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.

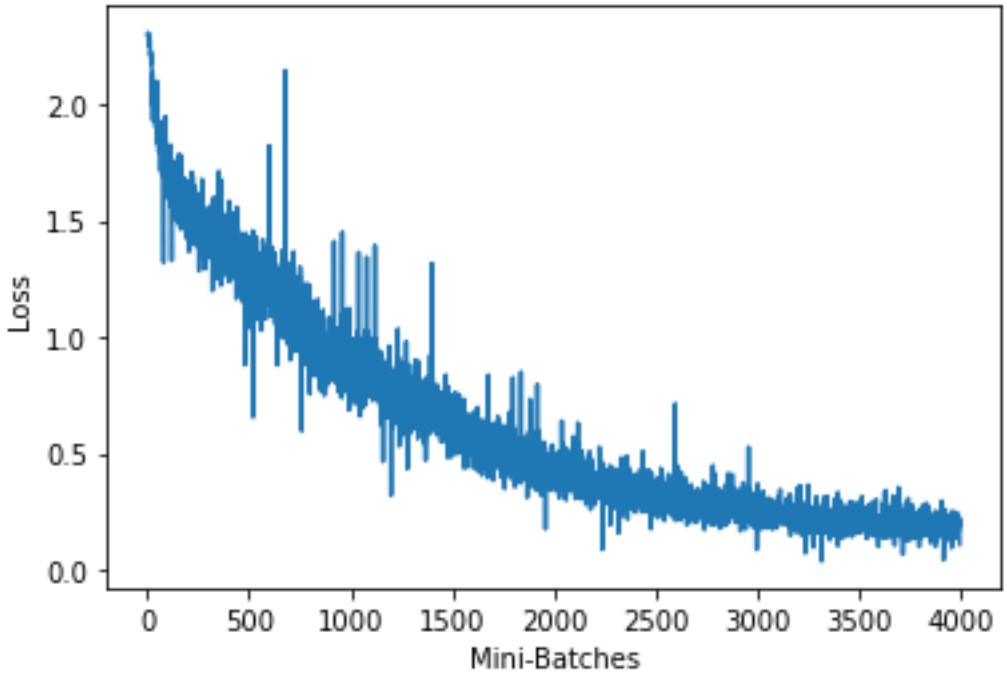
- Testing, again, was typically done.

- 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

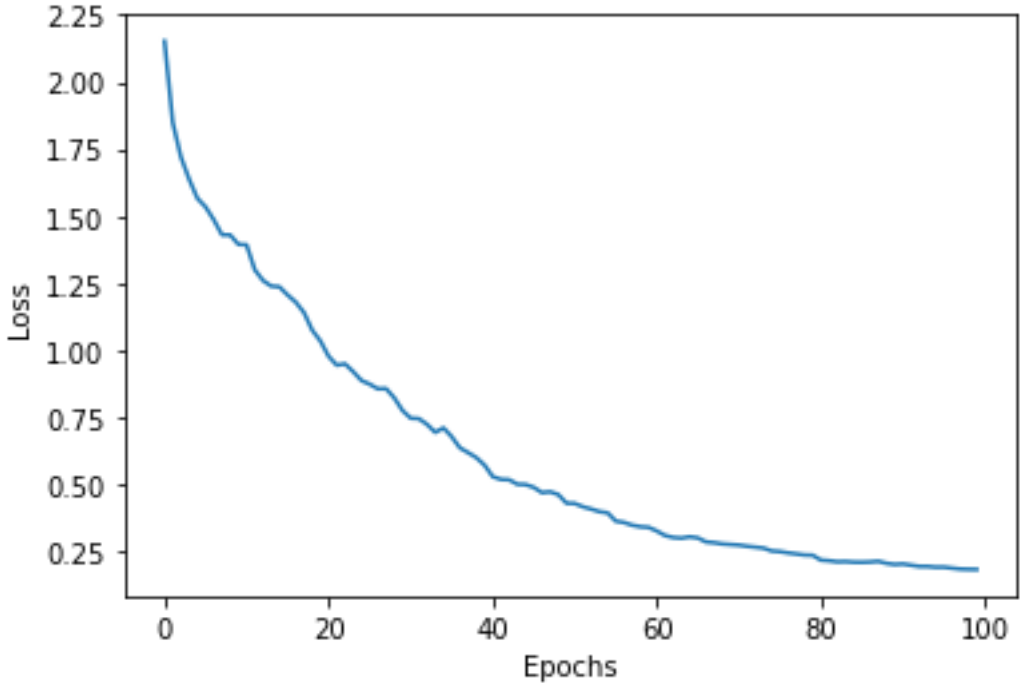
without using loggers or TensorBoard. All the results are available in the notebooks, the report and in separate folders in the submission.

