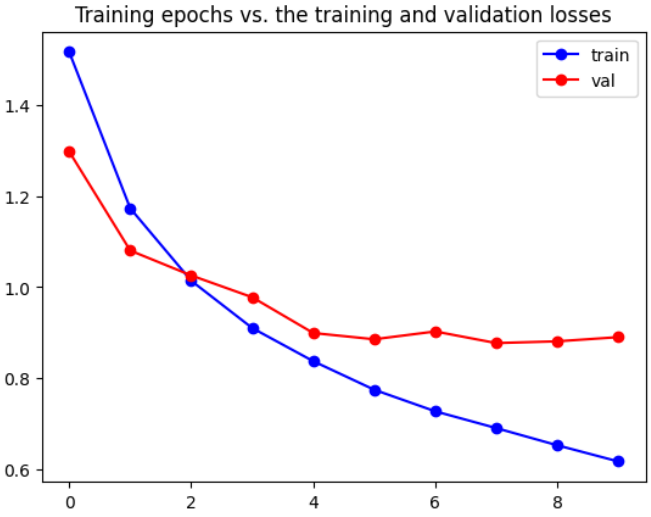
PyTorch Implementation

Data Augmentation chosen is normalization

Choice between Optimizers

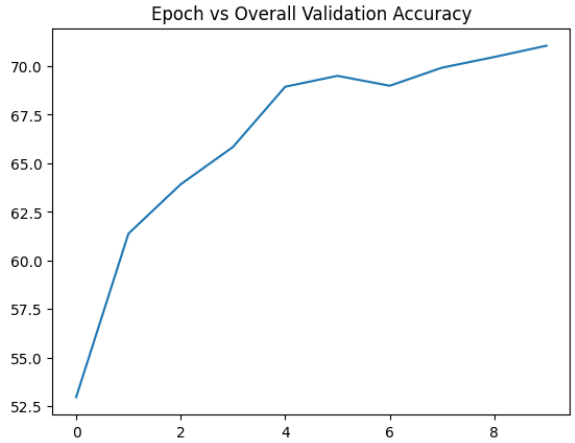
Adam

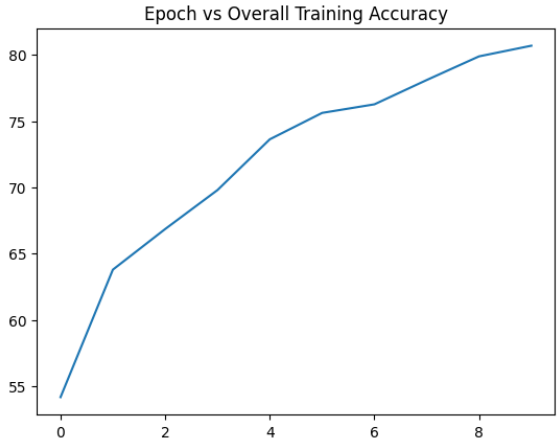
LR = 0.001, Epochs = 10, Batch = 32 **(Base case for comparison)**




```
71.05
```

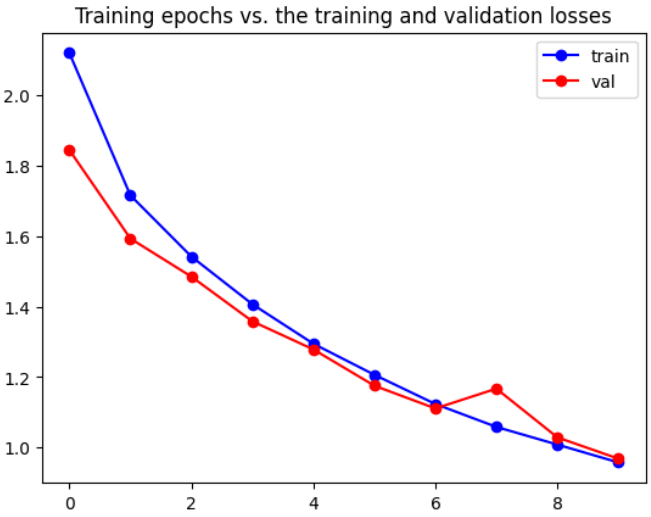
```
Accuracy for class: plane is 68.7 %
Accuracy for class: car   is 85.2 %
Accuracy for class: bird  is 54.3 %
Accuracy for class: cat   is 49.5 %
Accuracy for class: deer  is 75.6 %
Accuracy for class: dog   is 61.9 %
Accuracy for class: frog  is 84.2 %
Accuracy for class: horse is 75.5 %
Accuracy for class: ship  is 84.1 %
Accuracy for class: truck is 71.5 %
```





