CS 4476 PS5

Younes Djemmal
ydjemmal3@gatech.edu
903794219

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?

StandardScaler actually does the same thing but taking into account the case where the dataset size is large and performing operations for large matrices is not possible; it will basically split up the dataset and perform the operations gradually instead of in one bulk.

Looping over the whole dataset in ML to calculate mean and std will take up a lot of disk space and power which is not possible in most cases. StandardScaler is designed to do this in a much faster and simpler way.

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

The reason why we normalize our data is to make sure the gradient converges faster by havinf similar value ranges for all of our features. Otherwise we'd be learning things from data that have different scales and it can be more difficult.

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

The loss function is our objective function, meaning it’s the function that we have the goal of minimizing to solve the problem at hand. The parameters of the model are determined by minimizing this loss function.

It’s simply a measure of performance for our model and the lower the better. The model aims to get the lowest loss possible.

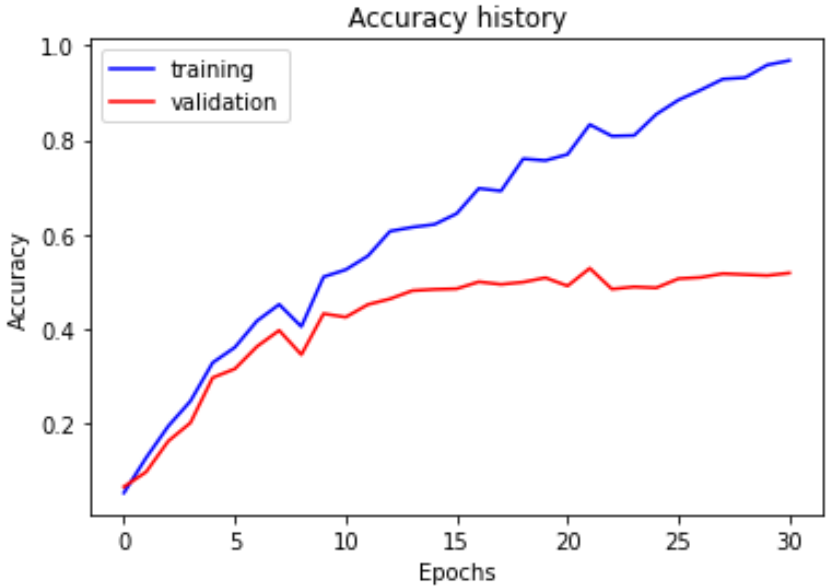
Part 3: Explain the reasoning behind the loss function used

I used Cross-entropy as a loss function.
I measures the difference between our true labels yi and y_hat, the vector of predicted probabilities after optimizing our weights. We get a measure of dissimilarity between y and y_hat and the goal of our model is to minimize this loss.

Part 5: Training SimpleNet

<Loss plot here>

<Accuracy plot here>



Final training accuracy value: 96,78%

Final validation accuracy value: 51,87%

```
print('Train Accuracy = {:.4f}; Validation Accuracy =
```

```
Train Accuracy = 0.9678; Validation Accuracy = 0.5187
```

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

After this project, I feel confident to go and take on more and more DL problems as I have aquired all the steps of the pipeline. I now know how to prepare a dataset for a given ML problem. How to design the architecture of my CNN model and train the model on my dataset. I have also learnt tricks to reduce overfitting. Morover, I now can transfer the knowledge learnt by other models out there and apply to my problem and build upon that.

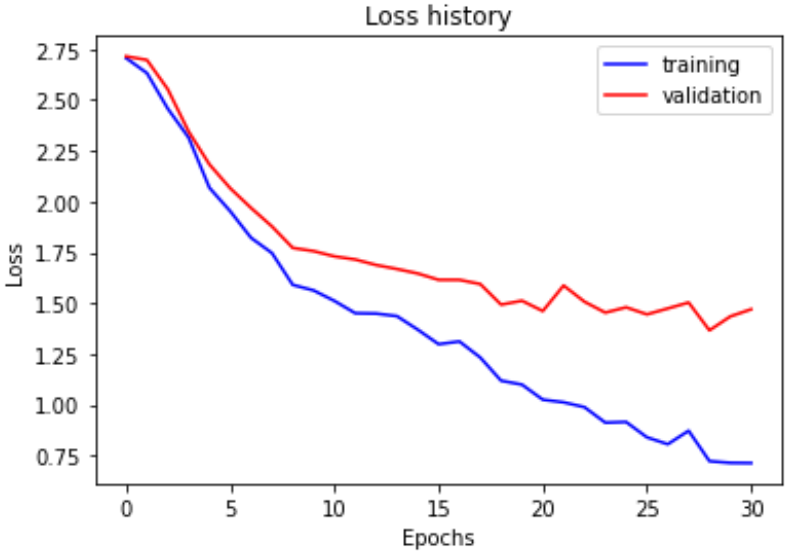
EC1.1: Screenshot of your get_data_augmentation_transforms()

