NN for Images – Ex1

David Guedalia

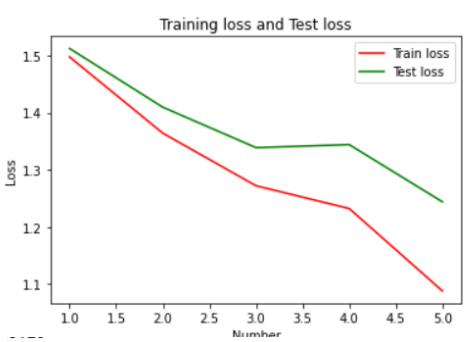
1. Programming Task

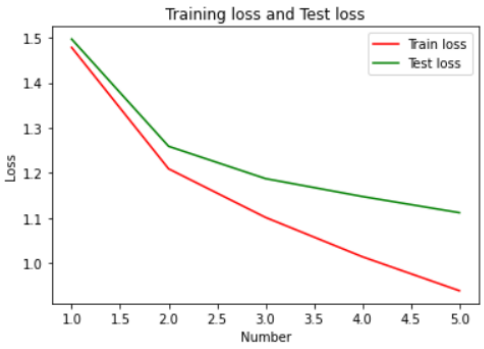
1.1 Architecture

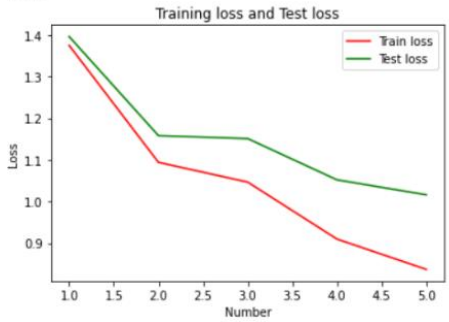
- On this section I trained 8 different networks of basic architecture with different numbers of filters.

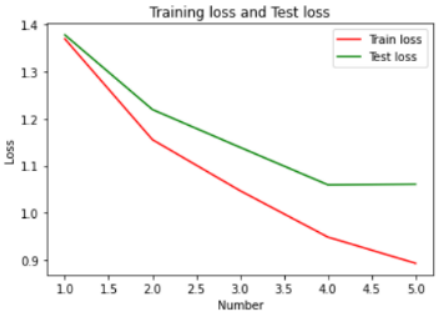
  Where the sizes were $N \in \{2, 5, 10, 15, 20, 25, 30, 35\}$

  The change was of the number of filters of the first convolution layer.
- I also changed the number of epoch on the train network to 5 as I saw it changes the loss and the accuracy as we can see on the next graphs:



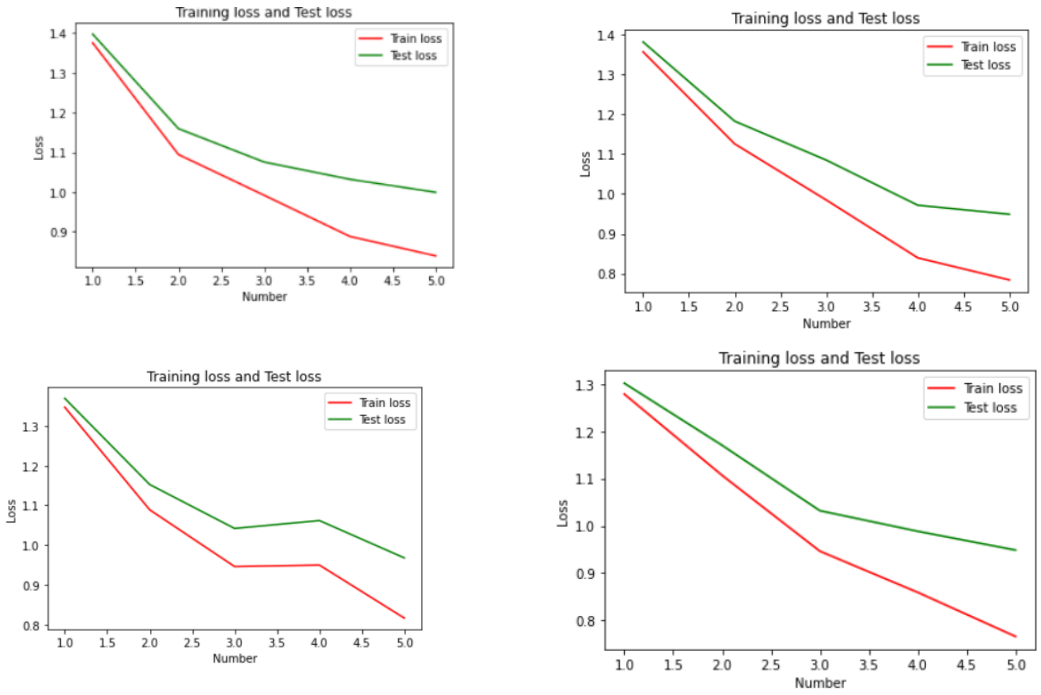






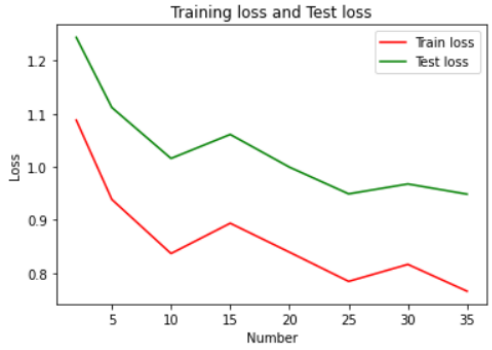
We can see that as the number of epochs grows the loss is getting lower but the differences between the train loss and the test loss is bigger what can show overfitting because we gwtting small loss on the train set but on the test set we getting high loss.

We can see the folloes graph that the filters are bigger:



- We can see from the plot, that if the network is too complex it gets overfitting on the other hand make it too simple we getting
- For every network I saved the train loss and the test loss and plot it as function of *N* and can analyses the best of filters.
- We can see that if conv1_out=.25. we get the best results i.e. train loss ≈ 0.8 and test loss ≈ 0.95. (The big difference is because it is average and there were lots of run with high epoch that made the overfitting)

  We also can see when conv1_out=15 We get a pick higher of loss.



1.2 Linear Model vs. Non-Linearity

I created a linear model that is like the first section architecture just omitted the non-linear layers, were I replaced Max Pooling with Average Pooling, as average pooling is linear.

Make these changes made the net decreased drastically – this can be explained as we removed the non-linear layers, so basically, we have a linear model and adding more wights doesn't make it higher expressiveness.

When I try to change the number of FC it didn't help to the model.

I took the best model – with 2 epoch and conv1_out=.25 from the first section and I get these results:

The non-linear model:

Train loss: 1.045

Test loss: 1.195

Accuracy: 58.71

The liner mode:

Train loss: 1.790

Test loss: 1.835

Accuracy: 38.69

Also, I try with a conv1_out= 100 linear model but also didn't help and I got these results:

Train loss: 1.760

Test loss: 1.84

Accuracy: 39.63

As I explained before as all layers are linear, they can replace to one linear operator that will be the same model, and the non-linear operations as activations function express new set of functions and cannot be replace with a linear operation.

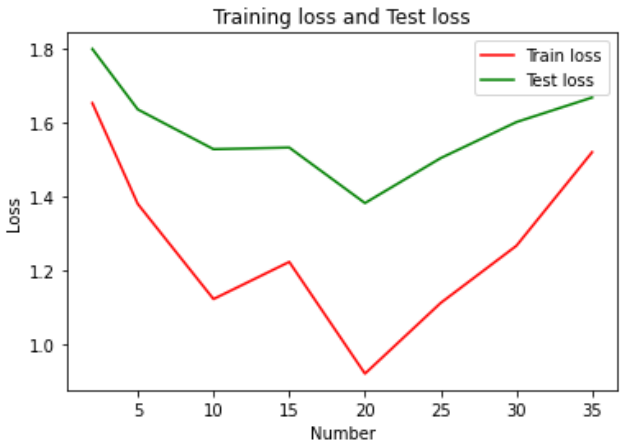
1.3 Cascaded Receptive Field

In this section I implement both ways of the shallower network and had bad results comparing the net on section 1, the reason of that is that as the net is bigger it can be more complex function and also creates features with much more scales.

In case of images, we can think of various images that can represent the same object such as blurring image or close/far image different color but the same shape etc.

So we need a deep net to identify object in various scales.

