**Homework 1 – Foundations of Deep Learning**Adi Album & Tomer Epshtein

Question 1:

|  |  |
| --- | --- |
| Method | Test Accuracy |
| Linear SVM | 29.8% |
| RBF SVM | 44.6% |

Question 2:

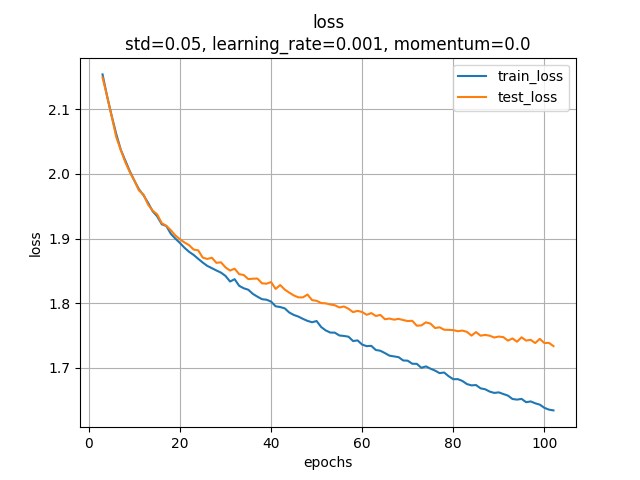
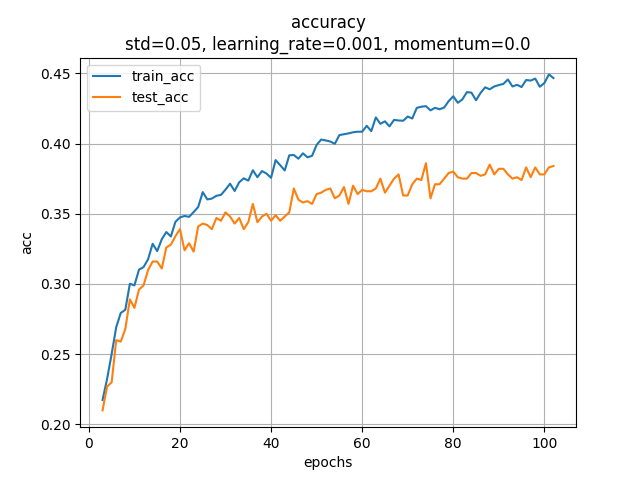
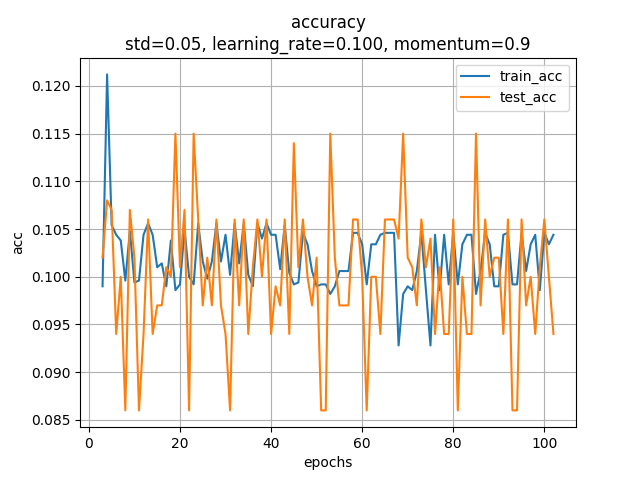
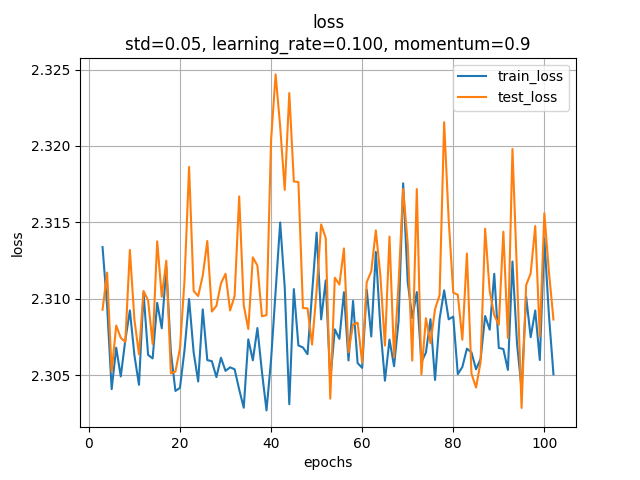
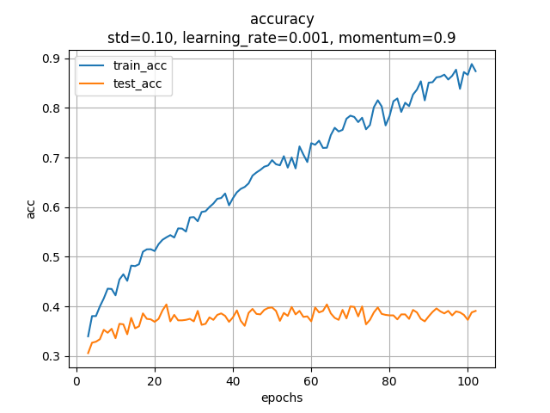
1. **Baseline**

We performed hyperparameter grid-search over the following hyperparameters:

* + std – Standard deviation of random normal weights initialization
  + learning\_rate – Learning rate parameter for SGD optimization
  + momentum – SGD’s momentum parameter