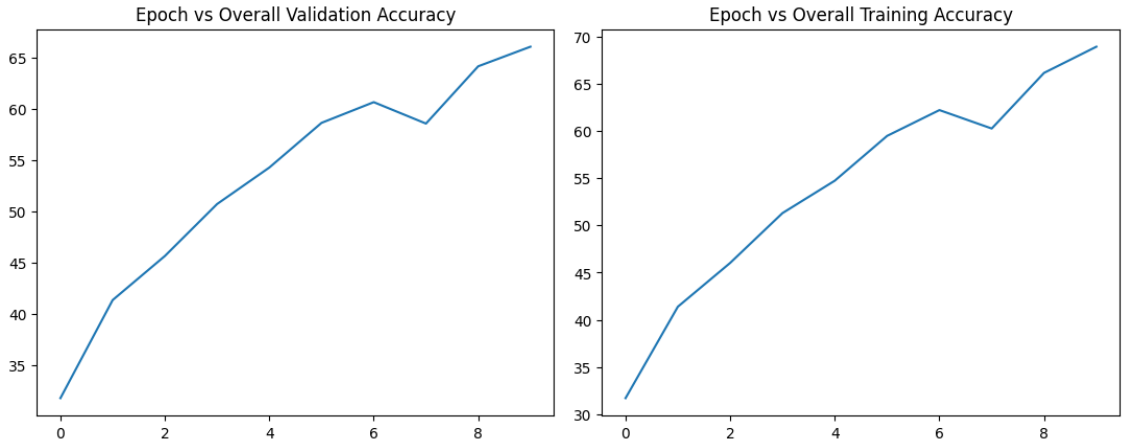
SGD

LR = 0.001, Epochs = 10, Batch = 32, Momentum = 0.9




```
66.06
```

```
Accuracy for class: plane is 64.6 %
Accuracy for class: car   is 78.6 %
Accuracy for class: bird  is 45.5 %
Accuracy for class: cat   is 56.2 %
Accuracy for class: deer  is 58.7 %
Accuracy for class: dog   is 51.2 %
Accuracy for class: frog  is 72.5 %
Accuracy for class: horse is 74.1 %
Accuracy for class: ship  is 80.5 %
Accuracy for class: truck is 78.7 %
```



Choice between LR

LR = 0.001

Refer base case (Pg1)

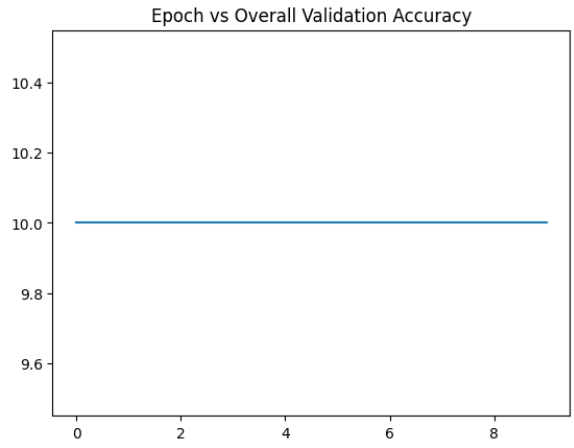
LR = 0.01

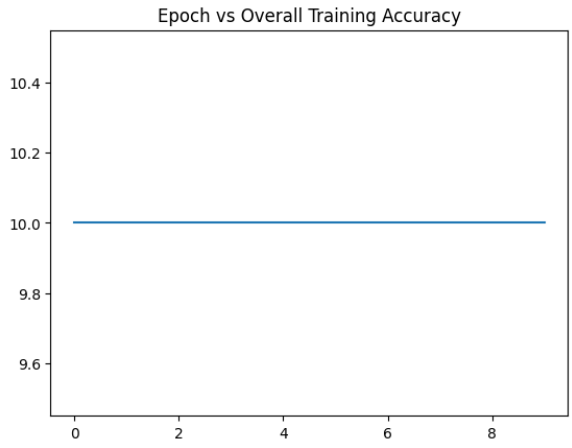
Epochs = 10, Batch = 32, Adam

Learning rate is too large, updates may be overshooting local minima and it is unable to converge.


```
10.0
```

```
Accuracy for class: plane is 0.0 %
Accuracy for class: car   is 0.0 %
Accuracy for class: bird  is 0.0 %
Accuracy for class: cat   is 0.0 %
Accuracy for class: deer  is 100.0 %
Accuracy for class: dog   is 0.0 %
Accuracy for class: frog  is 0.0 %
Accuracy for class: horse is 0.0 %
Accuracy for class: ship  is 0.0 %
Accuracy for class: truck is 0.0 %
```

