Experiments On Deep Networks

Tomer Borreda and Nir Endy

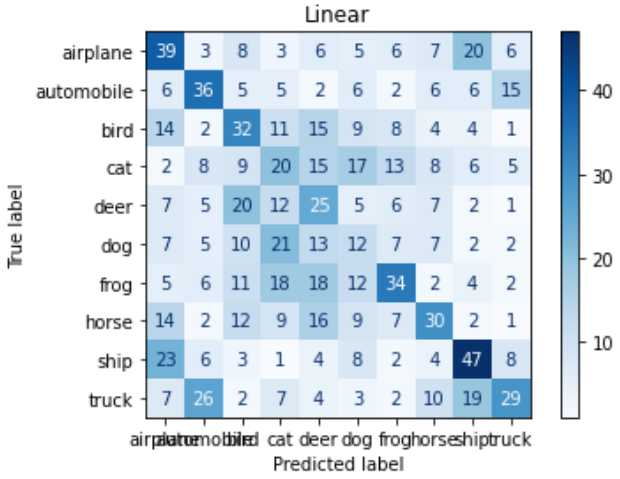
- By observing different options we decided that 50 epochs are enough to show convergence for all variants.

Part 1:

Linear kernel:

Accuracy: 0.304

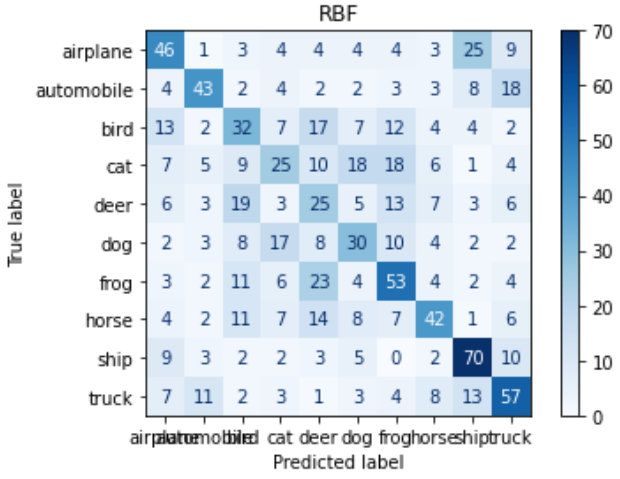
Confusion matrix:



RBF kernel:

Accuracy: 0.423

Confusion matrix:

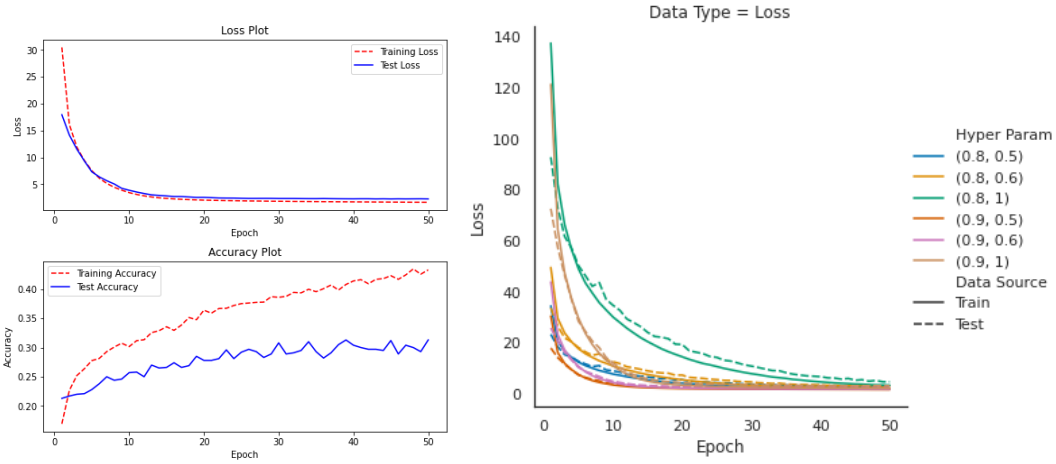


As expected, the RBF kernel achieved higher accuracy compared to the linear one.

