Supervised Deep Learning

Edoardo Antonaci 1234431
edoardo.antonaci@studenti.unipd.it

Abstract

In this report we face two supervised deep learning problems : regression and classification. For the first one we use fully connected layers in order to approximate an unknown function and for the second one we use a convolutional layers to recognize 0-9 hand-digits numbers.

1 Introduction

The goals of this homework are two:

1. Regression Task : it consists by using 100 points from training dataset and their relative labels y in order to approximate an unknown polynomial function f(x). For this task we explore different hyperparameters like numbers of epochs,regularizations and learning rate. This is done by using a gridsearch fixing the best layers configuration and its hidden neurons.
2. Classification Task: now we implement conv2D architecture + FC networks to recognize and classify hand-digit numbers from 0 to 9. Using the simplest architecture found,we also use a grid search to fix hyperparameters obtaining very good results.

2 Methods

2.1 Regression

For regression goal we use three FC hidden layers with ,respectively,100,120 and 100 neurons . For each output the ReLu activation function is applied, even if, attempts with Sigmoid function were also done.
As the first initialization of the network weights we use the default one even if both the sparse and the normal ones are possible for each FC layer
Fixing the number of layer,their neurons and Adam as optimizer we focus on hyperparameters like training epochs,learning rate and weight decay in order to know which combination fits better.
This goal is reached using a gridsearch and cross-validation technique : the training dataset is split in three batchs and ,by turn,one of these is used as validation test and the others as training.
Best configuration will be the model affected by minimum error as average of this three tentatives:

Module Nh1	Module Nh2	Module Nh3	Epochs	Learning rate	Optimizer	Weight decay
100	120	100	5000	0.0001	Adam	0

Table 1: Optimal parameters for the regression task.

A note must be done about earlystopping technique: if no validation error improvement is seen in a period of epochs,it stops the training procedure.
We choose 2000 epochs as "patience time" and the improvement has a threshold equal to 10^{-4}.
Finally training and validation behaviour loss graph is shown:

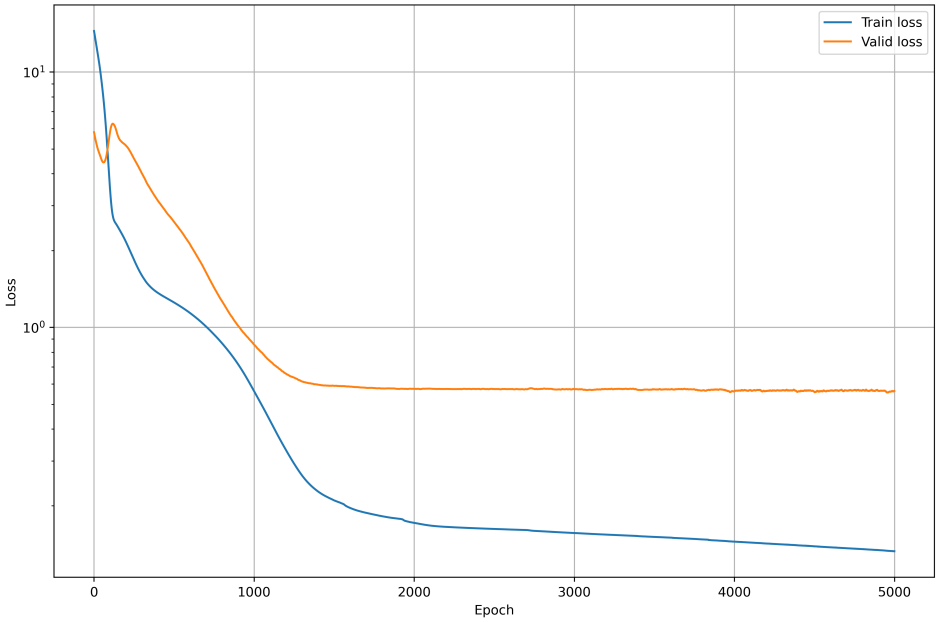


Figure 1: Training and loss regression graphs

2.2 Classification

In this case we try to classify the MNIST dataset using an architecture composed by : one conv2D layer followed by FC hidden layer. This choice is made for reason of simplicity : between all the models tested we select the simplest one found.
In the definition of the network we also apply different modules:

1. dropout : to disattivate neurons with a given probability
2. max pooling: to extract most relevant features given a specific window size.
3. ReLu and softmax activation function: the last one due to the fact the output should be like a "one-hot" encoder array (all zeros and 1 in j-th position) .

To not waste time and resources,as done in regression task, the early stopping is used with the relative tollerance of 1e-4 and 10 epochs as patience time.
To automatize the search of best hyperparameters we apply also the GridSearchCV, specifing different ranges like the numbers of channels after Conv2D,optimizer learning rate and weight decay. This time numbers of epochs is fixed and also other conv2D parameters like stride,kernel size and padding.
As in regression task ,the network weights initialization can follow normal or uniform distribution even if best results are reached by default normalization.
After the cross validation best model found is:

Module Conv2D	Module Nh1	Epochs	Learning rate	Optimizer	Weight decay
10	1440	1500	0.001	Adam	0.001

Table 2: Optimal parameters for the classification task.

