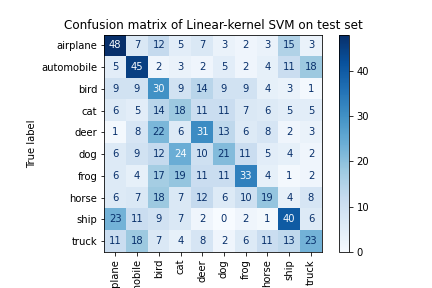
Question 1:

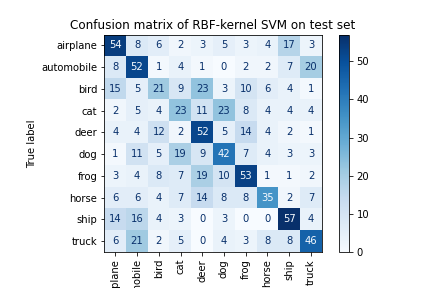
SVM with linear kernel results:

1. On Train:

* Accuracy: 1.0 (100%)

1. OnTest:

* Accuracy: 0.308 (~31%)
* Confusion matrix:

SVM with RBF kernel results:

1. On Train:

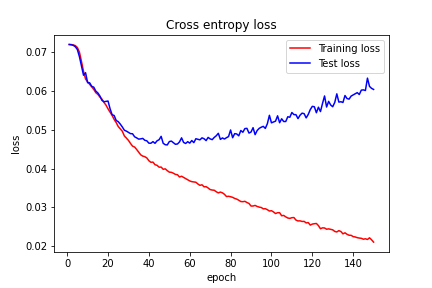
* Accuracy: 0.7146 (~72%)

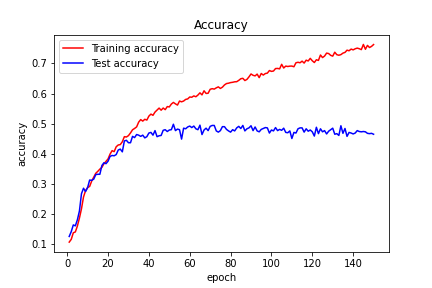
1. OnTest:

* Accuracy: 0.435 (~43%)
* Confusion matrix:

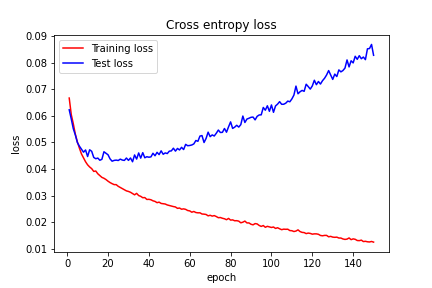
Question 3:

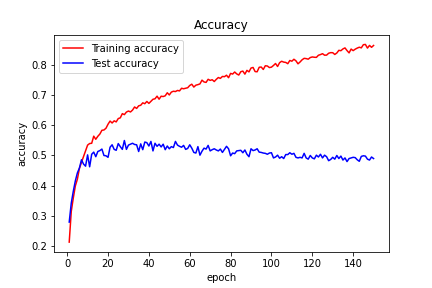
1. The best momentum was 0.9, here is the results using it with lr = 0.001, normal initialization with mean 0 and std 0.02, 150 epochs:





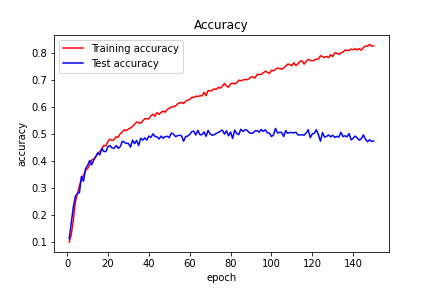
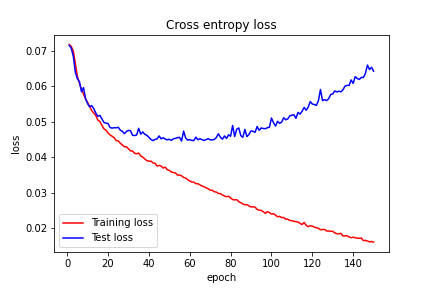
Adam optimizer results, using same initialization, same learning rate, same number of epochs:



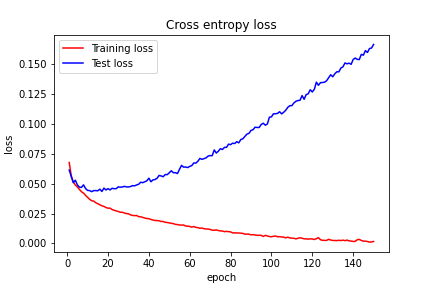


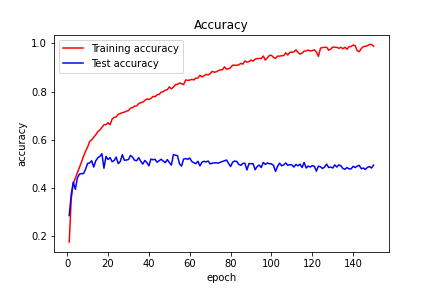
It’s easy to notice that ADAM optimizer is converging much faster, both in reducing the train loss and bot in getting more accurate on train data (about 3-4 times faster).

1. Same settings of SGD as in section 2, but with Xavier initialization:



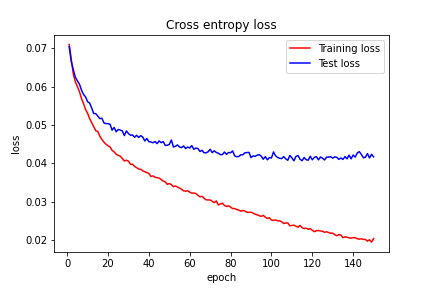
Same settings of Adam optimizer as in section 2, but with Xavier initialization:

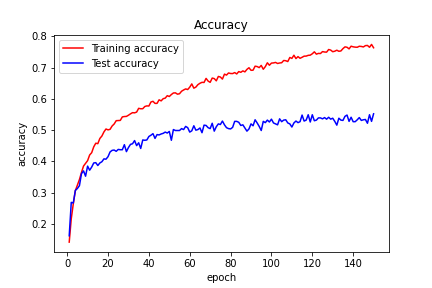




Xavier initialization is accelerating the convergence time of loss going to zero and of accuracy going to one.

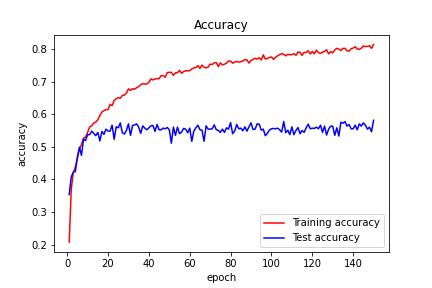
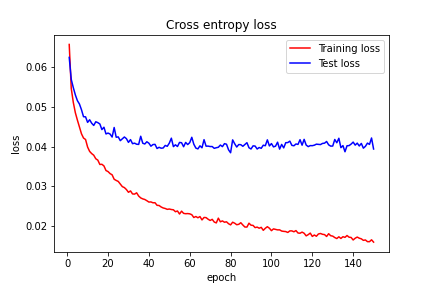
1. We took the same setting as in section3 and tried few different combinations of weight\_decay and dropout. We didn’t found values that achieve major improvement in the converging rate of the train loss to 0 or accuracy to 1 (actually it caused worse converging to 0 and 1), but we did notice that this values are probably reducing overfitting and enhancing generalization. We can see that by the lower difference between test and train losses and accuracies.

SGD with same configuration as in section3, but with dropout = 0.3, wheigt\_decay=0.02



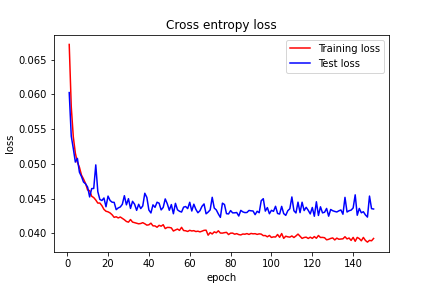
For ADAM optimization, any combination of these two didn’t improve results by much, but using only one of them (each one) did caused major improvement: using only dropout = 0.4 caused accuracy on test to be better (reached arround 0.6) and using weight\_decay=0.2 caused much better test loss:

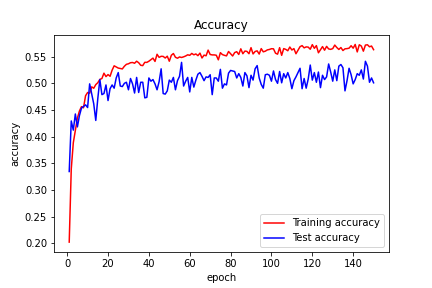
ADAM as in section3 with droupout = 0.5:



We can see the the loss and accuracy on the train are worse than in section3, but on the test the loss and Accuracy are better! Which means that it improve generalization, maybe run it more epochs would end with much better resluts

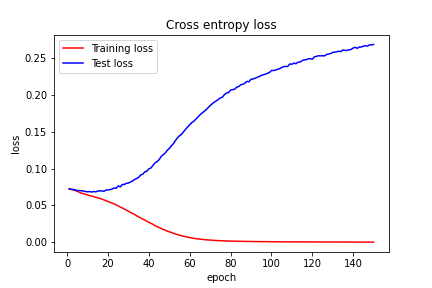
ADAM as in section3 with weight\_decay=0.2

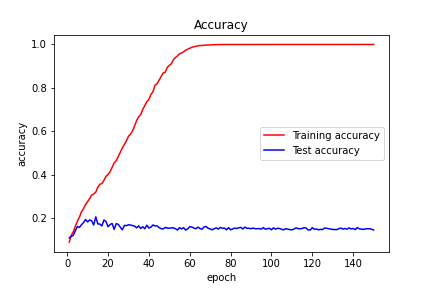


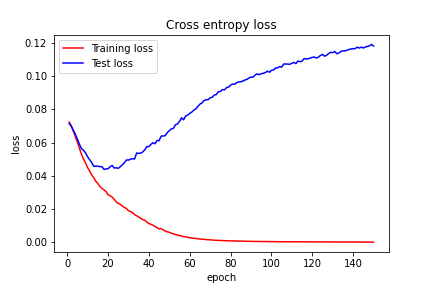


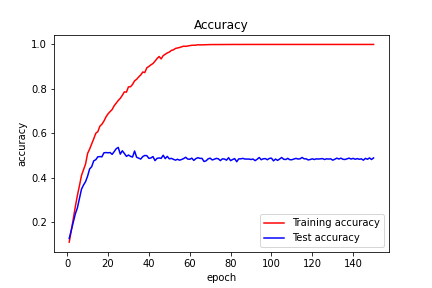
1. We see the same phenomena on SGD and on ADAM, we can notice that because ZCA preserve high-order statistics on the original data, especially because it’s pictures, we get better result with it on the test data using ZCA over PCA.

Here is the results when running SGD in the same settings as in section3 with PCA:

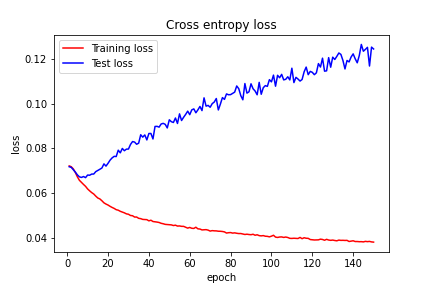


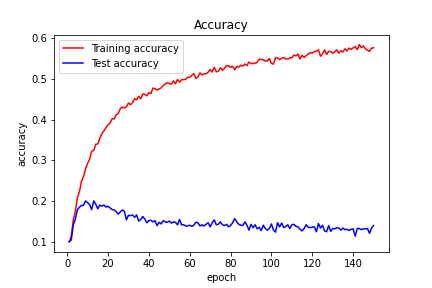
Here is the results when running SGD in the same settings as in section3 with ZCA:



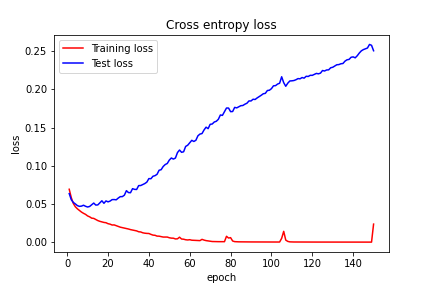


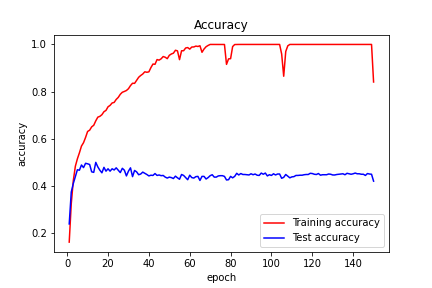
Here is the results when running ADAM in the same settings as in section3 with PCA:



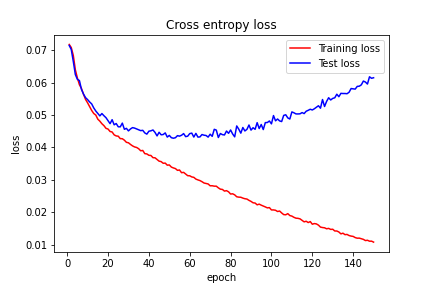


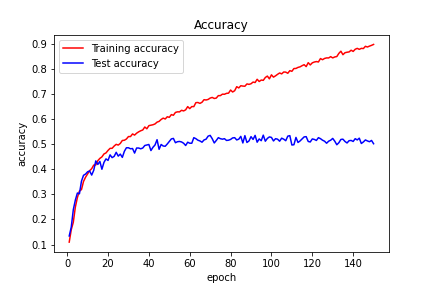
Here is the results when running ADAM in the same settings as in section3 with ZCA:

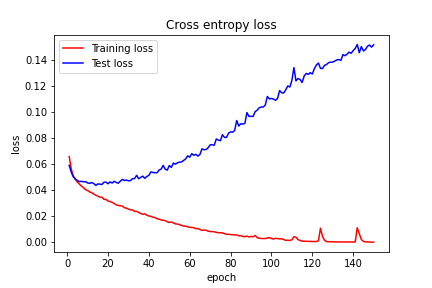


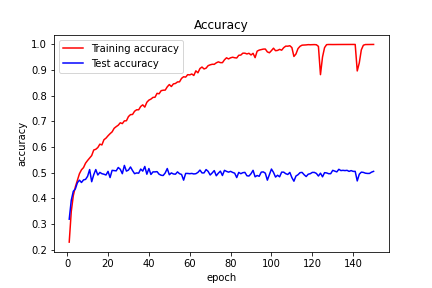


1. This change cause faster convergence but it seems that it cause more overfitting.

Results of running SGD with same settings as section3 but with kernel size 5x5 (the results are quite similar)

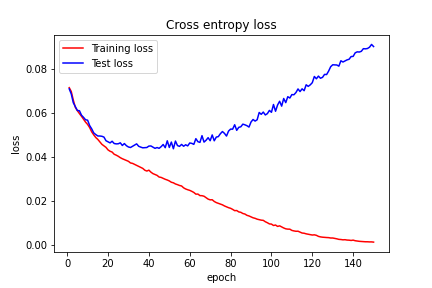


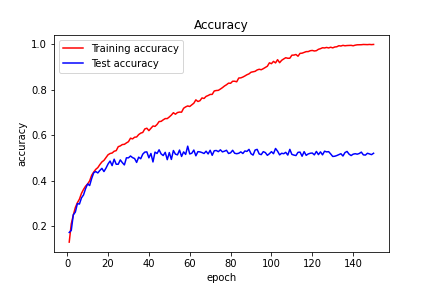
Results of running ADAM with same settings as section3 but with kernel size 5x5 (the results are quite similar): 

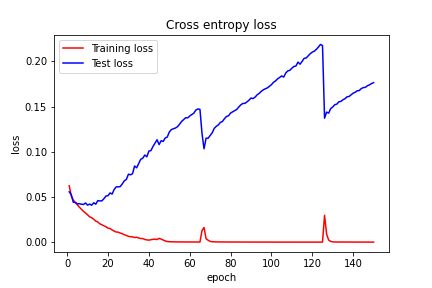


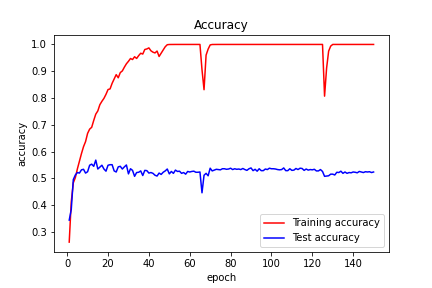
1. We can notice that (both for SGD and ADAM) it accelerated convergence of loss on train to zero and accuracy on train to one, but the test loss in the final result became worse, and test accuracy is quite the same (overfitting)

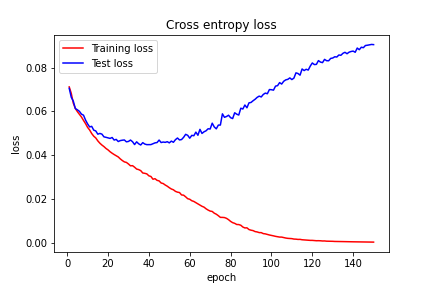
Results when Running SGD with the same setting as section3 but amount of filters: (256,64):

