Name: Eran Shasha

Id: 315241364

Part 1 sec 2 - SVM:

Linear SVM train accuracy: 0.986 test accuracy: 0.297

Rbf kernel SVM train accuracy: 0.7234 test accuracy: 0.469

Part 2 sec 1 - Baseline:

The grid search was conducted for the following parameters and 50 epochs:

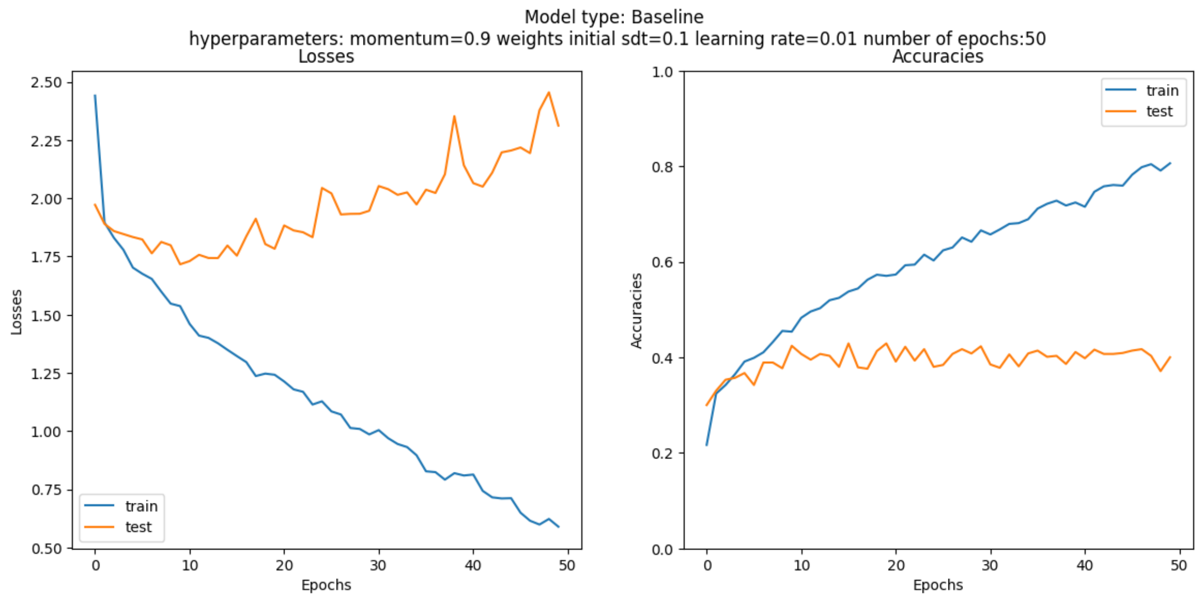
Momentum coefficient: [0.5, 0.9, 0.99]

Weights std: [0.01, 0.1, 1]

Learning rate: [0.001, 0.01, 0.1]

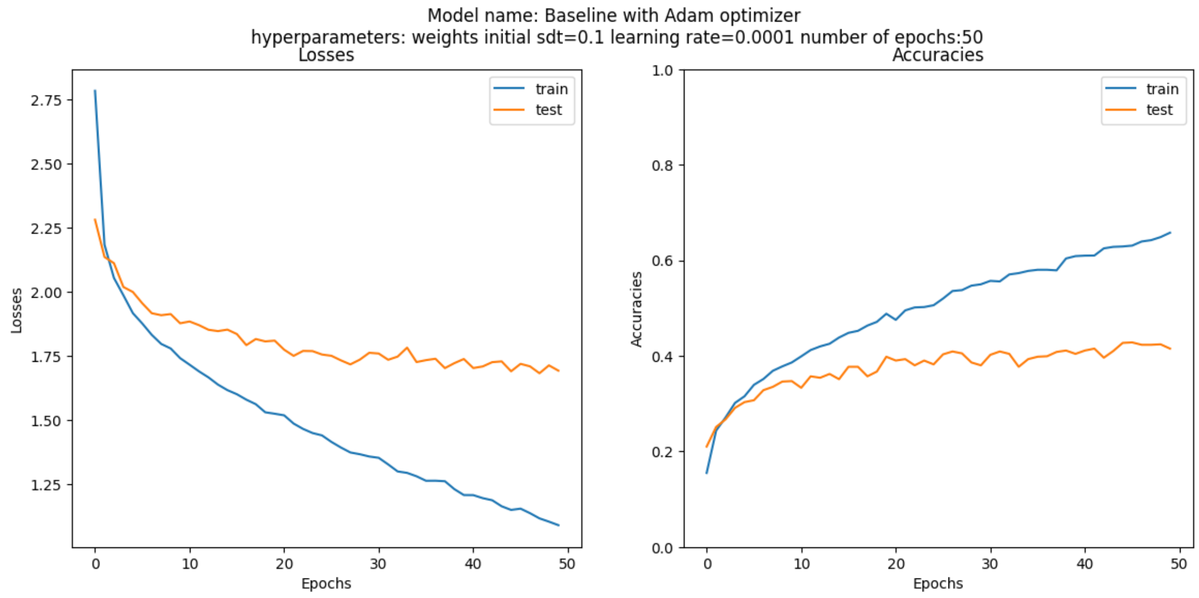
The best hyperparameters found were:

Momentum = 0.9 , learning rate = 0.01, weight std = 0.1



Train loss is: 0.5896 test loss is: 2.3119 train accuracy is: 80% test accuracy is: 40%

Part 2 sec 2 – Adam optimizer:

