processing, getting the model outputs, calculating loss and performing a backward pass to calculate the gradients and then finally updating the model parameters. We calculate and records the loss and accuracy after each epoch.

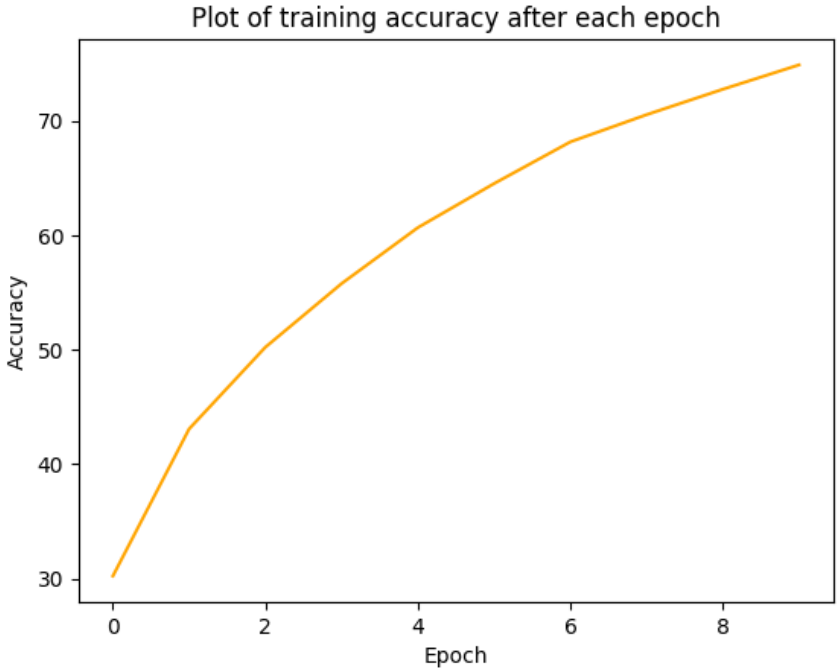
Testing: We set the model in evaluation mode using model.eval() [*which is also a functionality provided by the nn.Module class*]. We iterate over the testing dataloader and get the testing loss and accuracy for each epoch.

OUTPUTS

INITIAL MODEL

The initial model with the standard AlexNET architecture and a batch size of 128 and 10 epochs produced an accuracy of **70%**.





I have added dropout layers with 0.5 probability after each of the first 2 fully connected layers, which means each neuron has a 50% probability of being dropped. Dropout layers help to prevent overfitting of the data by randomly removing neurons, which trains the model to not be overly reliant on only some neurons making it more robust and generalizable.

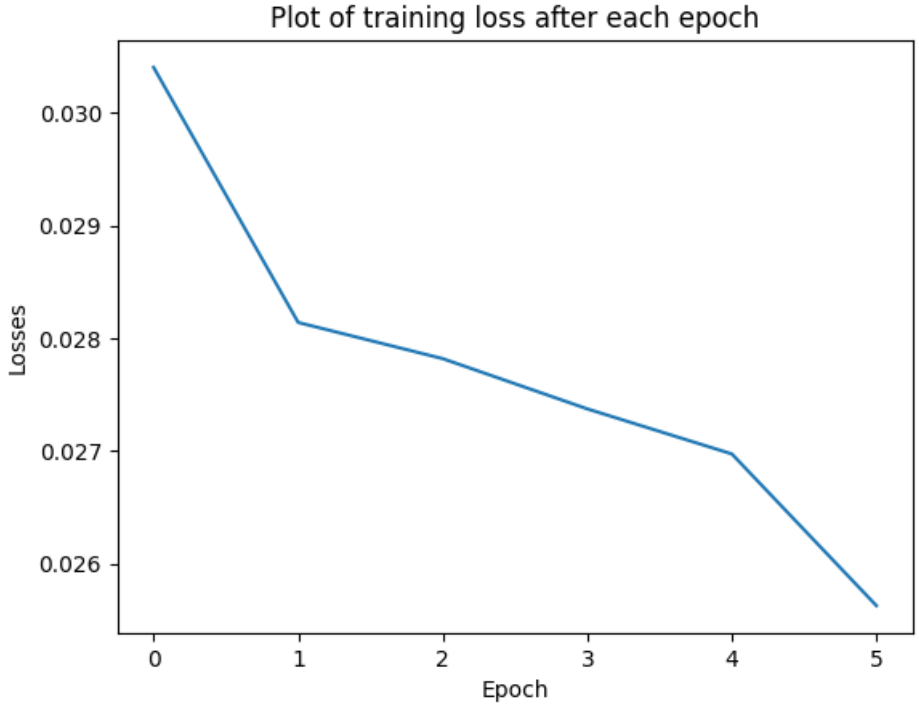
The original paper has used Local Response Normalization (LRN) and Batch Normalization technique, I have also added batch normalization after the first two convolution layers.

