CS 6476 Project 2

Safin Salih
ssalih6@gatech.edu
902111076

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?

When you care about computationial speed, you'd want to avoid looping and instead you want take advantage of vectorization and broadcasting. Which is exactly what StandardScaler does.

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

We normalize the input data so that the features in our image data are all on the same scale or in similar distribution .This will make convergence faster. So, standardization gives features a comparable scaling without highlighting outliers.

Pairs must answer these question separately

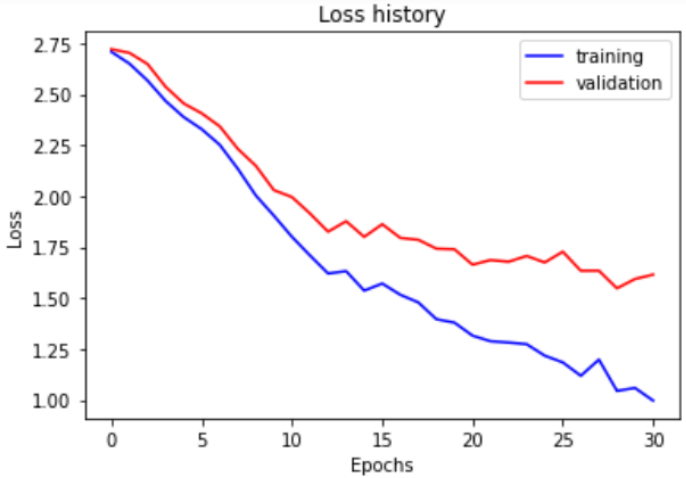
Part 3: Loss function. Why did we need a loss function?

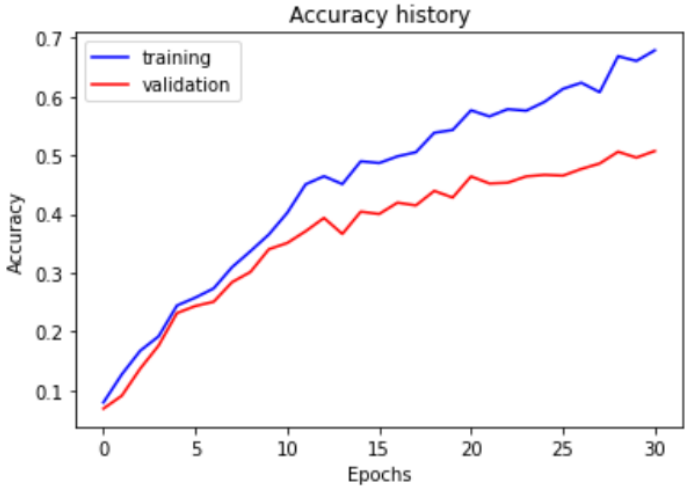
For Neural Network(Without loss of generality), in the backpropgation step we compute the gradient of the loss function/objective function with respect to each weights of layers by chain rule, our goal being to minimize loss function. Which makes the loss function a signal/feedback on how we should tune our weight, so we have a (slightly) better outcome in the next iteration.

Part 3: Explain the reasoning behind the loss function used

Because without the loss function we have no direction how we should incrementally change our weights in the network.

