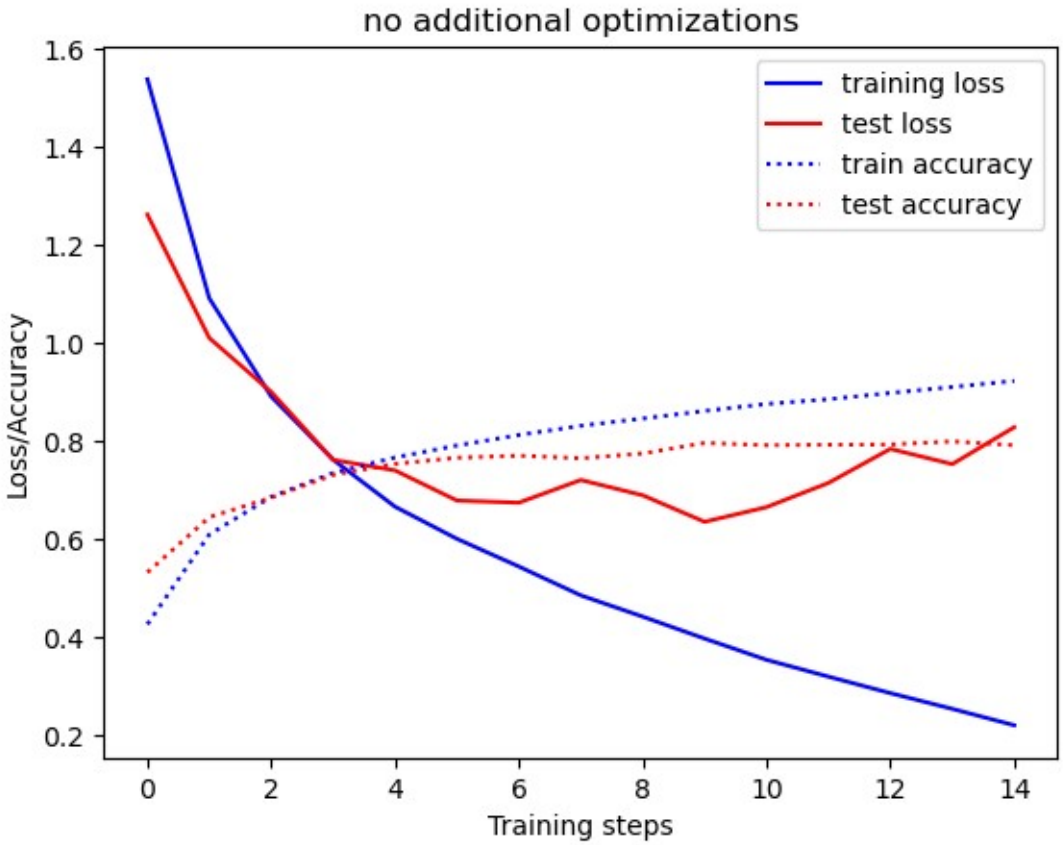
Different regularization techniques used:

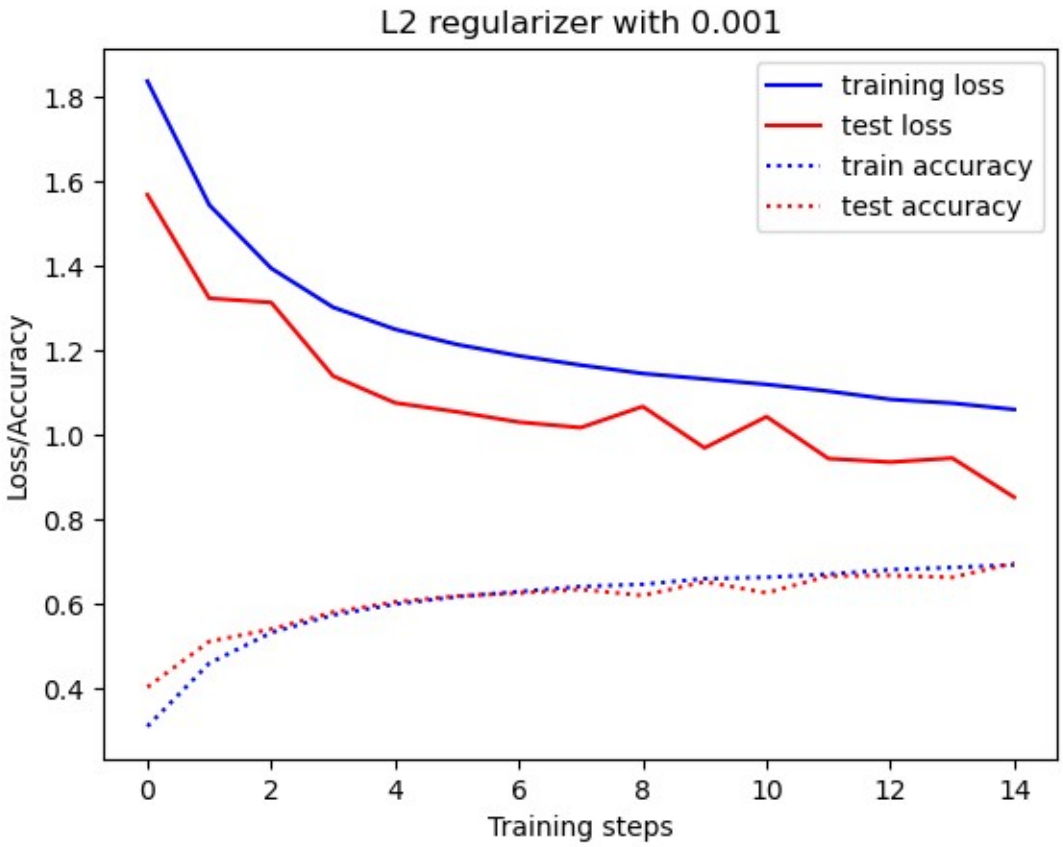
1. Normalizing the input
already done in the model, as it is kinda needed for the model and the rest of the techniques to work

2. Using ReLU as it is less affected by vanishing gradients as much
already implemented in the model from the start



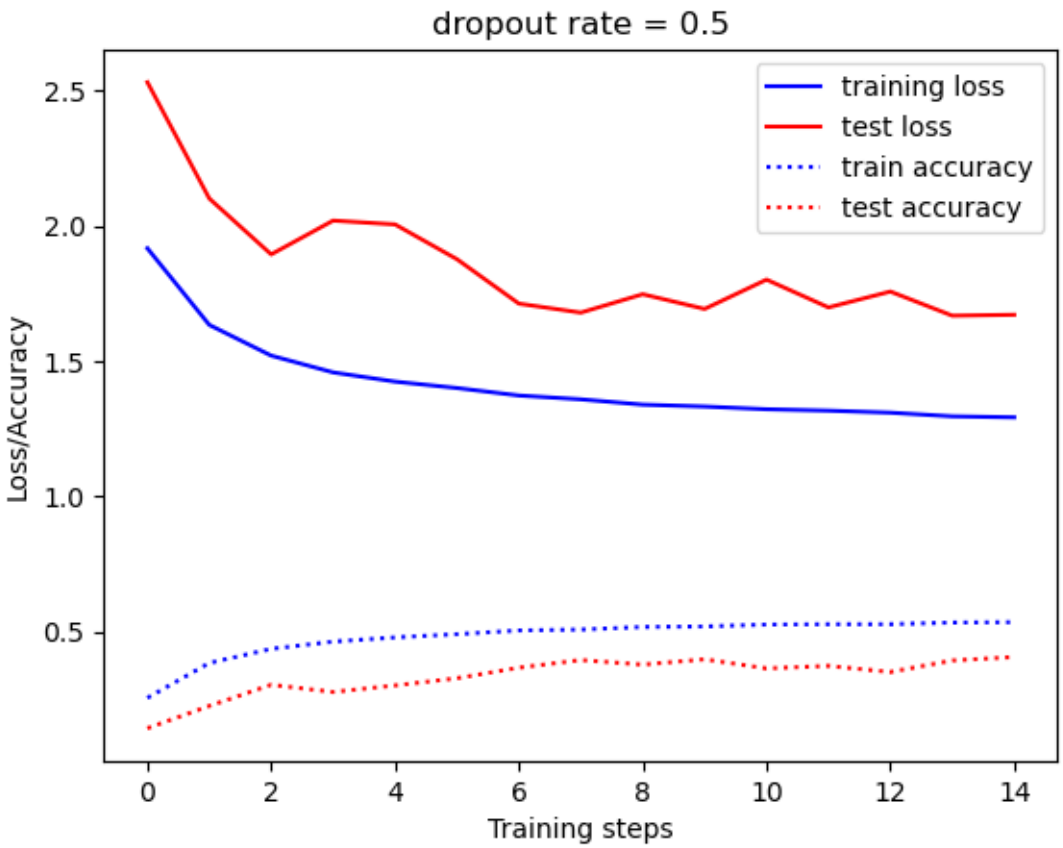
Remark:
there is a possible overfit as the train accuracy continues to grow to about 0.9 while the test accuracy flats out around 0.75. Also the test loss starts to increase again after 7-9 epochs.

