ANIRUDH SETH
aniset@kth.se

DD2424:Assignment 2

Deep Learning in Data Science

2020-04-08

A 2 layer neural network to classify images from the CIFAR-10 dataset was implemented from scratch.Figure 1 shows the computational graph for the network that was implemented.

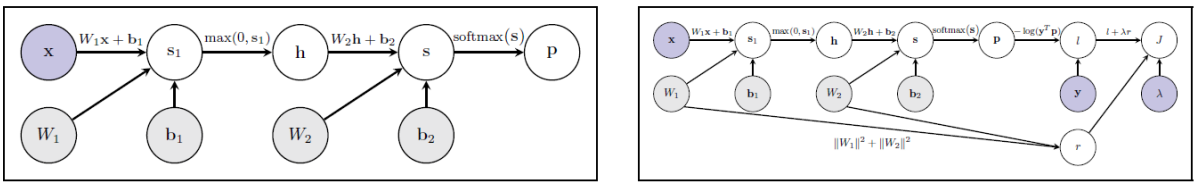

Figure 1: Computational Graph for the 2 layer Neural Network.Classification function(left) and the Cost function(right)

Checking the gradients

The analytical gradients of the weights and the bias for both the hidden layers of the network were compared to the numerical approximations and achieved a mean difference in the range of $\sim$e^{-08} .The results in Table 1 were computed for the first epoch for one batch to validate the gradient computation.The results can be replicated by using seed value of 123 and step size of $h = 1e^{-5}$.

Variable	Gradient Difference(Mean)
W_1	-4.1423e^{-07}
W_2	-2.9207e^{-07}
B_1	-3.9912e^{-08}
B_2	-4.2879e^{-07}

Table 1: Mean gradient difference between analytical and numerical approximations.

Implementing Cyclic Learning Rate

As seen in Assignment 1 , too small/large running rate may result in poor results. In this assignment we implement a cyclic learning rate as [**1**]. The idea is too periodically change the learning rate from η_{min} and η_{max} with a cycle of $2\eta_s$ steps.Figure 2 shows some examples[**2**]. For this assignment we implement the triangular schedule with no delay.

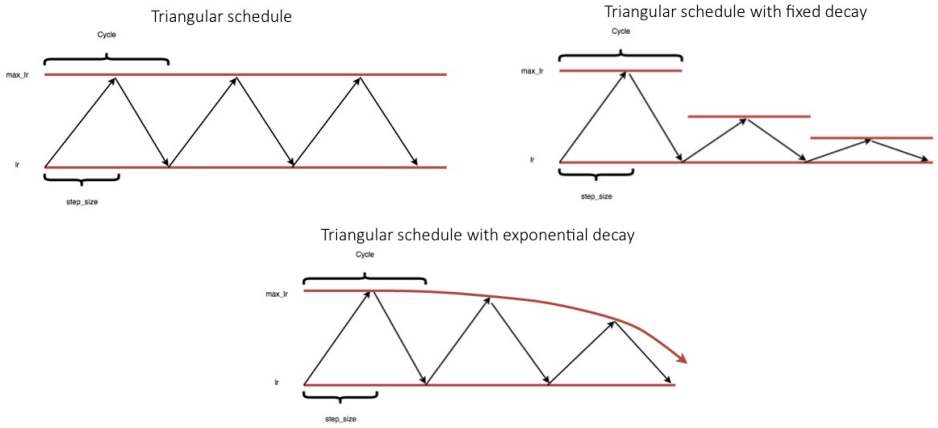

Figure 2: Some examples of Cyclic Learning Rates.

Training with default values for parameters

The results of the training are summarized in Table 2.The loss, cost and accuracy on the training and validation set were computed at each step of training and can be seen in the following plots.The performance of the models are within $\sim 1\%$ from the results in the assignment.