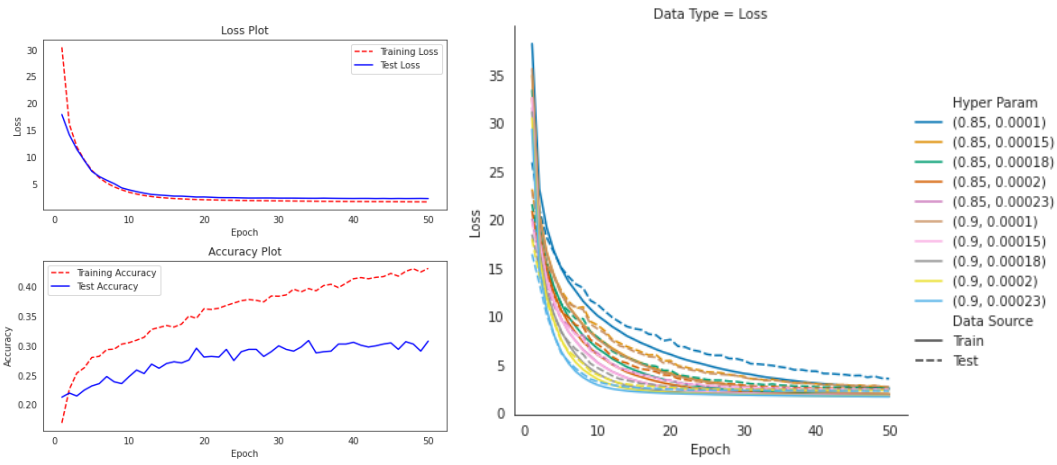
Part 2:

(Notice that full results and more graphs are available on the jupyter notebook)

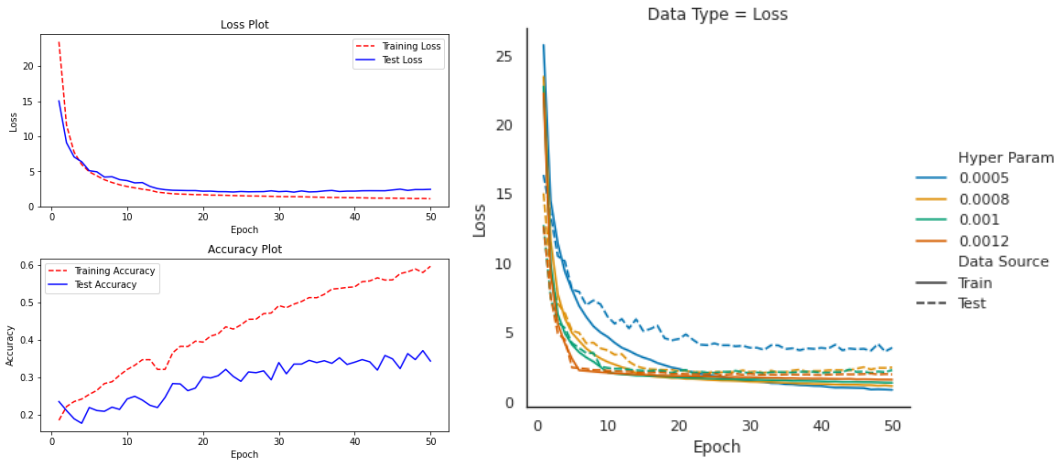
1. Here we use momentum SGD optimizer with different hyperparameters.
   a) We performed Grid search for *momentum* and *standard deviation* with constant *learning rate*(0.0002).
   The best accuracy obtained was 0.313 with *momentum* equal to 0.9 and *standard deviation* equal to 0.5.



   b) We performed Grid search for *momentum* and *learning rate* with constant *standard deviation*(0.5).
   The best accuracy obtained was 0.308 with *momentum* equal to 0.9 and *learning rate* equal to 0.0002.

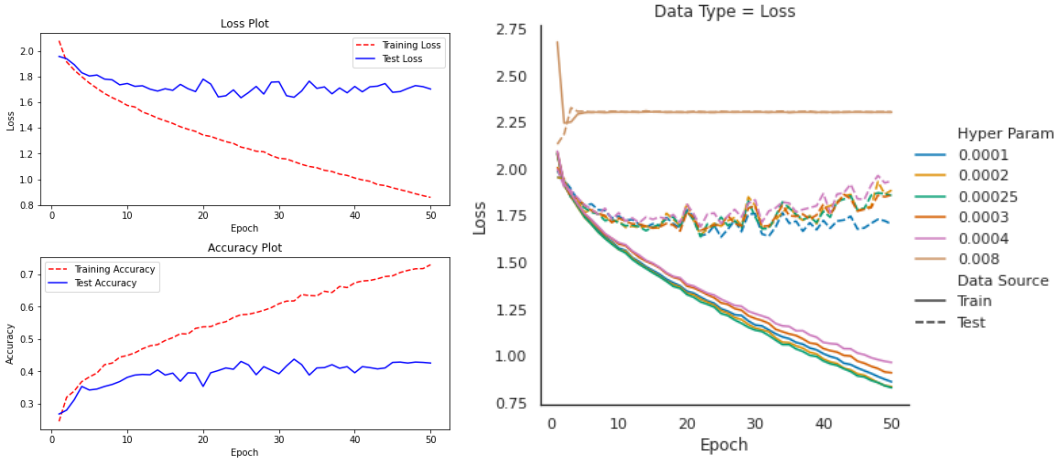


2. We tried different hyperparameters using ADAM optimizer.
   The best accuracy obtained was 0.343 with *learning rate* equal to 0.0008.
   (constant *initialization std* 0.5)