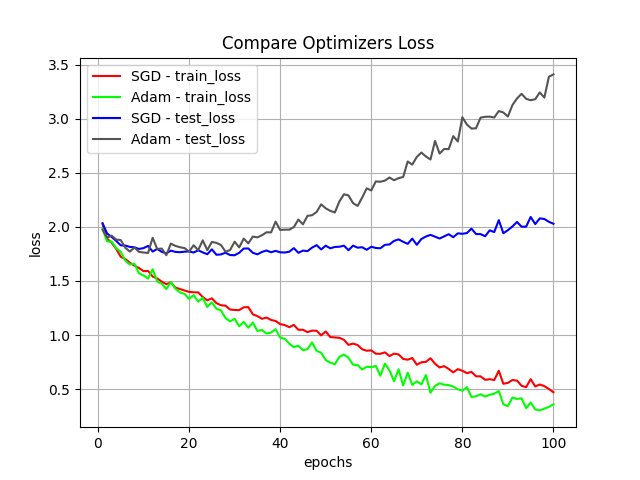
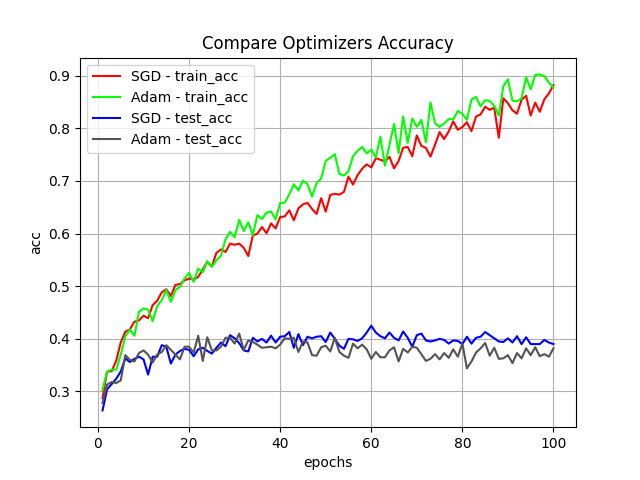
We performed the grid search over:

We ran each of the combinations for 100 epochs. Here are samples of results we obtained:

* 1.    
     Here for example the training loss decreases very slowly. This happens because the optimization parameters (i.e. learning rate and momentum) are too small.
  2.   
     Here on the other hand we see no convergence. This happens because the optimization parameters (i.e. learning rate and momentum) are too large.
  3.    
     This is our selected hyperparameter combination. We have an obvious overfitting of our model on the training data. This is the selected combination because the training loss converges nicely, and with some regularization or increase of dataset this will hopefully lead to better results.  
       
     These are the hyperparameters we used for the rest of our solution.

**(2) Optimization**

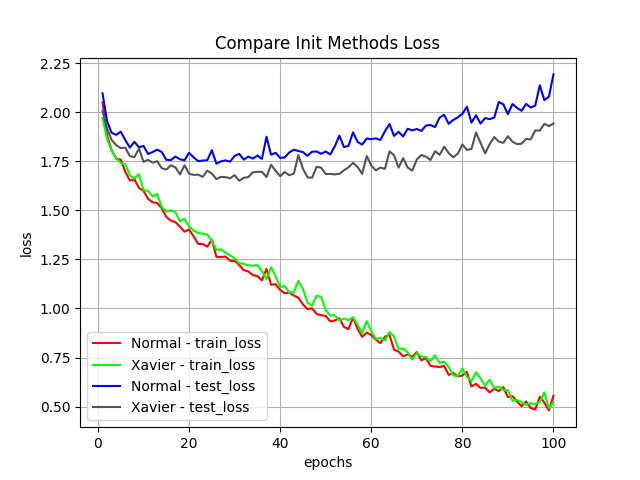
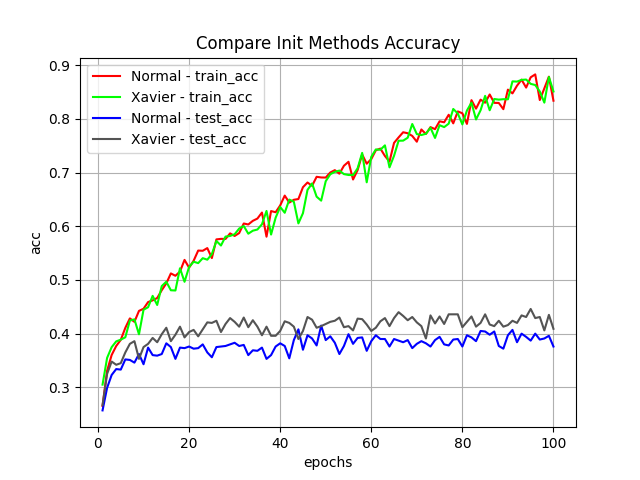
Comparison of SGD Optimization with above hyperparameters and Adam Optimization (without performing hyperparameter grid search).

It seems Adam optimizer obtains a lower train loss but a larger test loss (overfit), with similar convergence time. It is possible that with better hyperparameters for Adam optimization results could improve.