ETA Min = 1e-5, ETA Max = 1e-1, Lamba=.01 , Batch=100		
Accuracy	**Step Size : 500, No of Cycles : 1**	**Step Size : 800, No of Cycles : 3**
Training Set	0.6213	0.7784
Validation Set	0.4597	0.4608
Test Set	0.4602	0.4524

Table 2: Accuracy achieved with with default values of parameters

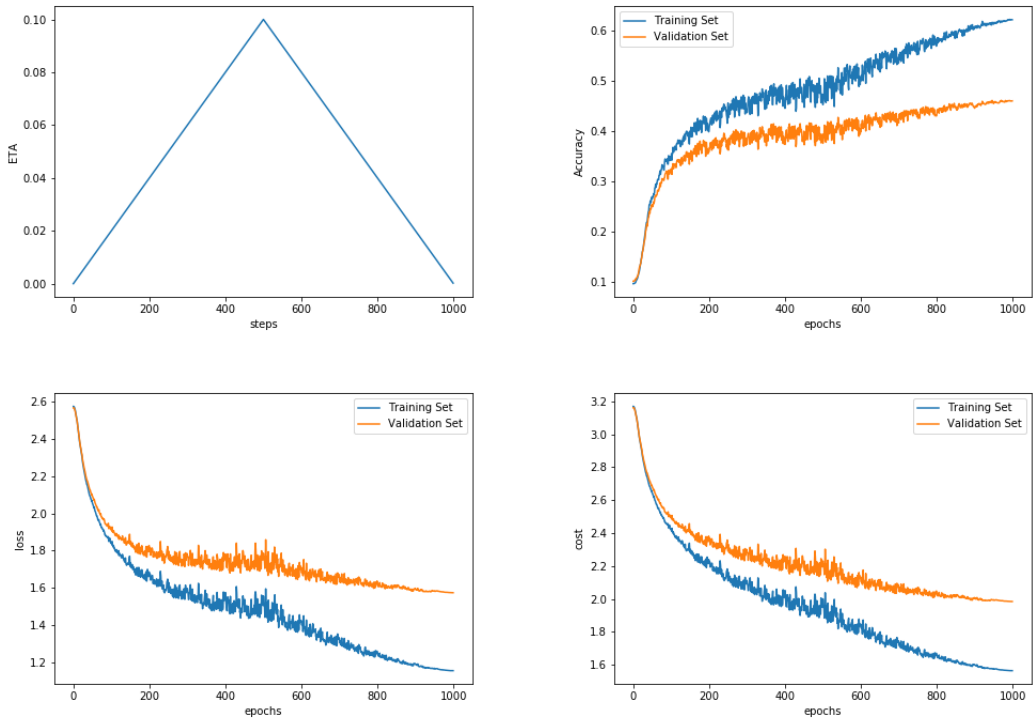

Figure 3: Replicating the results of Figure 3 from assignment.(*Values plotted at each time step)