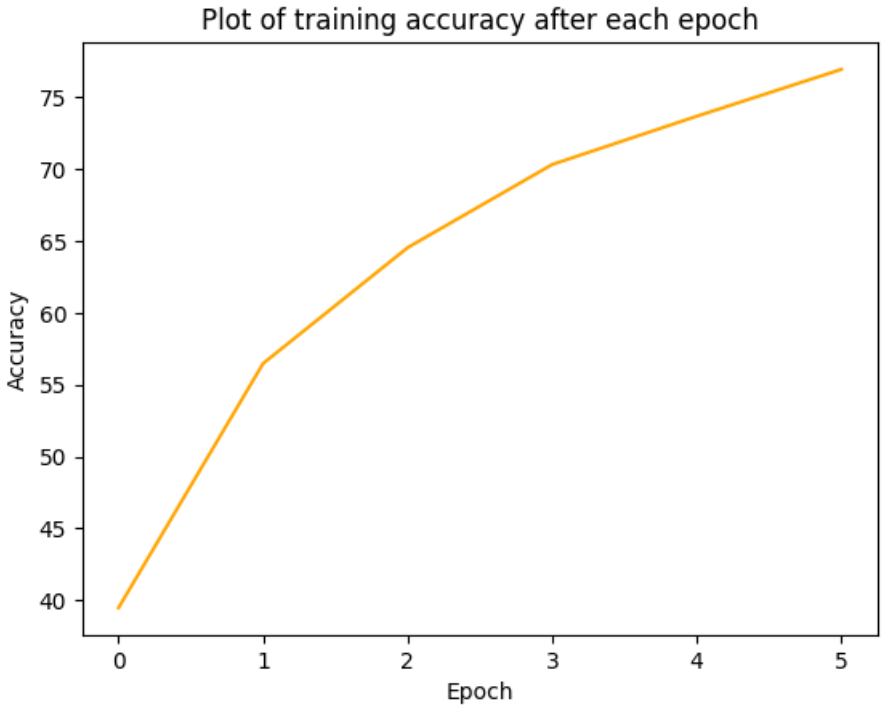
I have used the He initialization technique, which initializes the weights from a normal distribution based on the number of inputs to that layers and is typically used for convolution and fully connected layers that use ReLU activation.

While training the paper had suggested a batch size of 64 or 128, I have used a batch size of 128.

USING DIFFERENT NUMBER OF CONVOLUTION LAYERS

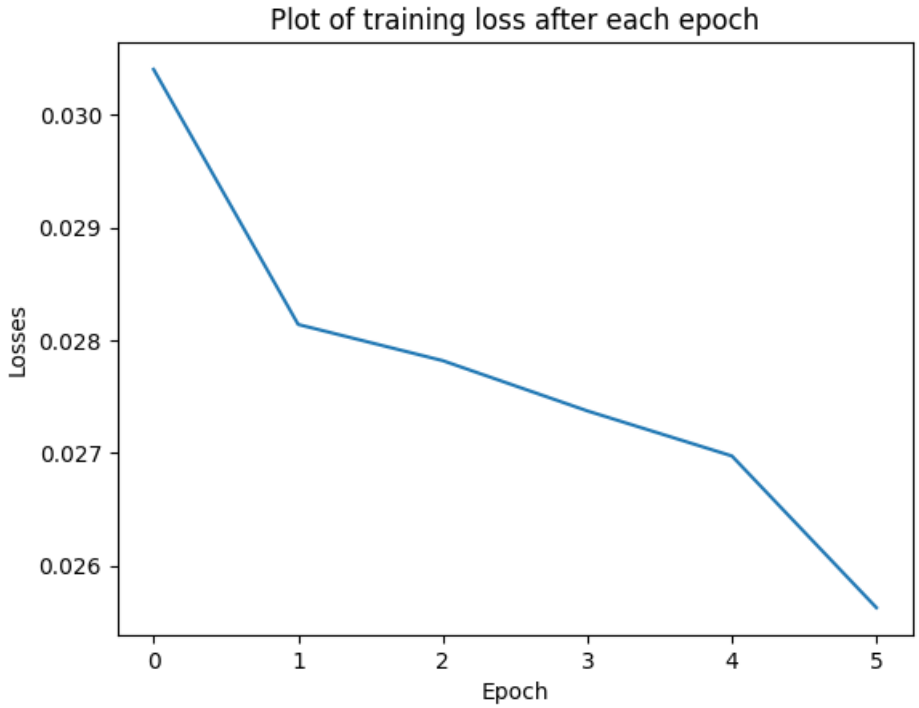
- **Using 4 Convolution layers**: Removed the fifth Conv2d layer and changed the output channels of the fourth Conv2d layer from 384 to 256.