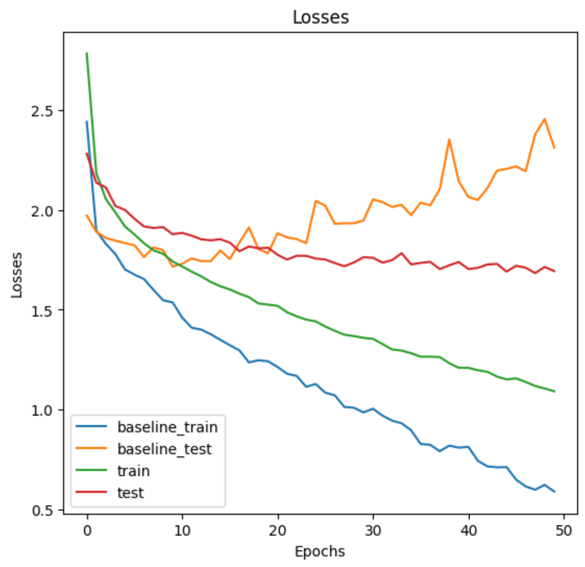


Train loss is: 1.0918 test loss is: 1.6934 train accuracy is: 65% test accuracy is: 41%

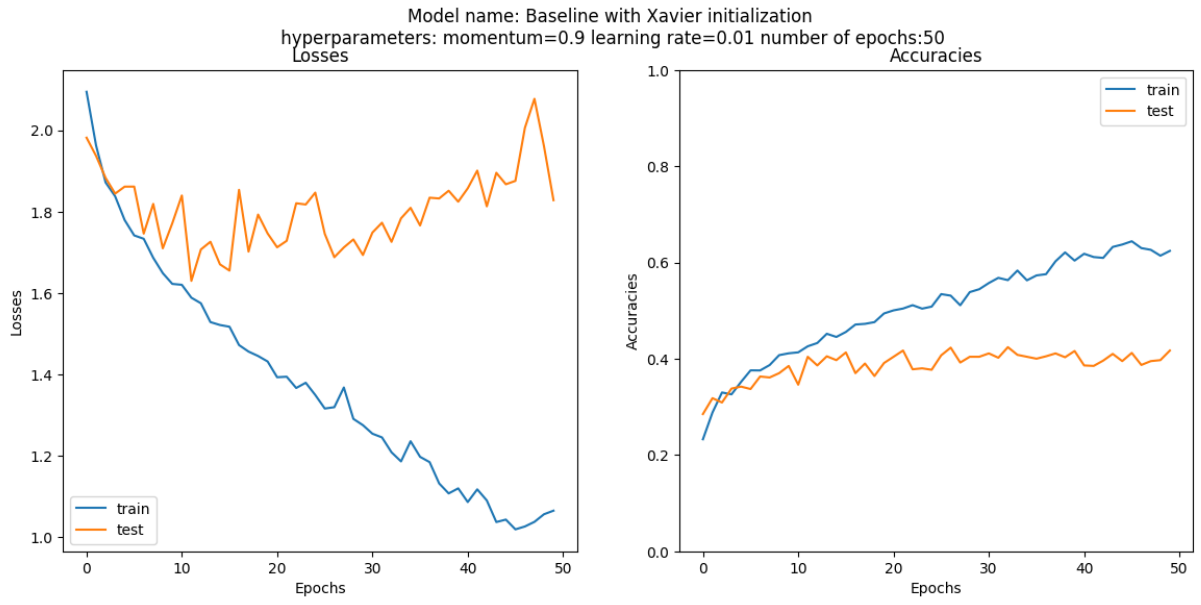
My expectation was that Adam optimizer would result in faster convergence.

Comparing Adam to baseline was difficult however, since Adam optimizer requires much lower learning rate. I decided to use learning rate of 1e-4.

We can see that even though the learning rate was 100 times lower, we still converged at a similar rate (around 20 epochs) and with less overfit.



Part 2 sec 3 – Xavier initialization:

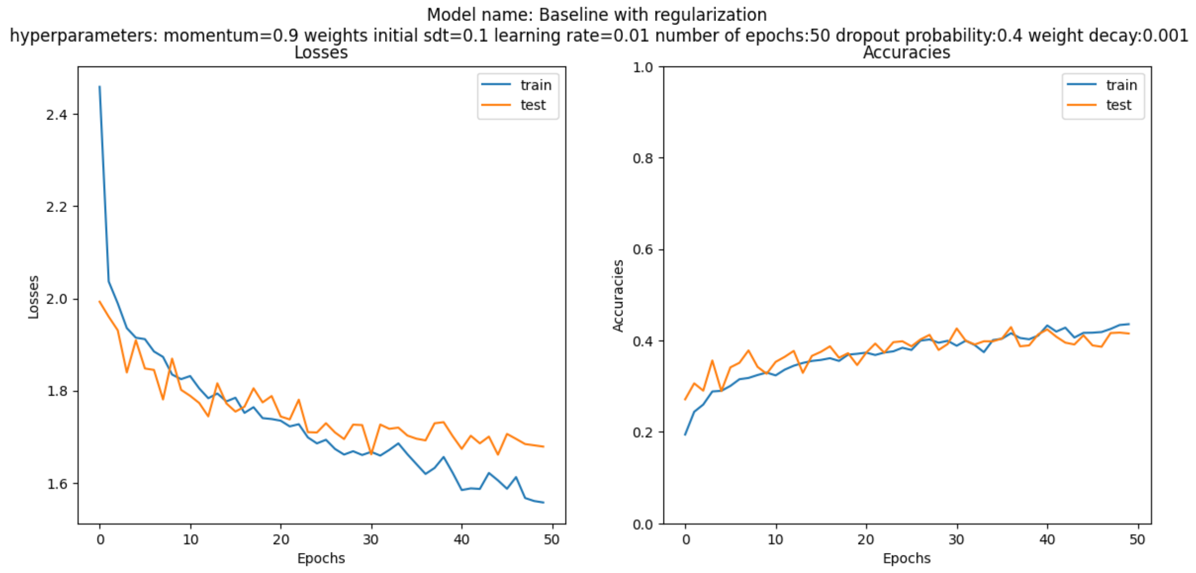


Train loss is: 1.0655 test loss is: 1.8282 train accuracy is: 62% test accuracy is: 41%

My expectation was that Xavier will converge faster at the beginning since the weights are initialized with respect to the size of the input.

In the end it did not seem to make much difference. I did see less overfit which could be regarded to a lower initialized weights values.

