Ivancich Stefano 1227846

Neural Network and deep learning course 2020/21
Homework 1

---

1. Regression Task

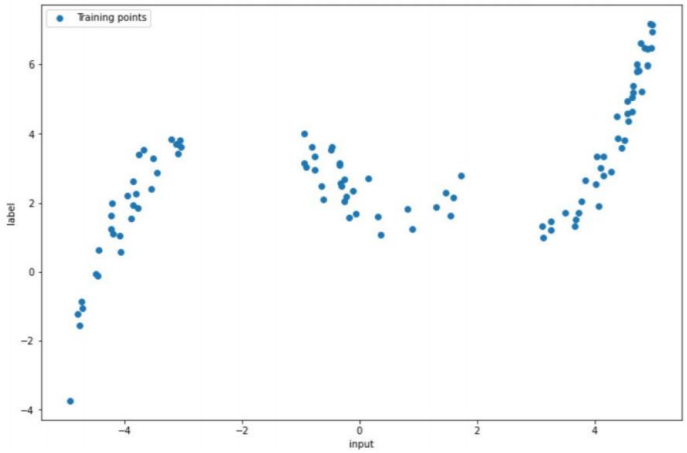
The goal is to train a neural network to approximate an unknown function:

$$f: \mathbb{R} \to \mathbb{R}$$
$$x \to y = f(x)$$
$$network(x) \approx f(x)$$

As training point, it's only given a noisy measure from the target function.

$$\hat{y} = f(x) + noise$$

The training set its composed of only 100 points in the range of $(-4.92, 4.97)$ for the x coordinate and $(-3.74, 7.19)$ for the y coordinate.
This information tells us that we probably have to use cross validation in order to avoid overfitting because the training set is very small and that we can't use tanh or sigmoid as activation functions in the output layer of the network because the output can be greater than 1.
This was the training set given:



Network architecture: we choose to use a network with 3 fully connected layers.
A **grid search** on the following hyperparameters with a cross fold validation of 3 folds was ran:

- **First layer number of neurons:** 8, 16, 32, 48
- **Second layer number of neurons:** 8, 16, 32, 48
- **Third layer number of neurons:** 1 (=output)
- **Layers activation:** ReLu or no activation (because the output can be grater that 1 so we cannot use sigmoid or tanh in the last layer)
- **Optimizer:** Adam
- **Learning rates:** 0.1, 0.01, 0.001
- **Regularization:** "L2" with values 1e-3, 1e-4, 1e-5 and 0 (no regularization)
- **Max epochs:** 3000 (we did not choose to tune this value, because the early stopping will take care of it)
- **Early stopping:** 100 epochs without improvement

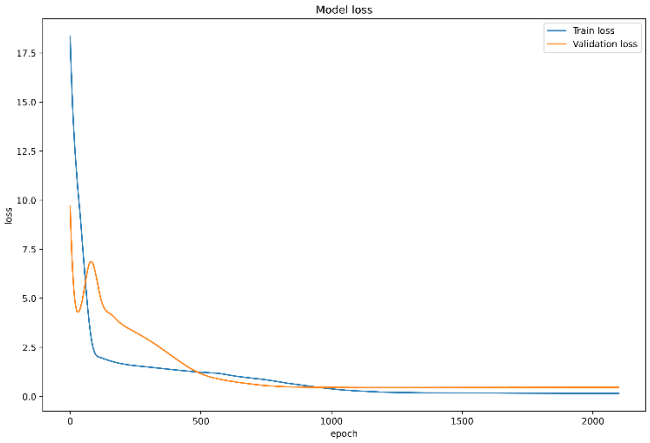
The validation errors of all training were saved, and the "best network" was chosen to be the one with smallest average validation error.

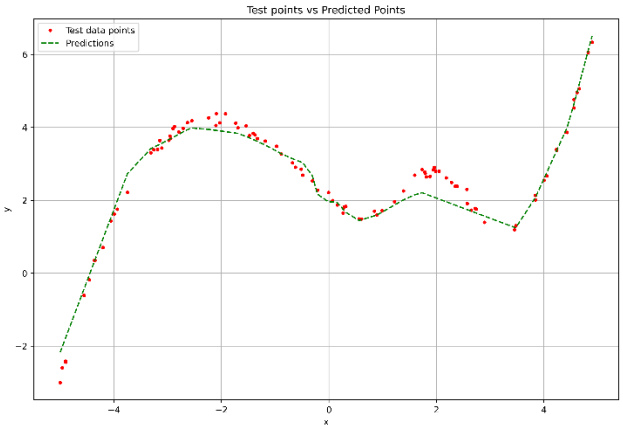
The **best hyperparameters** turned out to be:

- **First layer number of neurons:** 32
- **Second layer number of neurons:** 32
- **Layers activation:** ReLu except for the last layer that has no activation
- **Optimizer:** Adam
- **Learning rates:** 0.001
- **Regularization:** 1e-5 (L2)

This NN was trained again using the whole train dataset with a splitting of 80-20 train-val, giving the following results:

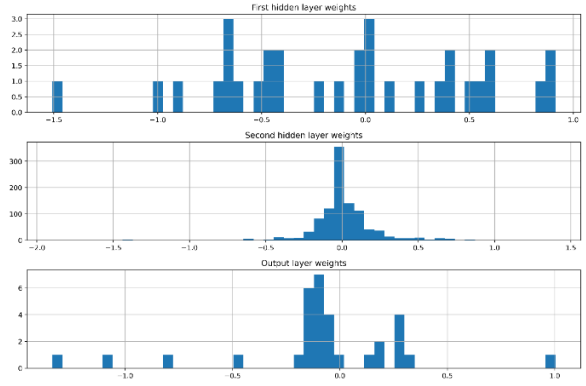
- **Train Loss:** 0.159
- **Val Loss:** 0.457
- **Test Loss:** 0.11



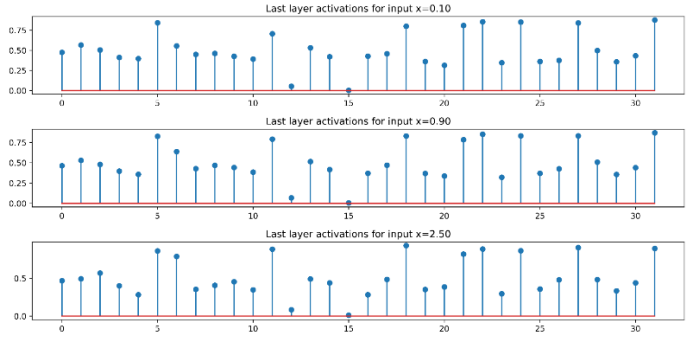


As we can see the model has problems predicting the points around $x = -2$ and $x = 2$, that is because in the training set those range of points are missing, and given the fact that the Train loss and the test loss are quite near to each other, this implies that this model is generalizing the training data quite well and probably with this kind of architecture we can't get much better results.

The following **Weights histograms** tell us that the weights are in an acceptable range. (Not vanishing or exploding)



Finally, the **layer activations** for different parameters are telling us that all neurons are being used, so probably would be difficult to use a smaller network.



2. Classification Task

Here we are facing a supervised classification problem, whereas our input (handwritten digits) must be correctly classified as a number between 0 and 9.

We tried 2 different **network architectures**:

- The same 3 FC layers network of the regression task
- A network composed by 2 Convolutional layers both followed by a max pooling layer and a the end 2 fully connected layers.

2.1. Fully connected network

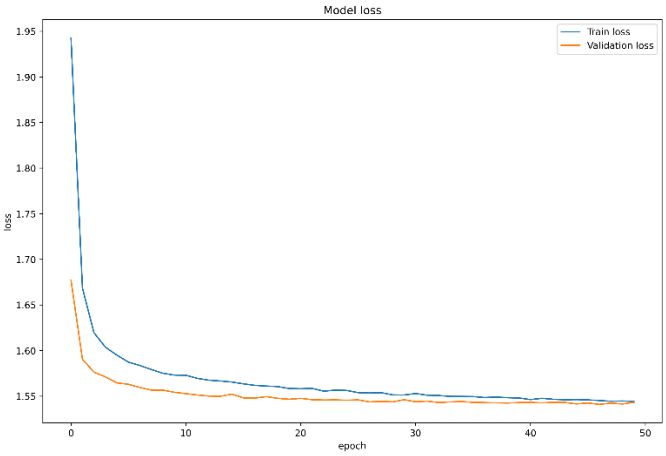
The 3FC network was trained exactly like in the regression task except for the cross validation that was not used in this case, since we have 60.000 images, we did not think that we can overfit. See above for the hyperparameters details.