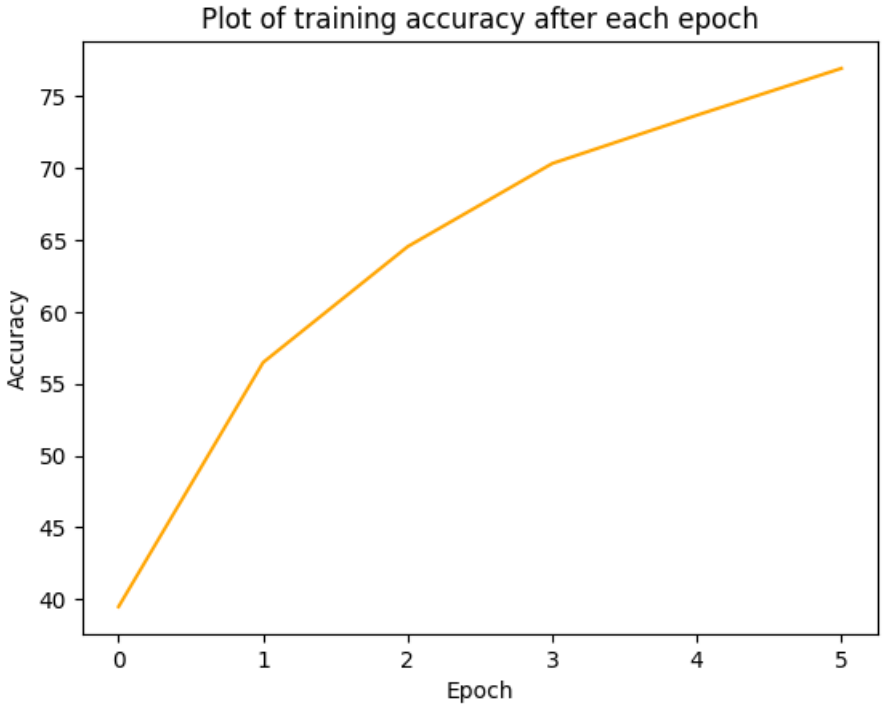




This model performed much better and produced an accuracy of 73%.

- **Using 3 Convolution layers**: Removed the fourth Conv2D layer which leaves only the first 3 convolution layers.