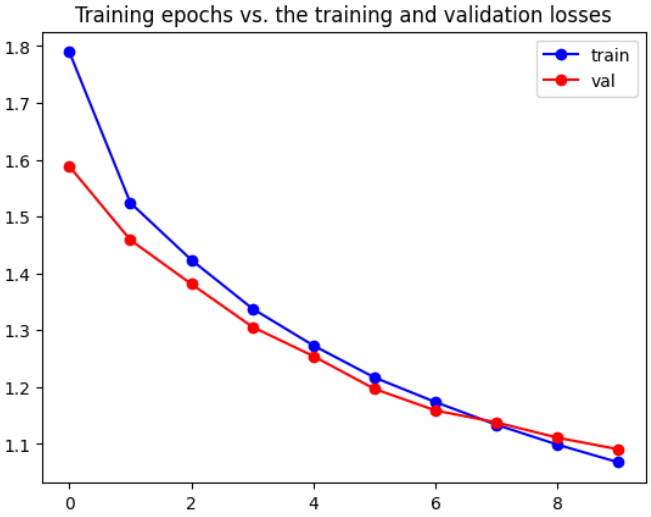




LR = 0.0001

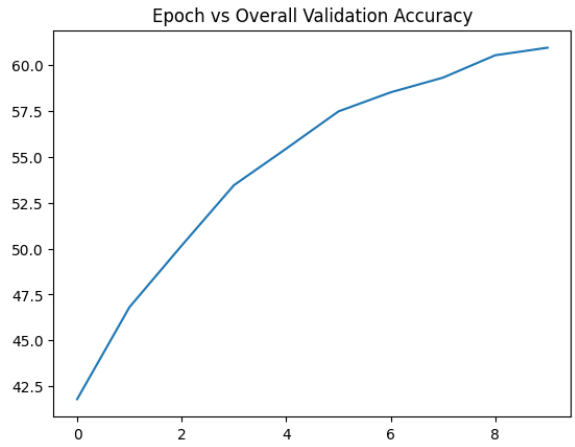
Epochs = 10, Batch = 32, Adam

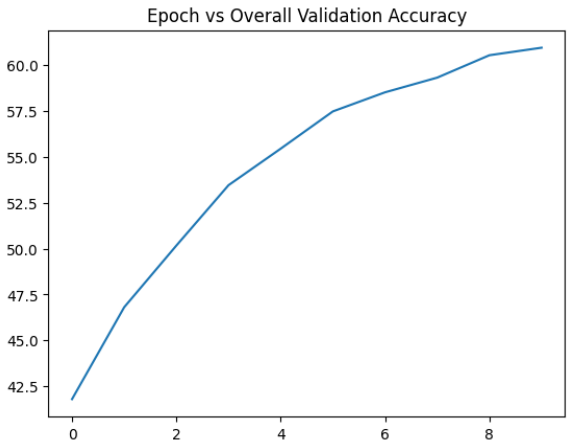
Learning rate is too low for solution to converge sufficiently in 10 epochs.




```
60.95
```

```
Accuracy for class: plane is 58.1 %
Accuracy for class: car   is 67.9 %
Accuracy for class: bird  is 42.8 %
Accuracy for class: cat   is 50.0 %
Accuracy for class: deer  is 43.3 %
Accuracy for class: dog   is 42.5 %
Accuracy for class: frog  is 81.2 %
Accuracy for class: horse is 67.8 %
Accuracy for class: ship  is 85.2 %
Accuracy for class: truck is 70.7 %
```





Analysis

Training and Validation Loss Curves

For LR = 0.001, both training and validation losses decrease consistently, with final values being approximately 0.6 and 0.9 respectively.

For LR = 0.0001 too, both training and validation losses decrease consistently, but drop by much smaller amounts, with final values being around 1.1 approximately for both.