Results - 1

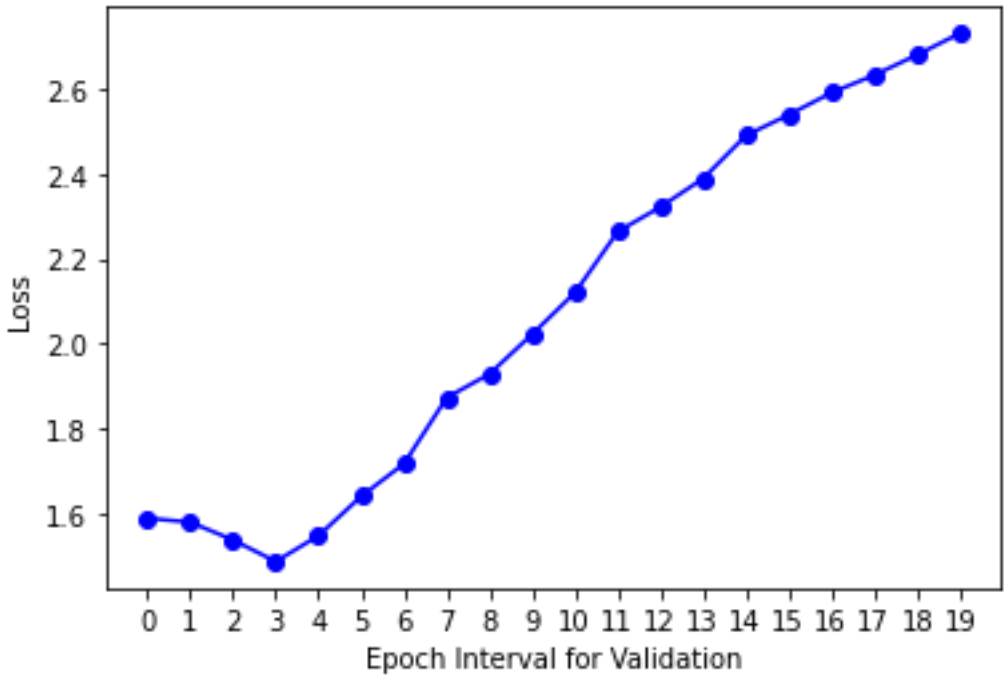
For 100 training epochs.



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).

	airplane_p	bird_pred	car_pred	cat_pred	deer_pred	dog_pred	horse_pre	monkey_p	ship_pred	truck_pred
airplane_t	341	28	21	9	11	2	11	2	44	31
bird_true	29	166	12	64	54	60	33	60	7	15
car_true	31	11	301	17	7	5	9	7	42	70
cat_true	12	87	30	147	52	50	34	61	9	18
deer_true	14	54	15	62	198	42	59	34	11	11
dog_true	11	54	3	71	57	121	63	113	2	5
horse_true	3	34	8	34	76	58	228	48	2	9
monkey_t	5	61	6	61	31	59	55	216	2	4
ship_true	46	7	34	9	8	4	5	1	331	55
truck_true	32	19	100	15	8	11	11	5	65	234