Part 5: Training SimpleNet

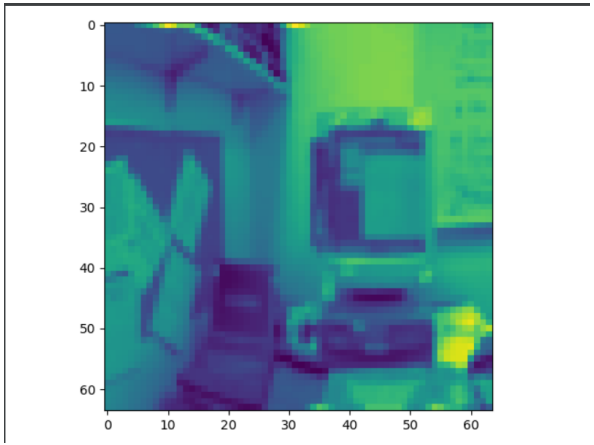


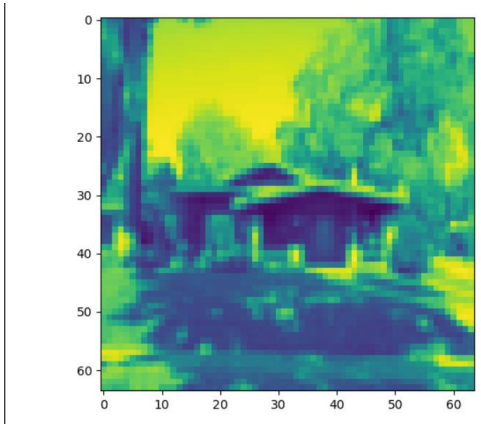


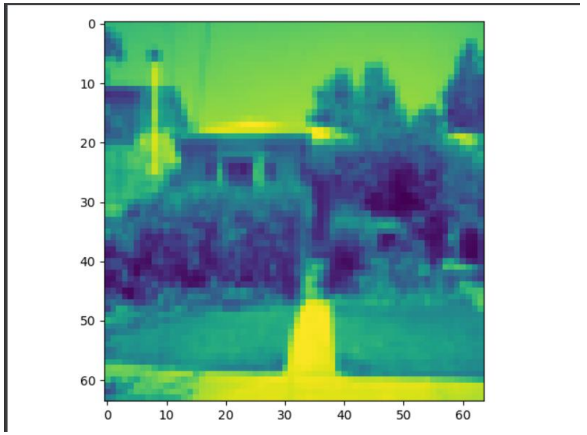
Final training accuracy value:0.6787

Final validation accuracy value:0.5073

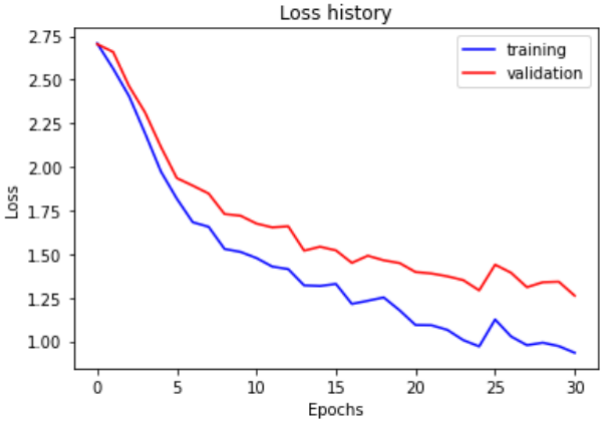
Part 6: Screenshot of your get_data_augmentation_transforms()

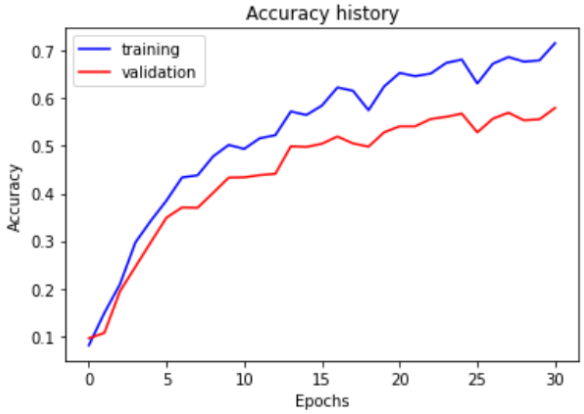






Part 7: Training SimpleNetDropout





Final training accuracy value:0.6972

Final validation accuracy value:0.5780

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?

Adding DropOut in the ffc layer helped in terms of accuracy in the validation set, also training loss and validation loss seem closer to each other at each iteration in training in SimpleNetDropOut.