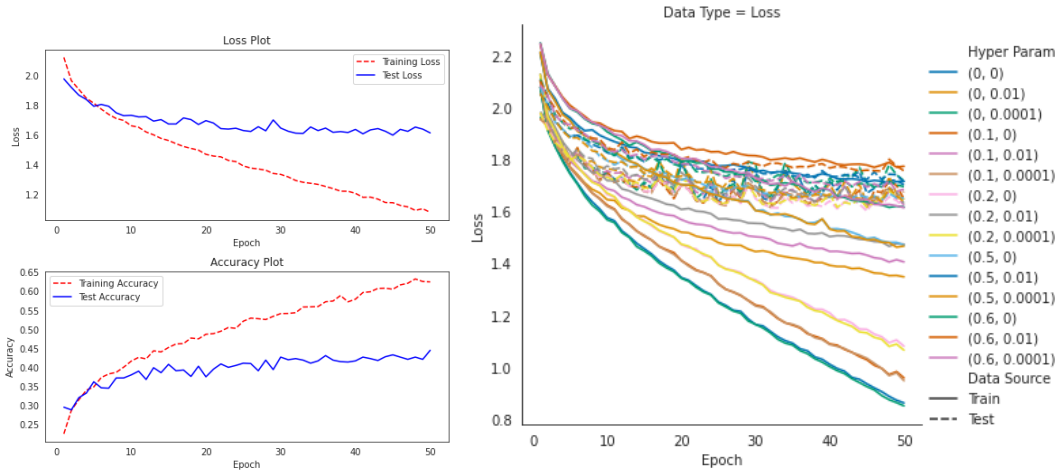
   As can be seen, ADAM achieves better accuracy and converges slightly faster. This can be expected because unlike momentum SGD, ADAM updates the momentum while optimizing. A more intuitive explanation would be that ADAM behaves like a heavy ball with friction which would make the ball reach a flat minima and therefore it would be more stable than momentum SGD.

3. We tried different hyperparameters using Xavier initialization.
   The best accuracy obtained was 0.426 with *learning rate* equal to 0.0001.



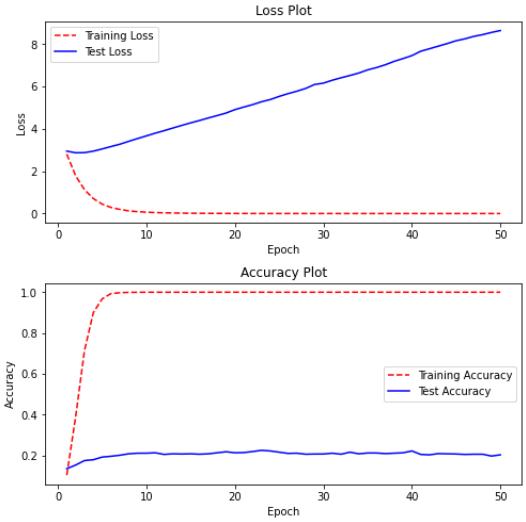
   Xavier initializes the weights at a better starting point which from it is more likely to reach a better minimum. This might require fine tuning and therefore a smaller learning rate can achieve that.

4. We tried different hyperparameters using Dropout and L2 Penalty.
   For dropout we set a *drop_prob* parameter - The probability a neuron will drop.
   And for the L2 Penalty we set a *Lambda* parameter.
   The best accuracy obtained was 0.445 with *drop_prob* equal to 0.2 and *Lambda* equal to 0. (constant *learning rate* 0.0001)



   The accuracy achieved is higher than the previous section and convergence is slightly faster. This can be explained by effective reduction of overfitting via dropout.
   Note that the grid search over the hyperparameters resulted with *Lambda* equal to 0, meaning no L2 penalty.

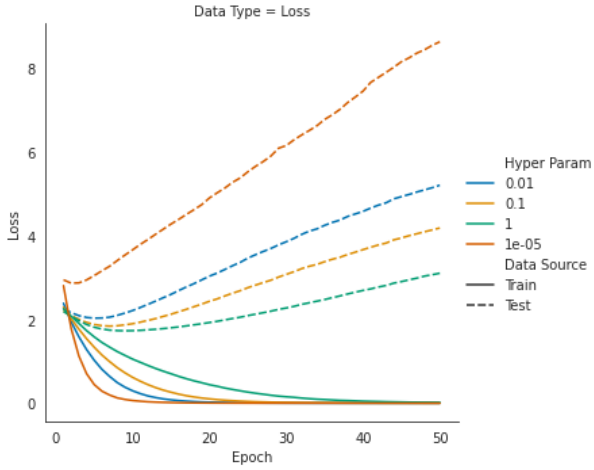
5. Here we performed PCA whitening on the data prior to training (PCA was performed on training data and with those values whitening was applied to test data).
   After trying many different hyperparameters we could not obtain good results. There is overfitting on the training data and low accuracy for the test data.