For LR = 0.01, losses increase/decrease randomly. Learning rate is too large to train the model and it tends to overcorrect at each step.

Class-wise Accuracy

For LR = 0.01, the entire model is biased. Hence, this model performs very poorly with 0% accuracy for all other classes (100% accuracy for deer class is not a positive, only shows classification of all inputs solely as deer).

For LR = 0.001 and 0.0001, Class-wise accuracy is better distributed with value ranges being similar among classes. Out of these, LR = 0.001 predicts most classes more accurately.

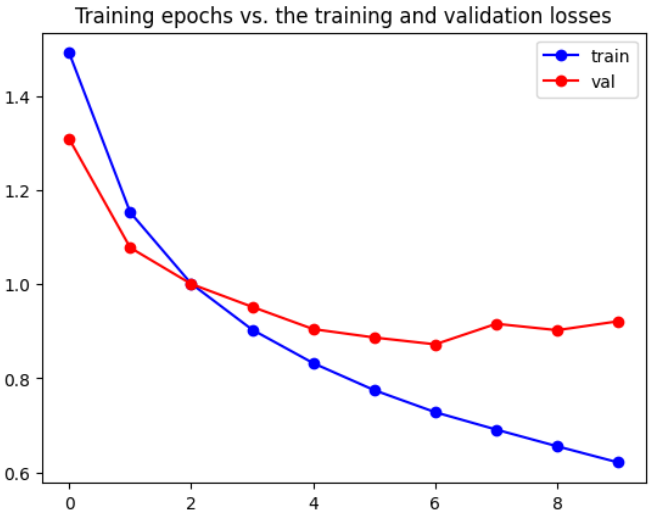
Choice between Batch Size

Batch Size = 32

Refer base case (Pg1)

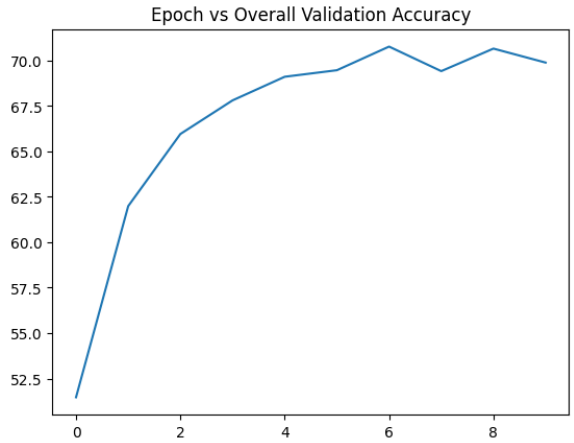
Batch Size = 16

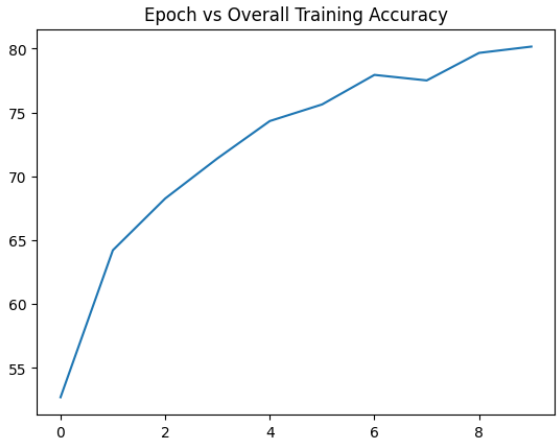
LR = 0.001, Epochs = 10, Adam




```
69.89
```

```
Accuracy for class: plane is 81.4 %
Accuracy for class: car   is 78.1 %
Accuracy for class: bird  is 54.4 %
Accuracy for class: cat   is 48.5 %
Accuracy for class: deer  is 68.9 %
Accuracy for class: dog   is 49.1 %
Accuracy for class: frog  is 79.7 %
Accuracy for class: horse is 76.7 %
Accuracy for class: ship  is 82.4 %
Accuracy for class: truck is 79.7 %
```