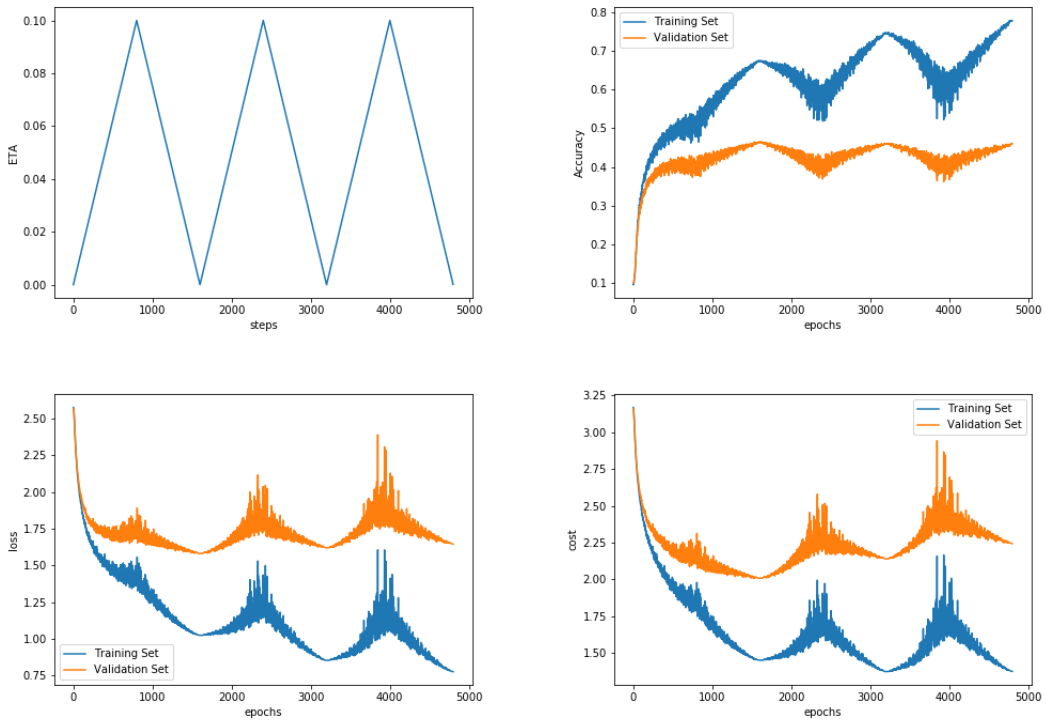

Figure 4: Replicating the results of Figure 4 from assignment.(*Values plotted at each time step)

Hyperparameter Tuning - λ

For the above experiments ,the default value of $\lambda = 0.01$ was used. In this section i used the coarse to fine random search to find the optimal λ.
16 values were sampled from a uniform distribution in a very broad range of values for $\lambda_{min} : 1e^{-5}$ to $\lambda_{min} : 1e^{-1}$. Accuracy on the validation set(size:5000) was calculated during training (size:45000) and the results are reported in Figure 5.Table 3 shows results from top 4 models. The ETA from top two models were then used to do a finer search for the optimal parameter. 8 points were sampled(uniform distribution in the new fine range) and accuracy on validation set was computed. $\lambda : 0.00878554$ gave the highest accuracy.

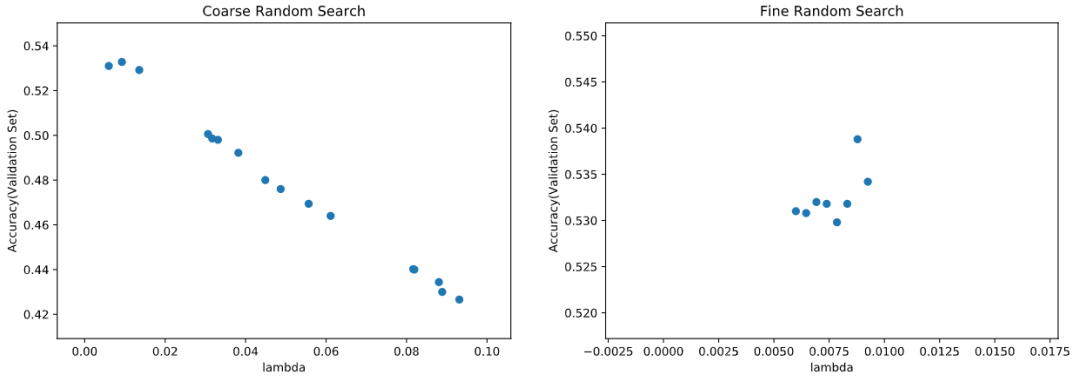

Figure 5: Random search for λ

Training Size :45,000 Validation Size: 5000			
Epoch:80 ETA_Min:1e-5 ETA_Max:1e-1 Ns:1000 Batch:100			
Coarse Search		Fine Search	
ETA	Accuracy (Validation Set)	ETA	Accuracy (Validation Set)
0.00599464	0.531	0.00878554	0.5388
0.00925069	0.5328	0.00925069	0.5342
0.01357111	0.5292	0.00692494	0.532
0.03065967	0.5006	0.00832039	0.5318

Table 3: Model trained with tuned parameters

Training with tuned hyperparameters
The above model was finally trained with the optimized hyper parameters with the same settings as above. The number of epochs was now increased to 100.The results and the plots are reported below.