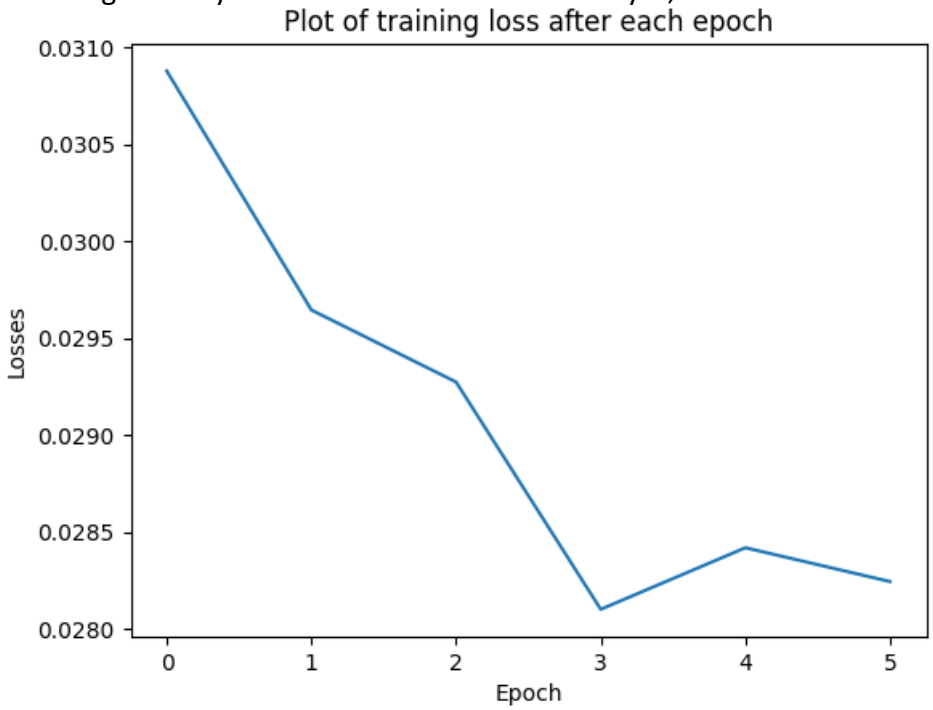


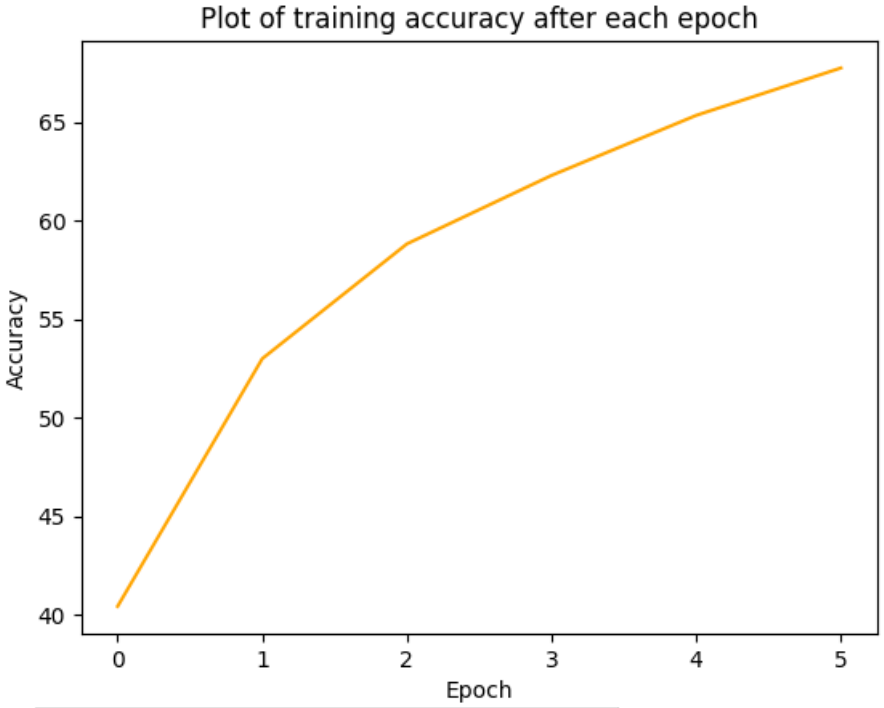
This model produced an accuracy of 67%.

Decreasing the depth of the model by removing some convolution layers, has increased the accuracy of the model. This could be because of a lower model complexity which works better when we have a smaller dataset. The CIFAR10 dataset is a relatively small dataset and would work well with a smaller model.