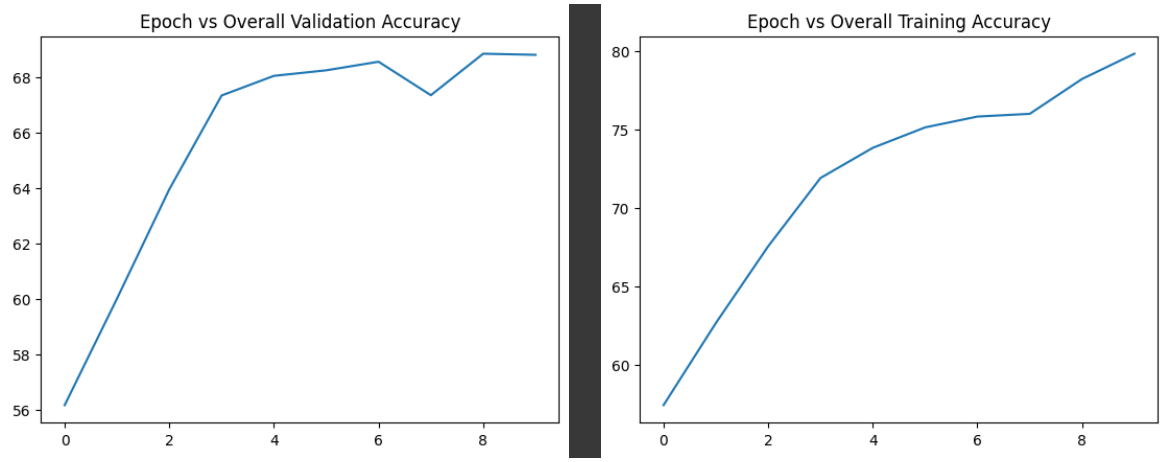


Batch Size = 8

LR = 0.001, Epochs = 10, Adam

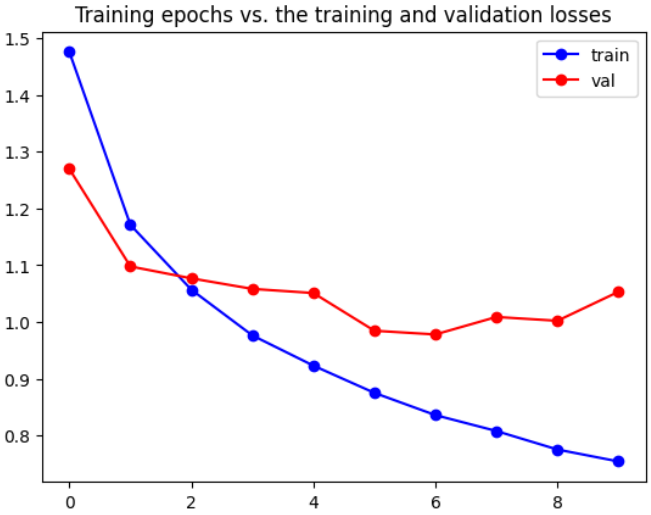

```
68.82
```

```
Accuracy for class: plane is 75.8 %
Accuracy for class: car   is 75.9 %
Accuracy for class: bird  is 49.1 %
Accuracy for class: cat   is 51.0 %
Accuracy for class: deer  is 66.9 %
Accuracy for class: dog   is 58.7 %
Accuracy for class: frog  is 79.9 %
Accuracy for class: horse is 74.9 %
Accuracy for class: ship  is 76.8 %
Accuracy for class: truck is 79.2 %
```