Training Size :45,000 Validation Size: 5000	
Epoch:100 ETA_Min:1e-5 ETA_Max:1e-1 Ns:1000 Batch:100 Lambda:0.00878554	
Set	Accuracy
Training Set	0.6139
Validation Set	0.5384
Test Set	0.5221

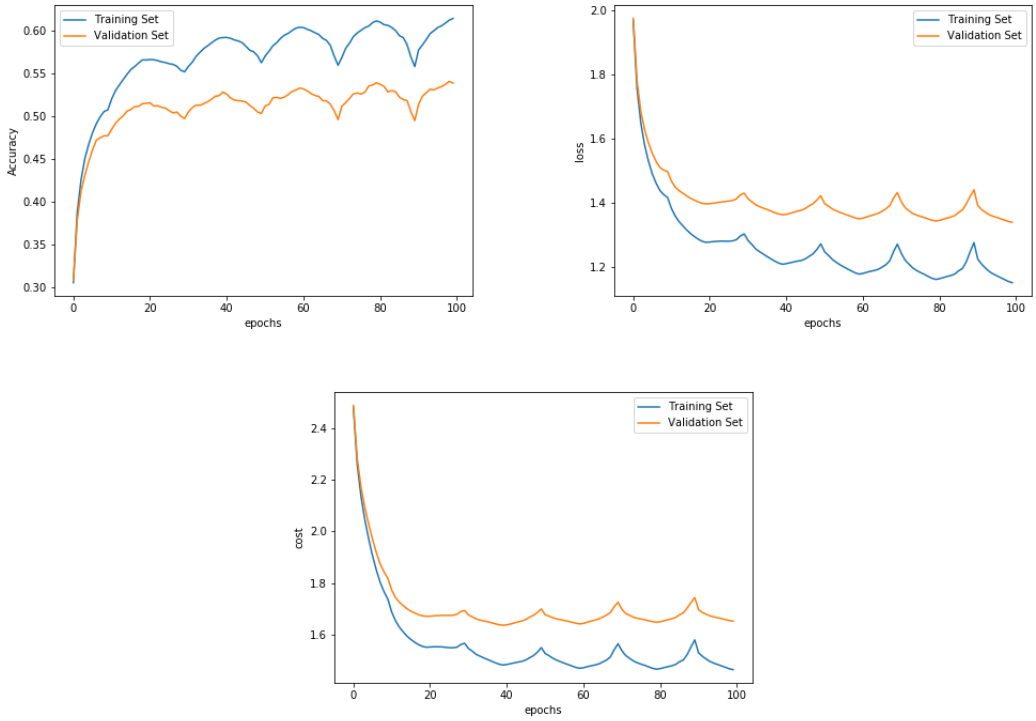

Figure 6: Model Trained with tuned hyperparameters

Results : My model with the tuned hyper parameters achieved a test accuracy of 52.21% which is relative increase of 13.45% from the default values.The curves for the loss, cost and the accuracy seem to follow a trend similar to the number of cycles of ETA performed. The accuracy gets better with more cycles during training.

Bonus Exercise

Implementing LR range test to learn ETA_{min} and ETA_{max}

For the bonus exercise, i use the entire dataset. There are 50000 training samples and 10000 test samples. The LR range test in [**1**] was implemented for two different models , one with lesser number of hidden nodes

(than the one mentioned in the assignment) and the other with more number of hidden nodes. I trained each model for several epochs while letting the learning rate increase linearly between low and high LR values ($1e^{-5}, 1$).The accuracy versus learning rate plots for each model are presented below. The learning rate value when the accuracy starts to increase and when the accuracy slows, becomes ragged were used for the final training. The goal is to achieve similar/better test accuracies after learning the tuned parameters for these new models.

Model 1 : 80 hidden nodes

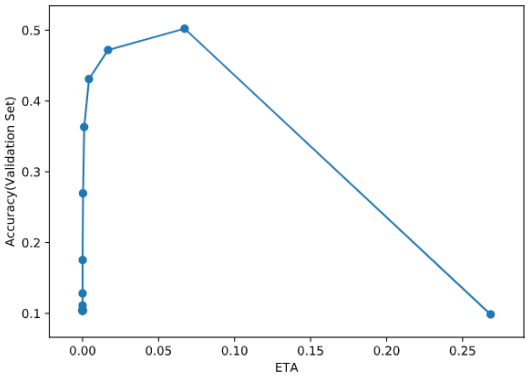

Figure 7: LR Range test (Accuracy vs ETA)

The ETA_{min} was chosen as 0.000262144 which achieved an accuracy of 0.2696 and ETA_{max} was chosen as 0.067108864 which achieved accuracy of 0.5022. The model was then trained for more epochs with the tuned parameters. The final test accuracy of 52.06% .Results are presented below.