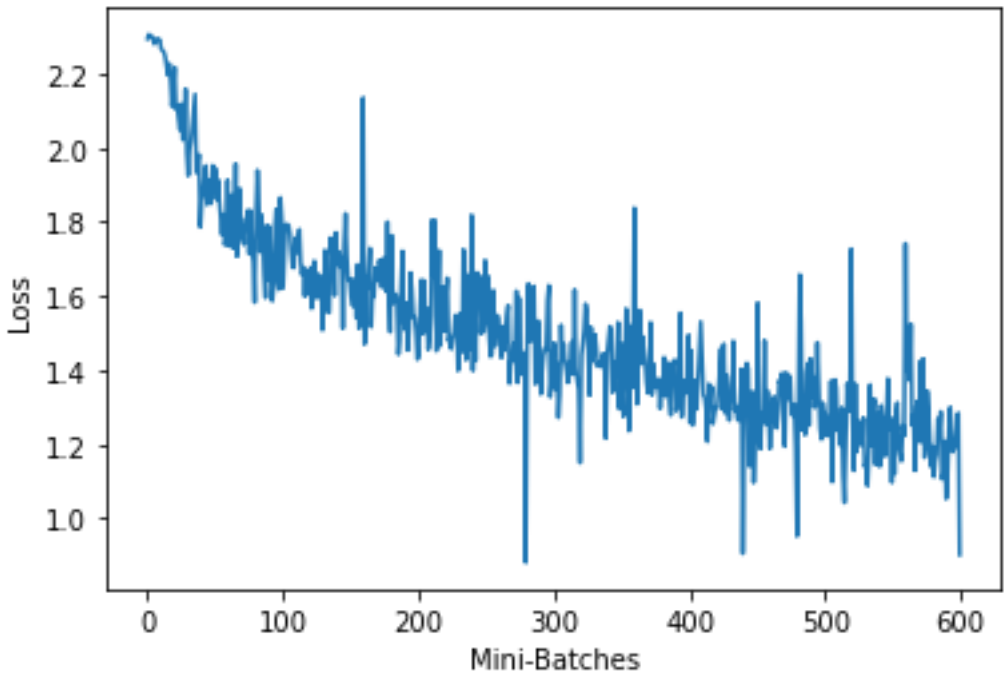
Confusion Matrix is as seen above – also saved in the .csv format.

	accuracy
airplane_a	0.682
bird_acc	0.332
car_acc	0.602
cat_acc	0.294
deer_acc	0.396
dog_acc	0.242
horse_acc	0.456
monkey_a	0.432
ship_acc	0.662
truck_acc	0.468

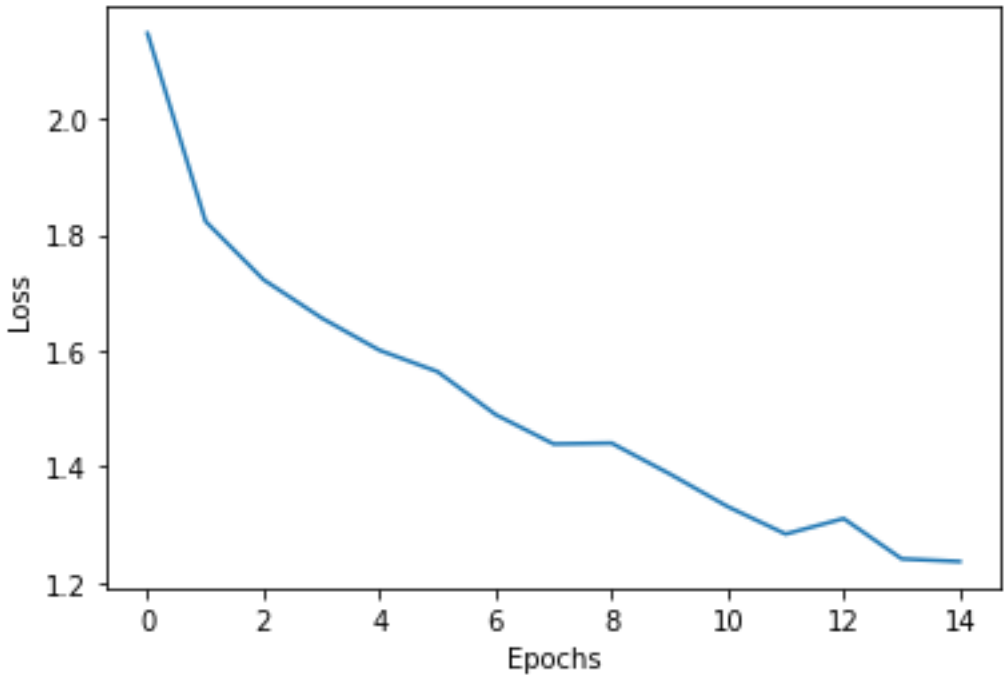
The classification accuracy is as seen above. Also in the .csv format.