**(3) Initialization**

Comparison of different weight initialization techniques – Random normal with std=0.1 or Uniform Xavier initialization:

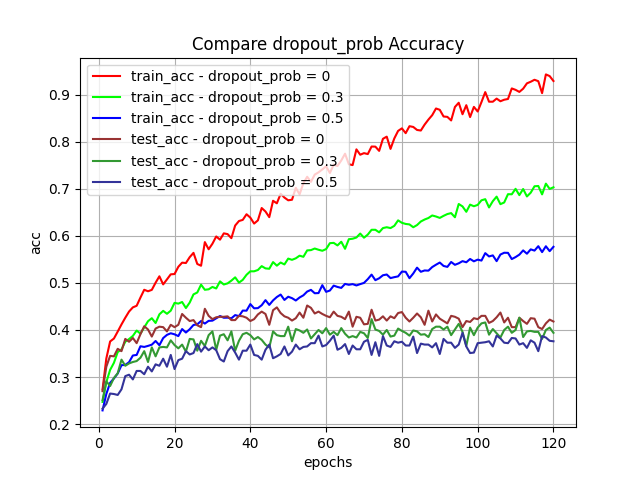
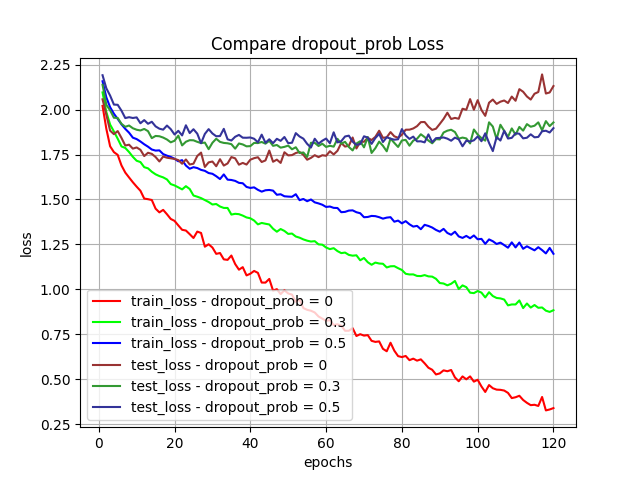
 

Both Normal and Xavier initializations seem to obtain similar train losses and accuracies but the network with Xavier initialization seems to generalize better with both lower test loss and higher test accuracy. Xavier initialization has shorter convergence time.

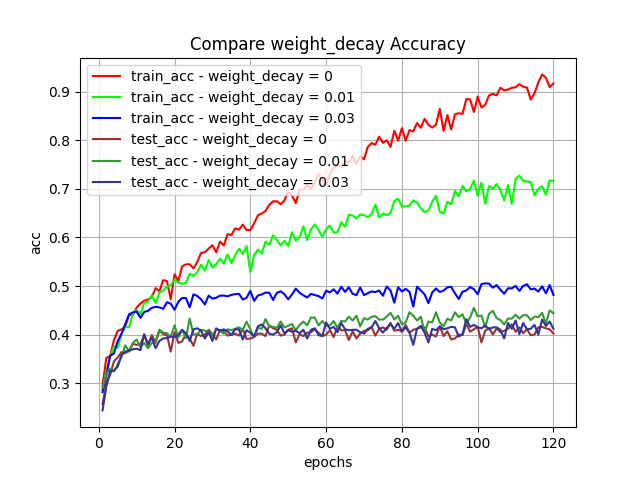
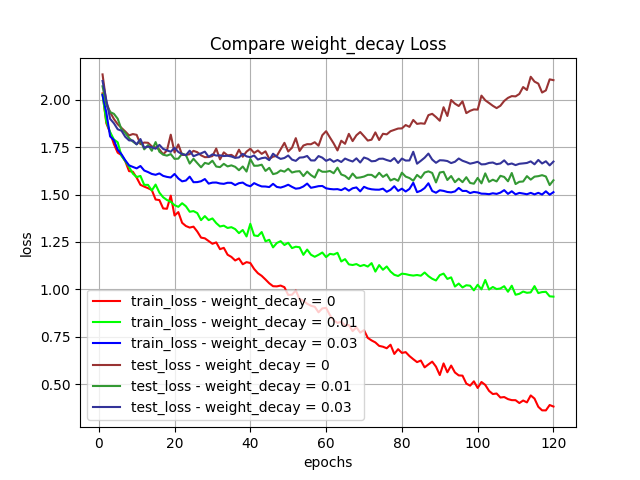
**(4) Regularization**

Comparison of different regularization techniques – None, Weight Decay and Dropout.

Dropout:



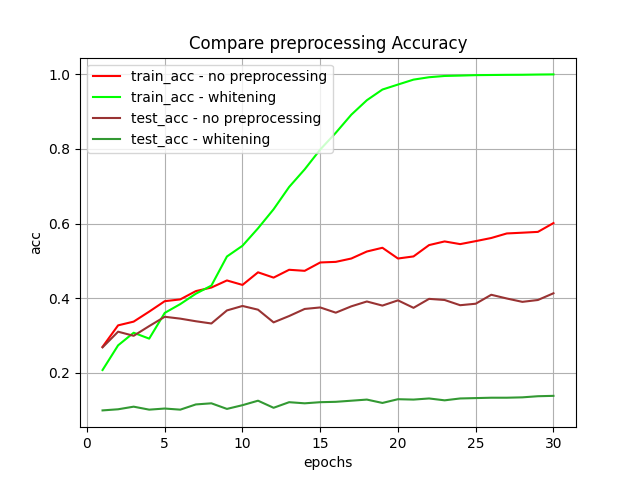
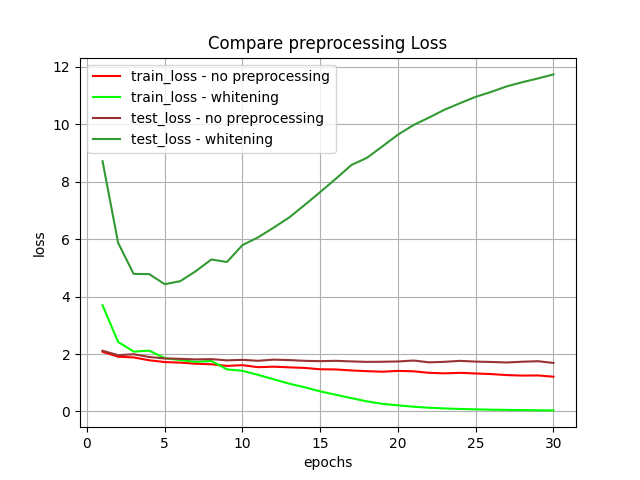
No dropout gave the best results in terms of convergence time and generalization, in terms of train loss and acc, and also test loss and acc. Eventually though no-dropout presented “aggressive” overfitting where with dropout we obtained a more subtle convergence.

