CS 6476 Project 2

Karan Sarkar
sarkak2@gatech.edu
903569592

Pairs must answer these question separately

Part 1: Standard Scaler: Why did we use StandardScaler instead of looping over all the dataset twice for mean and standard deviation? Why a simple loop will not be a good choice in a deployed production grade ML system?

Loops in native Python are very slow. Using broadcasting in StandardScaler delegates the looping into lower level C++ code which is much faster.

Part 1: Why do we normalize our data (0 mean, unit standard deviation)?

Normalization improves the numerical stability of the model. Large unnormalized data values can lead to large gradient values for the weights. Having widely varying optimal values for the weights can mean convergence will take longer as no learning rate works will for every weight value.

Pairs must answer these question separately

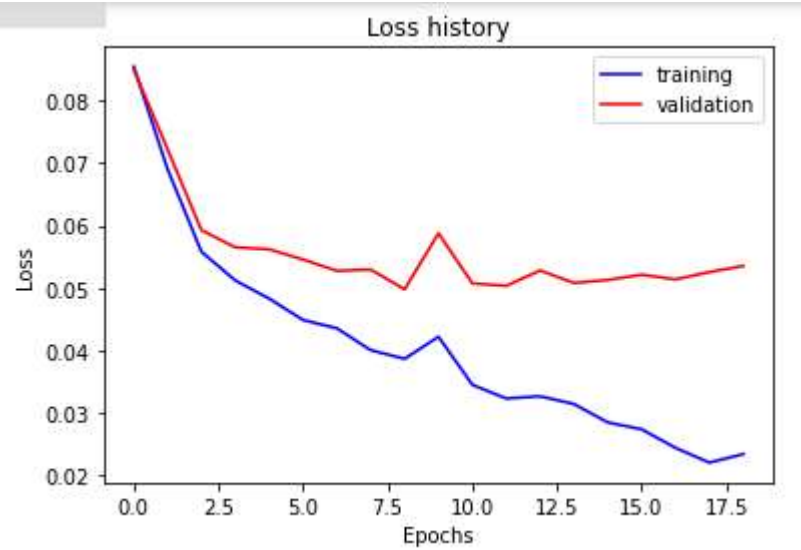
Part 3: Loss function. Why did we need a loss function?

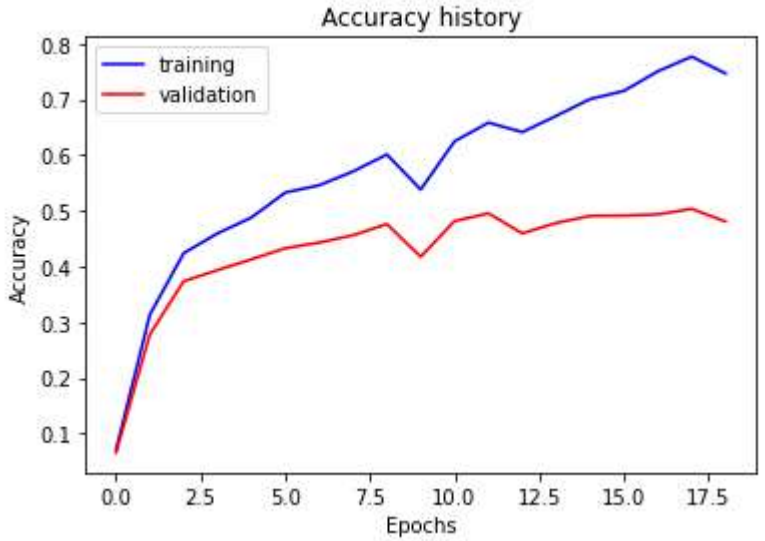
We need a loss to measure how far our current solution is from optimal. This way we can tell how to improve the existing solution.

Part 3: Explain the reasoning behind the loss function used

We used the nn.CrossEntropy loss function. This function represents the log likelihood of observing the data and labels given the current hypothesis function. If normalization was asked for, we divide by the number of data points.