Result – 2

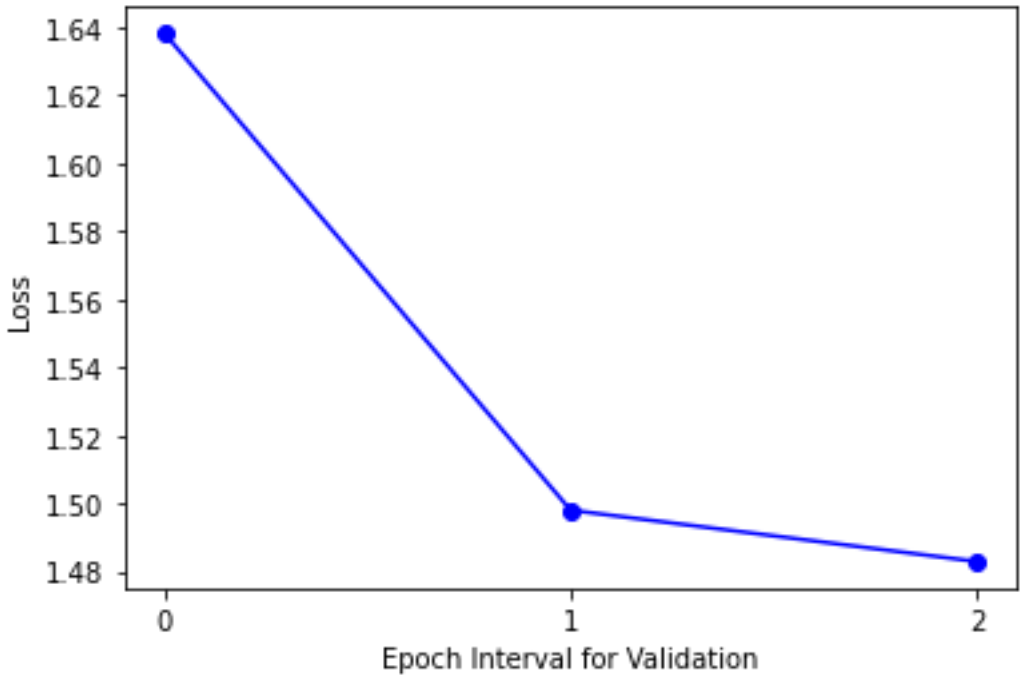
These are the results for training for 15 epochs as could be deliberated from the 100 epoch case, since the validation error there increased post 15 epochs.



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.

	airplane_p	bird_pred	car_pred	cat_pred	deer_pred	dog_pred	horse_pre	monkey_p	ship_pred	truck_pred
airplane_true	350	16	19	3	19	5	3	2	73	10
bird_true	25	188	14	55	82	36	22	61	11	6
car_true	27	10	299	14	15	5	7	9	64	50
cat_true	7	69	12	130	127	34	20	67	15	19
deer_true	12	47	8	33	292	25	48	21	11	3
dog_true	10	59	1	54	95	79	87	112	2	1
horse_true	6	23	4	15	114	28	238	60	6	6
monkey_true	5	56	4	47	82	42	56	202	4	2
ship_true	46	8	19	6	7	1	3	2	374	34
truck_true	40	24	99	21	20	4	7	9	124	152

Confusion Matrix for this case.

