Machine Learning 2 HW 2 Q1

CNN

In this part I've started out from the architecture we were introduced to in the tutorial and continued from there.

The first change I've applied is increasing the number of **convolution layers to 4** and on top of that I've put them on the same sequential function. The other sequential function took care of transforming the convoluted output to a fully connected list.

I've wrapped both sequentials in a single Forward function.

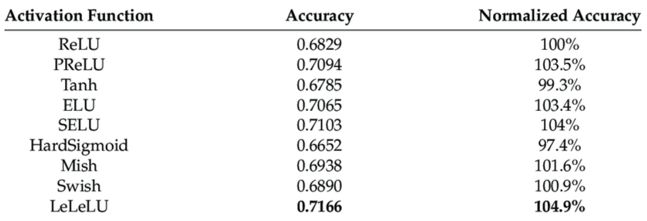
Layer-wise I did the following:

Added a dropout function between the third and fourth layer.

Played around with different final convolution out-channels until I've reached good performance resulting in the final **dimension of 95** without reaching 50K parameters for the network.

In addition I've changed the activation function from ReLU to **PreLU**.

I've decided to take the last step after finding a publication in regard to the best activation function for cifar10[1],I've tried the other options but in my case PReLU preformed the best.



Activation Function	Accuracy	Normalized Accuracy
ReLU	0.6829	100%
PReLU	0.7094	103.5%
Tanh	0.6785	99.3%
ELU	0.7065	103.4%
SELU	0.7103	104%
HardSigmoid	0.6652	97.4%
Mish	0.6938	101.6%
Swish	0.6890	100.9%
LeLeLU	**0.7166**	**104.9%**

Training-wise I've opted to use a few regularization functions:

I've added **learning rate scheduling** which will change the rate after every batch.

A **weight decay** was added to the optimizer in order to prevent the weights from becoming too large. **A gradient clipping** was added for the function of limiting the range of the gradient for the purpose of avoiding undesirable large changes to the model parameters.

During my attempts I've tried to use a smaller number of layers but always hit an accuracy wall at around 77%.

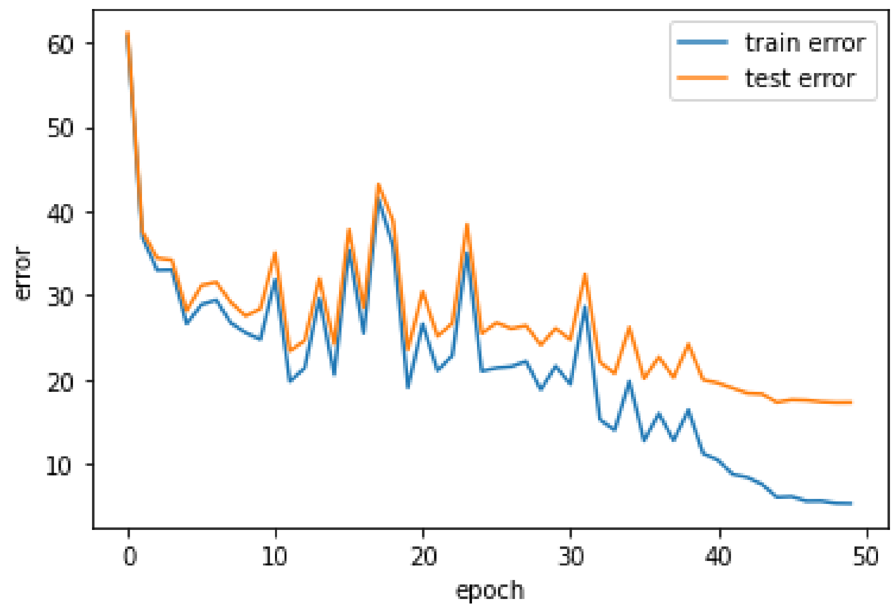
My current **batch size is 400**, lower sizes also gave out a low accuracy model.

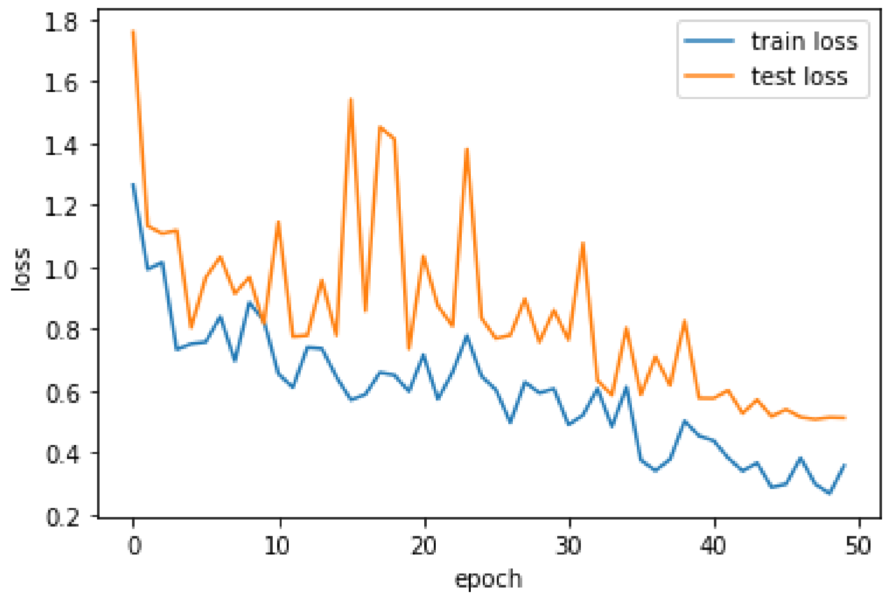
I've concluded that the best optimizer in my case is the Adam optimizer although a couple of sources on Google stated that the best one is Adagrad which in reality only lowered the accuracy of the model.

[1]Maniatopoulos, Andreas-Antonios & Mitianoudis, Nikolaos. (2021). Learnable Leaky ReLU (LeLeLU): An Alternative Accuracy-Optimized Activation Function. Information. 12. 10.3390/info12120513.

The lowest error I've reached is 17%

Here are the performance graphs for the train and test sets:


train error
test error
error
epoch


train loss
test loss
loss
epoch