Training Size :45,000 Validation Size: 5000	
Epoch: 48 ETA(Cyclic): 0.000262144 - 0.067108864 Lambda: 0.00599464 Batch Size: 100	
Set	**Accuracy**
Training Set	0.6359
Validation Set	0.5286
Test Set	0.5206

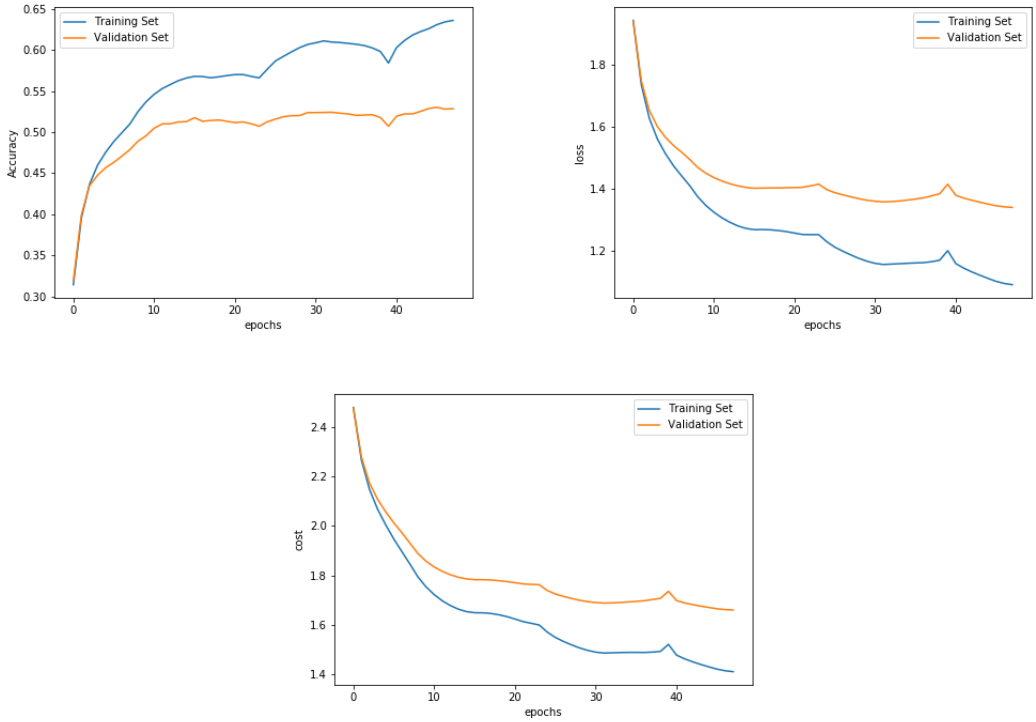

Figure 8: Model 1 Trained with tuned hyperparameters

Model 2 :40 hidden nodes

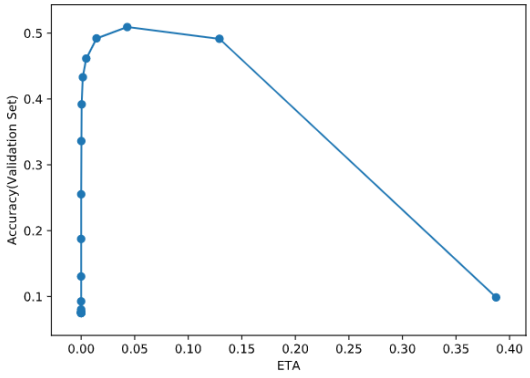

Figure 9: LR Range test (Accuracy vs ETA)

The ETA_{min} was chosen as 0.000262144 which achieved an accuracy of 0.3896 and ETA_{max} was chosen as 0.067108864 which achieved accuracy of 0.49.The model was then trained for more epochs with the tuned parameters. The final test accuracy of 50.69% was achieved.Results are presented below.