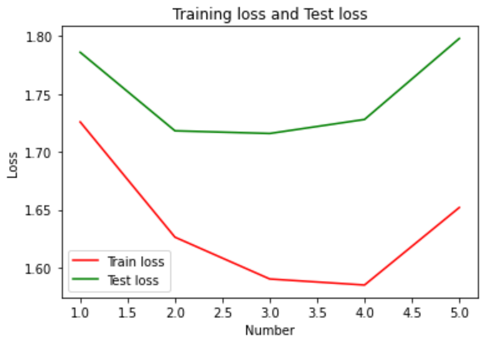
This graph is the number of filters as function of the loss:

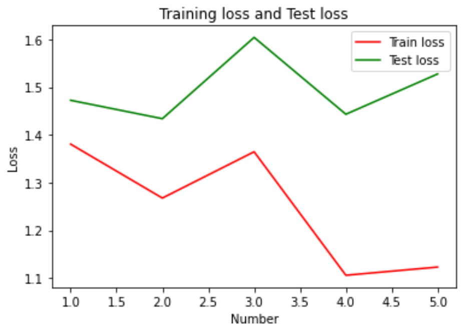


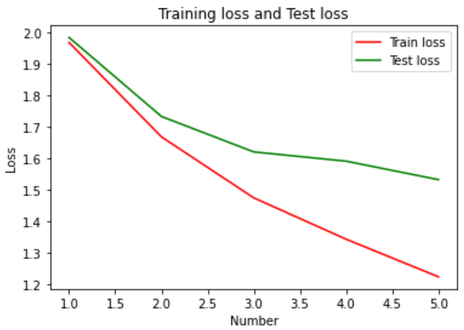
We can see as more filters is not reduce the loss but we can notice as more filters the difference between the test loss and the train loss becomes smaller.

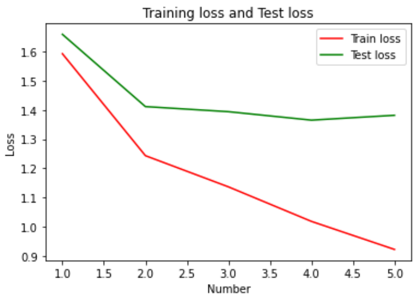
Also, we can still see that the loss is much higher than the first section.

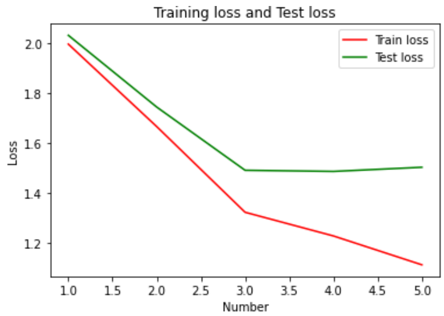
I add plots of different sizes of filters.

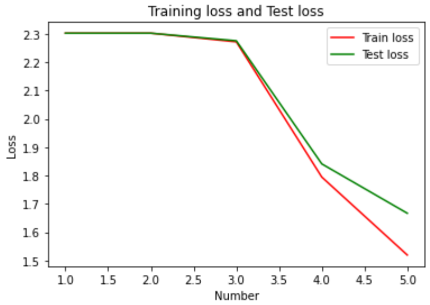












2. Theoretical Task

Question 1:

Question 2:

When we reshaping a 2D activation map and feeding it into an FC layer the order of the input is no matter, but it needs to be consistent between the train and the test data, and FC takes all connections between the previous layer and the output layer, and that's why neuron in 2D activation can be anywhere.

Although FC layers use independent weights, shuffle the input we will get different order of the learned weights, but it is not meaningful change

Question 3.a:

ReLU activation is not LTI, and I will show it with simple example:

$$0 = ReLU(0)$$

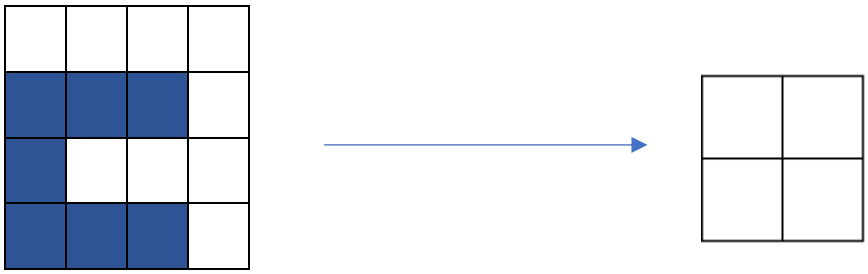
$$= ReLU\big(1 + (-1)\big) \neq ReLU(1) + ReLU(-1)$$

$$= 1 + 0 = 1$$

So we conclude is not ReLU sctivation.

Question 3.b:

Strided pooling layer is not LTI, although is linear but not translation-invariant, I will show it with example of strided pooling:



And it easy to see that with max pooling we getting the same results when we translated the pixels.

Question 3.c:

The bias is not LTI, if we will add $bias \neq 0$ to a zero vector we get vector $[b, \dots, b]$, but we know that linear operation goes from zero to zero so we can conclude that adding bias is not linear so it is not LTI.