Part 5: Training SimpleNet





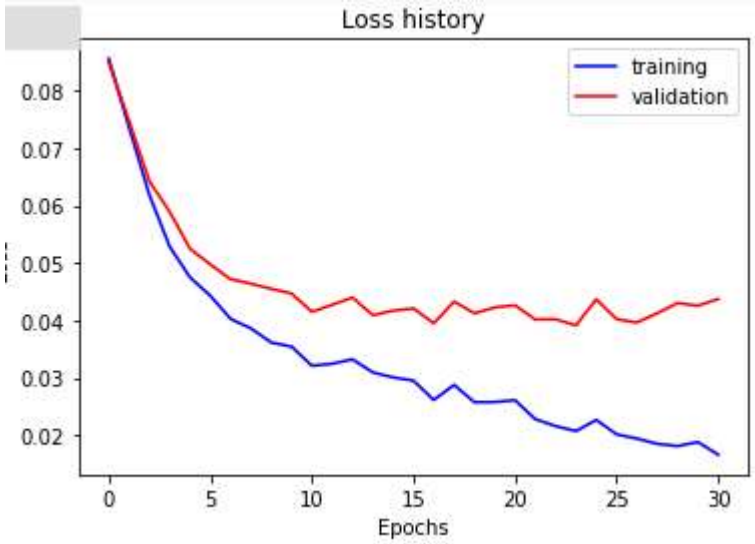
Final training accuracy value: 0.7467

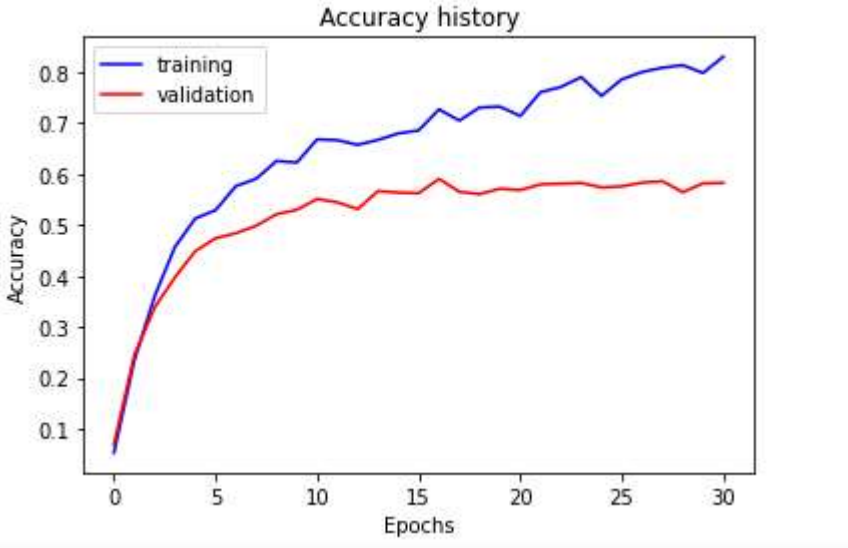
Final validation accuracy value: 0.4807

Part 6: Screenshot of your get_data_augmentation_transforms()

```
return transforms.Compose([
    ##########################################################################
    # Student code begin
    ##########################################################################
    transforms.Resize(inp_size),
    transforms.ColorJitter(brightness=0.01, contrast=0.01, saturation=0.01, hue=0.01
    transforms.ToTensor(),
    transforms.Normalize(pixel_mean, pixel_std),
    transforms.RandomHorizontalFlip(p=0.5)
    ##########################################################################
    # Student code end
    ##########################################################################
])
```

Part 7: Training SimpleNetDropout





Final training accuracy value: 0.8308

Final validation accuracy value: 0.5827

Pairs must answer these question separately

Part 7: SimpleNetDropout: compare the loss and accuracy for training and testing set, how does the result compare with Part 1? How to interpret this result?

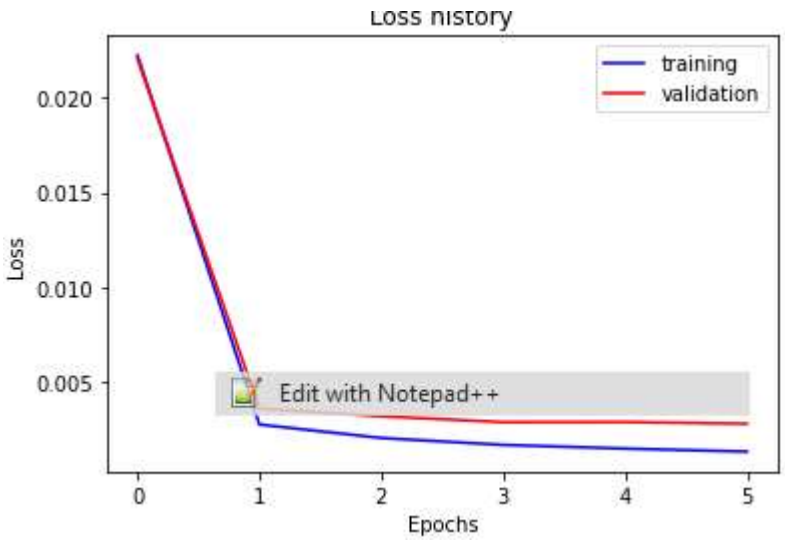
I got a much improved accuracy on the testing set and a similar accuracy on the training set. This is because the use of dropout and data augmentation reduced the amount of overfitting and improved generalization.

