Deep Learning - HW2

Tidhar Suchard, Danielle Ben Shlomo

GitHub repository: https://github.com/daniellebs/deep-learning-assignment-2.git

Model architecture description and training procedure:

The model we built contains the following layers:
If we mark 3 layers of SpatialConvolution, SpatialBatchNormalization and ReLU as ConvBNReLU(a->b) where a is the input size and b is the output size on the convoluiton, and the convolution layer has a window of size 3, with padding of 1 on each side, and the step size is 1, the model will be:
ConvBNReLU(3->16)
ConvBNReLU(16->16)
MaxPooling with step size of 2 (vertical and horizontal steps).
ConvBNReLU(16->32)
ConvBNReLU(32->32)
MaxPooling with step size of 2
ConvBNReLU(32->32)
ConvBNReLU(32->32)
MaxPooling with step size of 2
ConvBNReLU(32->32)
View
ReLU
Dropout of 0.5
Linear layer where it's output is the number of classes (10)
At last, we added a LogSoftMax layer, required by the criterion we chose to use: ClassNLLCriterion (Negative Log Likelihood), a criterion used for classifications problems.
The number of parameters for this model is 49,914 (< 50,000).
During the construction of the model we tried less convolution layers with larger depth for each layer, and received worse results. We eventually reached the conclusion that larger number of convolution layers will produce better results. We also reached the conclusion that the current order of the layers is the preferred one.

As well as trying to optimize the layer architecture, as part of preparing the data for training we normalized it around 0 to try and further improve our result.
We used 62 epochs to train the network (The minimal error for the test set was given using this number of epochs), and the size of each batch is 64.

<u>Data Augmentation:</u>

As seen in the results section, we tried using several data augmentation methods: horizontal flip, vertical flip, and rotation (we rotated some photos by $\frac{1}{9} * \pi$ radians).
We chose to use only hflip (horizontal flip) as data augmentation in our final model since it produced the best accuracy for the test set.

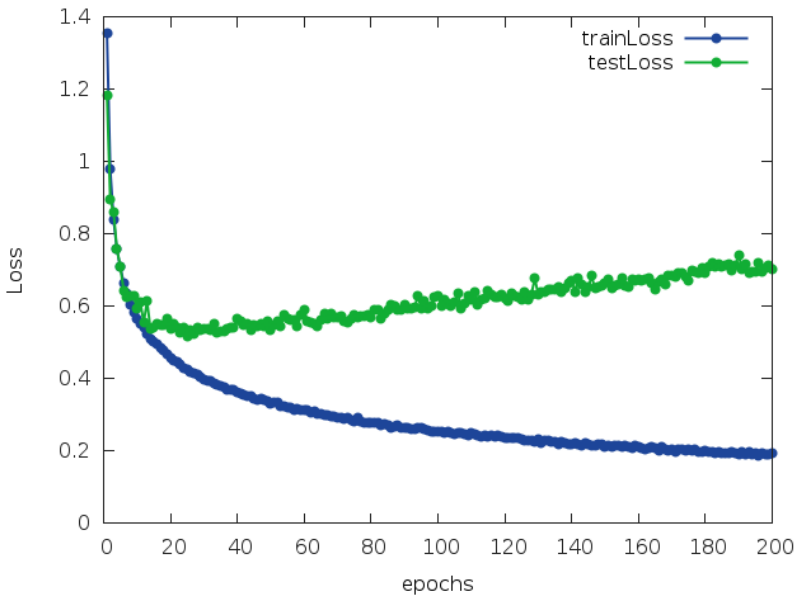
Optimization Methods:
Again, as can be seen in the results section we tested both adam and sgd methods, and found adam to be a lot more helpful for receiving better accuracy, and so this is the optimization method used in our final model.

The best accuracy received with our final trained network on the test set of cifar10 is **82.953%**.