Weight Decay:  
  


Here weight decay with parameter gave us the best results for generalization with slower convergence time.

**(5) Preprocessing**

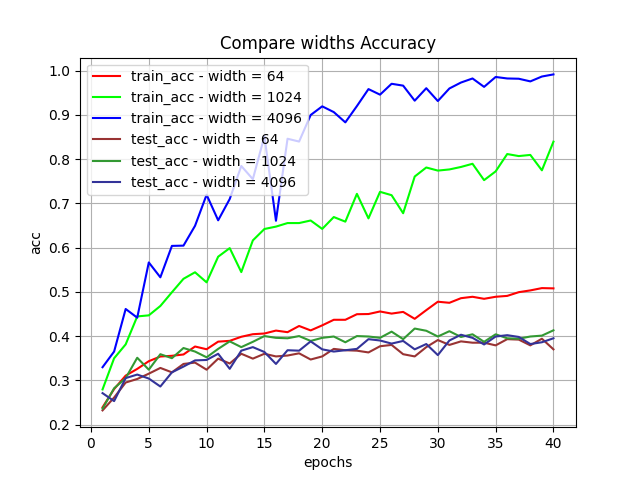
Comparison of different preprocessing techniques – with or without whitening.



This gave us dramatic results: whitening provided us with a speedy convergence on training but with no generalization ability.  
This makes sense as whitening ignores the data representation corresponding to image pixels and transforms every feature so general data will distribute well. This had a huge impact on the networks ability to generalize.

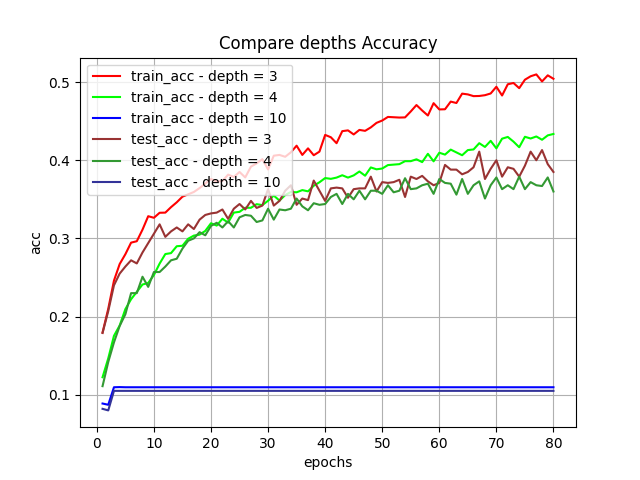
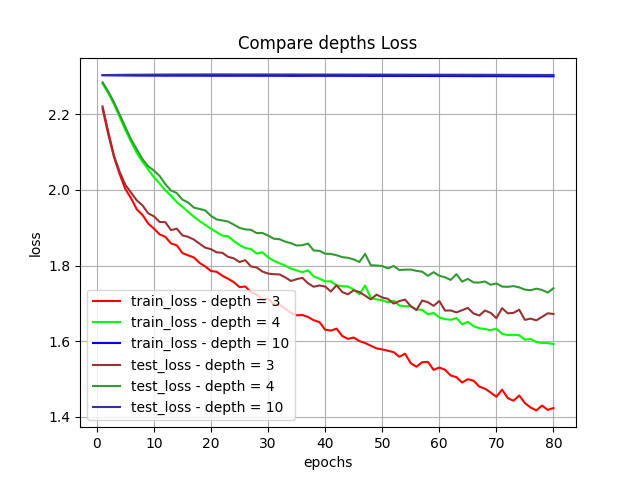
**(6) Network Width**

Comparison of network widths = 64, 1024, 4096.



Here width of 1024 gave best results with fastest convergence time and generalization (test loss and accuracy).

**(7) Network Depth**



With network with depth 10 we saw ‘vanishing gradients’ phenomenon, as the backpropagating gradients were too small the network doesn’t have the ability to train. Networks with depths 3 and 4 obtained similar results with depth 3 giving slightly better generalization results with lower test loss and higher test accuracy.

