Ethan C & Sahith B
CSCI 4963: ML for Autonomous Systems
November 4, 2022

Homework 5

Question 1.

How many parameters does your network have? Please provide your calculations, layer by layer. You can ignore batch normalization parameters – just count the convolution and fully connected ones.

The first three layers are convolution layers. The computation for convolution layers is (kernel*(# of filters in previous layer)+1)*(# of filters in current layer).

Conv layer 1 has 96 filters, the input has 3 filter layers (RGB) and the kernel size is 11x11. The number of parameters = ((11*11*3)+1)*(96) = 34944

I'm not counting batch normalization and pooling = 0 so these layers (and future norm/pooling layers) dont matter in calculation.

Conv layer 2 has 256 filters, the previous layer had 96 filters, and the kernel size is 5x5. The number of parameters = ((5*5*96)+1)*(256) = 614656

Conv layer 3 has 384 filters, the previous layer had 256 filters, and the kernel size is 3x3. The number of parameters = ((3*3*256)+1)*(384) = 885120

The final two layers are fully connected. The number of parameters in a fully connected layer are (previous layer of neurons + 1)*(current layer of neurons)

Dense layer 1 has 254 neurons and the previous layer (a max pooling layer) was a 13x13x384 tensor, which when flattened turns to 64896 neurons total. (64896+1)*254 = 16483838

Dense layer 2 has 10 neurons and the previous layer had 254. (254 + 1)*10 = 2550

```
Conv1  =    34944
Conv2  =   614656 +
Conv3  =   885120 +
Dense1 = 16483838 +
Dense2 =     2550 +
-----------------
Total  = 18021108
```

The final total is just about 18 million parameters.

Question 2.
The initial problem with the provided small CNN was that its single convolutional layer wasn't capturing the relationships between the features of the data enough and the overall model wasn't generalizing its learning well. To fix this, I added two more convolutional layers with extra filters to learn the data more wholly and I added pooling and dropout layers to help the model only learn the important aspects of the training data, which allows for better generalization.

There was still some overfitting from these changes though, most likely because of the increased complexity of the model, so I fixed this issue by changing the optimizer and adding regularization to the training. The optimizer change was mostly due to the fact that I've had previous success with the Adamax optimizer over other ones, but the regularization directly addressed the overfitting, and the final model was able to achieve an test accuracy of 70% on CIFAR10.