3. L2 regularizer adds a penalty to the loss value for large weights
large weights will push the acctivation of the unit to the extremer ends of the activation function and leave less influence to the actual input.
Here it is used with a value of 0.001 (as used in the example)



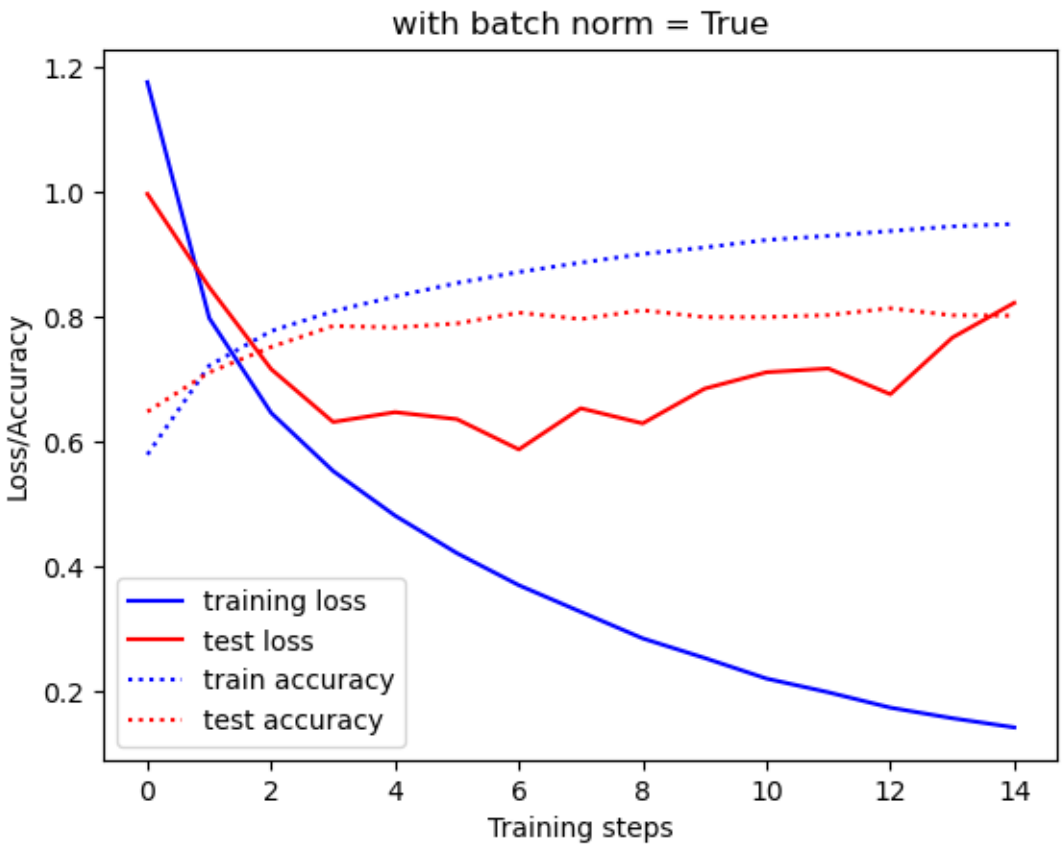
Remark:
Test loss is no longer increasing and the accuracies do not diverge at the end.
It seems like the train accuracy sunk and the test accuracy stayed similiar at about 0.7

4. Using dropout units
randomly (based on a probability value) dropping certain units, that do not belong to the output units.
Here used with a probability of 0.5 as it was proposed in various forums