Question 3:

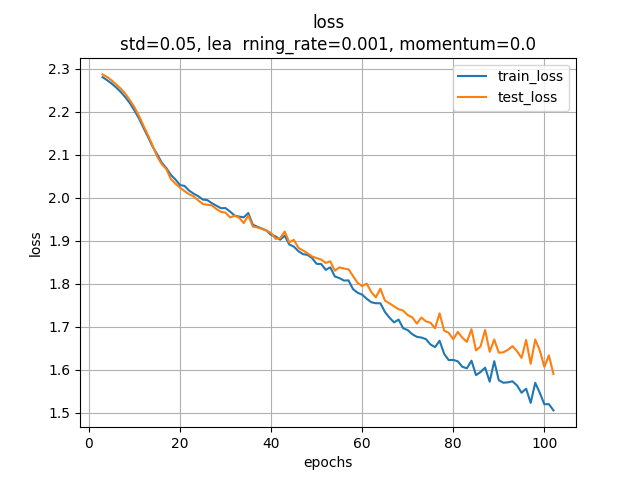
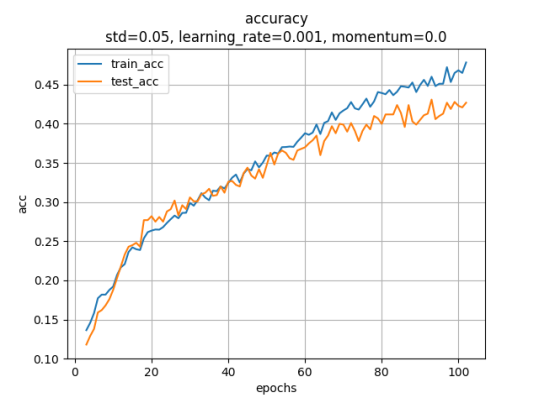
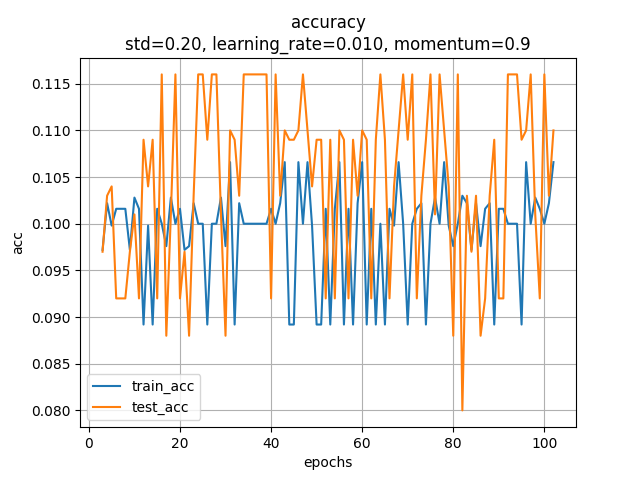
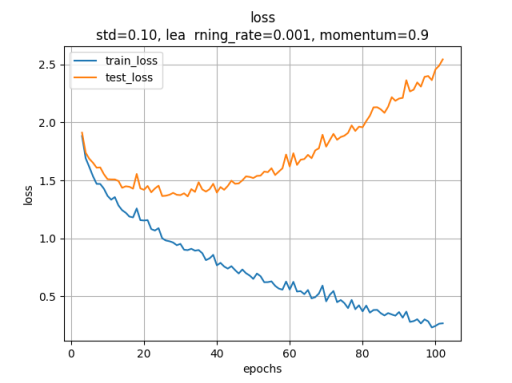
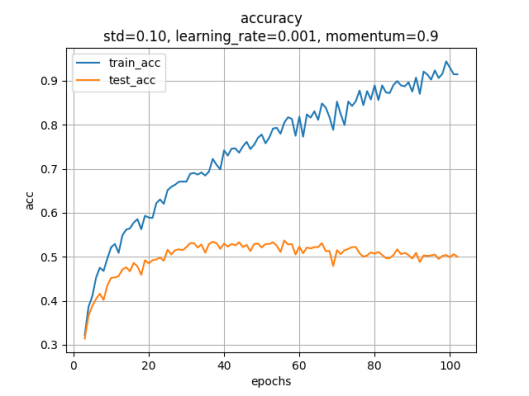
1. **Baseline**

We performed hyperparameter grid-search over the following hyperparameters:

* + std – Standard deviation of random normal weights initialization
  + learning\_rate – Learning rate parameter for SGD optimization
  + momentum – SGD’s momentum parameter

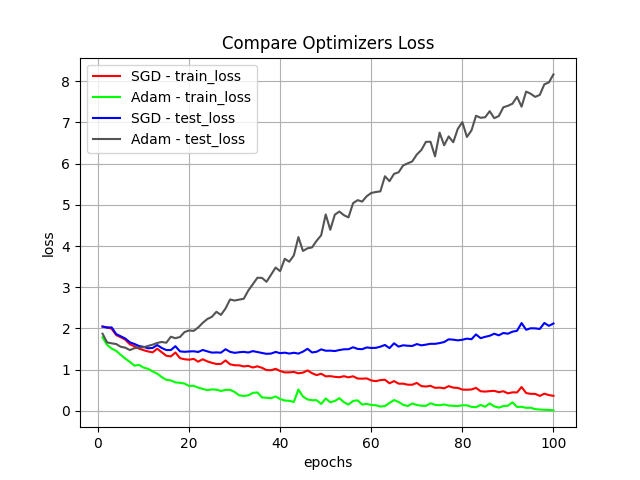
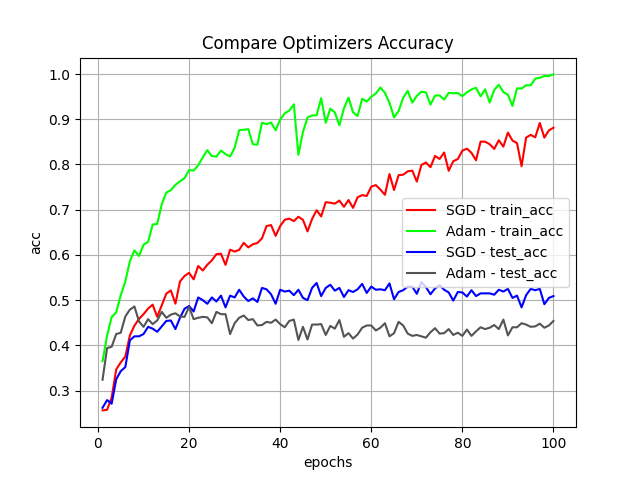
We performed the grid search over:

We ran each of the combinations for 100 epochs. Here are samples of results we obtained:

* 1.    
     Here for example the training loss decreases very slowly. This happens because the optimization parameters (i.e. learning rate and momentum) are too small.
  2.   
     Here on the other hand we see no convergence. This happens because the optimization parameters (i.e. learning rate and momentum) are too large.
  3.    
     This is our selected hyperparameter combination. We have an obvious overfitting of our model on the training data. This is the selected combination because the training loss converges nicely, and with some regularization or increase of dataset this will hopefully lead to better results.  
       
     These are the hyperparameters we used for the rest of our solution.

**(2) Optimization**

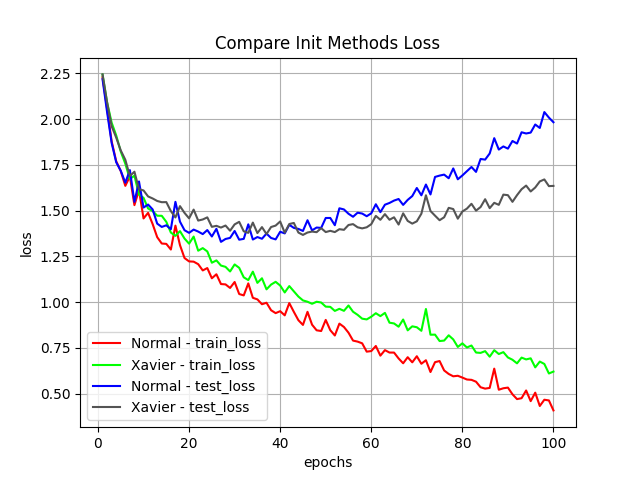
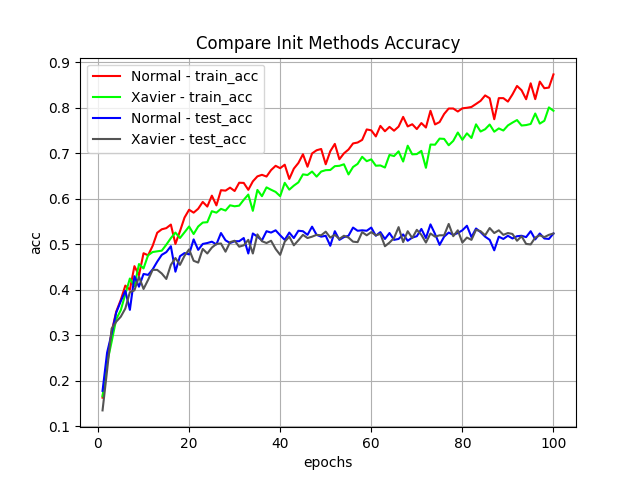
Comparison of SGD Optimization with above hyperparameters and Adam Optimization (without performing hyperparameter grid search).

It seems Adam optimizer obtains a lower train loss but a larger test loss (overfit), with similar convergence time. It is possible that with better hyperparameters for Adam optimization results could improve.

**(3) Initialization**

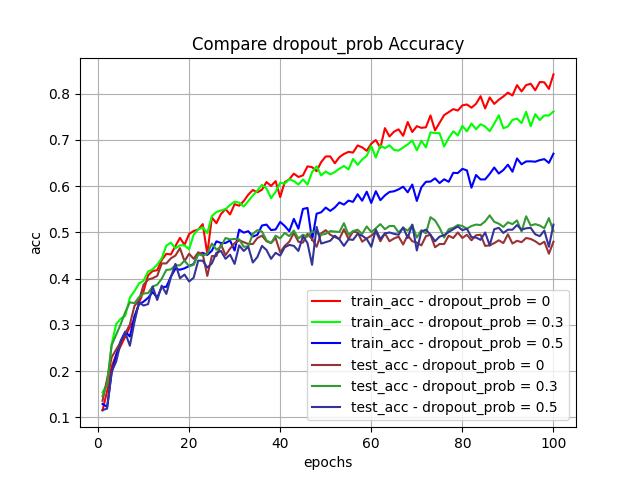
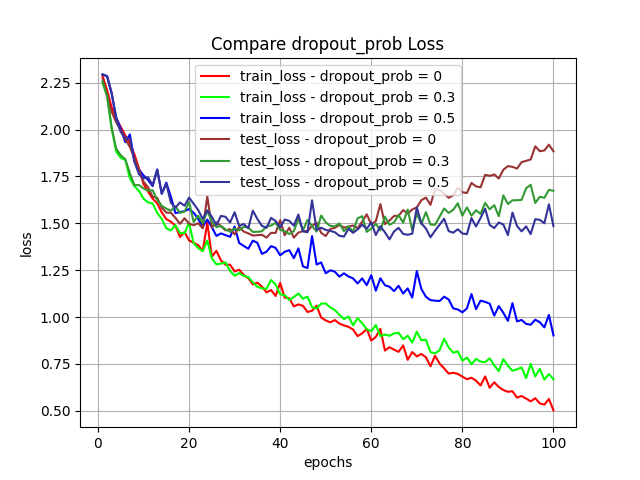
Comparison of different weight initialization techniques – Random normal with std=0.1 or Uniform Xavier initialization:

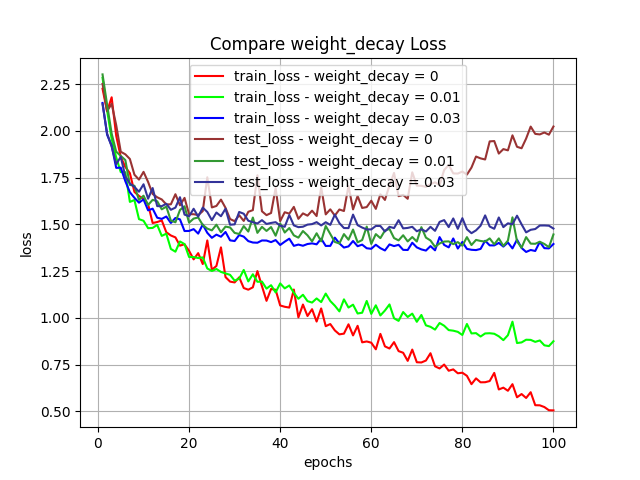
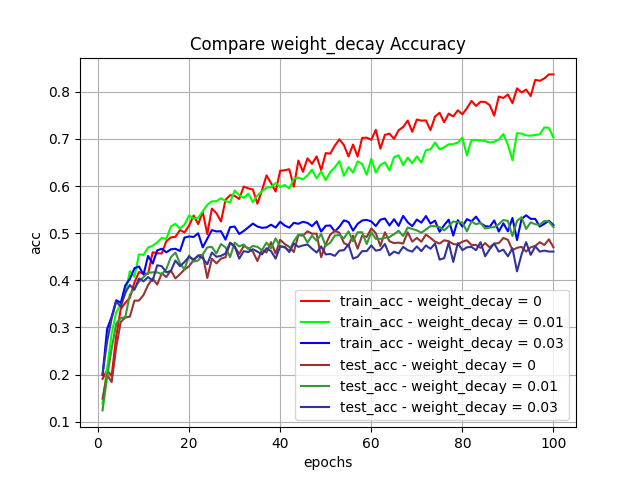
Clearly about training data Normal initialization works better, both the train loss and the train accuracy higher than Xavier initializations, but both Normal and Xavier initializations seem to obtain similar test accuracies. In addition, the network with Xavier initialization seems to generalize better with lower test loss and the same test accuracy. Xavier initialization has shorter convergence time.

**(4) Regularization (**Comparison of different regularization techniques – None, Weight Decay and Dropout)

Dropout:



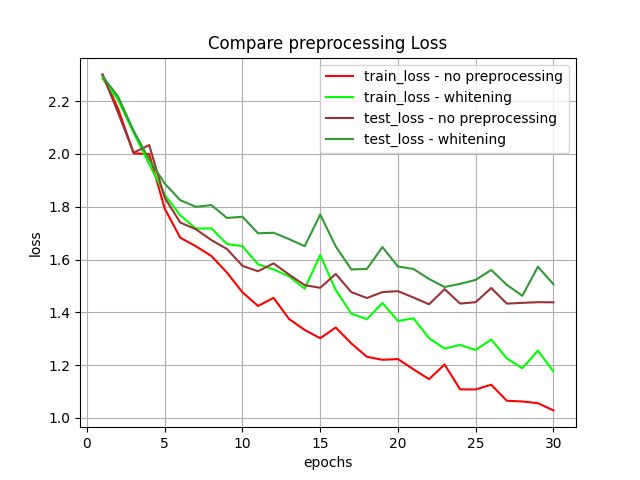
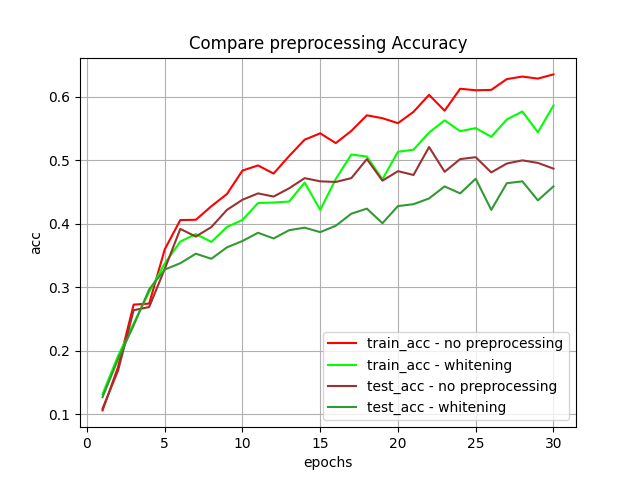
No dropout gave the best results in terms of convergence time and generalization, in terms of train loss and accuracy, and also test loss and accuracy. Eventually though no-dropout presented “aggressive” overfitting where with dropout we obtained a more subtle convergence.