Part 2 sec 4 - Regularization:



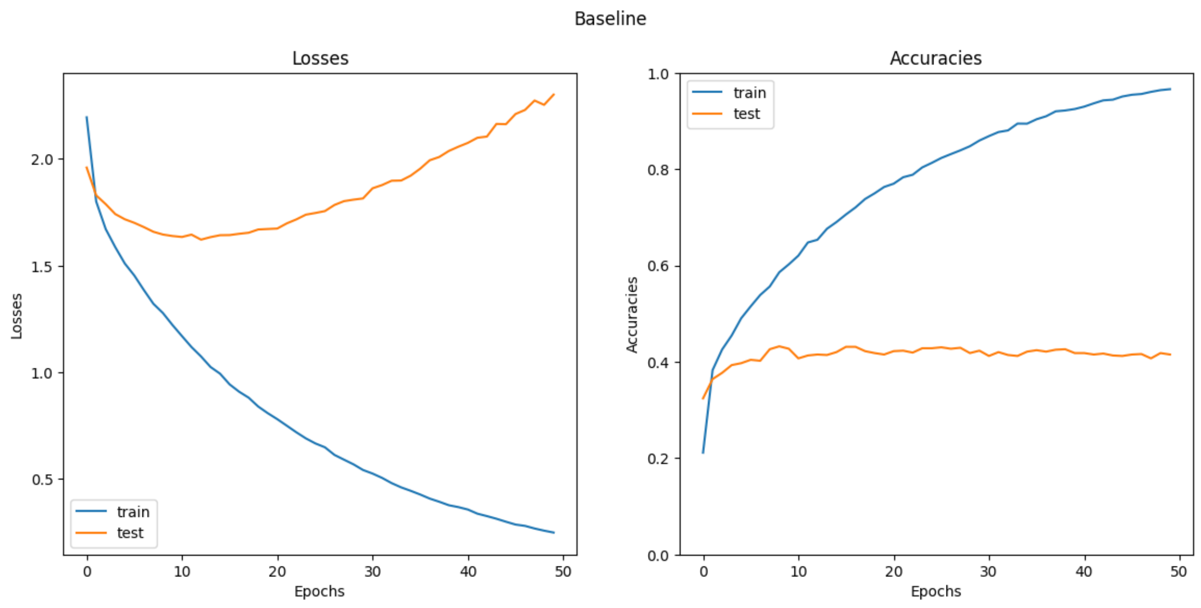
Train loss is: 1.5573 test loss is: 1.6785 train accuracy is: 43% test accuracy is: 41%

My expectation was to see much less overfit and a slightly better performance on the test.

I managed to get no overfit at all, while maintaining the same accuracy.

Runtime was a bit longer, probably due to the extra dropout layers.

Part 2 sec 5 - Preprocessing:



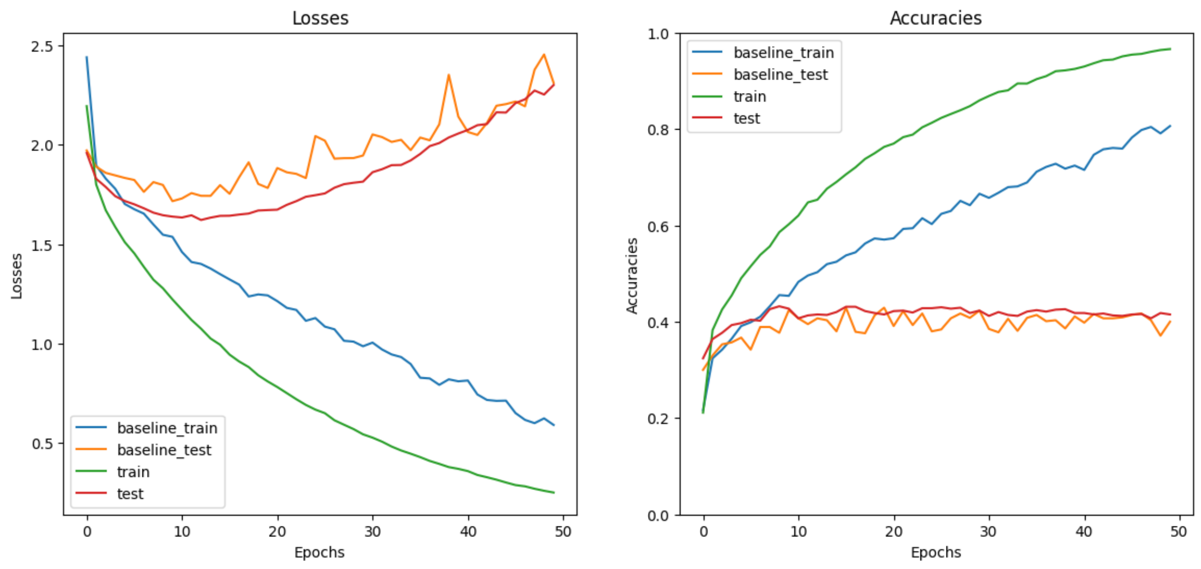
Train loss is: 0.2495 test loss is: 2.3011 train accuracy is: 96% test accuracy is: 41%

I used PCA to find the number of most significant dimensions which contained 85% of the information in data which was 52 dimensions. I then processed the data by reducing the number of dimensions and whitening.

My expectation was to see:

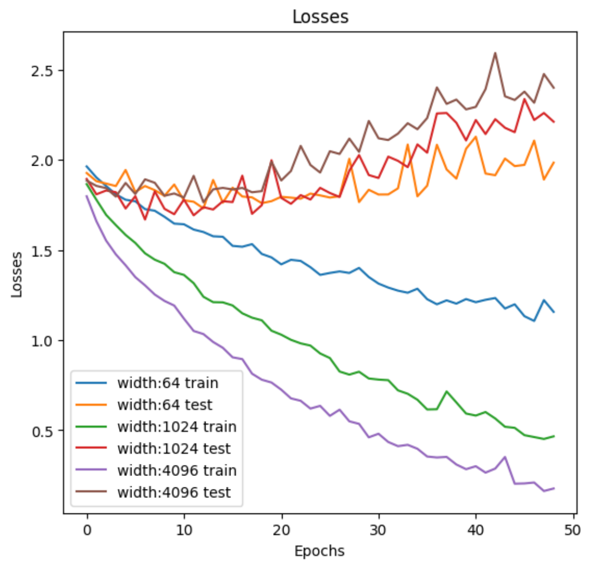
1. Much lower runtime due to the dimension reduction.
2. Better performance on test as we remove some of the noise from the data by reducing the dimensions. We also should see some improvement due to whitening.
3. Higher overfit as the ratio between the size of the data and the number of parameters in the network changed drastically.