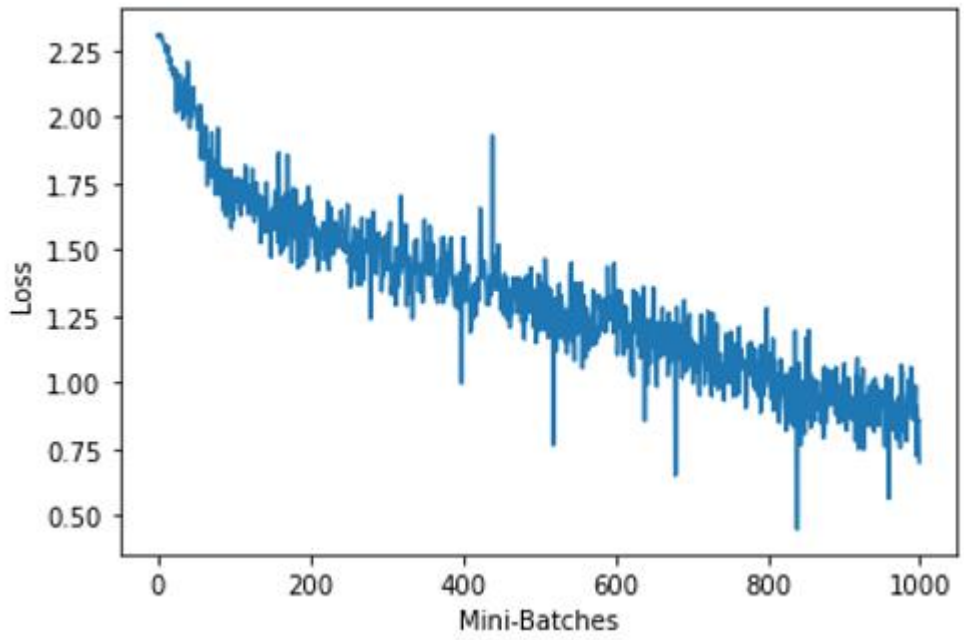
	accuracy
airplane_a	0.7
bird_acc	0.376
car_acc	0.598
cat_acc	0.26
deer_acc	0.584
dog_acc	0.158
horse_acc	0.476
monkey_a	0.404
ship_acc	0.748
truck_acc	0.304

Accuracy for this case.

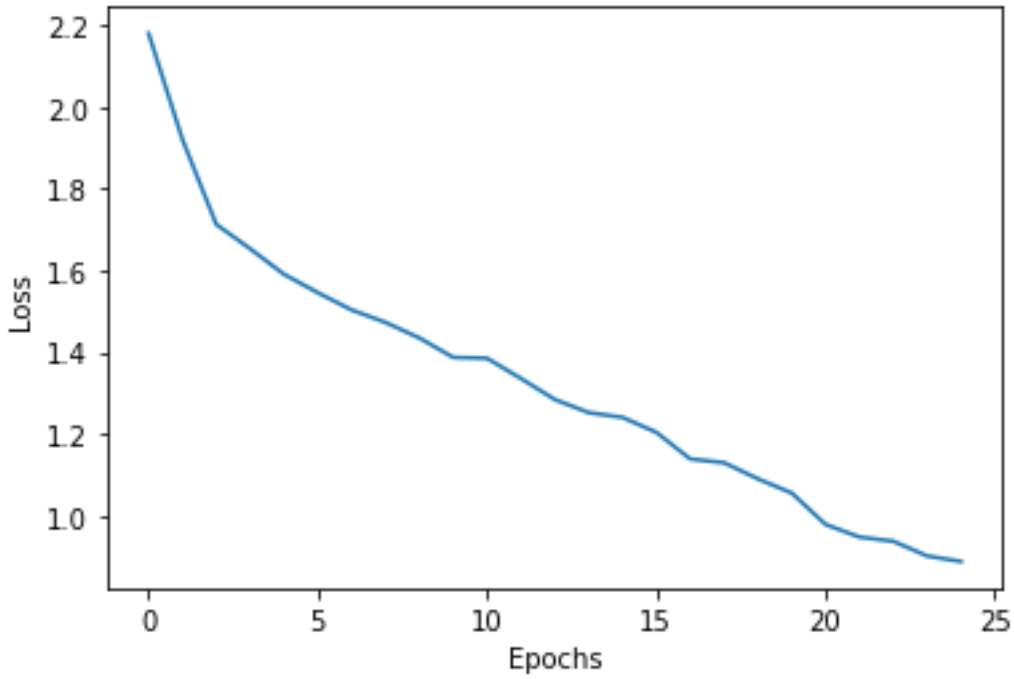
As can be seen, by not overfitting the data, the accuracy is somewhat increased. However, we can surely increase it much more and the full potential of using a complicated neural network for training hasn’t been completely extracted.