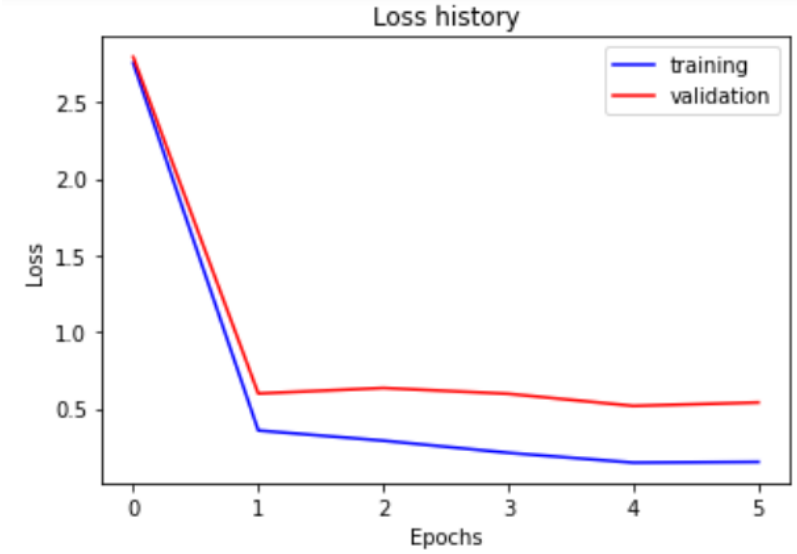
This means that there was some overfitting occuring in Part one; which means the network wouldn’t be able to generalize really well in new unseen data, so adding a regularizer term certaintly helped in terms of overall accruacy.

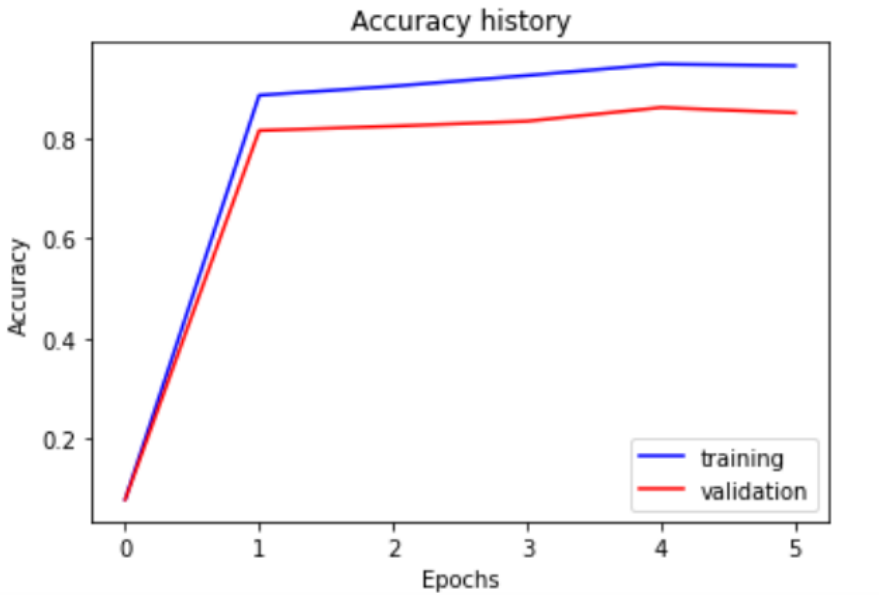
Pairs must answer these question separately

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

Dropout regularization helps reduce overfitting and improves generalization in a neural network. While data-agumentation provides more data images by altering them slightly such that the relevent information is still holds in the augement image. Which results in the neural network gaining more information and hence improve overall accuracy.

Part 8: Training Alexnet





Final training accuracy value: 0.9447

Final validation accuracy value:0.8507

Pairs must answer these question separately

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

Fine tuning is the process of taking a neural network that was trained on some task and using the weights of that neural network model trained on different task given that they share some similarities (for example images) . This helps speed up training time since you’re the weights in the network start in a ‘good’ position during backpropgration.

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?

In a pretrained model you are freezing the earlier layers by making the weights unchangeable. We do this so it can retain already learned basic representations, and the later layers are not frozen so that they can learn specific representation of the task at hand. We can do this because someone previously trained AlexNet on a different task (usually Imagenet) , and we download the weights of that model and adjust which layers to freeze and unfreeze at our will.

Pairs must answer these question separately

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

I learned different methods of achieving higher accuracy and better generalization; for example, data augmentation, dropout, pretrained models, playing around with different parameters in the optimizer. During training relying on my GPU instead of CPU made a difference in terms of computational speed, but nonetheless searching for the right parameters is rather time-consuming process. Training a deeper network yield better result but tends to overfit hence regularization is necessary . I also learned that plotting your results (line graph) gives you a better understanding if your over fitting and when to stop training.

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>