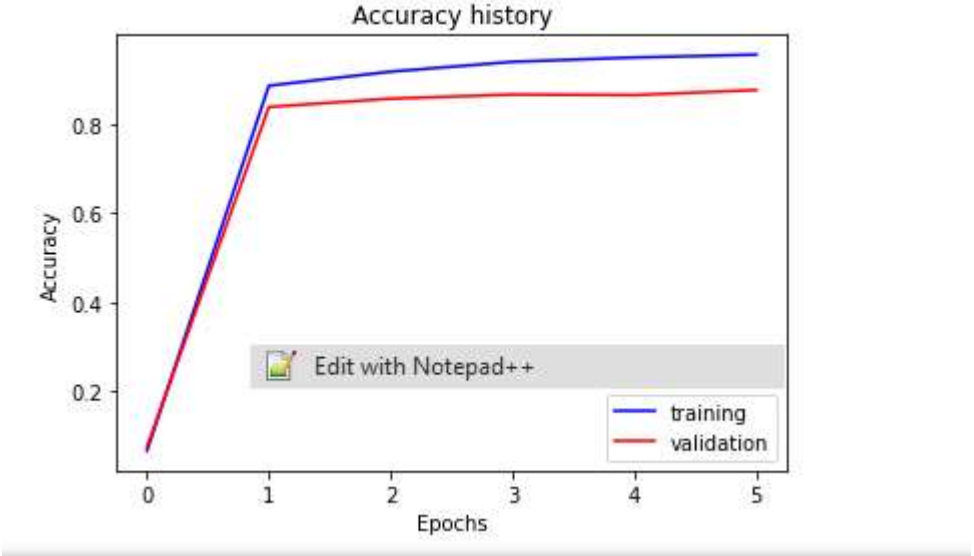
Pairs must answer these question separately

Part 7: SimpleNetDropout: How did dropout and data-augmentation help?

Dropout and Data augmentation reduced the amount of overfitting and improved generalization. Dropout effectively reduces the complexity of the model and makes overfitting less likely. Data augmentation doubled the amount of usable data and allowed more training to be done before reviewing the same data points again.

Part 8: Training Alexnet





Final training accuracy value: 0.9554

Final validation accuracy value: 0.8760

Pairs must answer these question separately

Part 8: AlexNet: what does fine-tuning a network mean?

Fine Tuning a network means to use transfer learning to use a pretrained network. You use a pretrained network that has been trained on a huge data set for the lower layers and then just train the upper layers manually. This gives a superior performance because you are reusing information that has already been learned.

Pairs must answer these question separately

Part 8: AlexNet: why do we want to “freeze” the conv layers and some of the linear layers in pretrained AlexNet? Why CAN we do this?

If we were to train using all of the layers, we would surely overfit. It also reduces the amount of computation that needs to be done. We can do this because the the lower level features should be similar for a variety of tasks. Because the lower layers correspond to lower level features, we can freeze the lower layers.

Pairs must answer these question separately

Conclusion: briefly discuss what you have learned from this project.

I learned that pretraining can result in a huge increase in performance. I also learned that dropout is very effective at reducing overfitting without sacrificing accuracy. It seems to work better than weight decay.

Pairs must answer these question separately
(for observations and conclusions)

EC1: Ablation study #1

<Write your hyperparameter values, accuracy plot, training time, inference time results in a table and provide observations and conclusions>

Pairs must answer these question separately
(for observations and conclusions)

EC1: Ablation study #2

<Write your hyperparameter values, accuracy plot, training time, inference time results in a table and provide observations and conclusions>

Pairs must answer these question separately
(for observations and conclusions)

EC1: Ablation study #3

<Write your hyperparameter values, accuracy plot, training time, inference time results in a table and provide observations and conclusions>

EC2: Quantization. Paste the code of the quantize function here.

<Screenshot here>

EC2: Quantization. Briefly discuss the steps you followed. You cannot use code and should describe the intuition behind each step.

<text answer here>

EC2: Quantization. Paste your results here

Size comparison: <text answer here>

Speed comparison: <text answer here>

Accuracy comparison: <text answer here>

EC3: Analysis. What are your confusion matrices for each model? What information do they give you and what do you think about it?

<confusion matrix for SimpleNet>

<confusion matrix for SimpleNetDropout>

EC3: Analysis. What are your confusion matrices for each model? What information do they give you and what do you think about it?

<confusion matrix for MyAlexNet>

<information they give you and your ideas about it>

Pairs must answer these question separately

EC3: Analysis. What additional metrics are you using? What are the scores and how are they evaluating the model's performance? What do you think could possibly improve the performance?

<text answer here>