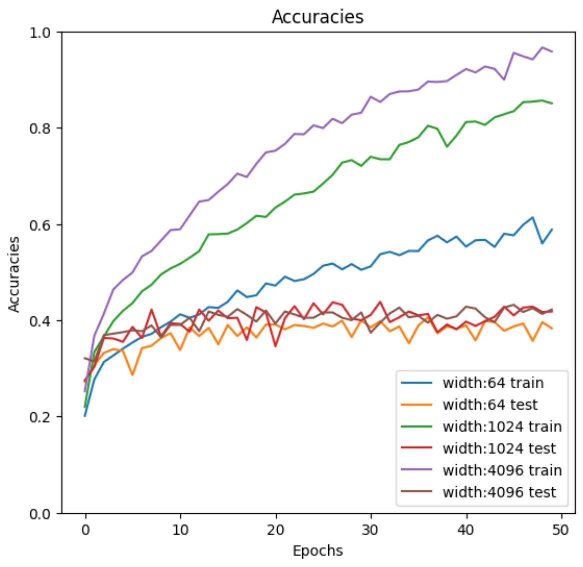
In this case I ended up seeing all three:



(Can see lower test loss, higher overfit, and runtime was nearly 5 times faster)

Part 2 sec 6 – Network width:



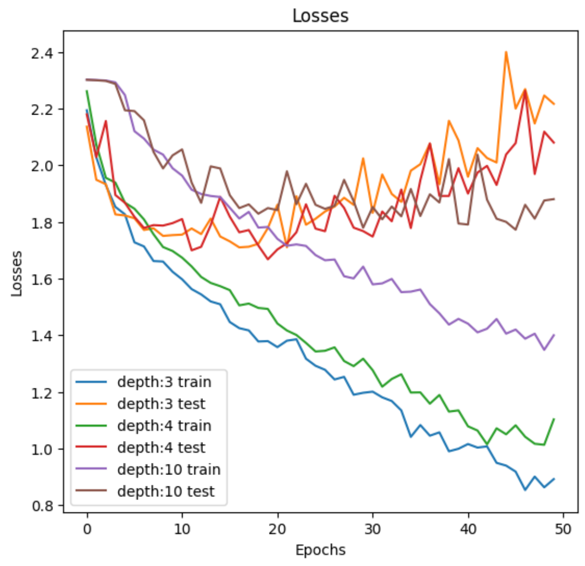


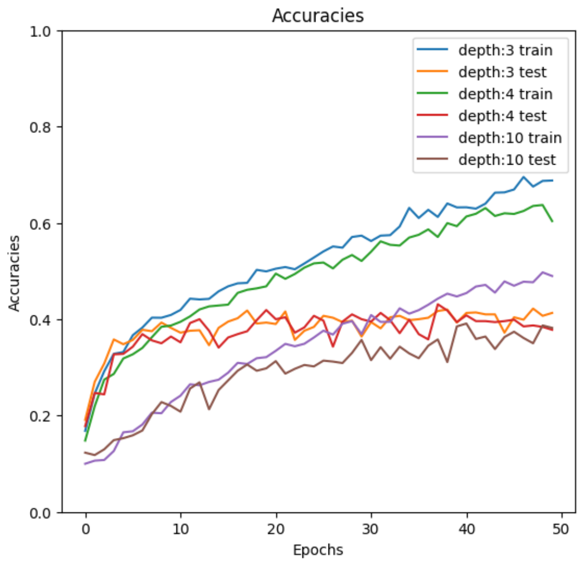
My expectations were:

1. The test accuracy and loss get better as we can generalize the data better with more parameters.
2. Overfit will be higher as the ratio between the size of the data and the number of parameters in the network changes.

I ended up not seeing much difference in test accuracy. I did see the overfit gradually increase with the network width.

Part 2 sec 7 – Network depth:





My expectations were to see slower convergence as the network gets deeper, as well as more overfit.

We can see that at depth 10 learning gets much slower, however I did not see more overfit.

<u>Part 3 sec 1 – Cnn Bseline:</u>

The grid search was conducted for the following parameters and 50 epochs:

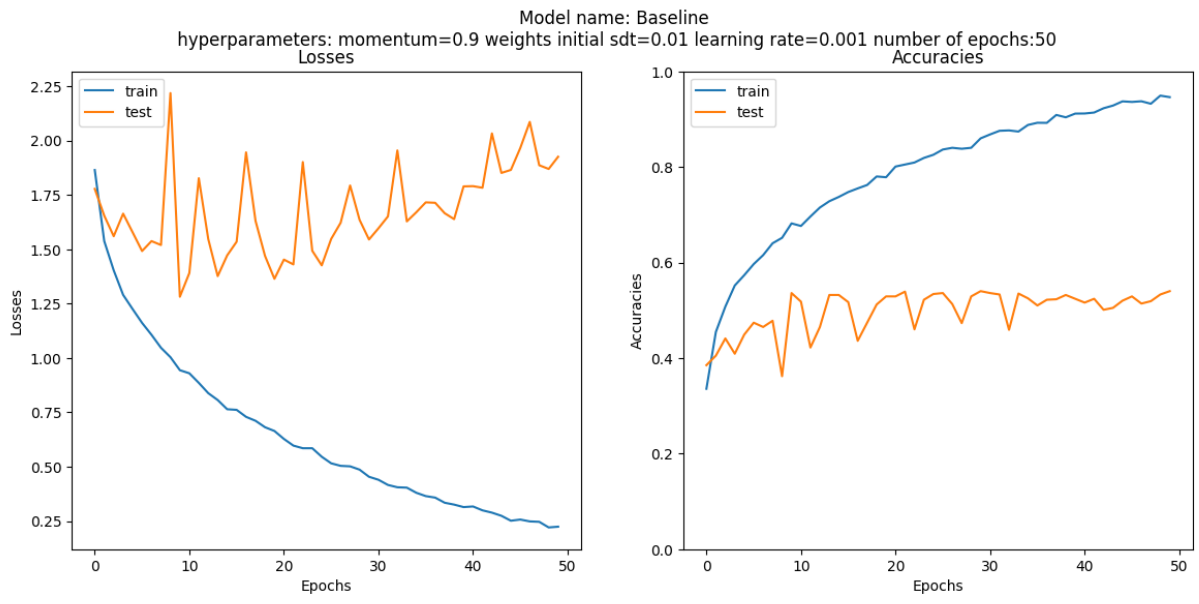
Momentum coefficient: [0.5, 0.9, 0.99]