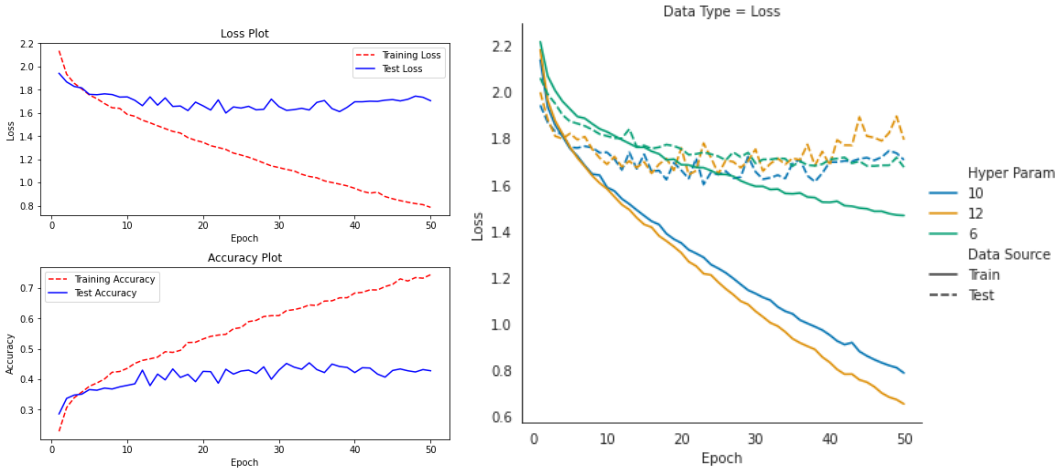
   A possible explanation for this phenomenon is that whitening using all principal components, i.e. manipulating the data in a way that there is a constant variance

among them all, results in shrinking the magnitude for the most variant data patterns, and expending the magnitude for the least variant ones. This may improve the net expressibility for the training data, but worsen its generality. That explains the high performance on the training set and the low performance on the testing set.
A possible solution that we haven't implemented, is to reduce the dimension of the data using the PCA dimension reduction method, and then perform PCA whitening on the data.

We also tried different epsilon values. The obtained results are better but the data is still overfitted.



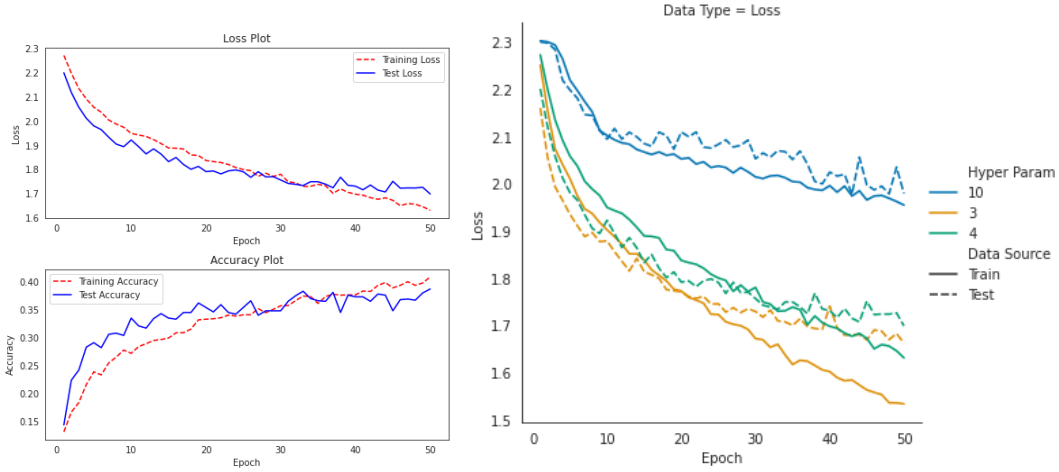
6. Using the best parameters from the previous sections we tried different widths: 2^6, 2^{10}, 2^{12}.
The best accuracy obtained was 0.428 with *width* equal to 2^{10}. (constant *learning rate* 0.0001, *drop_prob* equal to 0.2 and *Lambda* equal to 0)



As we can see, with width of 2^{12}, the test loss started increasing from epoch 35.
A possible explanation for these results would be that a network with larger

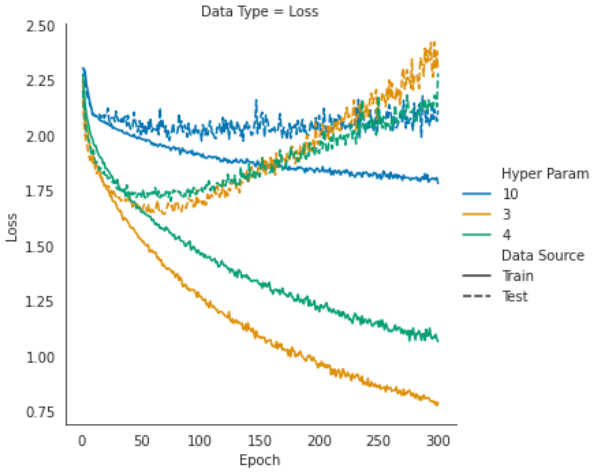
width might cause overfitting and therefore not generalize well. Performing grid search over widths obtains the best width that would not overfit.

