Denis Deronjic 1231829

**HOMEWORK 1**

**REGRESSION TASK**

The problem of this task is to train a neural network that is able to approximate an unknown function f such that:

f: R -> R

Given a training set of 100 points with the domain [-4.92, 4.97] in x and [-3.74, 7.19] in y

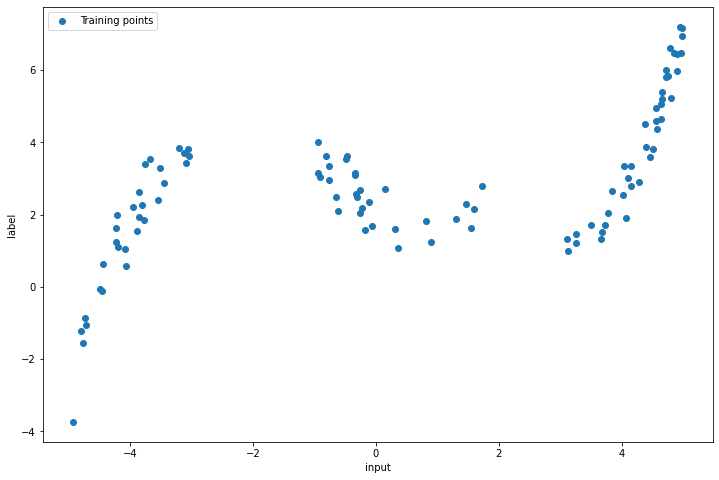


Figure 1

As we can see in the picture above, some data are missing around -2 and 2, so our regression model should be able to approximate the function in those points even though data are missing.

Due the lack of data, the neural network regressor should be trained with a cross validation technique. The model selection is made by Randomized Grid Search which is a computationally less demanding algorithm than the grid search.

The network is a feed forward neural network with 2 hidden layers. The number of input and output is 1 since the function to approximate is R->R. The hidden layers are composed respectively by 512 and 256 neurons. The activation function used in the network is tanh except in the output layer that has no activation.

The loss function used is Mean Squared Loss and the optimizer used is Momentum SGD. Since the data are small and the network is not too deep, momentum optimizer can be used here even though it’s slower compared with advanced optimizers. Since the SGD optimizer has an higher stocaticity, it can potentially be able to find a lower loss solution than any other optimizer.

The cross validation and grid selection are made by skorch library, which is a wrapper of sklearn for pytorch. 5 cross validation kfold are choosen.

The hyperparameters to optimize are:

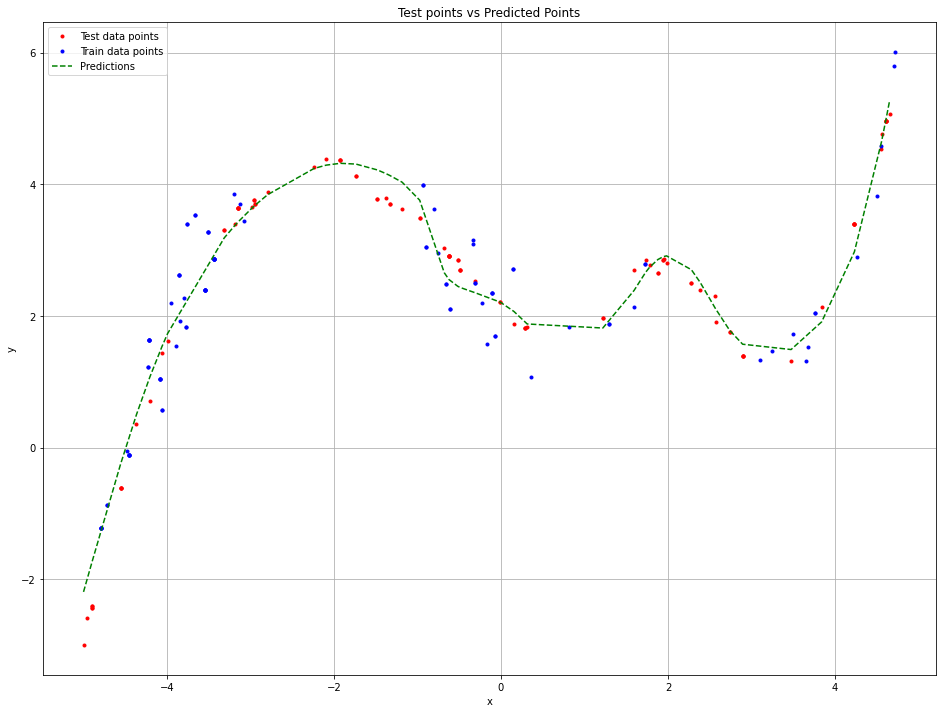
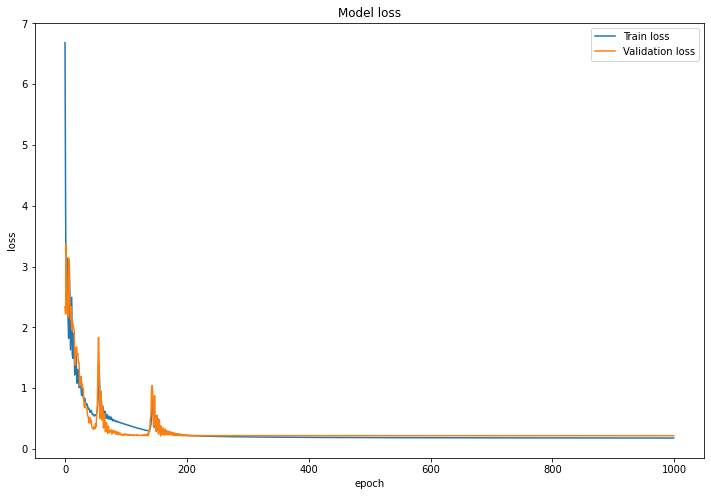
* Optimizer’s learning rate
* Optimizer’s momentum
* Number of epochs
* Optimizer’s weight decay

The best hyperparameters found are:

* Optimizer’s learning rate = 0.01
* Optimizer’s momentum = 0.9
* Number of epochs = 1000
* Optimizer’s weight decay = 0.001

Train Loss: 0.186860

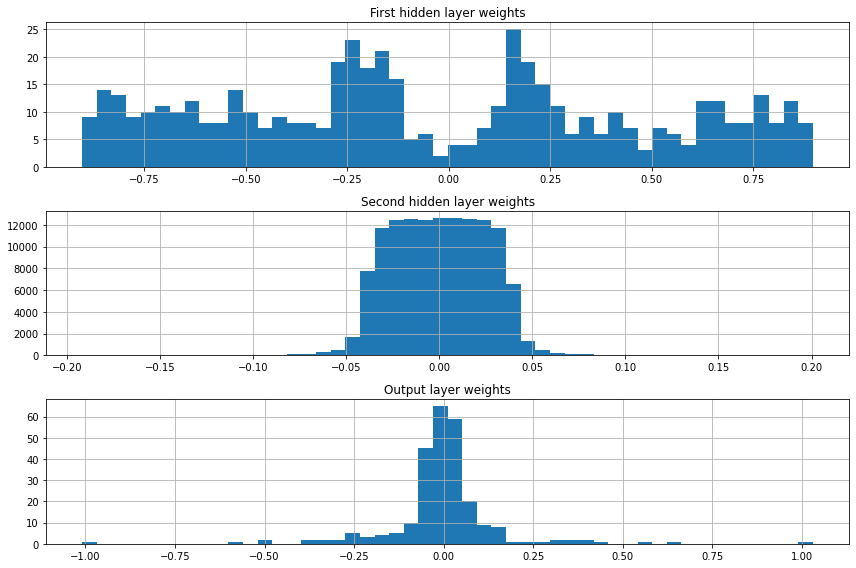
Test Loss: 0.109783



The model generalized well the function as it can be seen above. Looking at the loss function the model doesn’t not overfit and the model can predict the values around the critical points -2 and +2. This is a great insight since we can suppose that the prediction could be get more accurate with a better hyperparameters tuning or using a different model.