Weights std: [0.01, 0.1, 1]

Learning rate: [0.001, 0.01, 0.1]

The best hyperparameters found were:

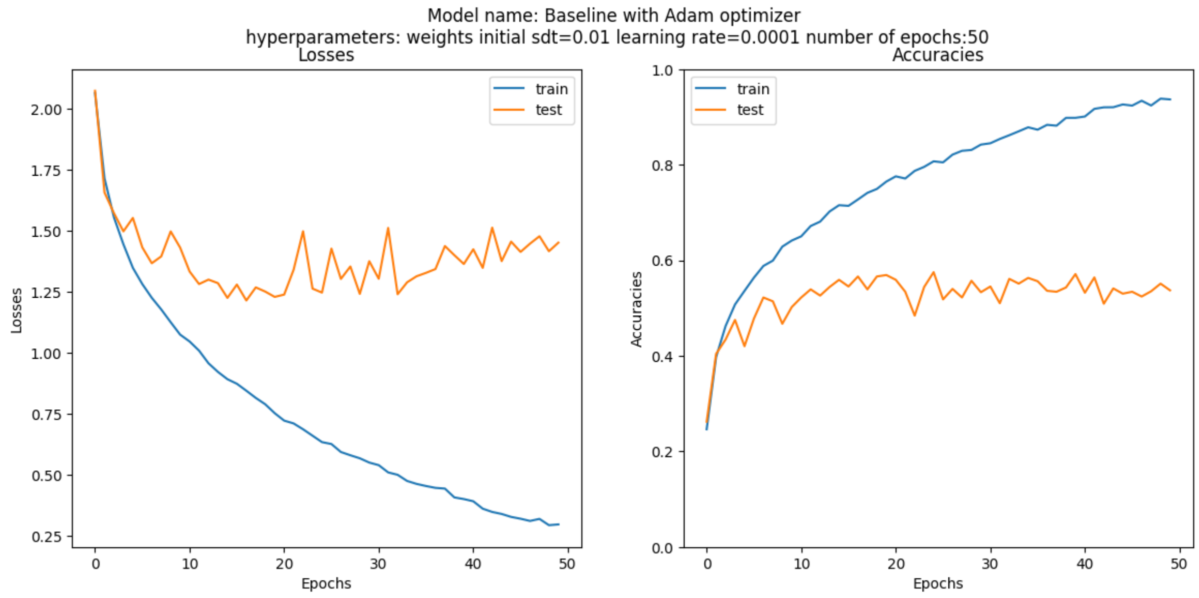
Momentum = 0.9 , learning rate = 0.001, weight std = 0.01



Train loss is: 0.2243 test loss is: 1.9257 train accuracy is: 94% test accuracy is: 54%

We can see that the convolution network manages to achieve much higher accuracy than the linear network. I believe this is because the convolution network also considers spacial information and does not flatten the image. The linear network lost a lot of information by flattening the data right away.

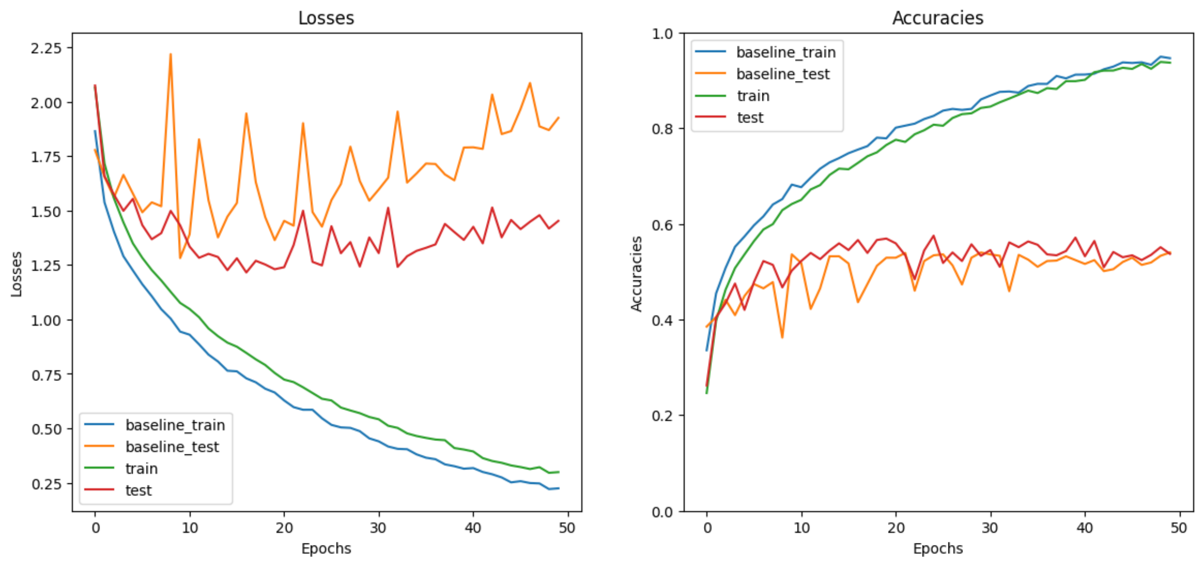
Part 3 sec 2 – Adam optimizer:



Train loss is: 0.2987 test loss is: 1.4527 train accuracy is: 93% test accuracy is: 53%

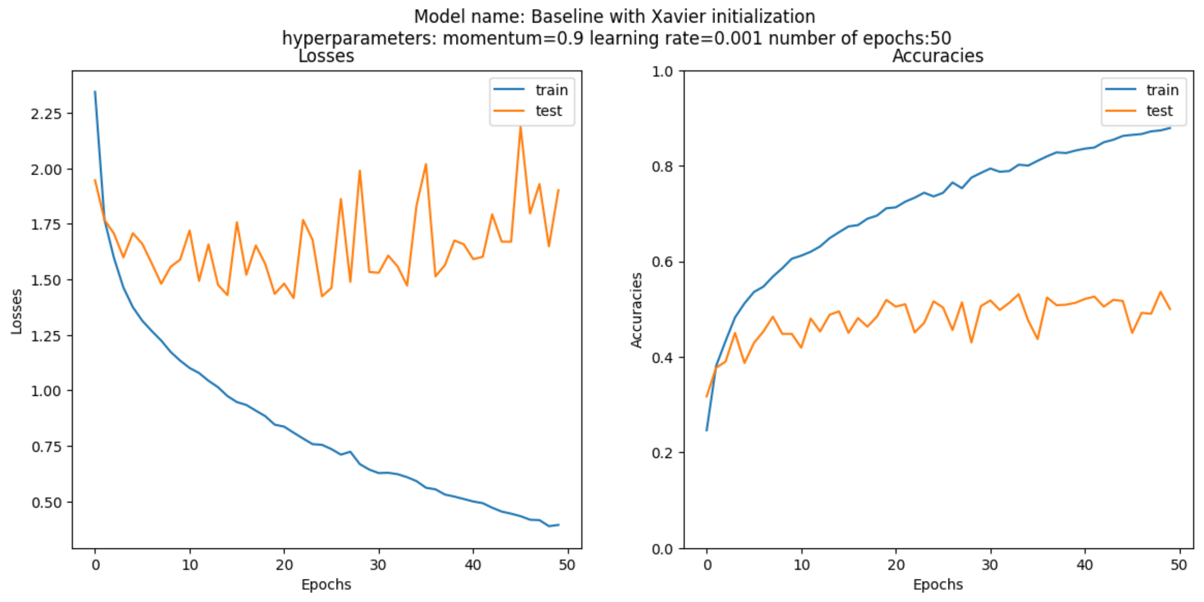
My expectation was that Adam optimizer would result in faster convergence.

Even though I used learning rate 10 times slower the network converged faster:



We can see that the test loss for the Adam optimizer gets lower faster than SGD. We also get better test loss in general. I believe this is due to the slower learning rate allowing to take smaller more accurate steps.

Part 3 sec 3 – Xavier initialization:



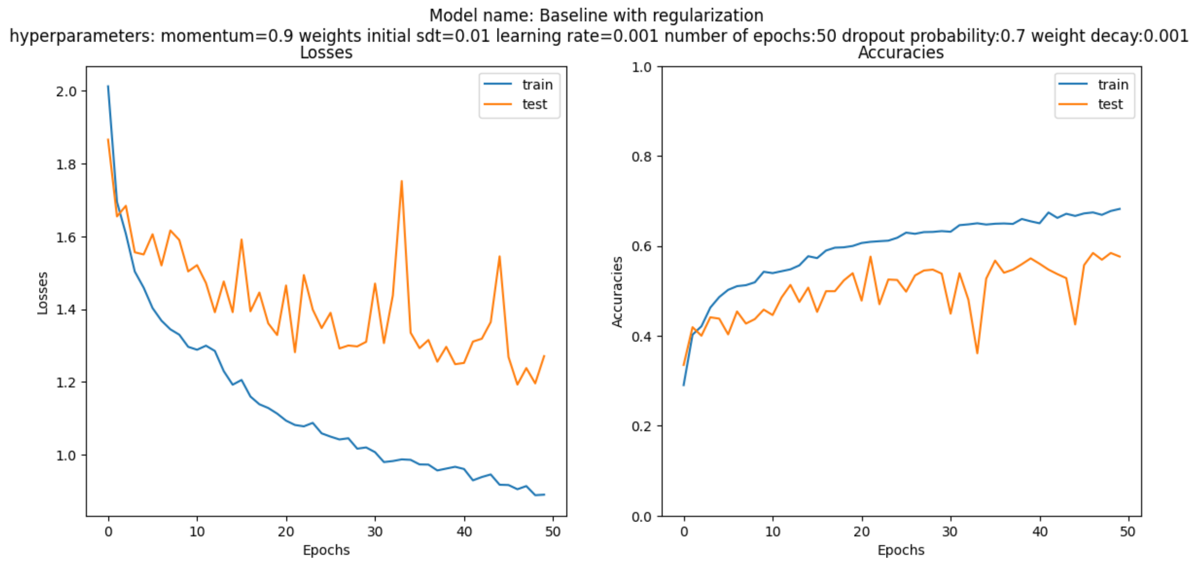
Train loss is: 0.3943 test loss is: 1.9021 train accuracy is: 87% test accuracy is: 50%

My expectation was that Xavier will converge faster at the beginning since the weights are initialized with respect to the size of the input.

Once again I did not see any improvement in the convergence time compared to the baseline.

I did see slightly less overfit.