<Screenshot here if attempted; do not delete the slide if not attempted>

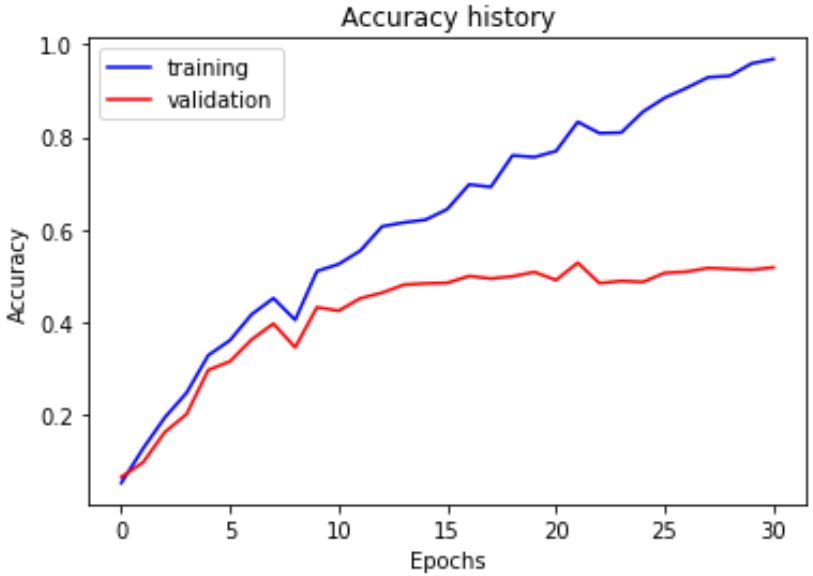
```
    Returns:
    - aug_transforms: transforms.compose with all the transf
    """

    return transforms.Compose(
        [
            ##################################################
            # Student code begin
            ##################################################
            transforms.ColorJitter(),
            transforms.RandomHorizontalFlip(),
            transforms.Resize(inp_size),
            transforms.Grayscale(1),
            transforms.ToTensor(),
            transforms.Normalize(pixel_mean,pixel_std)
            ##################################################
            # Student code end
            ##################################################
        ]
    )
```

EC1: Training to solve overfitting

<Loss plot here>



<Accuracy plot here>



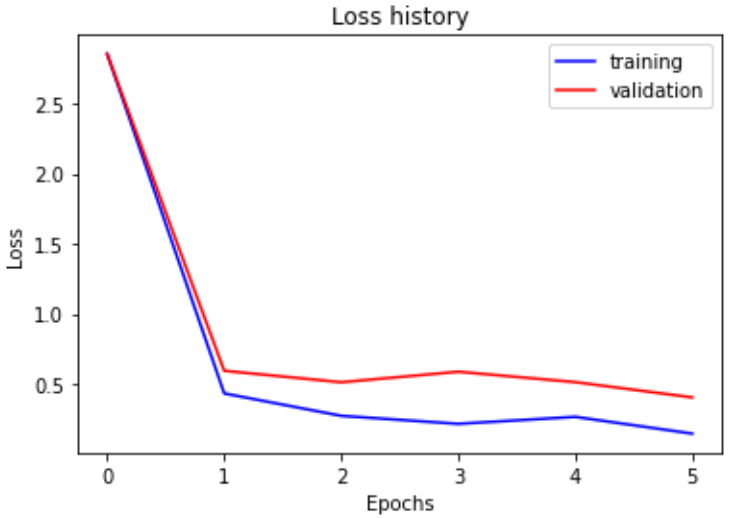
Final training accuracy value: 76,01%

Final validation accuracy value: 54%

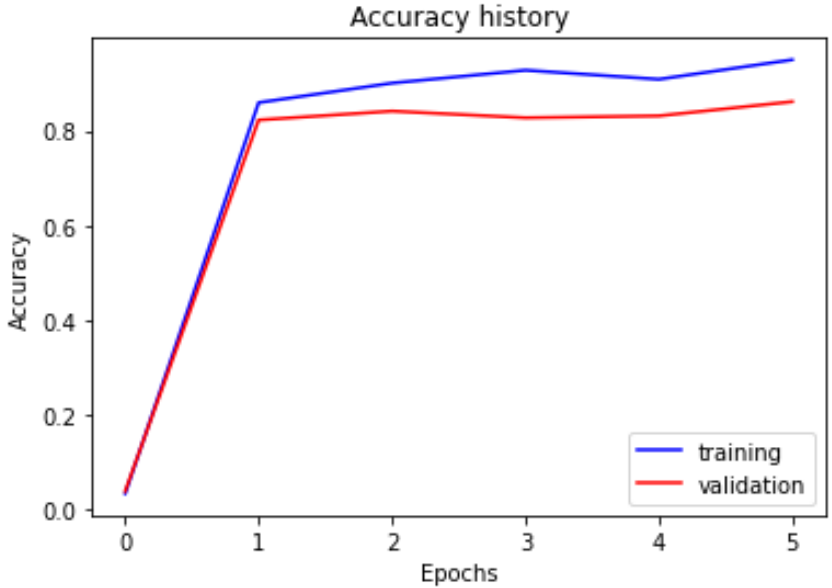
```
print( Train Accuracy = {:.4f}; Validation Accuracy =
Train Accuracy = 0.7601; Validation Accuracy = 0.5400
```

EC2 AlexNet: Training Alexnet

<Loss plot here>



<Accuracy plot here>



Final training accuracy value: 94,97%

Final validation accuracy value: 86,13%

```
Train Accuracy = 0.9497; Validation Accuracy = 0.8613
```