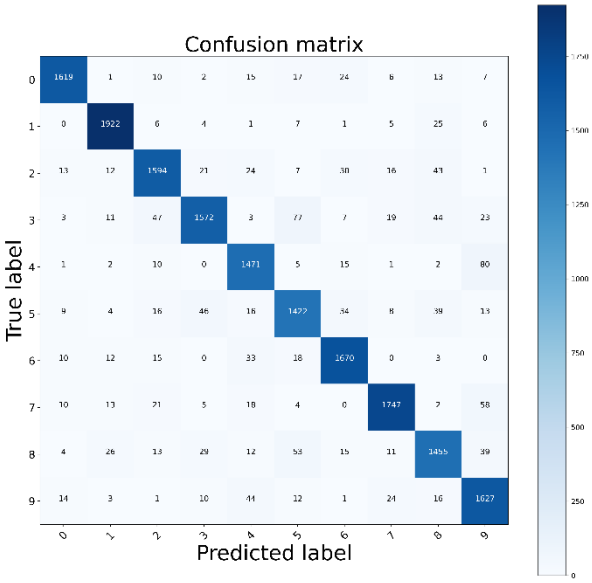
The best hyperparameters for the 3FC network turned out to be:

- **First layer number of neurons:** 8
- **Second layer number of neurons:** 48
- **Layers activation:** ReLu except for the last layer that has no activation
- **Optimizer:** Adam
- **Learning rates:** 0.001
- **Regularization:** 1e-5 (L2)

This NN was trained again using the whole train dataset with a splitting of 80-20 train-val, giving the following results:

- **Train Loss:** 1.5442
- **Validation Accuracy:** 0.919
- **Test Accuracy:** 0.92

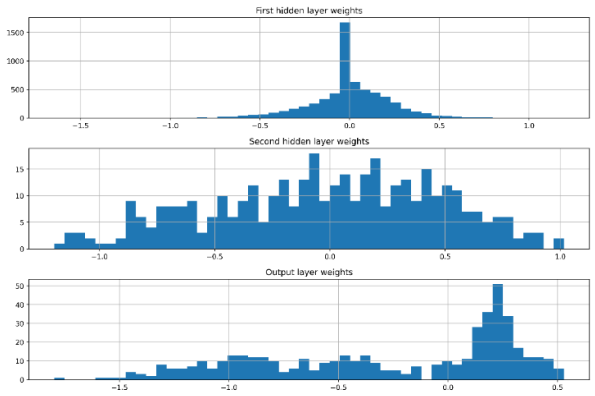




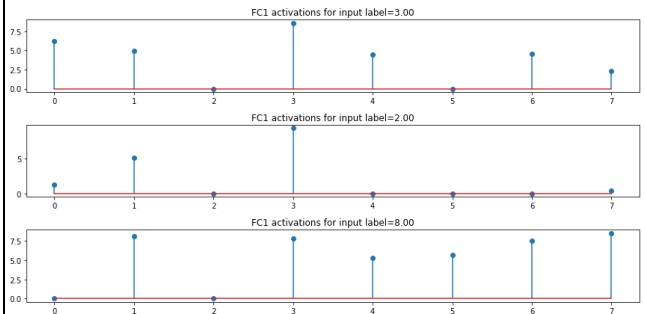
Most mispredicted labels:

- 124 errors: 4 - 9
- 123 errors: 3 - 5
- 92 errors: 5 - 8

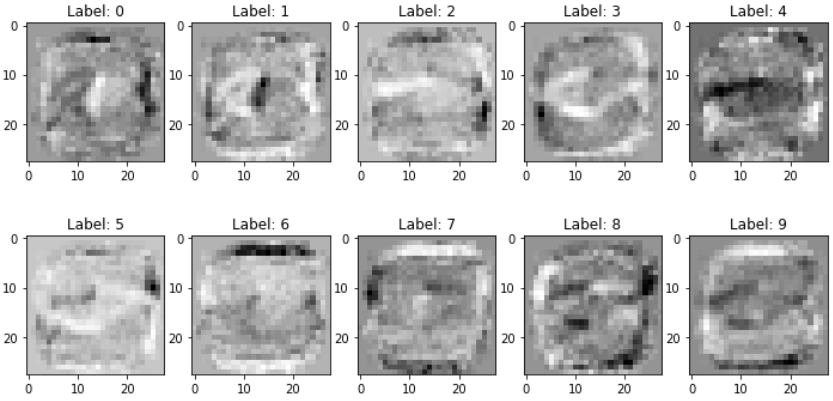
Weights histogram: are in an acceptable range. (Not vanishing or exploding)



Analyzing Activations: for different input images are telling us that all neurons are being used, so probably would be difficult to use a smaller network.