USING DIFFERENT NUMBER OF FULLY CONNECTED LAYERS

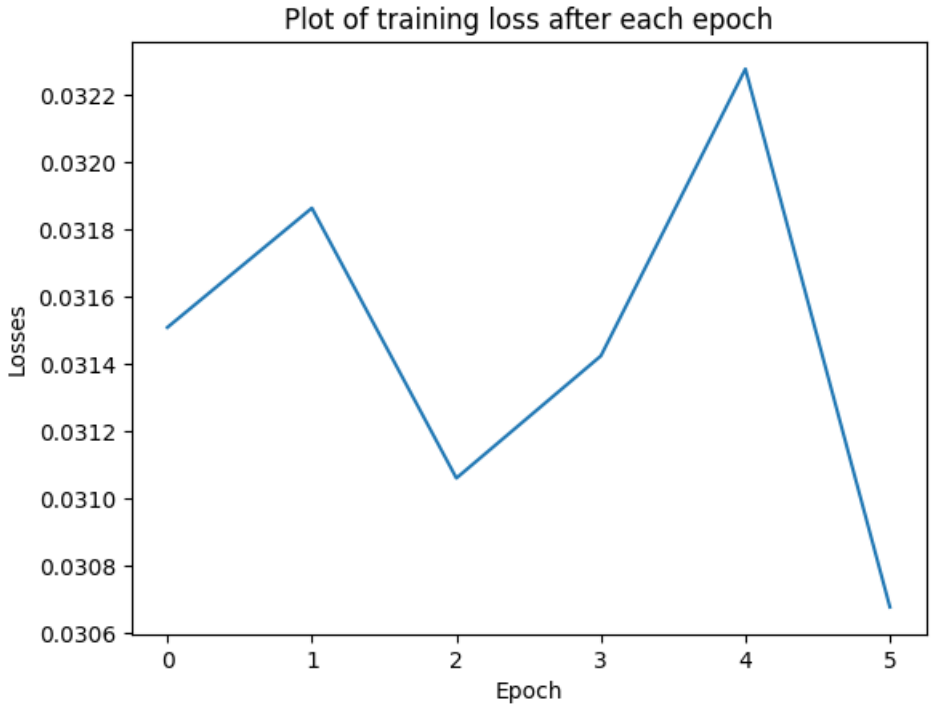
- Using 2 FC Layers: Removed the second FC layer, which leaves us with only 2 FC layer

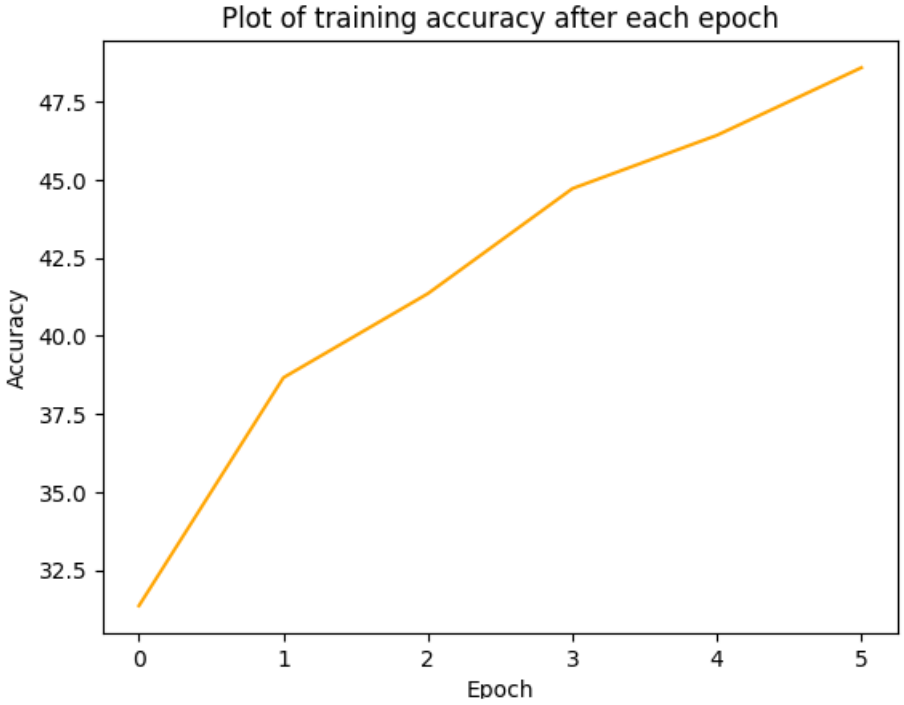




This modified model performed with an accuracy of 61.73%.

- Using 1 FC Layers: Using only 1 FC layer which is the last layer, and we use the sigmoid activation for classification of 10 classes.