In the following weight histogram and activations of the second layer of the network given some inputs. The first hidden layer weights are almost equally distributed in the range of -1 to 1. This means that the first hidden layer doesn’t contribute too much on the prediction. The second hidden layer is distributed near 0, with a range of -0.05 to 0.05. The second hidden layer gives more contributes on predictions. The weights of output layer are distributed mostly near 0. It is expected since when a prediction is made, only 1 label of 10 is chosen, the not chosen layers tend to have a low probability (i.e. a value near 0).

Looking at layers activations we can see that almost all the weights in each layer are activated given an input.

Immagine che contiene testo, parete

Descrizione generata automaticamente

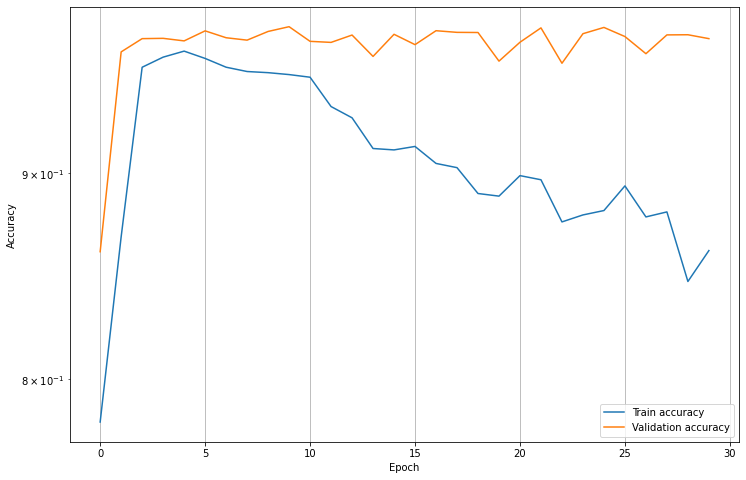
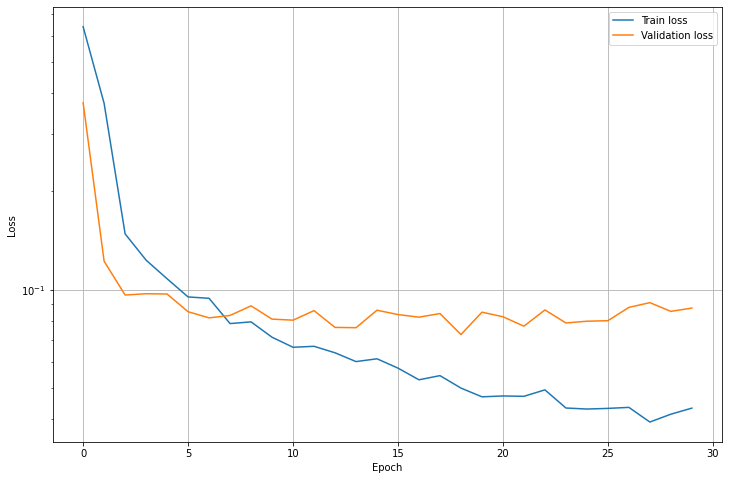
**CLASSIFICATION TASK**

The classification task is made using two different models: a feed forward neural network and a convolutional neural network.

**1 Fully connected network**

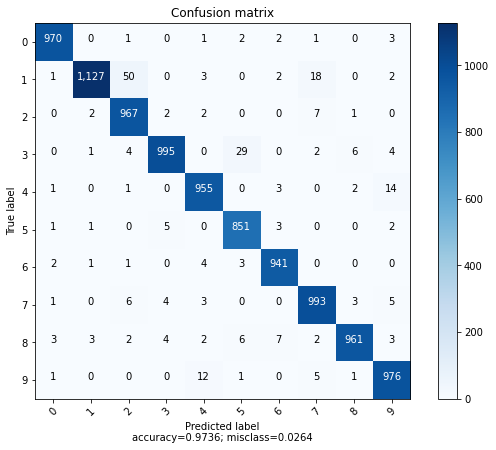
The architecture is a simple 3 hidden layer feed forward neural network with a dropout of 0.5 as a regularization method. The network was trained with a fixed validation set, an 80-20 train-val splitting.