Weight Decay:  
  

Here weight decay with parameter gave us the best results for generalization with slower convergence time (test accuracy with weight\_decay=0.01 seems to converge to the same value of train\_acc with weight\_decay=0.03)

**(5) Preprocessing**

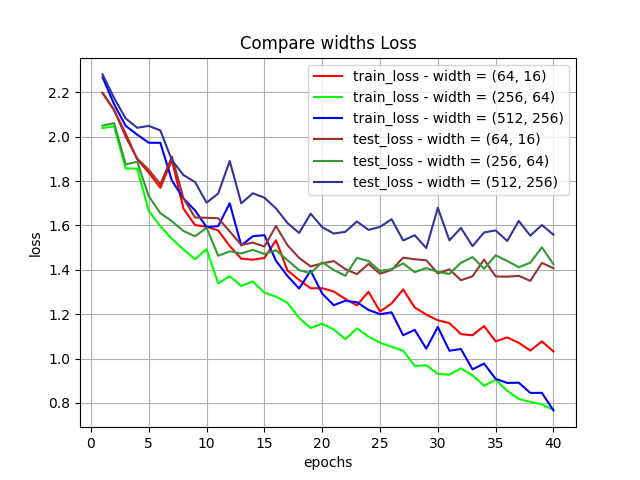
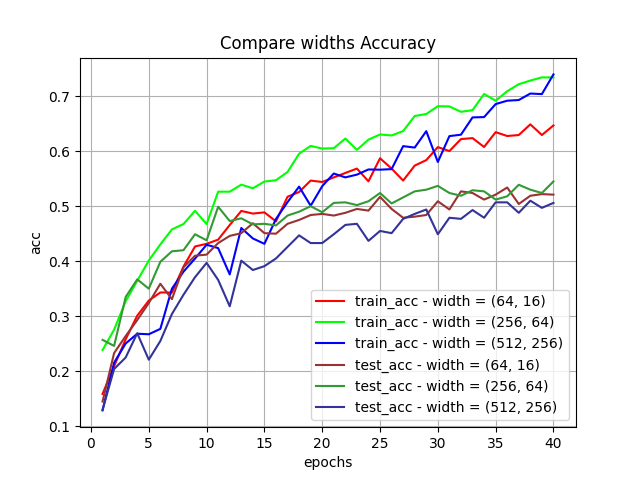
Comparison of different preprocessing techniques – with or without whitening.

Due to these results, we can see that the whitening didn’t help us and both the test\_acc is lower and test\_loss is higher with whitening rather than without whitening. Also the train\_loss converges faster without the whitening and the train\_acc is higher.  
This makes sense as whitening ignores the data representation corresponding to image pixels and transforms every feature so general data will distribute well. This had a huge impact on the networks ability to generalize.

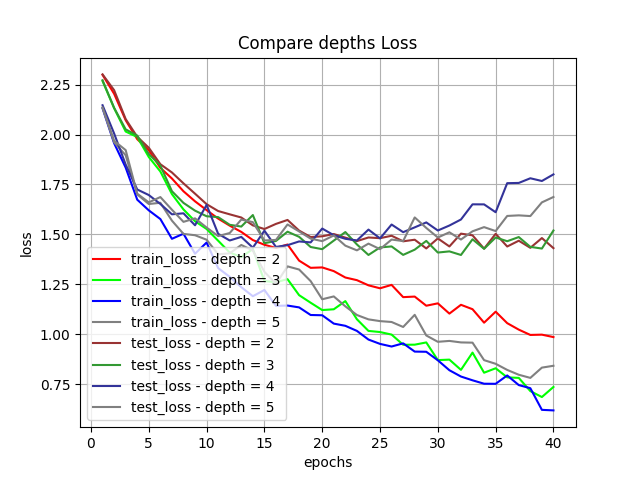
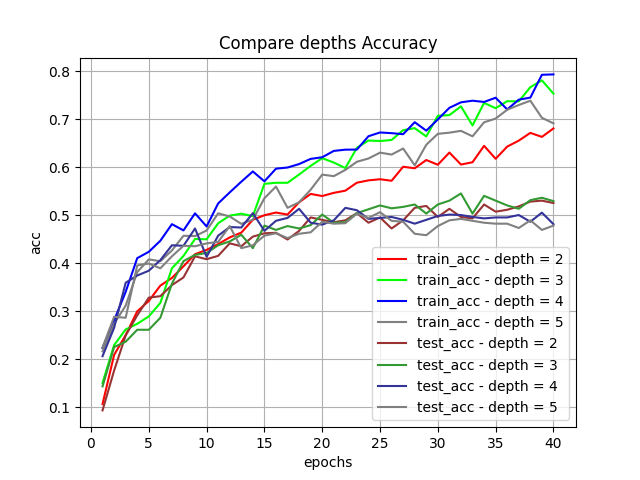
**(6) Network Width**

Comparison of network widths = (64,16), (256, 64), (512,256).

Here width of (256,64) gave best results with fastest convergence time and generalization (test loss and accuracy).

**(7) Network Depth**

We can see that network with depth 2 or 3 have the best results on the test data, with the highest test accuracy and the lowest test loss. In addition on the train data, the network with depth 3 has almost the best train accuracy and train loss (second only to depth 4).  
An explanation for the reason that the results of deeper networks generates worse than the shallower networks is the vanish-gradient problem, maybe that parameters of the network converges faster in deeper network because the gradient is lower and thus the training process not continue and the results converge… with other tools in order to avoid the vanish-gradient problem we would expect to get better results.