7. Using the best parameters from the previous sections we tried different depths: 3, 4, 10.
   The best accuracy obtained was 0.387 with *depth* equal to 4. (constant *learning rate* 0.0001, *drop_prob* equal to 0.2 and *Lambda* equal to 0)



As we can see, higher values of depth results in a slower learning and therefore higher loss after 50 epochs. This might result from the phenomena of vanishing gradients which occur at deeper networks and cause for slower learning for the deeper layers. There are many solutions for this problem which we did not implement here.

We also tried to run this for a longer time which resulted in overfitting the training dataset.



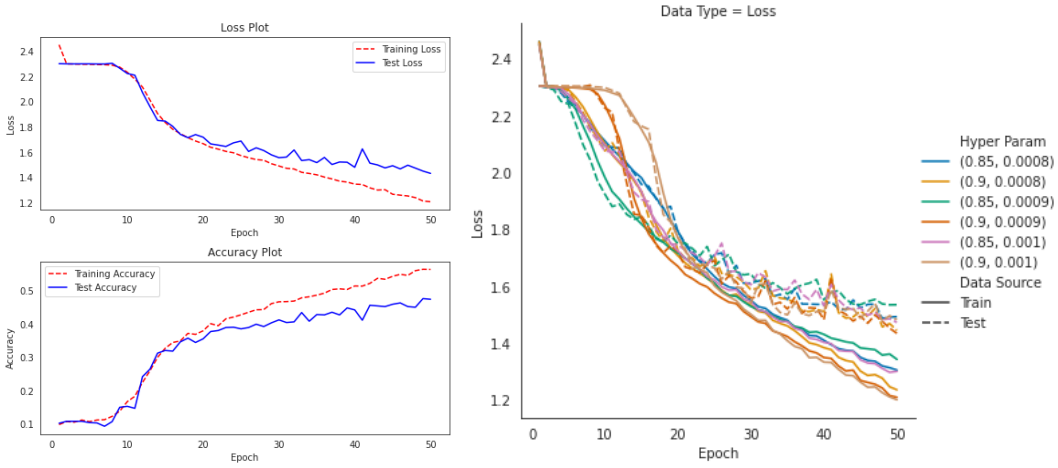
As can be seen, 50 epochs is approximately the number of epochs which from the network starts to overfit and therefore 50 epochs is a good choice.
Also, it might be that the overfitting occur so soon due to small size of the training dataset.

Part 3:

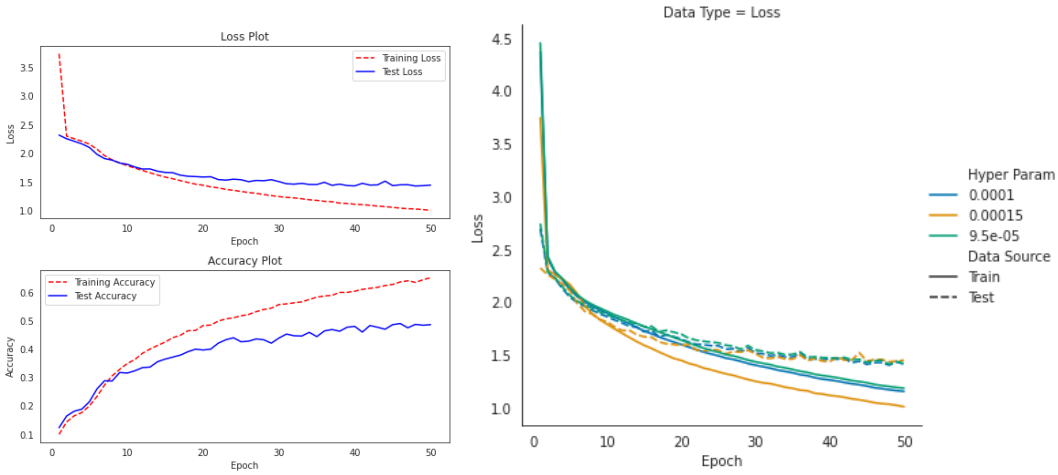
(Notice that full results and more graphs are available on the jupyter notebook)

1. Implementation of the CNN can be found in the jupyter notebook.

2. We tried different hyperparameters using momentum SGD and ADAM optimizers.
   For momentum SGD the best accuracy obtained was 0.475 with *momentum* equal to 0.9 and *learning rate* equal to 0.0009. (constant *initialization std* of 0.2)