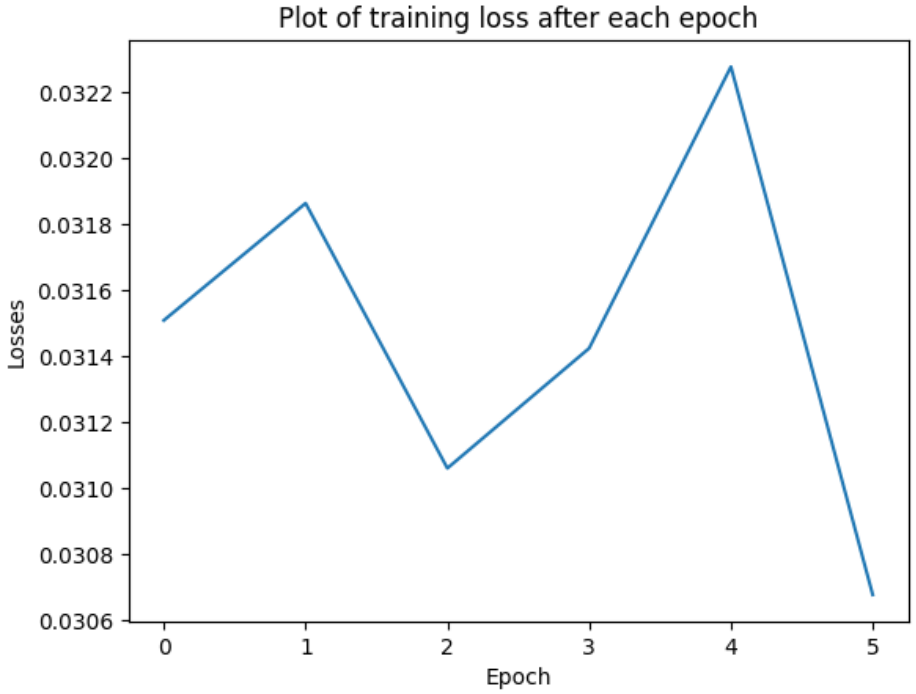
This model performed with an accuracy of 56.59%

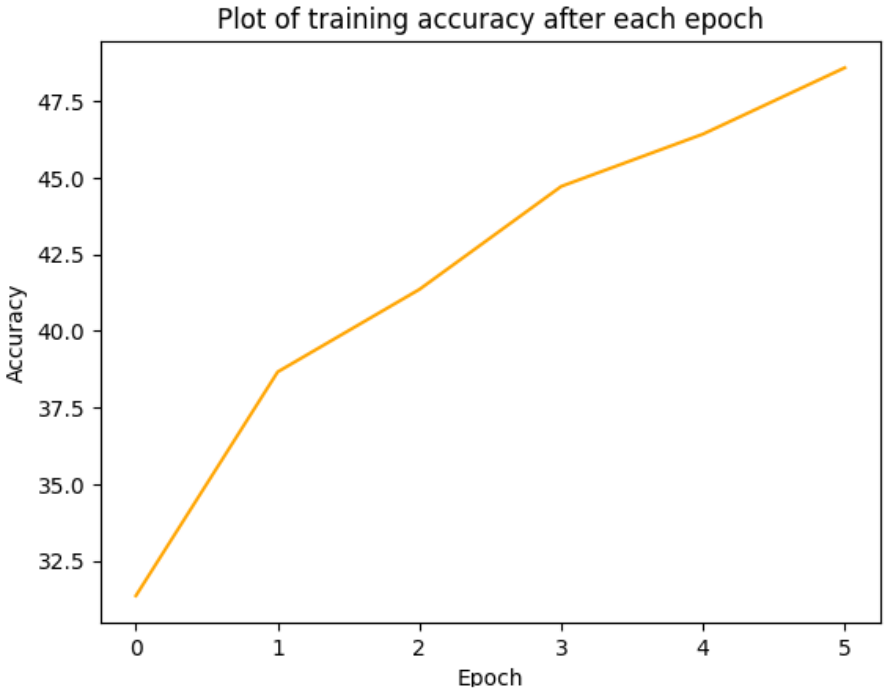
Decreasing the depth of the model by removing some Fully Connected layers, has decreased the accuracy of the model. This could be because lesser Fully Connected layers mean lesser neurons to process and make sense of the data which could lead to underfitting.

I had also tried to fit the model on 4 Fully Connected layers where it performed worse, which could be because of more complex model which might not be needed since this dataset is relatively small.