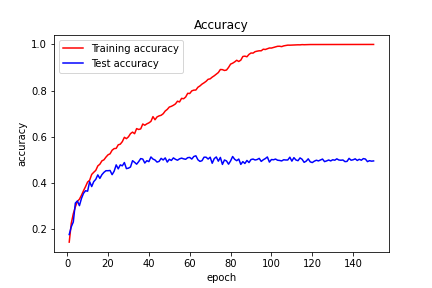




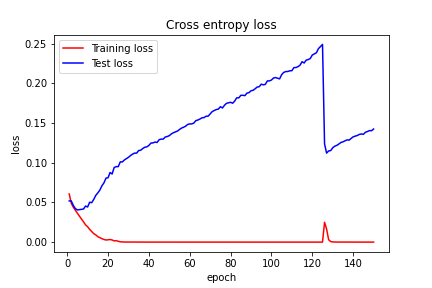
Results when Running ADAM with the same setting as section3 but amount of filters: (256,64): 

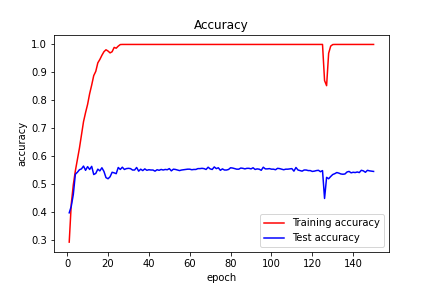


Results when Running SGD with the same setting as section3 but amount of filters: (512,256): 



Results when Running ADAM with the same setting as section3 but amount of filters: (512,256):





1. We run the architecture with k= {2,3,4,5} such that we added more layers to the architecture in section3, we added:

For k=3: relu(conv(64 filters, 3x3 kernel))

For k=4: another relu(conv(128 filters, 3x3 kernel)) on top k=3

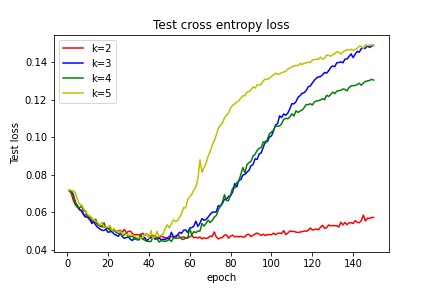
For k=4: another relu(conv(256 filters, 3x3 kernel)) on top k=4

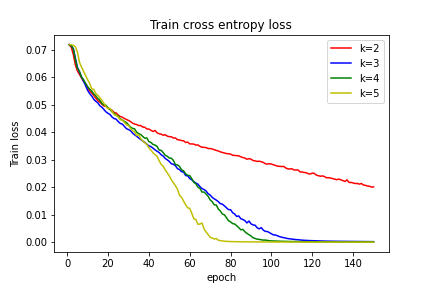
because the kernel reduces the image, we thought it will be good to increase the number of filters.

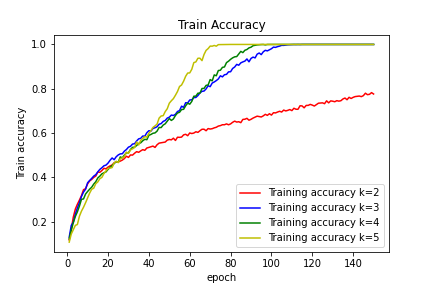
We see similar results to what we have seen before: as we bring k to be higher, we make the net larger and thus, it can represent more hypotheses, that’s why we can get to 0 loss on train and 1 accuracy on train. But, it’s probably cause overfitting (being too good on train, and not good enough on test), because the gap between the 2 is getting larger as k is large.

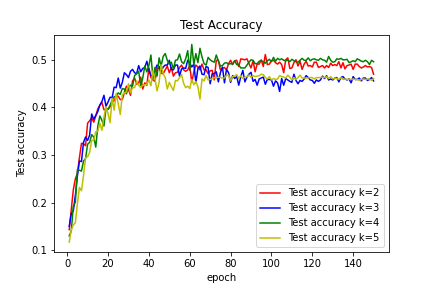
When we look on the bottom line – at the accuracy on test, we see that there is a tradeoff between being good on the train and overfitting, we specifically got best tradeoff with k=4, but not significantly better than the others. (we didn’t try dropout and other things)

Here are the results (running SGD):









We got the best results (test accuracy) in ADAM optimizer with dropout = 0.5, so out of interest we try that with different values of k:

We can notice that the dropout causing the optimization to be very fluctuated.