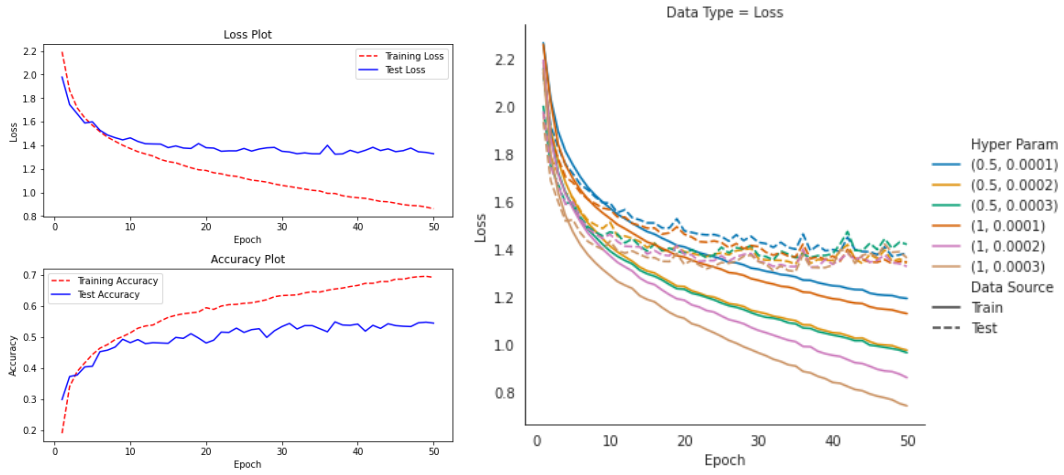


   For ADAM the best accuracy obtained was 0.486 with learning rate equal to 0.00015. (constant initialization std of 0.2)



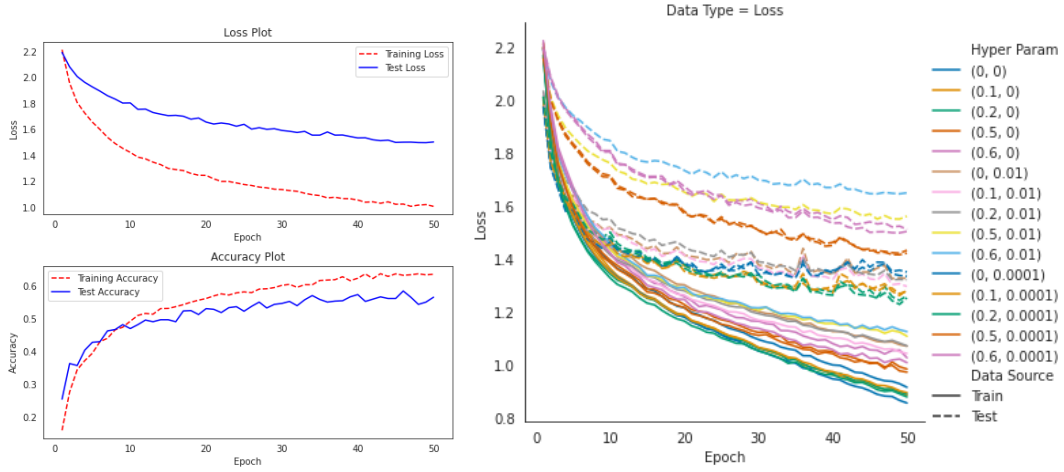
   With ADAM optimizer we are able to achieve faster convergence and better results.
   We see the same behavior as in the previous section, and as we explained, ADAM momentum updates result in a more stable optimization.

3. We tried different hyperparameters using Xavier initialization.
   The best accuracy obtained was 0.544 with *std-gain* equal to 1 and *learning rate* equal to 0.0002.



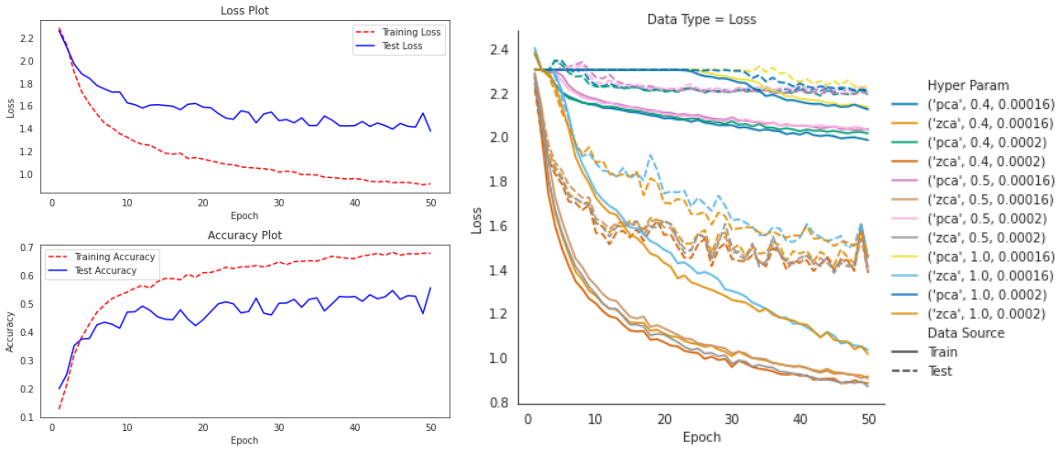
   The obtained results are better. Accuracy is higher and test convergence rate is slightly faster(~15 instead of ~25).