USING ReLU ACTIVATION FUNCTION

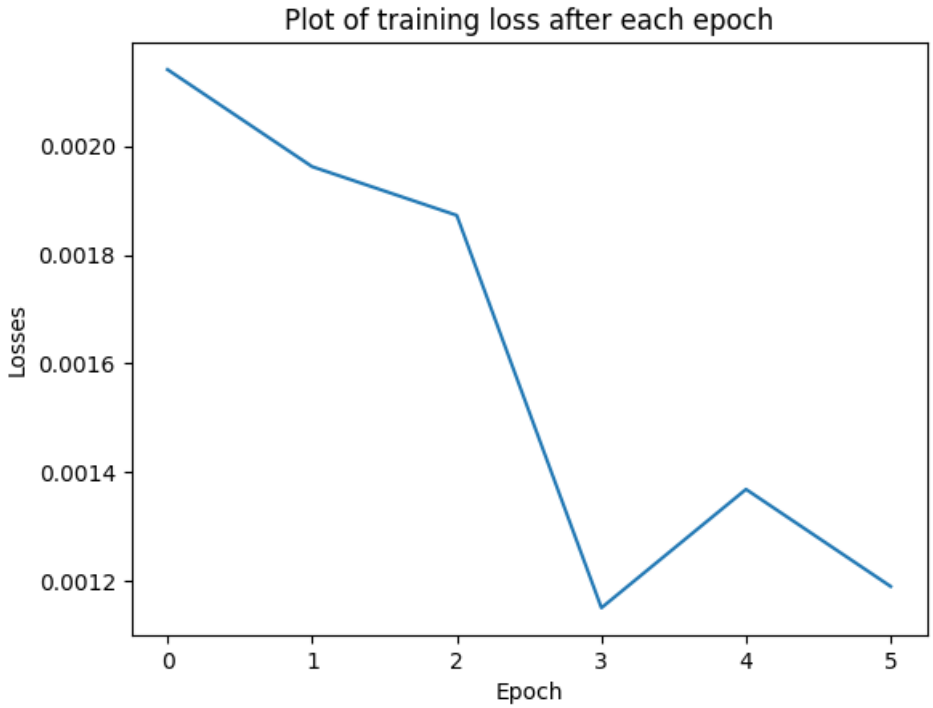
Used Leaky ReLU instead of ReLU for the convolution layers and the Fully Connected layers and got an accuracy of **62.82%** which is a slight degrade as compared to the initial model.