Variation Experiment

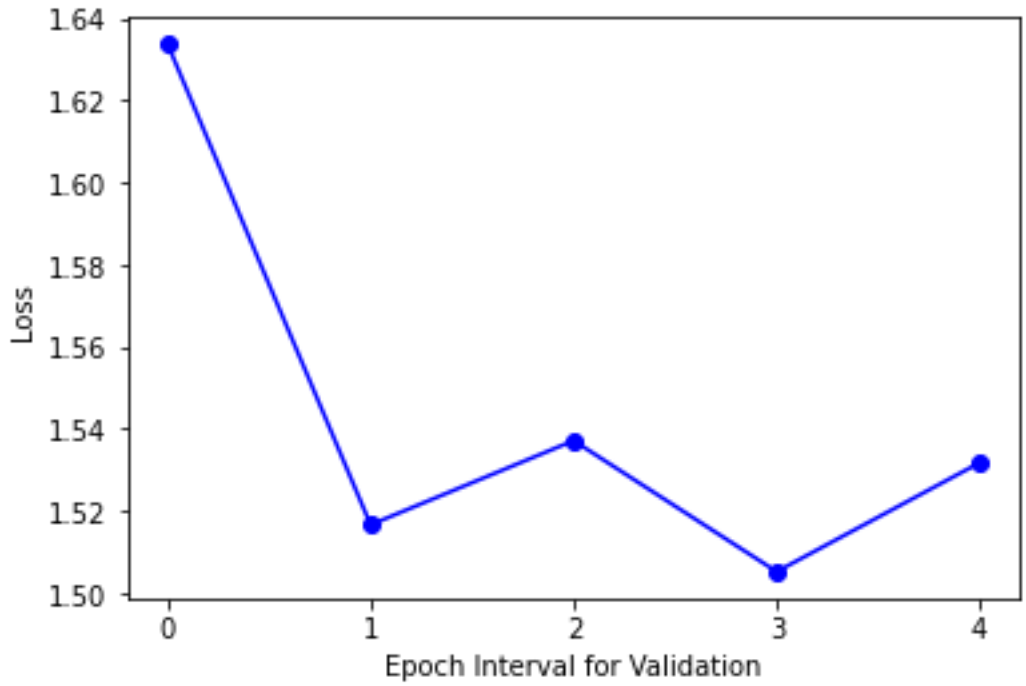
For the variation experiment, everything in the code is precisely same, however batch normalization, AdamW and 25 training epochs have been used here.



Minibatch loss versus epochs



Loss versus Epochs



Validation Loss versus Validation Epochs

	airplane_p	bird_pred	car_pred	cat_pred	deer_pred	dog_pred	horse_pre	monkey_p	ship_pred	truck_pred
airplane_t	362	18	19	3	6	4	3	3	34	48
bird_true	27	155	17	67	58	40	34	80	9	13
car_true	18	9	304	15	8	4	10	4	21	107
cat_true	7	43	19	150	63	40	63	69	8	38
deer_true	19	28	9	58	232	22	83	33	5	11
dog_true	5	42	4	71	66	107	111	85	0	9
horse_true	6	17	7	39	60	34	280	42	2	13
monkey_t	3	40	4	65	41	46	87	205	2	7
ship_true	50	0	24	9	5	2	1	3	307	99
truck_true	27	11	105	21	11	2	13	7	34	269

Confusion Matrix

	accuracy
airplane_a	0.724
bird_acc	0.31
car_acc	0.608
cat_acc	0.3
deer_acc	0.464
dog_acc	0.214
horse_acc	0.56
monkey_a	0.41
ship_acc	0.614
truck_acc	0.538

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.