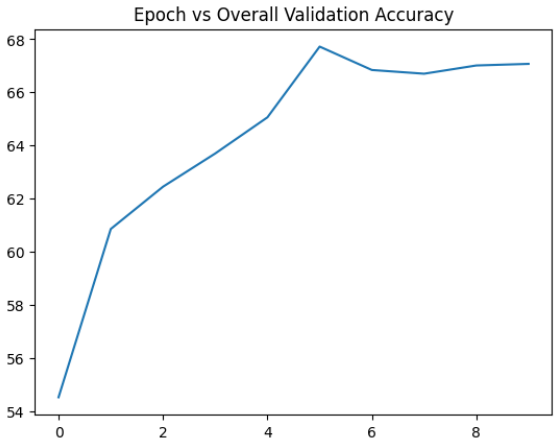
Batch Size = 4

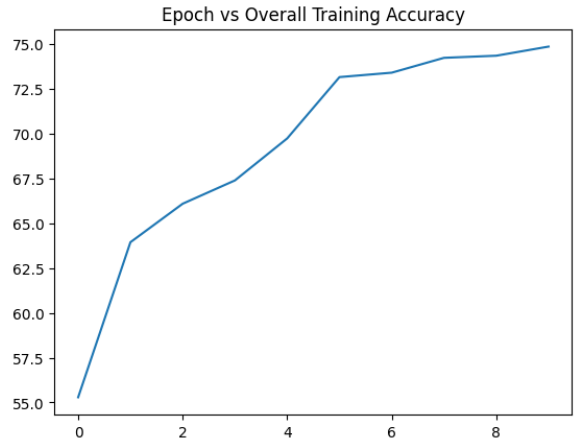
LR = 0.001, Epochs = 10, Adam



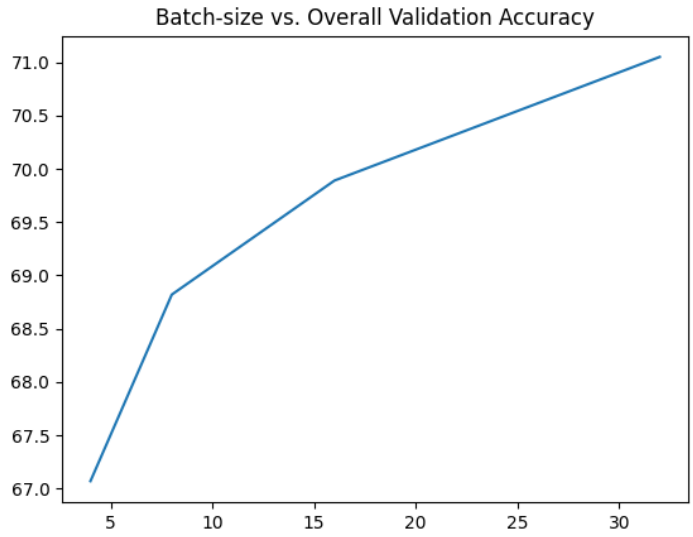

```
67.07
```

```
Accuracy for class: plane is 54.1 %
Accuracy for class: car   is 72.1 %
Accuracy for class: bird  is 48.8 %
Accuracy for class: cat   is 41.0 %
Accuracy for class: deer  is 70.3 %
Accuracy for class: dog   is 61.6 %
Accuracy for class: frog  is 78.3 %
Accuracy for class: horse is 74.6 %
Accuracy for class: ship  is 82.2 %
Accuracy for class: truck is 87.7 %
```





Batch-size vs. Overall Validation Accuracy



Increasing the batch-size also seemed to increase the computation time.

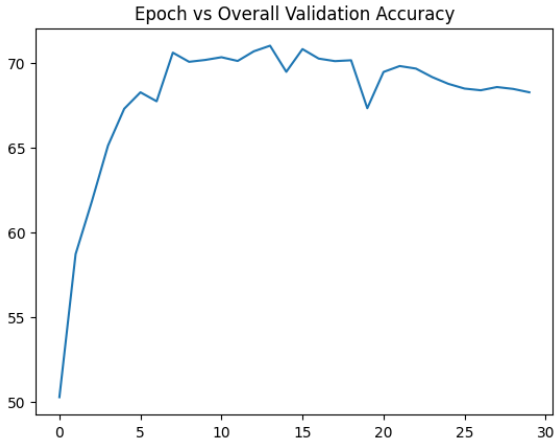
Variance with Epochs

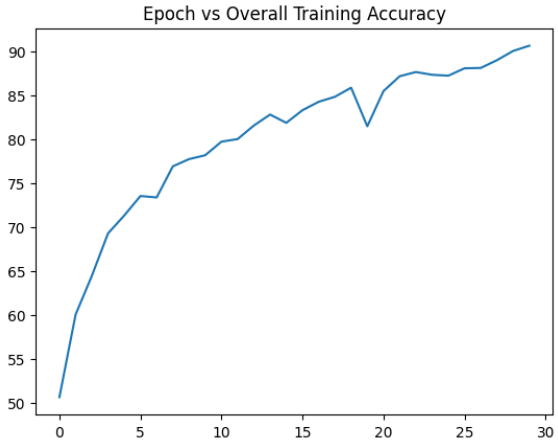
Over 30 Epochs

LR = 0.001, Batch Size = 32, Adam


```
68.27
```

```
Accuracy for class: plane is 70.7 %
Accuracy for class: car   is 81.2 %
Accuracy for class: bird  is 52.4 %
Accuracy for class: cat   is 53.1 %
Accuracy for class: deer  is 66.6 %
Accuracy for class: dog   is 52.1 %
Accuracy for class: frog  is 77.4 %
Accuracy for class: horse is 69.1 %
Accuracy for class: ship  is 78.9 %
Accuracy for class: truck is 81.2 %
```





Analysis

Increasing the number of training epochs will always improve the overall training accuracy. However, this leads to overfitting beyond a point and costs the overall validation accuracy. In the above two graphs, training accuracy increases over almost all of 30 epochs, however, validation accuracy reaches its best around 10 epochs, after which it proceeds to decrease.