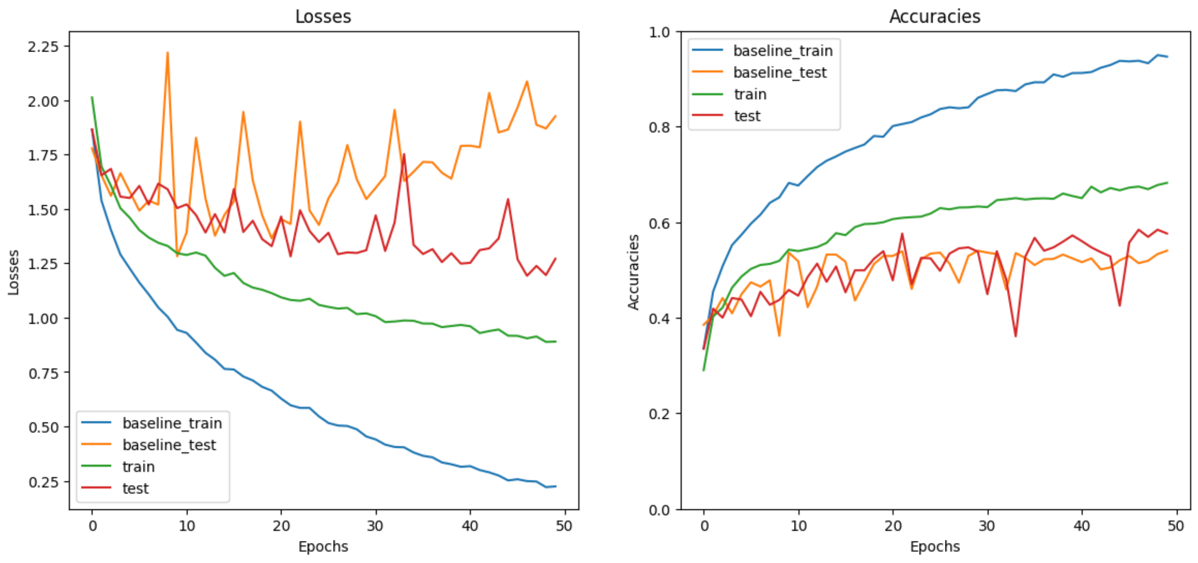
Part 3 sec 4 - Regularization:



Train loss is: 1.1126 test loss is: 1.3286 train accuracy is: 59% test accuracy is: 53%

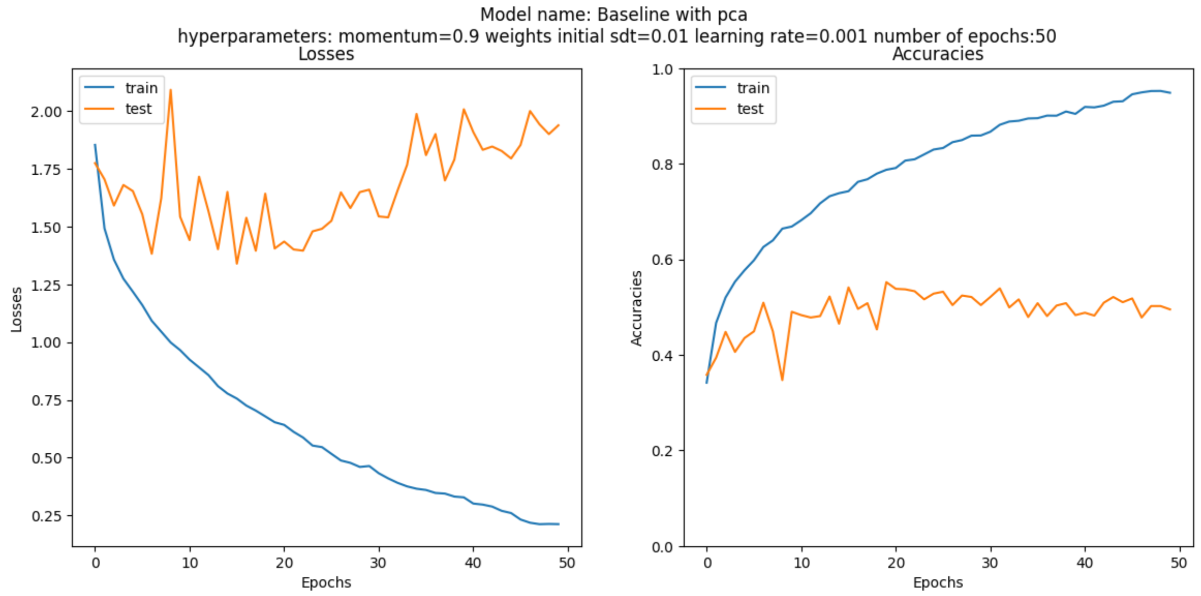
My expectation was to see much less overfit and a slightly better performance on the test.

We can see that the overfit was reduced by a lot (from around 80% to around 60%) as well as slightly lower test loss:



Runtime was increased due to additional dropout layers.

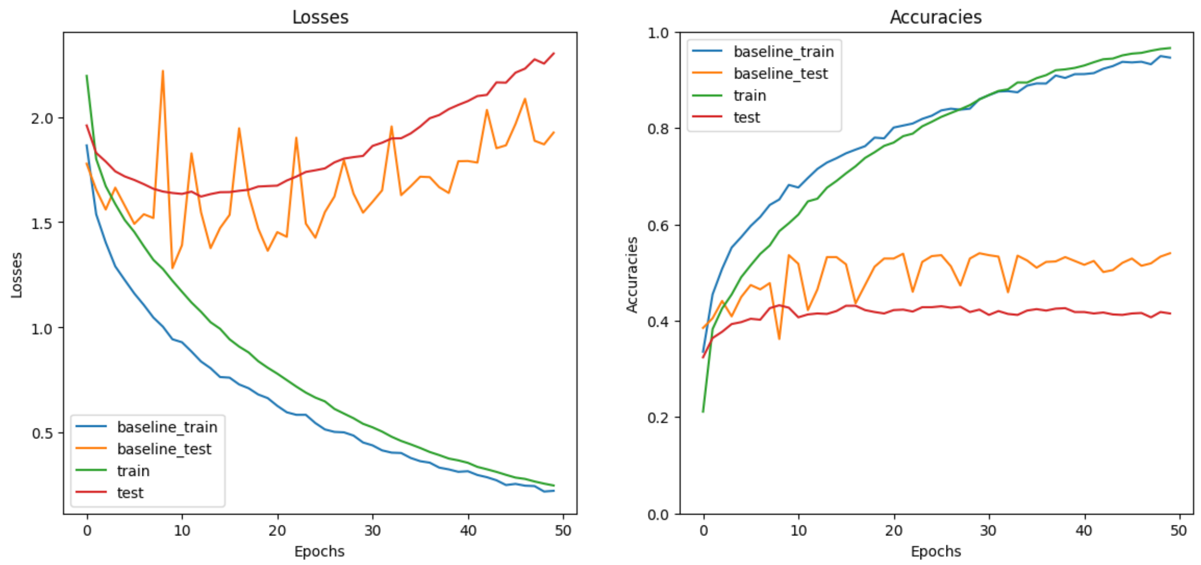
Part 3 sec 5 - Preprocessing:



To perform pca whitening on image I had to separate the image channels and perform pca whitening on each channel separately. I also reduced the number of components to 500 from 1024 in hope to remove some of the noise from the data.