Training Size :45,000 Validation Size: 5000	
Epoch: 60 ETA(Cyclic): 0.004782 - 0.12914 Lambda: 0.00599464 Batch Size: 100	
Set	**Accuracy**
Training Set	0.5995
Validation Set	0.523
Test Set	0.5069

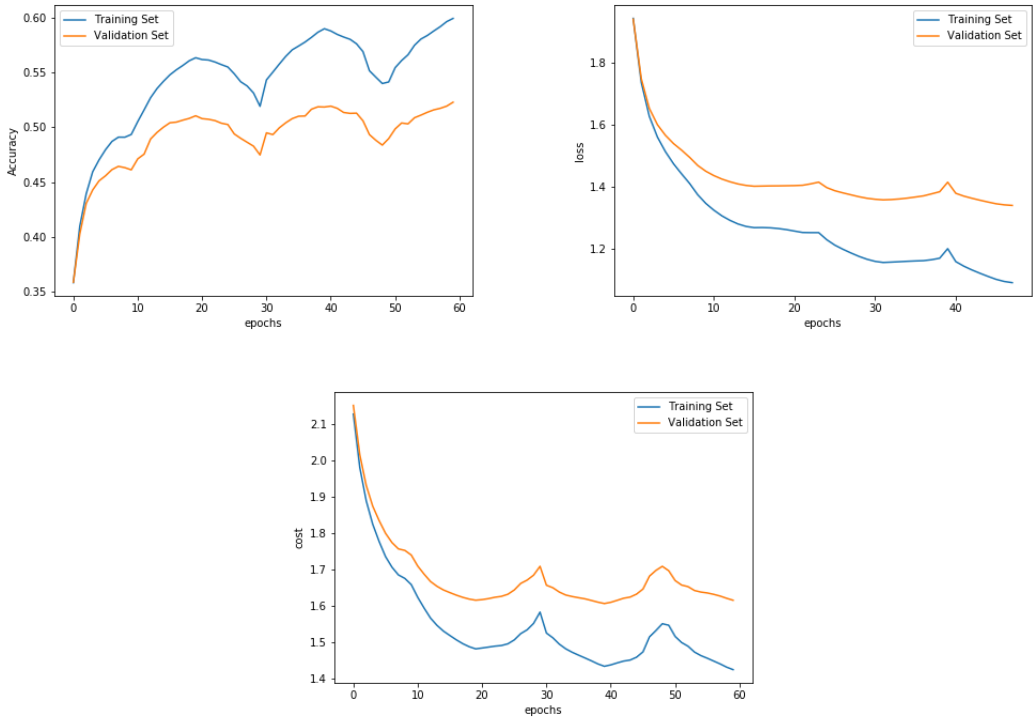

Figure 10: Model 2 Trained with tuned hyperparameters

Performance Optimization

Increasing Training Data and Exhaustive random search for hyper parameters

All 5 batches of the data were used , resulting in training size of 45,000 images , validation size of 5000 images and test size of 10,000 images. Increasing the training size increased the predictive power of the neural network. The optimal settings for the nodes, learning rate range, λ were computed by the methods discussed above.

Random Shuffling

Before each epoch the training examples were randomly shuffled.This is done because permuting the training data gives an unbiased estimate of the true gradient, that could improve the accuracy of the model.

Training Data Augmentation

Each batch dataset was augmented by randomly selecting a subset and addition of white noise (Gaussian).For follwing model, the batch data was augmented by a factor of 10%. A Gaussian noise $\mathcal{N}(0, 0.0001)$ was used to simulate noise/blurriness in the image.

Dropout

For a hidden layer with n nodes, the expectation of the number of neuron to be active at each Dropout is $p * n$, where p is the probability we drop the neuron. These dropped neuron won't propagate anything to the rest of the network.Since we force the network to train with only random p*n of neurons, then intuitively, we force it to learn the data with different kind of neurons subset. This results in better generality of the model.This was implemented along with the re scaling of the active neurons. Upon investigation ,it was observed that dropout often required more number of hidden nodes and epochs to get better results.