Hence, increasing the number of epochs does not always help.

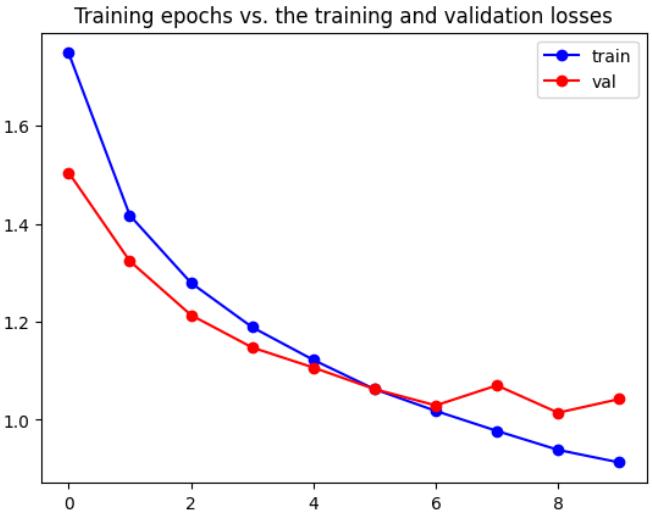
Effect of Data Augmentation

With Augmentation

Refer base case (Pg1)

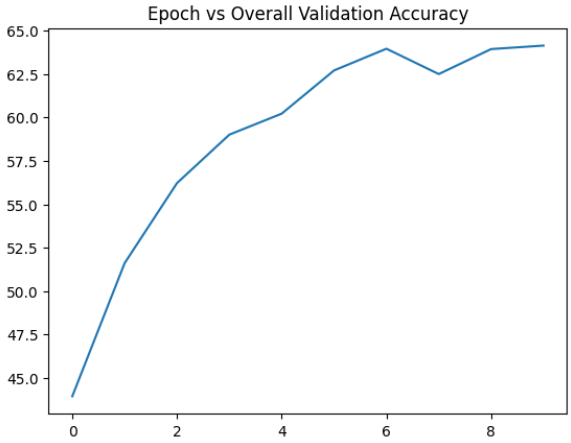
Without Augmentation

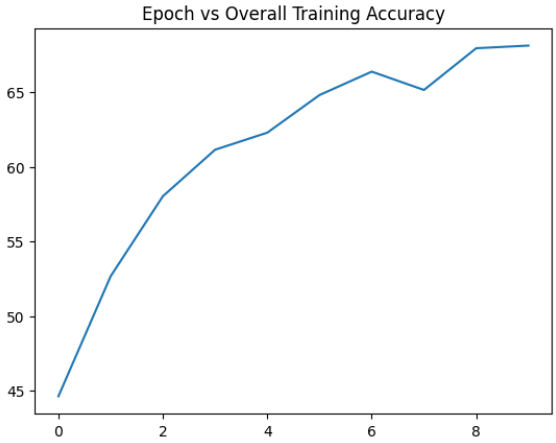
LR = 0.001, Adam, Batch Size = 32, CELoss




```
64.14
```

```
Accuracy for class: plane is 59.4 %
Accuracy for class: car   is 78.0 %
Accuracy for class: bird  is 43.4 %
Accuracy for class: cat   is 45.9 %
Accuracy for class: deer  is 66.4 %
Accuracy for class: dog   is 61.9 %
Accuracy for class: frog  is 76.6 %
Accuracy for class: horse is 75.0 %
Accuracy for class: ship  is 63.0 %
Accuracy for class: truck is 71.8 %
```





Analysis

Both class-wise and overall validation accuracy are significantly reduced. The reason for this is the increase in training loss term. The generalization bound remains similar but the error on the training set has increased and thus error on validation set has also increased.