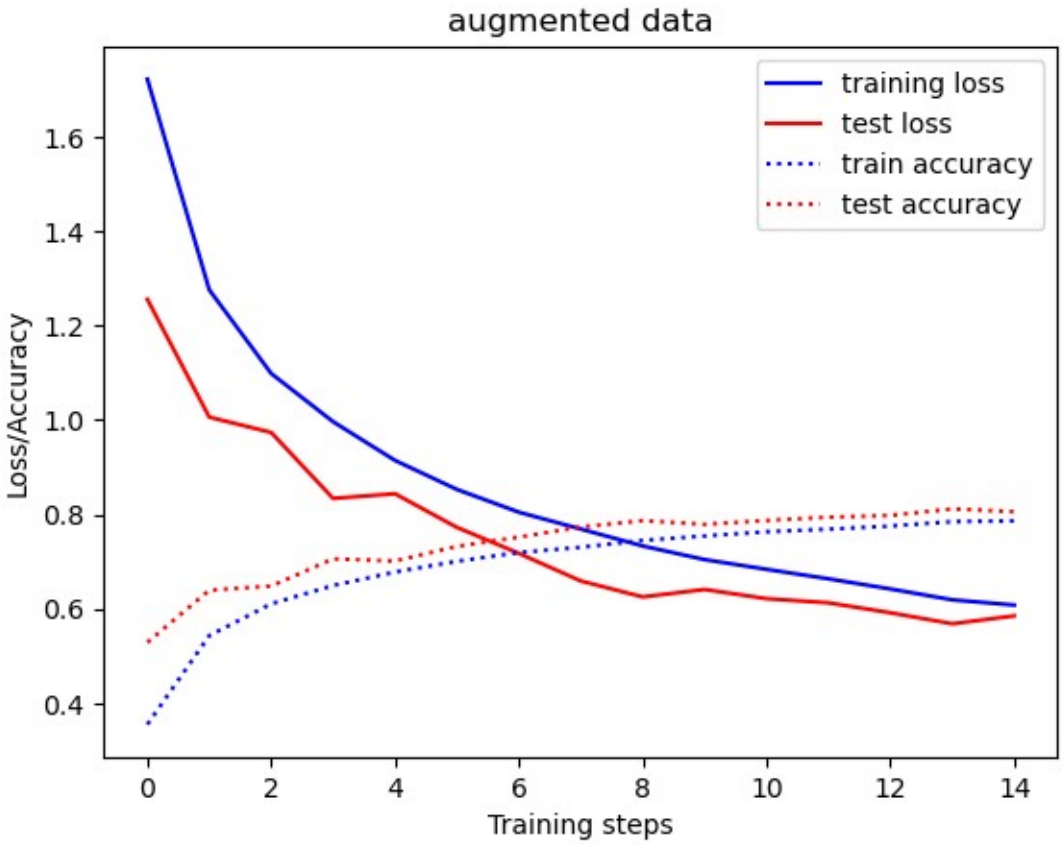
Remark:
Reduces the accuracy down to under 0.5, so definitely worse performance.
As implemented, it gives every layer the same chance of dropout, some sources claim you should differ that value per layer.

5. batch normalization makes sure the activation looks similar to a normal distribution advanced versions have new learnable hyperparameters. Here the defaults of TF are used.