It is used Adam optimizer with learning rate of 0.001 and the model is trained in 30 epochs.



In the left the train-val loss plot and on the right the train-val accuracy plot. The validation loss and accuracy don’t improve after 5-10 epochs. The training accuracy though is decreasing. This can be caused by the dropout layers.

The accuracy achieved in the test set is about 97.36%. Quite reasonable.



The receptive fields are calculated by matrix multiplication. Starting from the first layer, the matrix weights of the first layer are multiplied by the matrix weight of the second layer and so on to the output. Doing those multiplications, the matrix shape become [10, 1, 784]. In the following is possible to look the receptive fields for every label. A clear shape cannot be recognized.

Immagine che contiene testo, diverso, molti, lotti

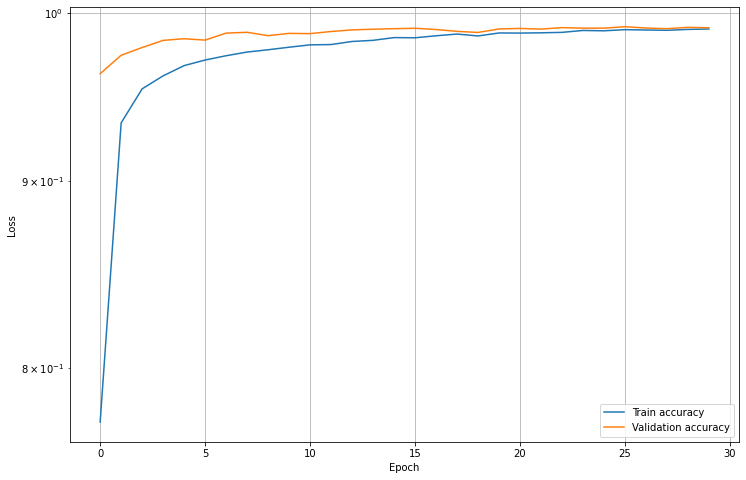
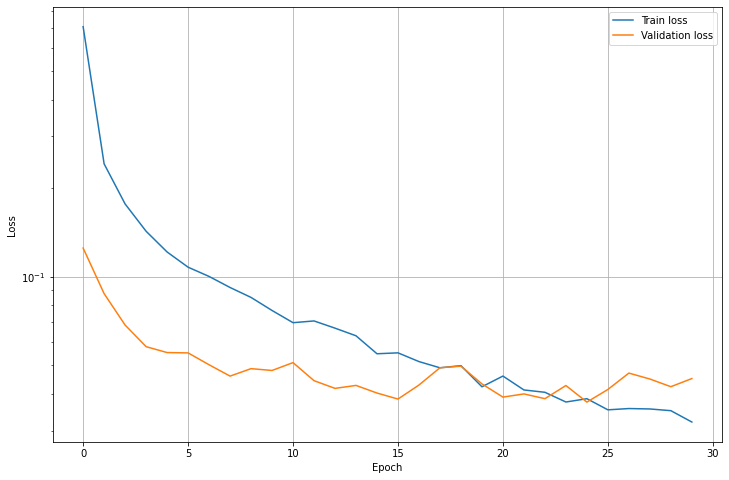
Descrizione generata automaticamente

**Convolutional neural network**

The architecture of the network implemented is the Lenet5 network with an addition of some dropout layers as regularization and kaming layer initialization.

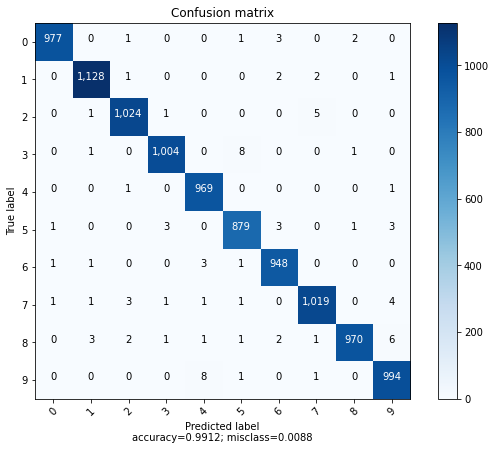
Like in the feed forward case, the network is trained using a 80-20 train-val split.