4. We tried different hyperparameters using Dropout and L2 Penalty.
   For dropout we set a *drop_prob* parameter - The probability a neuron will drop.
   And for the L2 Penalty we set a *Lambda* parameter.
   The best accuracy obtained was 0.566 with *drop_prob* equal to 0.6 and *Lambda* equal to 0.0001. (constant *initialization std* of 0.2 and *learning rate* 0.0002)



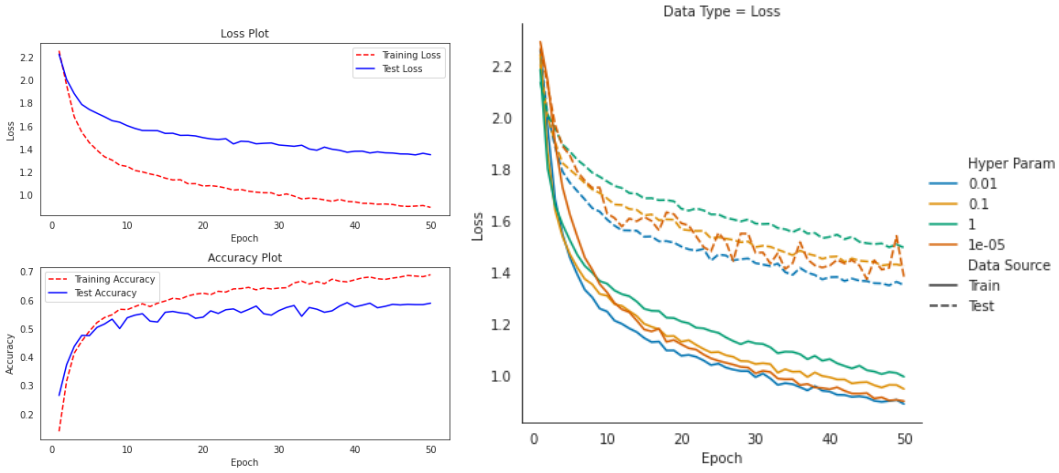
   The obtained results have higher Accuracy. As explained in the previous part, regularization reduces overfitting and improves test time results (Accuracy and loss).

5. When changing the data (using pca/zca) the weight initialization can affect the effectiveness of learning. Since different initializations require different learning rates we perform a grid search to find the best parameters.
The best accuracy obtained was 0.556 using ZCA with *sgd-gain* equal to 0.5 and *learning rate* equal to 0.00016. (constant *drop_prob* equal to 0.6 and *Lambda* equal to 0.0001)



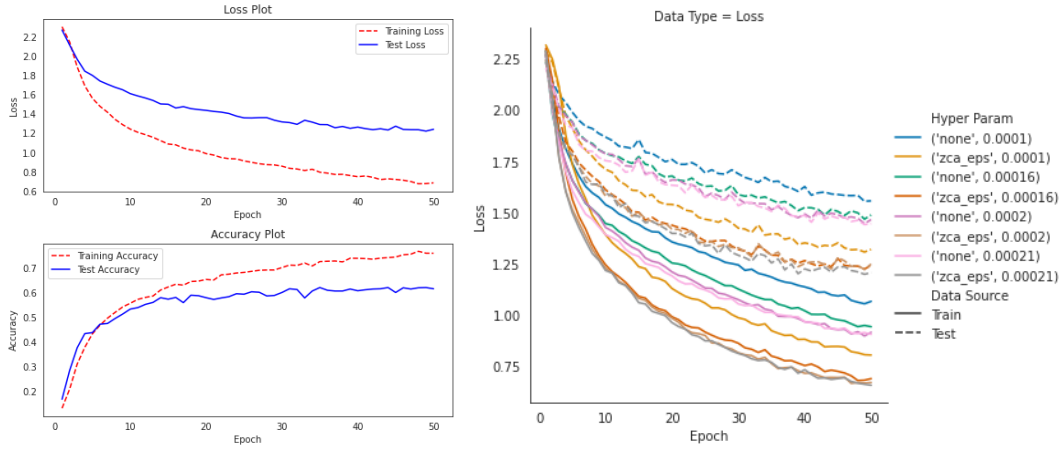
As expected by the theory, using PCA whitening for CNN will give bad performance. This is because convolutional layers are taking into account the feature "environment". In order to solve that, we used ZCA to return the data to the original basis.
Using ZCA the results are much better compared to PCA and similar to previous results.