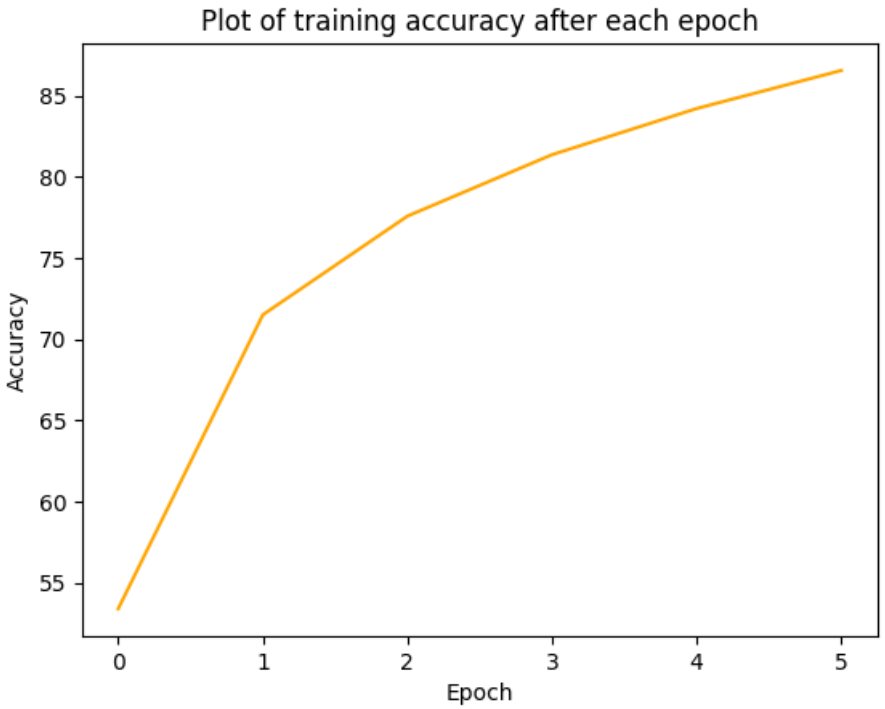




USING DIFFERENT LOSS FUNCTIONS

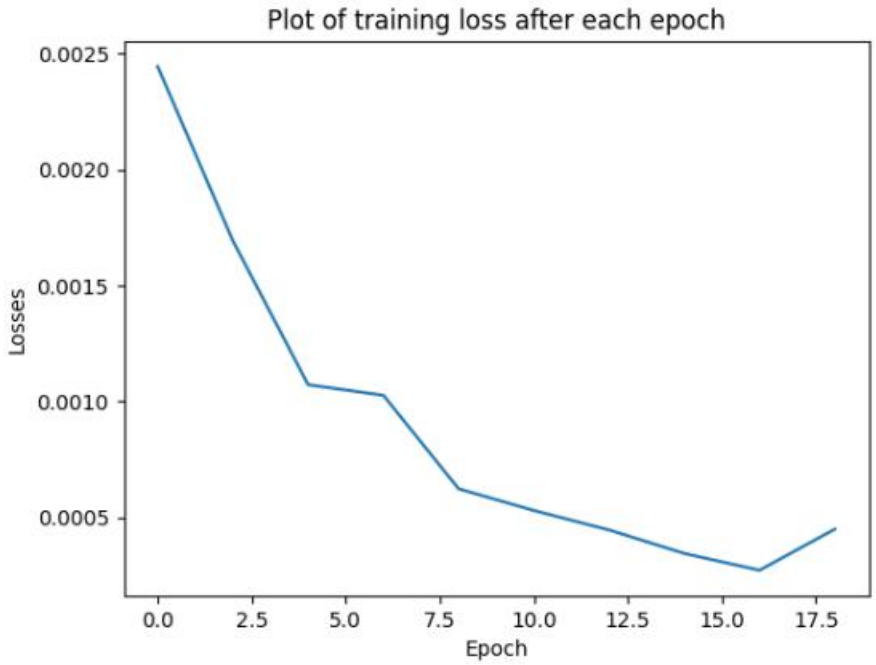
I used Binary Cross Entropy (BCE) loss instead of Cross Entropy Loss and got an accuracy of 80%, which is the best performing model until now.

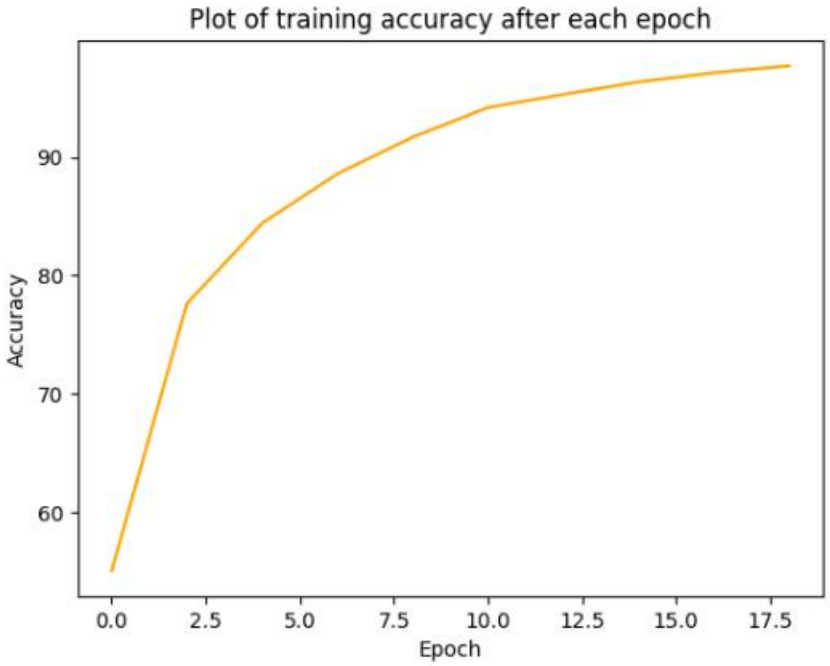




FINDINGS

The best performing model achieved an accuracy of **81.36%,** with three Conv2d layers, ReLU activation function and max pooling, and three fully connected layers (as in the original AlexNet model architecture). I used BCE loss function to calculate training loss.






```
Final Testing Accuracy :  81.369995
```

I used the Adam optimizing algorithm to update the weights. I tried various combinations of the hyperparameters and found that the following work best, learning rate of 0.0001, weight decay of 0.0005, momentum of 0.95 and a batch size of 128.

The better performance by decreasing the number of convolution layers could be attributed to decreasing the model complexity. Since the CIFAR10 is a smaller dataset, it can work better with a smaller model.

We saw that decreasing the number of fully connected layers moderately worsened the model performance. This could be because the model couldn't properly make sense of the data which might be leading to underfitting of the data. Increasing the number of fully connected layers further worsened the model performance which might be because of increased complexity.