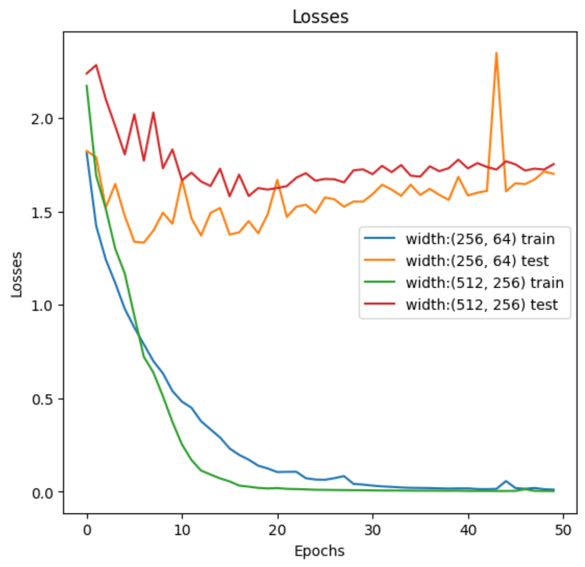
My expectation was for runtime to remain the same as we did not change the size of the data.

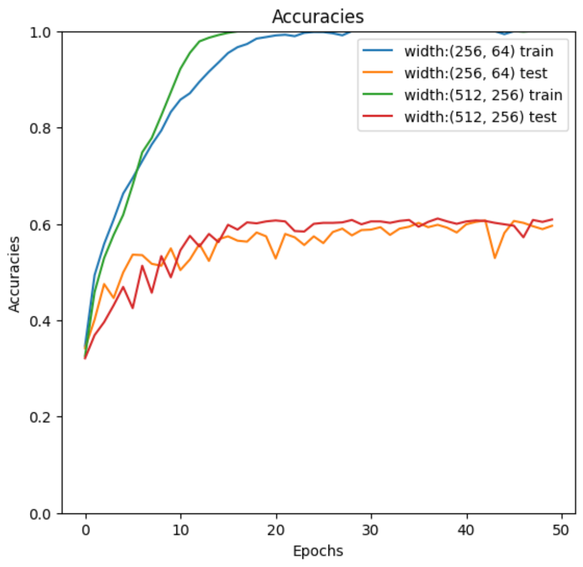
I also expected to see slightly better results due to the removal of the less relevant components and whitening.



Results ended up looking worth however, I assume since I by reducing the number of component I removed some of the information in the data.

Part 3 sec 6 – Network width:





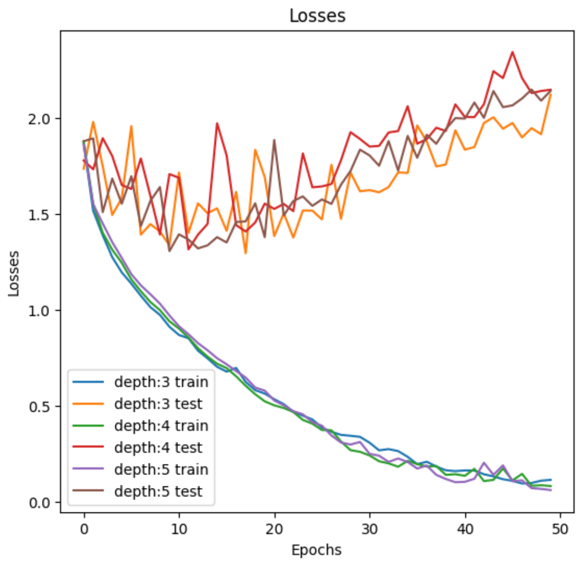
My expectations were:

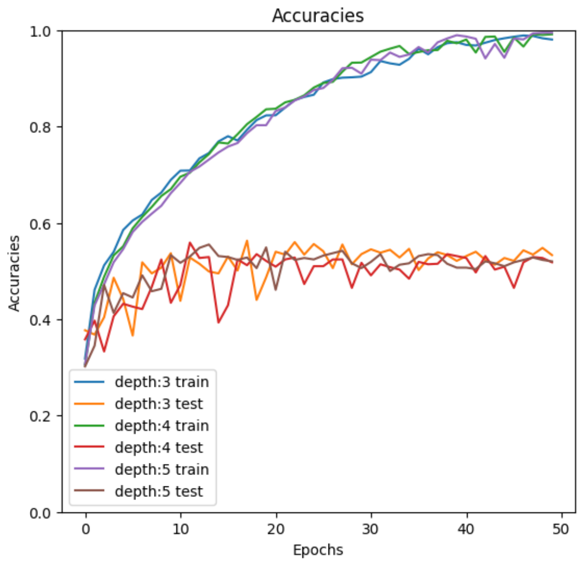
1. The test accuracy and loss get better as we can generalize the data better with more parameters.
2. Overfit will be higher as the ratio between the size of the data and the number of parameters in the network changes.

We can see that the test accuracy did in fact get better, reaching 60% accuracy.

We also see that the train accuracy reach 100% in both cases (faster on the wider net). This happens since the networks are now large enough to fit the train data and predict it perfectly.

Part 3 sec 7 – Network depth:





My expectations were to see slower convergence as the network gets deeper, as well as more overfit.

I did not end up seeing much difference between the different depths.