We also tried different values of epsilon in the ZCA implementation.
The best accuracy obtained is much higher and equal to 0.590. Here we used ZCA preprocessing with *epsilon* equal to 0.01.



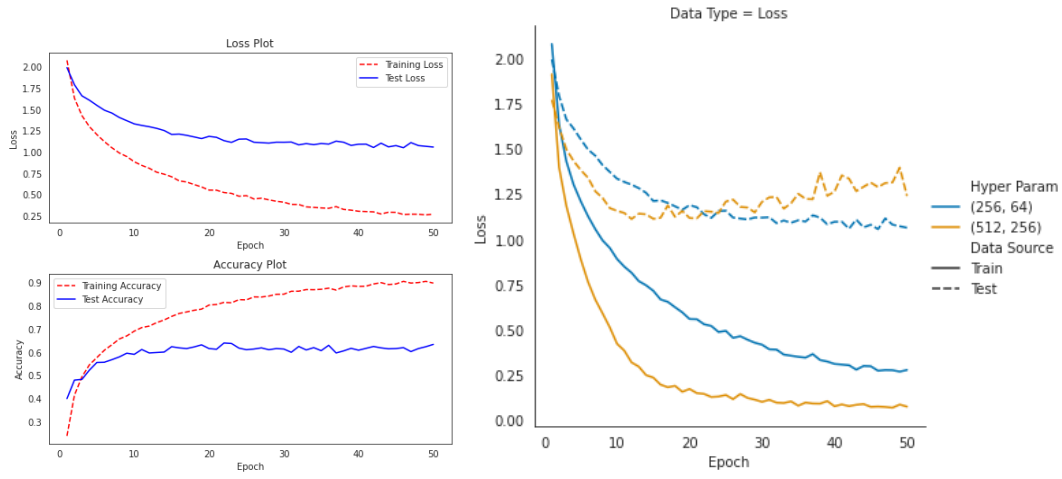
The *epsilon* is the factor that we add as part of the PCA whitening in order to avoid dividing by values that are too close to zero. We perform a grid search to find an epsilon that will be appropriate to our data.

6. While trying to achieve best results for the network with kernel size of 5x5 we noticed that using the data with ZCA whitening gives better results.
   The best accuracy obtained was 0.615 using ZCA preprocessing with *epsilon* equal to 0.01 and *learning rate* equal to 0.00016. (constant *drop_prob* equal to 0.6, *Lambda* equal to 0.0001 and *sgd-gain* equal to 1)



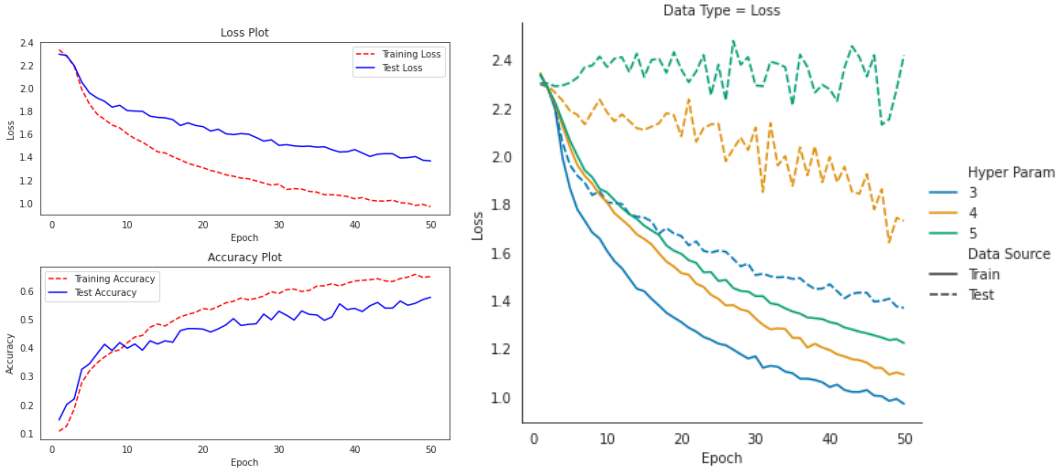
   Generally, it's common to use 3x3 kernel size, as it practically gives better results. Therefore, we expected that 5x5 would worsen the results, especially since CIFAR pictures are in low resolution. However, it seems that the 5x5 kernel improves the results, and that can be explained by the fact that 5x5 includes more complex information.

7. Using the best parameters from section 5 we tried the different widths mentioned.
   The best accuracy obtained was 0.635 with *widths* of (256,64).



   Similarly to section 6 in part two, we observe an upward tendency for the larger width which indicates overfitting of the training dataset.

8. Using the best parameters from section 5 we tried the different depths mentioned.
   The best accuracy obtained was 0.578 with *depth* of 3.



Similarly to section 7 in part two, we observe that a deeper network did not improve performance. We may need to use additional methods in order to be able to utilize the deeper network. (such as residual connections)