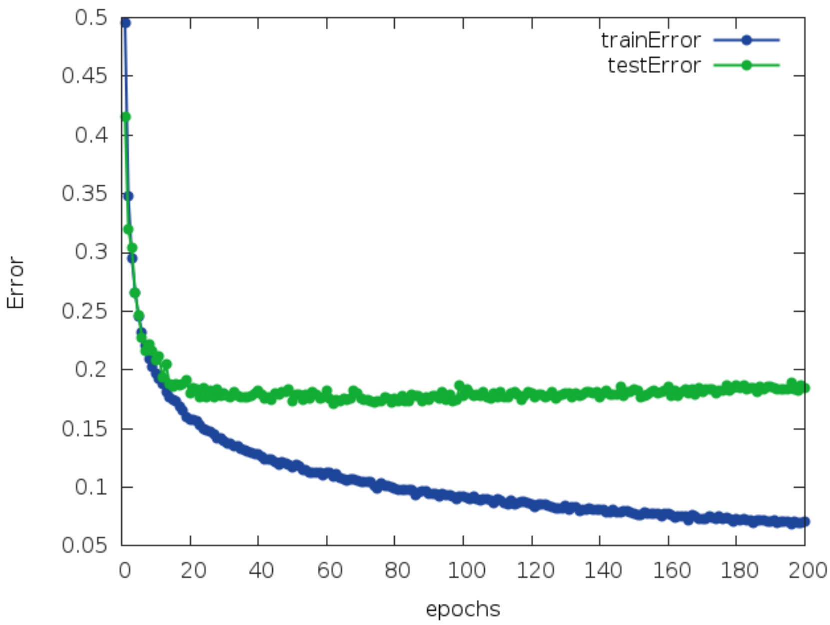
Results:

Results of final model:

Test VS train loss:



Test VS train error:



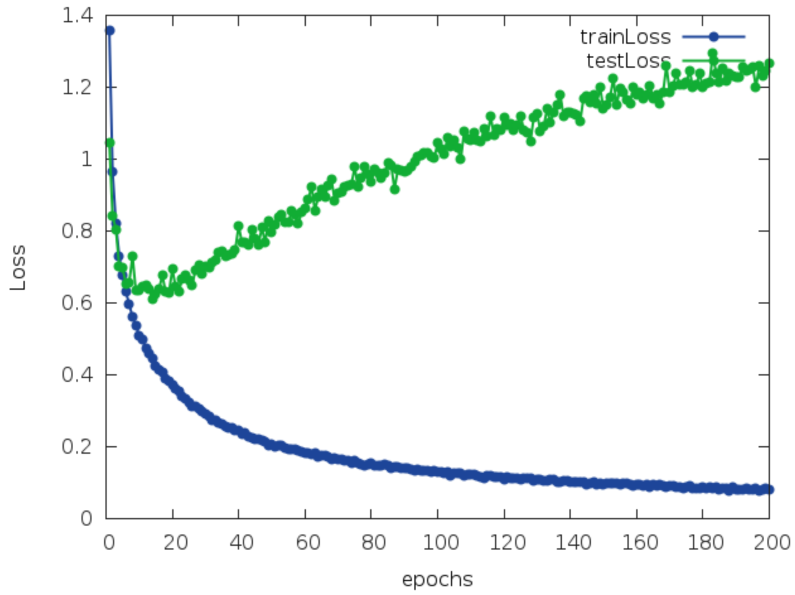
Evaluation of 2 optimization methods:

Adam:

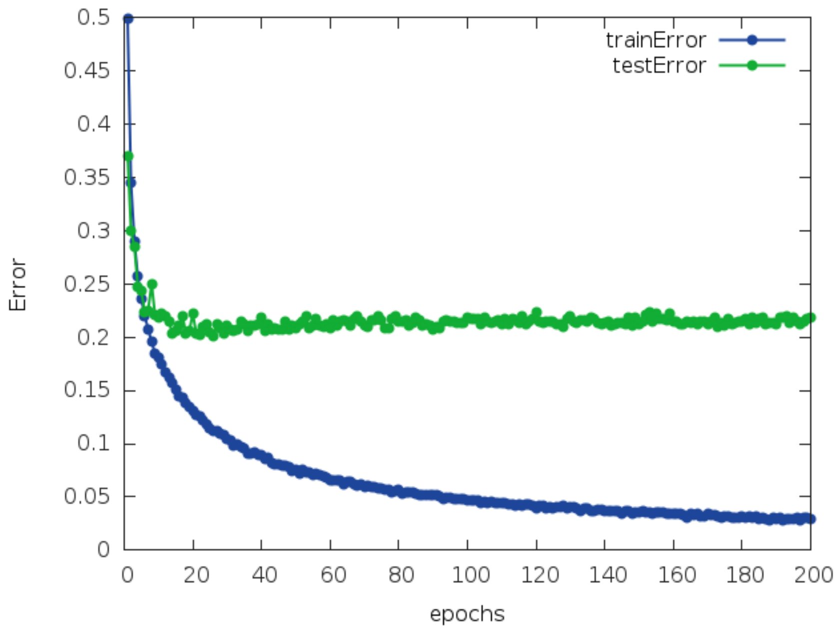
Evaluation of 3 data augmentation methods:

Horizontal Flip:

Test VS train loss:

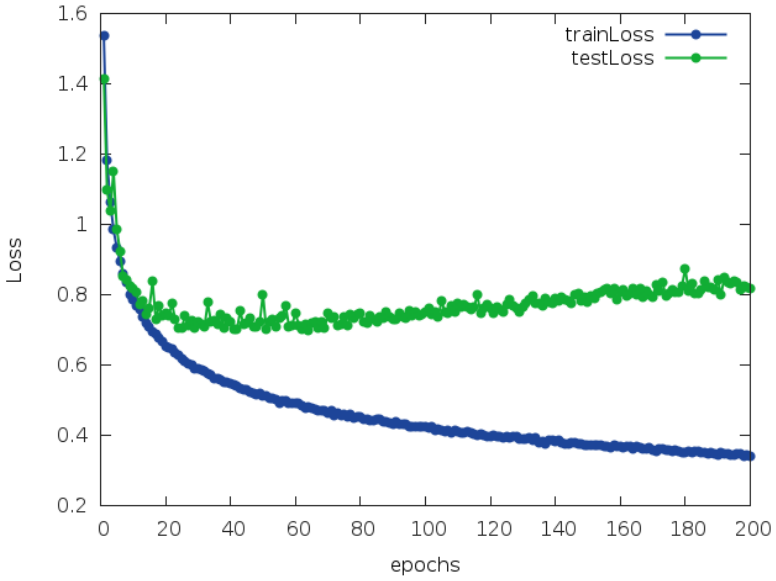


Test VS train error:

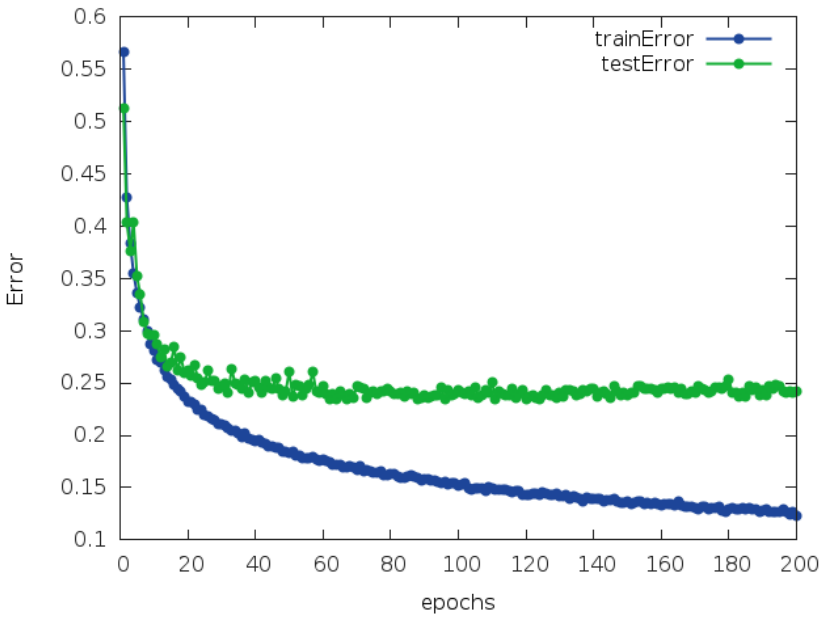


Vertical Flip:

Test VS train loss:

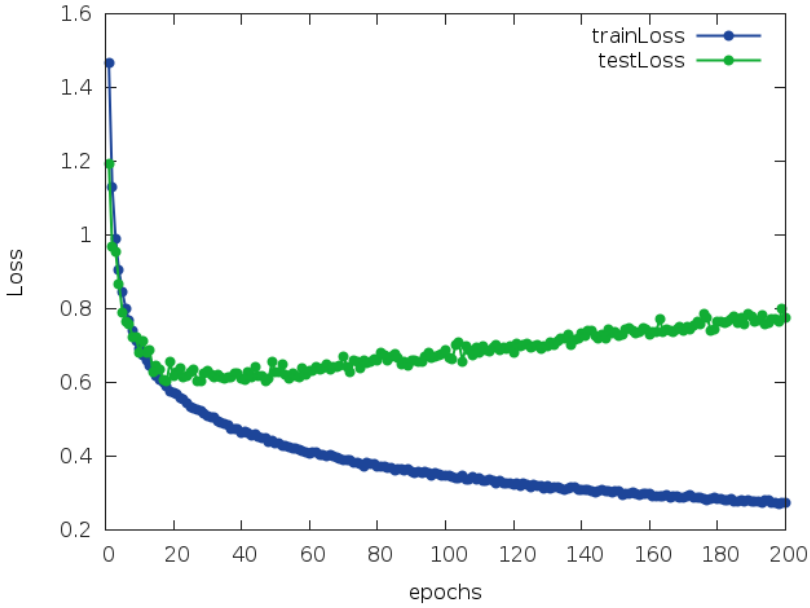


Test VS train error:

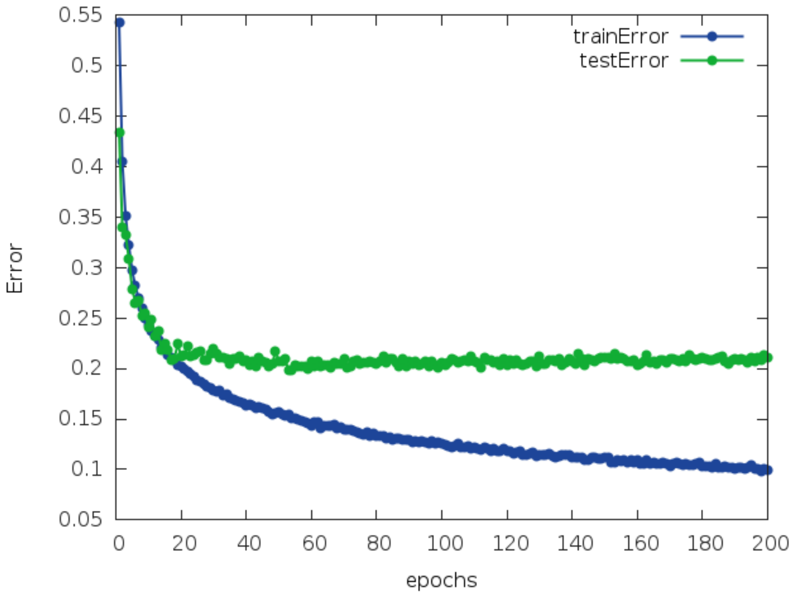


Rotate:

Test VS train loss:



Test VS train error:

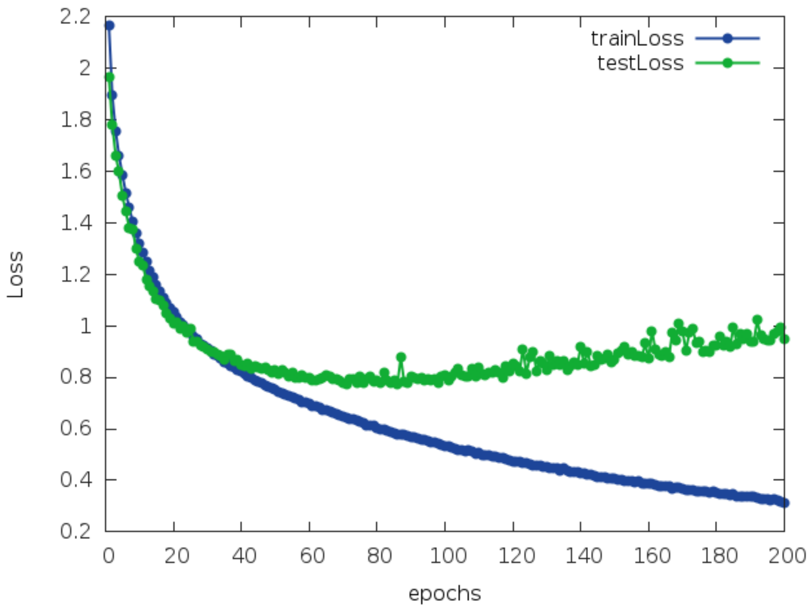


SGD:

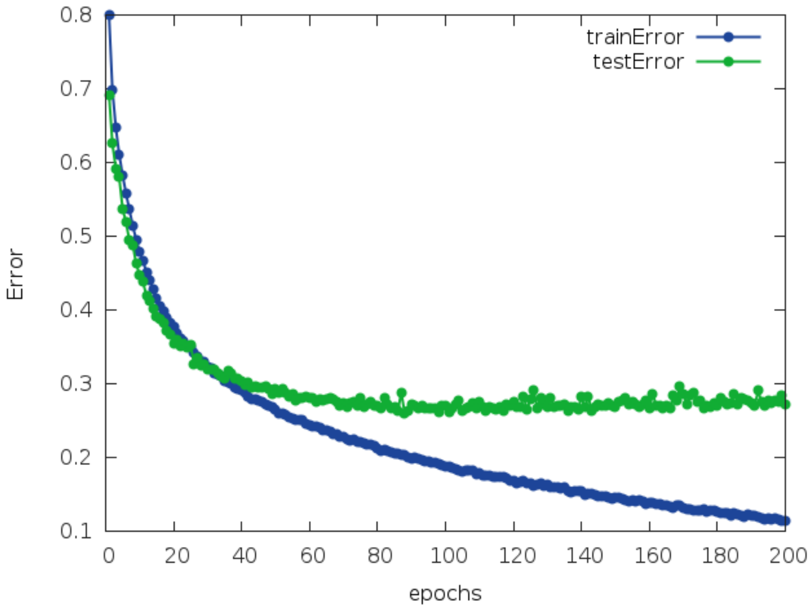
Evaluation of 3 data augmentation methods:

Horizontal Flip:

Test VS train loss:

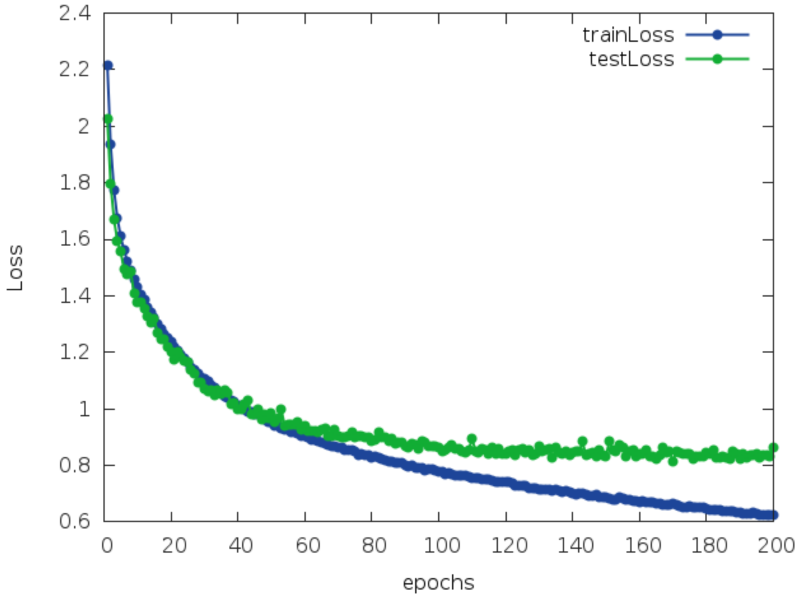


Test VS train error:

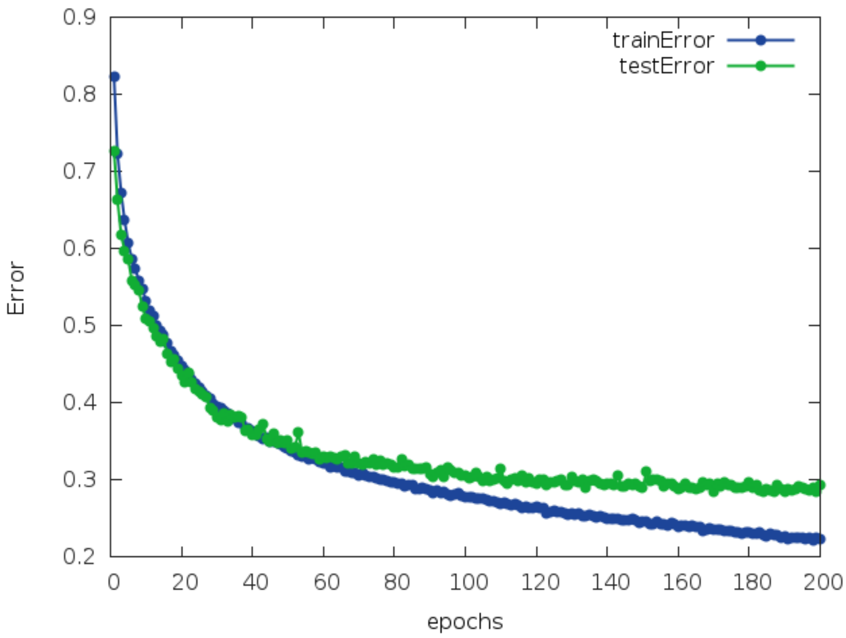


Vertical Flip:

Test VS train loss:

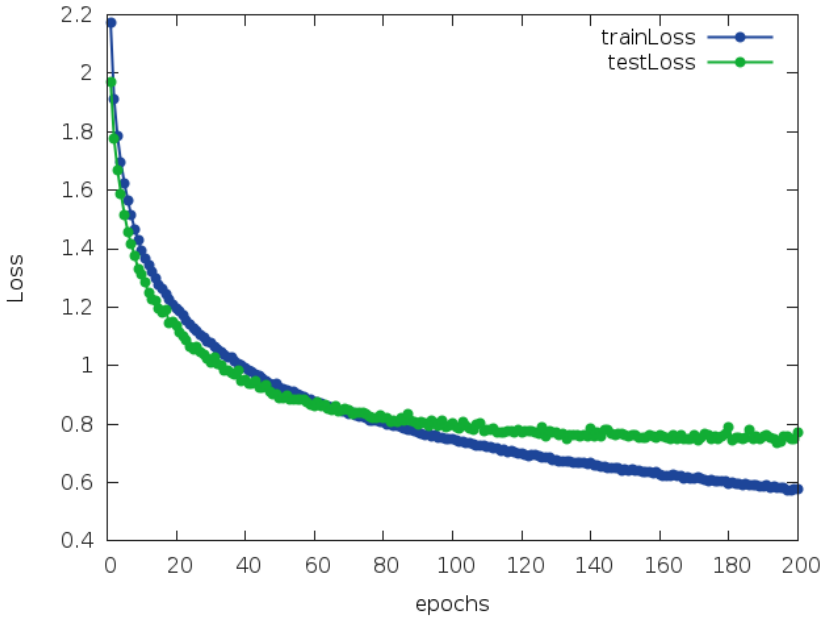


Test VS train error:

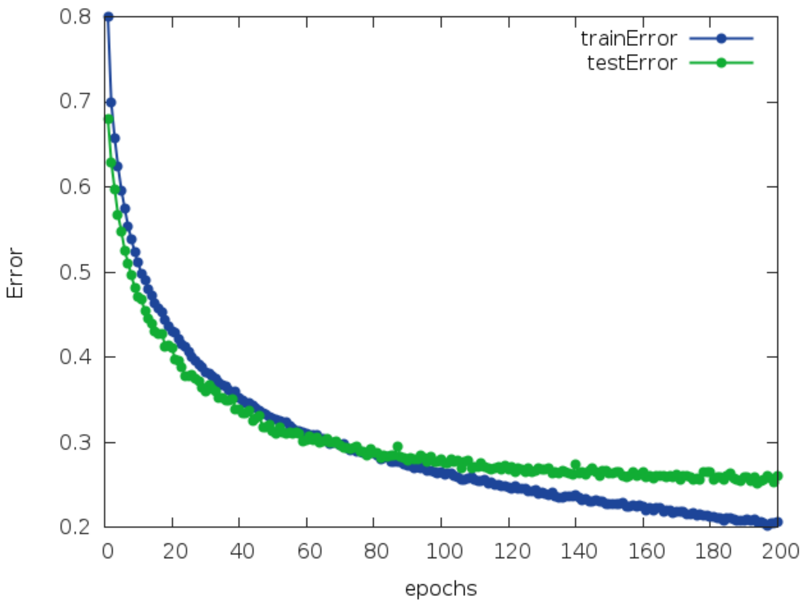


Rotate:

Test VS train loss:



Test VS train error:



Summary and conclusions:

Our final model concludes only horizontal flip as data augmentation and uses “adam” as an optimization method. Combined with small change in the way we ran the model, we got our final results.

All the results were evaluated using the same model architecture and training procedure. Using other networks may produce different results.

Vertical flip doesn't seem to have a big impact on improving our results, and they even got worse. Possibly because it’s less reasonable that objects appear upside-down in test set (e.g photo of ship never be with its mast downwards).

Rotation could be helpful for getting better results, but then raises the question what will be the size we use to rotate. For better handling in our code we chose to use only horizontal flip as data augmentation.