LR Scheduler

Regular

Refer base case (Pg1)

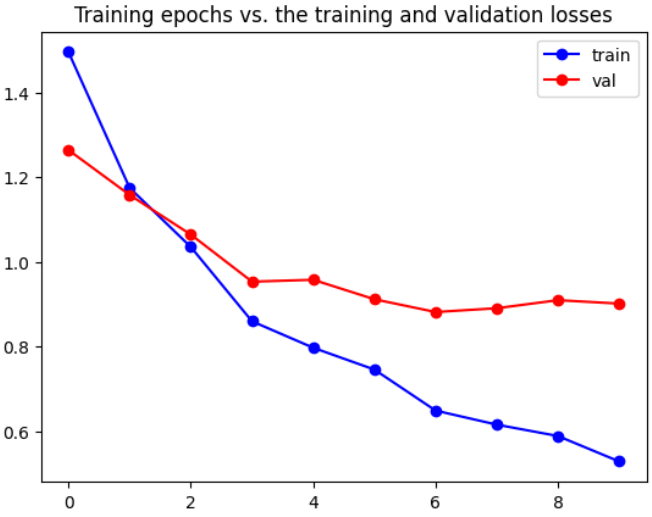
ExponentialLR

Batch Size = 32, Adam, Epochs = 10, Initial LR = 0.01, Gamma = 0.8

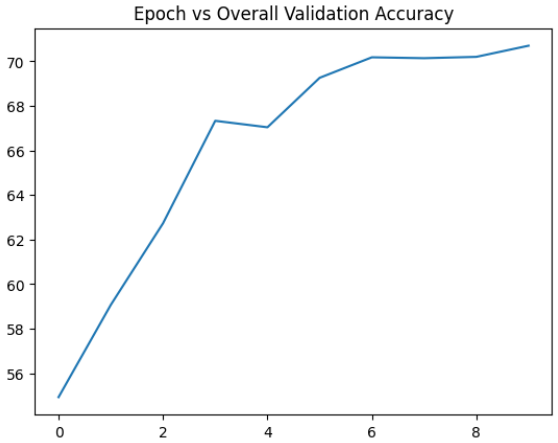
Primary issue faced – LR reduced to quickly, epochs beyond 5 had little to no impact on loss.

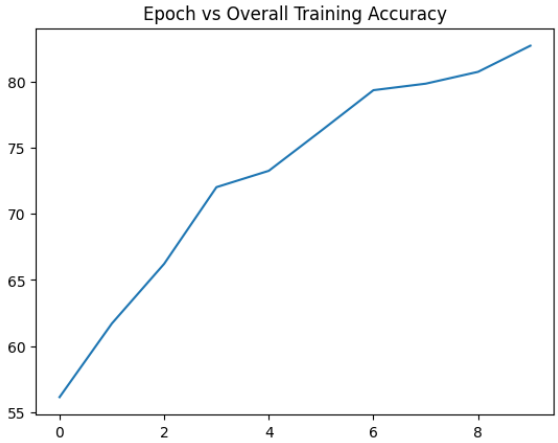
StepLR

Epochs = 10, gamma = 0.5, Initial 0.002, Step Size = 3, Batch Size = 32, Adam




```
70.7
Accuracy for class: plane is 71.2 %
Accuracy for class: car   is 85.1 %
Accuracy for class: bird  is 57.5 %
Accuracy for class: cat   is 46.6 %
Accuracy for class: deer  is 69.3 %
Accuracy for class: dog   is 64.0 %
Accuracy for class: frog  is 80.2 %
Accuracy for class: horse is 74.1 %
Accuracy for class: ship  is 79.5 %
Accuracy for class: truck is 79.5 %
```





Analysis

Show that less tendency to overfit in that case

Although performance with and without LRScheduler was similar for 10 epochs, StepLR usually performs better, with larger learning rates initially helping locate the minima quickly, followed by lower learning rates which make the error even further lower. One thing to note however is that there is a tendency to overfit, i.e. in LRSchedulers training losses are much lesser than an individual LR, but this may not improve the validation accuracy.

Loss Function

Cross-Entropy Loss

Base Case

KL Divergence Loss

Not able to conclude significantly, errors in type-casting or negative losses.

LR = 0.001, Adam, Batch Size = 32, Epochs = 10

NumPy Implementation

Code is not optimised enough to run within time constraints, as such I have submitted the ipynb file but it will not be compiled.

The Layers classes have been debugged and thoroughly verified – please grade accordingly. I have declared the few parts which are not yet implemented correctly.

Not Implemented –

- CELoss
- Adam Optimizer
- Relevant Plots