Receptive Fields: of the output layer were calculated by multiplying the first layer weight for the second layer weight and again for the last layer weight so the result would be a 784 array which then was reshaped in 28x28 matrix and then plotted.
As you can see in the fully connected layer network there are no clear shapes, nevertheless we can reach 92% accuracy.



2.2. Convolutional Network

Composed by 2 Convolutional layers both followed by a max pooling layer and at the end 2 fully connected layers.
The convolutional network hyperparameters for the training are the following:

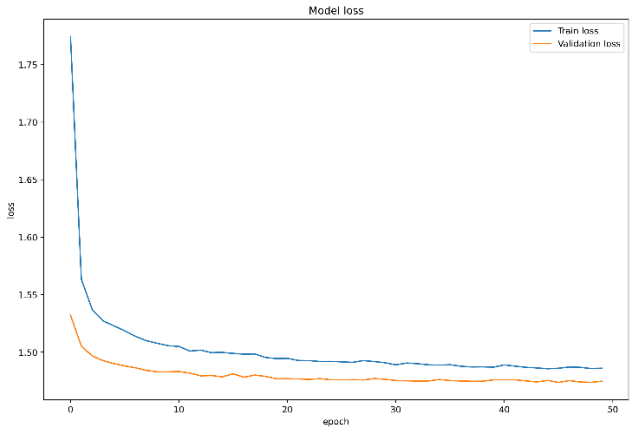
- **Conv1:** 32, 64, 128 filters of size 3x3
- **Maxpool2d_1:** size 2
- **Conv2:** 32, 64, 128 filters of size 3x3
- **Maxpool2d_2:** size 2
- **FC1 number of neurons:** 32,64,128 with dropout 0.5
- **FC2 number of neurons:** 10 (=output)
- **Layers activation:** ReLu for all except the last layer that is Softmax.
- **Optimizer:** Adam
- **Learning rate:** 0.1, 0.01, 0.02, 0.001
- **Regularization:** "L2" with values 1e-3, 1e-4, 1e-5 and 0 (no regularization)
- **Max epochs:** 3000 (we did not choose to tune this value, because the early stopping will take care of it)
- **Early stopping:** max 10 epochs without improvement

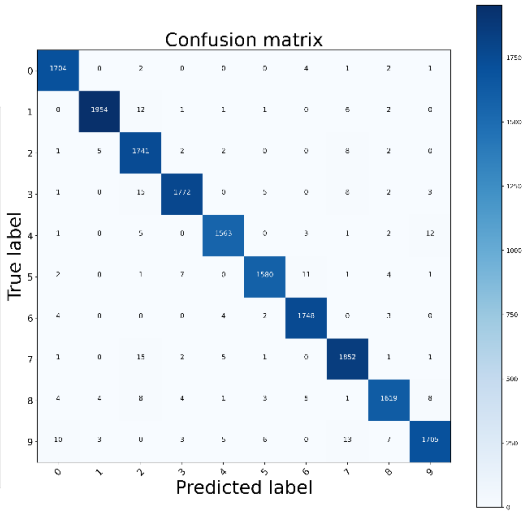
The best hyperparameters for the CNN network turned out to be:

- **Conv1:** 32 filters of size 3x3
- **Conv2:** 64 filters of size 3x3
- **FC1 number of neurons:** 64
- **Optimizer:** Adam
- **Learning rate:** 0.001
- **Regularization:** 1e-5 (L2)

This NN was trained again using the whole train dataset with a splitting of 80-20 train-val, giving the following results:

- **Train Loss:** 1.486
- **Validation Accuracy:** 0.987
- **Test Accuracy:** 0.985



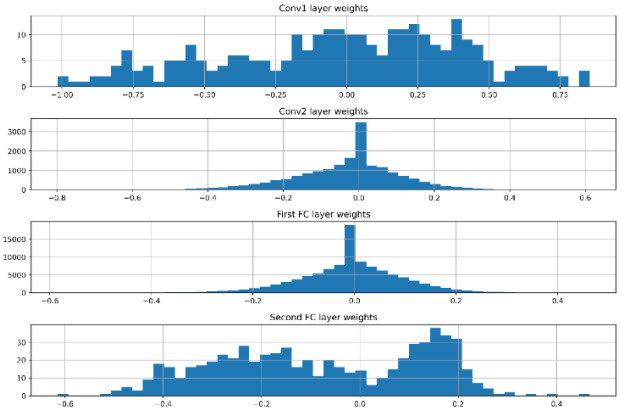


Most mispredicted labels:

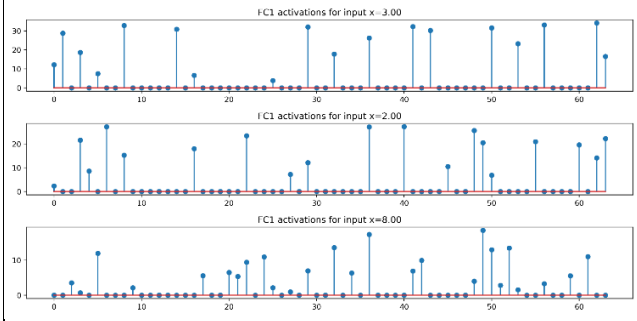
- 23 errors: 2 - 7
- 17 errors: 4 - 9
- 17 errors: 2 - 3

As you can see there are much less mispredictions than the Fully Connected network and has much better accuracy.

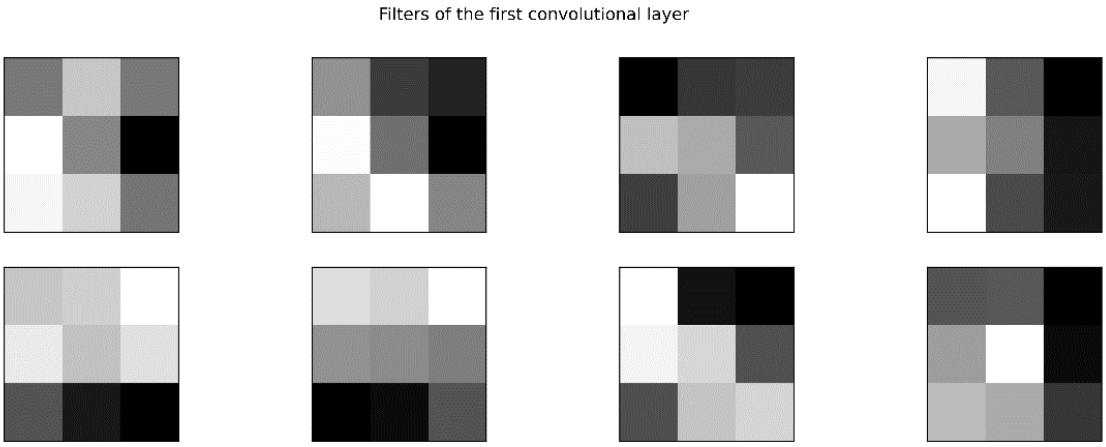
Weights histogram: are in an acceptable range. (Not vanishing or exploding)



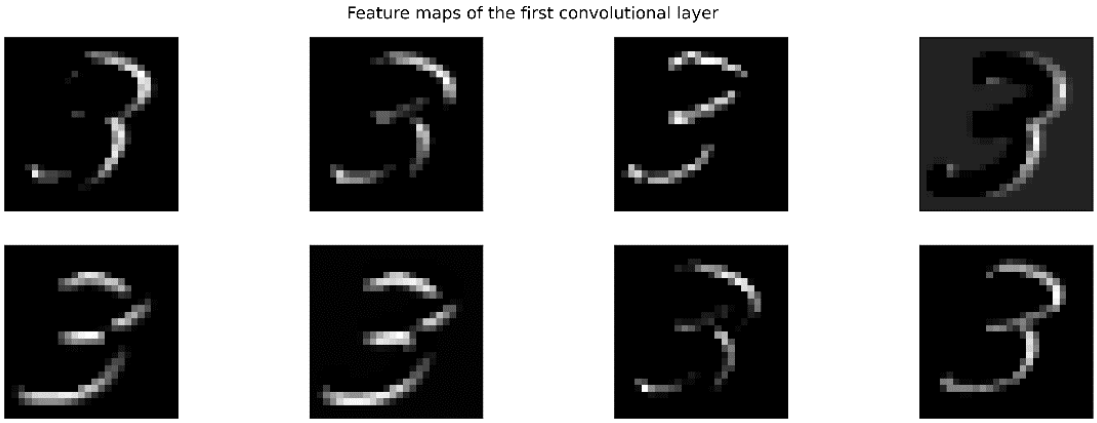
Analyzing Activations: for different input images are telling us that all neurons are being used, so probably would be difficult to use a smaller network.
Here some example for the first FC layer.



3x3 Filters of the first convolutional layer:



Feature Maps of the first convolutional layer, give us the internal representation of the input image after applying the 8 filters shown above.



Receptive Fields of the output layer shows clearly which neurons are activated for the input image **"3",** while in the fully connected network there was not a clear shape.