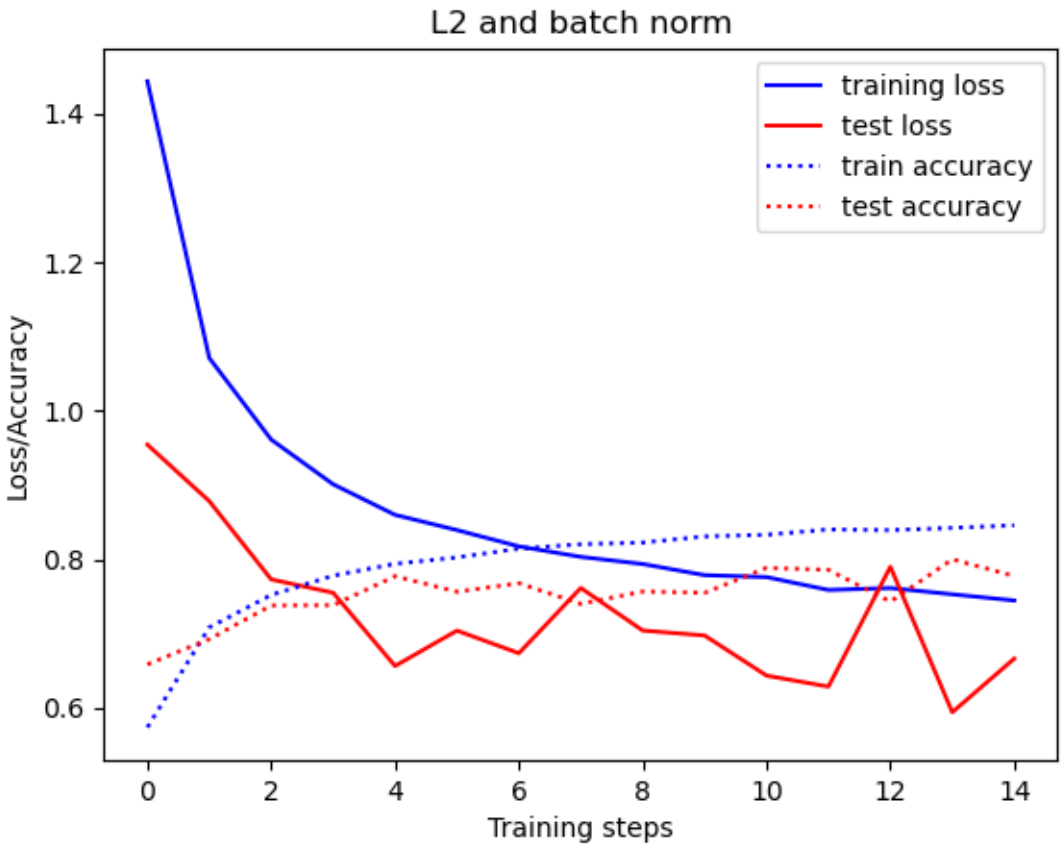
Remark:
Looks very similar to the first plot with no additional optimizations.
Test loss increases after a while and the accuracies diverge towards the end

6. Data augmentation means changing the data in certain ways to get more diverse training data it is only performed on the training dataset and you have to make sure to not change anything that is actually supposed to be used for training (e.g. swap color channels)



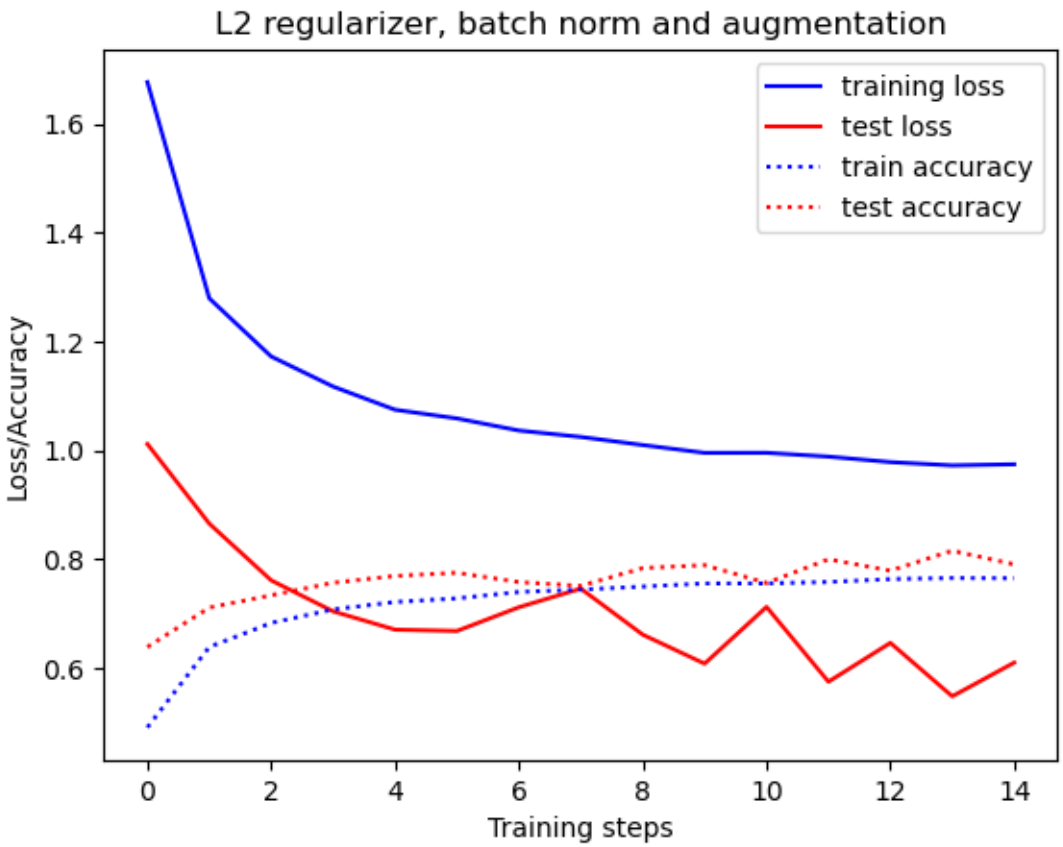
Remarks:
Helps to bring accuracies closer together. Test accuracy seems to be the highest so far with 0.8. Interestingly the train loss / accuracy is higher / lower than the corresponding test value, which is usually the other way around