The loss used is the cross-entropy loss and the adam optimizer with learning rate of 0.001 is chosen.



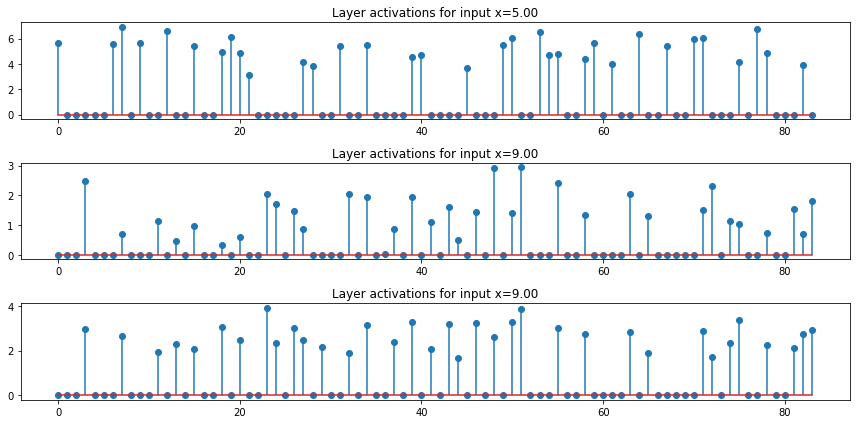
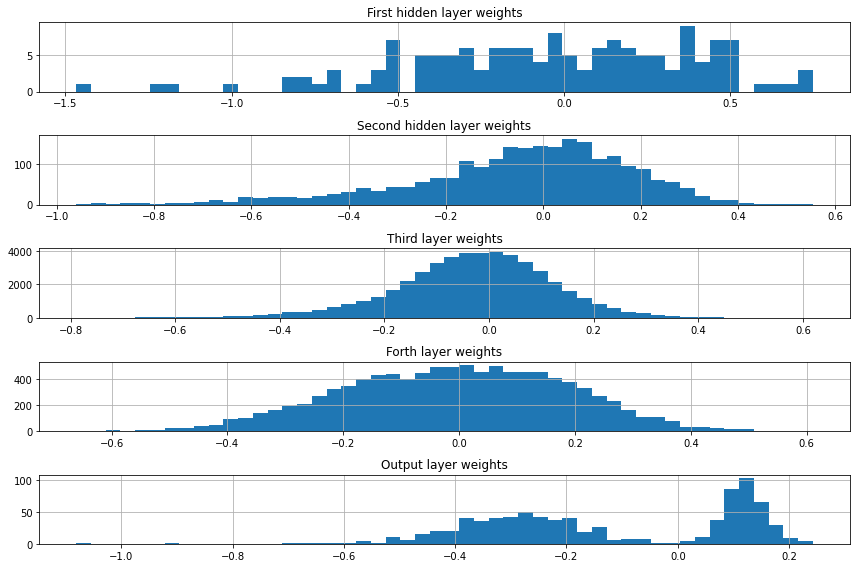
On the left the train-val loss plot, on the right the train-val accuracy plot.

The accuracy reached in the test set is 99.12% as you can see in the following confusion matrix.



In the following layers weight and the layer activation for the last feed forward layer in the network before the output. The same considerations as the fully connected layers can be done for weight histogram.

From layer activations we can see that given different inputs some neurons are activated and some neurons not. This behavior is achieved thanks to the parameter sharing.



In the following the 5x5 kernel sizes of the first and second convolutional layer and the correspondent feature maps of an image after applying the first and second layer’s kernels.

Immagine che contiene testo, clipart

Descrizione generata automaticamente

Immagine che contiene cruciverba

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo, kit da pronto soccorso

Descrizione generata automaticamente

The receptive fields are calculated applying a mask to the input image. This mask is calculated applying the gradient on a 1s image matrix based on the prediction that the network made of the input image. The receptive fields show the main feature of the image that the network recognized to make the prediction.

Immagine che contiene freccia

Descrizione generata automaticamenteImmagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene testo

Descrizione generata automaticamente