Momentum

The update for the weight equals the learning rate times the gradient, plus a momentum factor times the weight delta from the previous iteration. The idea is we do not want to drastically change directions at each time step because some directions could lead to valleys or local minima. Momentum (just like physics) forces

you to go to the direction in which most of its weight gradients push. This accelerates training and can result in better performance. I used $\eta = 0.97$ in below models.

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

weight increment — learning rate — weight gradient

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

momentum factor — weight increment, previous iteration

Figure 11: Back-Propagation Update Without and with Momentum

Combining Improvements

The above mentioned improvements were combined with a random search of parameters that give good results on the validation set.The final model was then trained for more epochs, results from the two best models are summarized in Table 4.

Epoch: 100 ETA(Cyclic): 0.000262144 - 0.067108864 Lambda: 0.00599464 Batch Size: 500 Step: 1000			
	Training Accuracy	**Validation Accuracy**	**Test Accuracy**
Model 1 (Nodes:120)	0.7083	0.5564	0.5534
Model 2 (Nodes:140)	0.7229	0.57	0.5567

Table 4: Good Models after implementing improvements

Below are the plots from the model that obtains the highest Test Accuracy of 55.67% which is a relative increase of $\sim 6\%$ from the model with the provided values for the hyper parameters.The values plotted are plotted after every 100th cycle in the training.

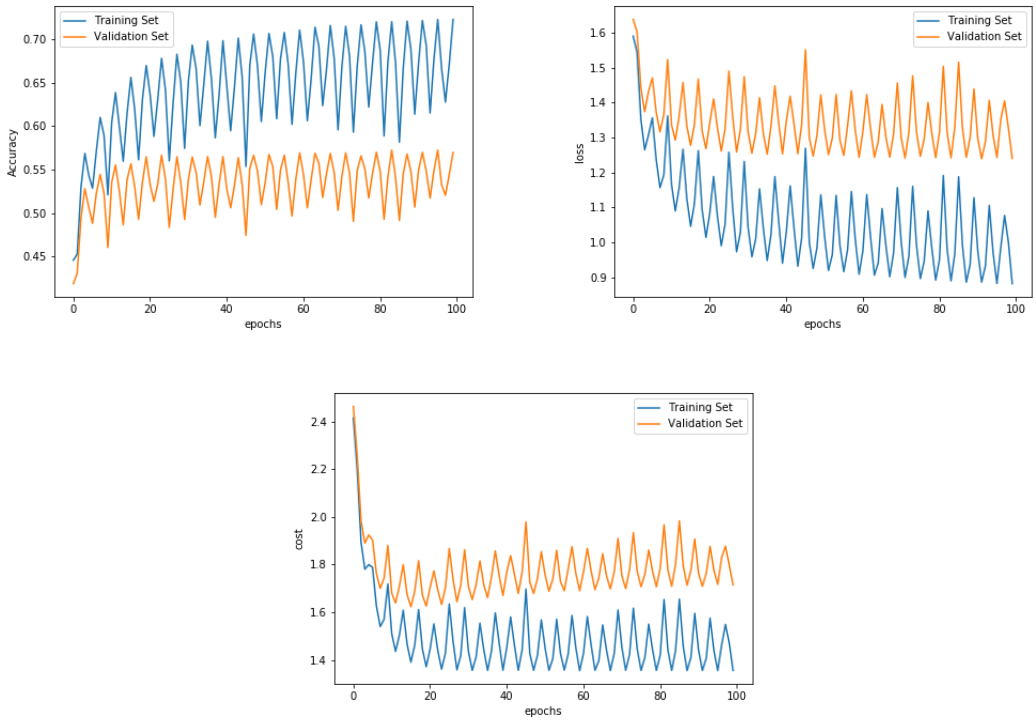

Figure 12: Model trained with improvements*(Values plotted after every 100th cycle)

References

[1] Smith, L. N. (2015). Cyclical learning rates for training neural networks. arXiv:1506.01186 [cs.CV].

[2] https://towardsdatascience.com/adaptive-and-cyclical-learning-rates-using-pytorch-2bf904d18dee

[3] https://visualstudiomagazine.com/articles/2017/08/01/~/media/ECG/visualstudiomagazine/Images/2017/08/0817vsm_McCaffreyFig1.ashx