**CSCI 677**

**HW4 Report**

**Dhruv Parikh**

**Main Experiment**

As described in the HW document, the entire report is divided into two parts – main experiment and variation experiments.

The current section will briefly describe the coding aspects of the main experiments along with the results obtained.

**Code Divisions**

i. Dataset and Dataloader

ii. Neural Net

iii. Training and Validation

iv. Testing

v. Metrics

All the sections are relatively straightforward and have been coded along the lines of instructions in the assignment and the PyTorch tutorials (specifically the CIFAR10 one).

A short description goes thus –

- For Dataset generation the Dataset PyTorch class was inherited and the regular definitions and methods added in. The data was downloaded on a local machine using code attached in the homework, then the splits were referenced to generate the datasets.

- One important thing to note here is that converting to range (0,1) and normalizing the image post that was done using two normalizing transforms – as such for validation and test datasets the ideal approach should have been to average the max, min, mean and std quantities for each image obtained from the training set and using that to convert the validation and test ranges into (0,1) and to normalize them. This approach could have been easily implemented; however, I chose not to, and the normalizing transforms used the max, min, mean and std metric from the validation and the test set itself. The reason for this was two-fold – to simplify the code and to avoid keeping a track of max, min, mean and std metrics from the train set. Had it not been for this simplification, possibly, separate dataset definitions from validation and test would’ve to be created.

- Thus, despite the code addition being easy, it was time consuming and so pre-processing on each set was done using their local metrics.

- Neural Net was defined typically – exactly as needed for the main experiment.

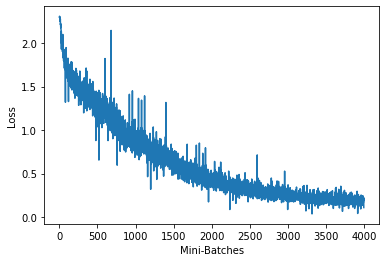
- Training and Validation was done on my local computer itself, the entire process took around 20-30 minutes so I chose not to perform the training on Google Collaboration.

- Testing, again, was typically done.

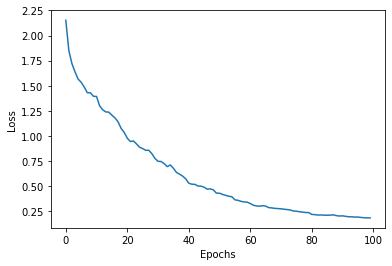
- Since the output metrics were limited to – training and validation loss tracking, classification accuracy and confusion matrix, to keep things simple for the assignment I chose to work without using loggers or TensorBoard. All the results are available in the notebooks, the report and in separate folders in the submission.

**Results - 1**

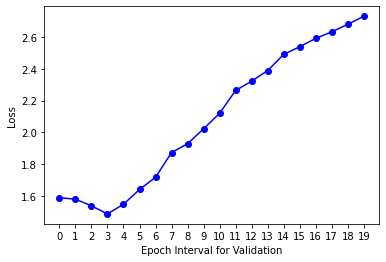
For 100 training epochs.



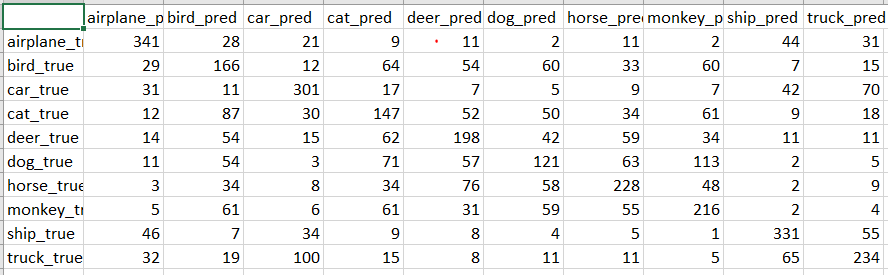
This is the minibatch loss versus minibatches for all the minibatches in the training data.



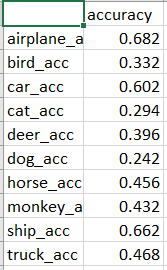
This is the loss versus epochs curve – as can be seen the loss consistently reduces with the number of epochs.



The above curve is the validation loss versus epoch (epoch interval) and it can be clearly seen that the data has been overfitted. The number of training epochs should be in the range of 15 to 20 epochs (each point in the curve is 5 training epochs).



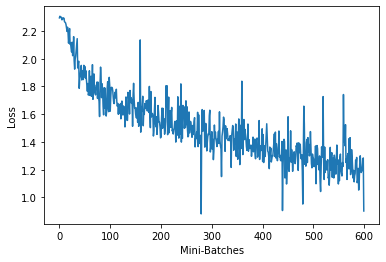
Confusion Matrix is as seen above – also saved in the .csv format.



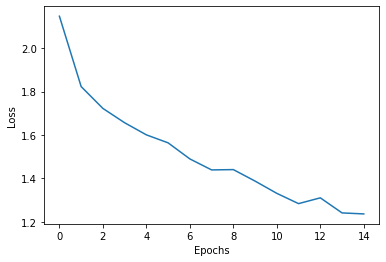
The classification accuracy is as seen above. Also in the .csv format.