7. Combining L2 regularization and batch normalization



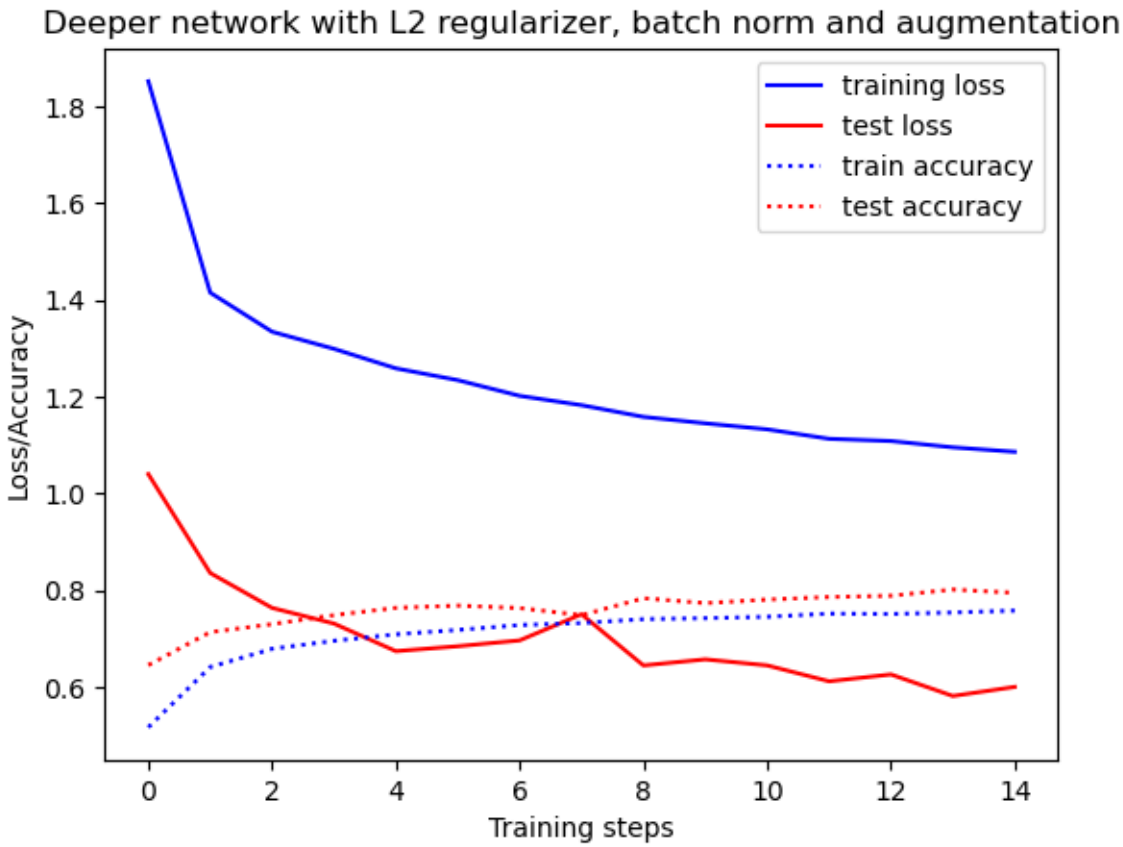
Remarks:
test values seem a bit unstable and are changing a lot (especially loss) but less obvious overfitting. Test accuracy stays under 0.8.

8. Combining L2 regularization, batch normalization and data augmentation



Remarks:
Best test accuracy so far (partly above 0.8). But training loss is again a lot higher than test loss.
No visible overfitting, train accuracy is even lower than test accuracy – seems to be a stable effect of data augmentation

9. A deeper model (one additional block) with using the above changes



Remarks:
Looks very similar to the shorter model. Accuracy doesn't increase much but seems consistently above 0.8.