**Result – 2**

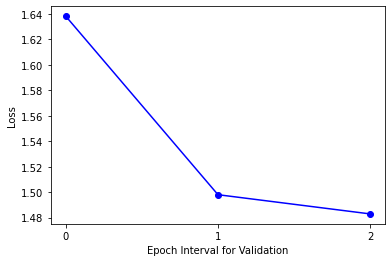
These are the results for training for 15 epochs as could be deliberated from the 100 epoch case, since the validation error there increased post 15 epochs.



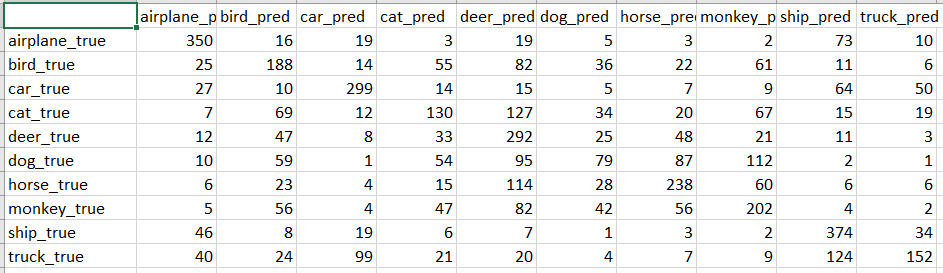
The minibatch error for this case.



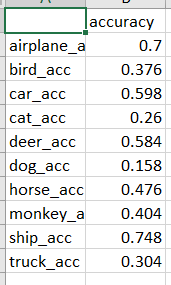
The epoch loss in this case.



As can be seen here, we have not overfit the data. We have stopped training before the validation error could possibly increase.



Confusion Matrix for this case.

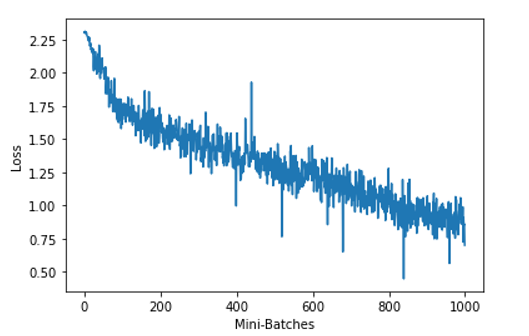


Accuracy for this case.

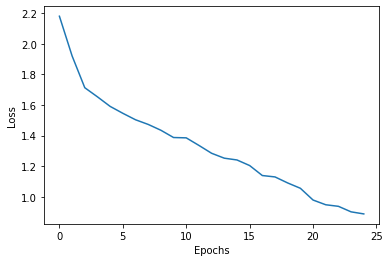
As can be seen, by not overfitting the data, the accuracy is somewhat increased. However, we can surely increase it much more and the full potential of using a complicated neural network for training hasn’t been completely extracted.

**Variation Experiment**

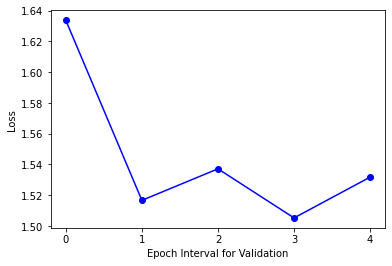
For the variation experiment, everything in the code is precisely same, however batch normalization, AdamW and 25 training epochs have been used here.



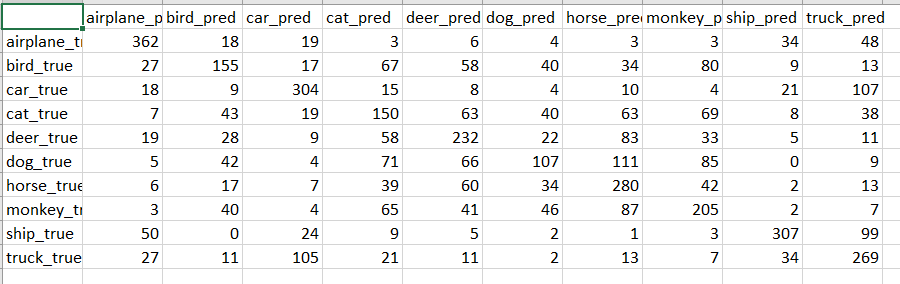
Minibatch loss versus epochs



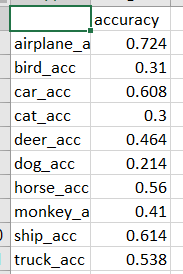
Loss versus Epochs



Validation Loss versus Validation Epochs



Confusion Matrix



Accuracy per class

**Notes –**

Most classes have decent accuracies except cats and dogs – this might be due to the inter-class confusion or due to the relative similarity between these classes. Training more will overfit the remaining data, but, at the same time may relatively improve results for the cats and dogs classes.

This is just one possible caveat though.