We show training and validation loss for classification task:

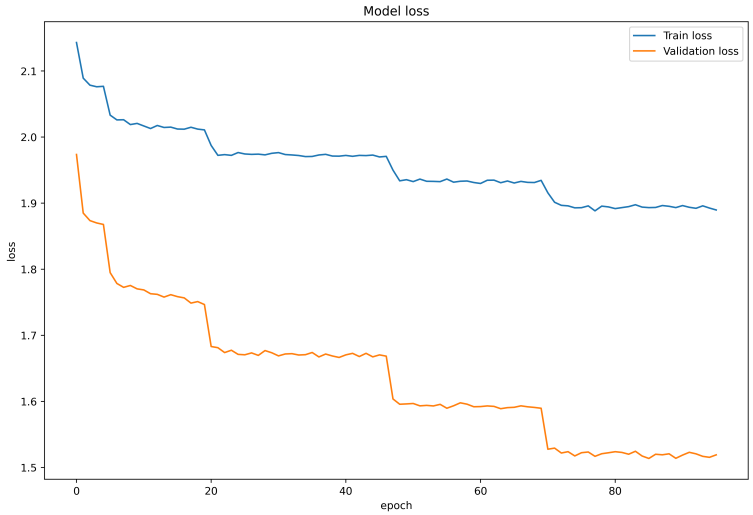


Figure 2: Training and validation loss

3 Results

3.1 Regression

We briefly report the results obtained testing the best model and we attach in Appendix section some details about network. So finally we have :

1. Train loss : 0.133
2. Validation loss : 0.563
3. Test loss : 0.123

as loss we mean that value obtained by "mean square error".
Then we plot all training and test data and we overlap the prediction done by network in all the range[-6:+6] (figure 3).

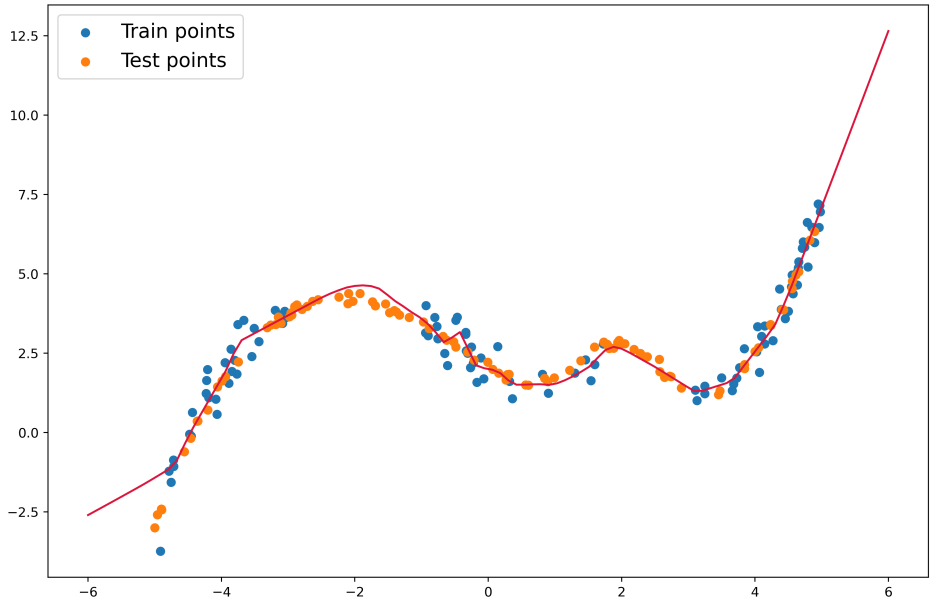


Figure 3: best function predicted

3.2 Classification

For classification task the best results reached are:

- Train Loss : 1.89
- Validation Accuracy : 0.977
- Test Accuracy : 0.981

And to better visualize misclassification hand-numbers images we plot the confusion matrix:

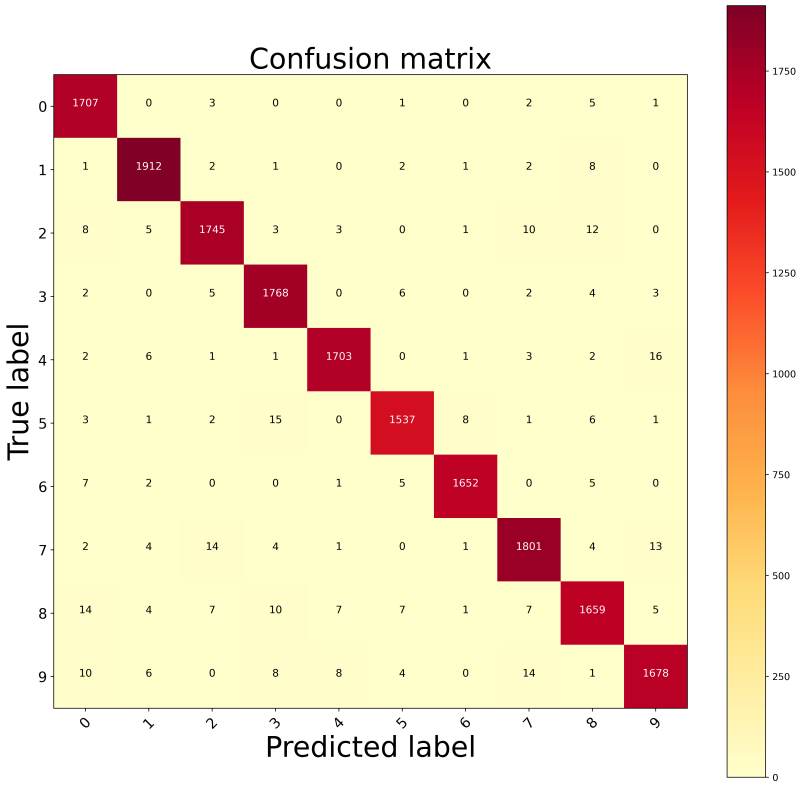


Figure 4: Confusion matrix

4 Appendix

4.1 Regression

Here we register the weights values after training procedure and plot them in a histogram (figure 7) and then we show the activation of the last layer after three different inputs given(figure 6).

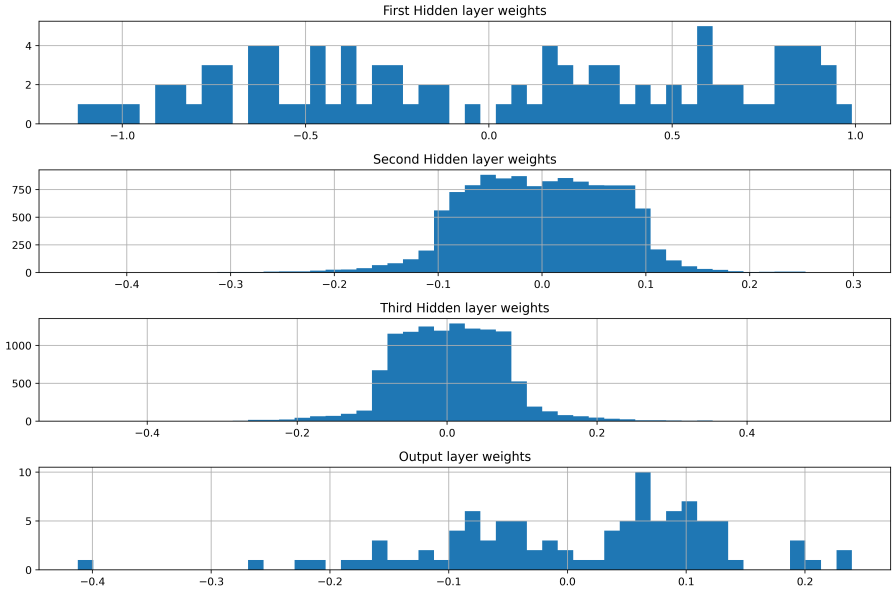


Figure 5: weight histogram values

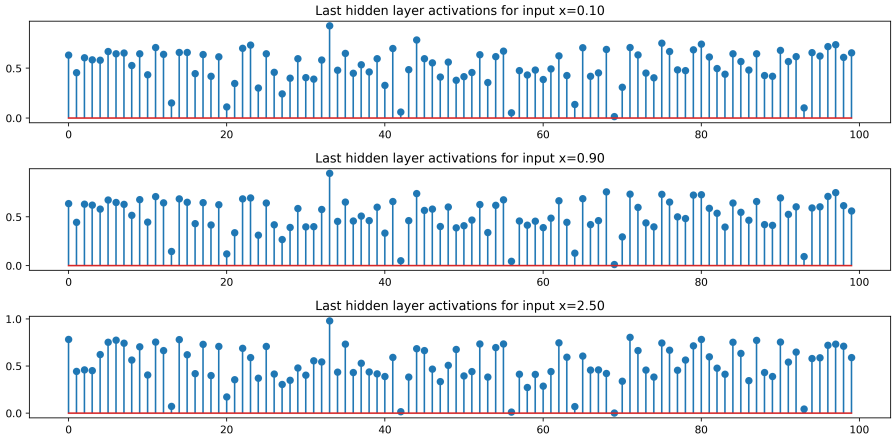


Figure 6: last hidden layer activation by inputs

4.2 Classification

As before we register the weigths values and plot them in histogram(Figure 7) and then we plot(Figure 8) last layer output for different inputs.
Finally we show the filters of conv2D after "screening" the first example from test dataset and relative feature maps.

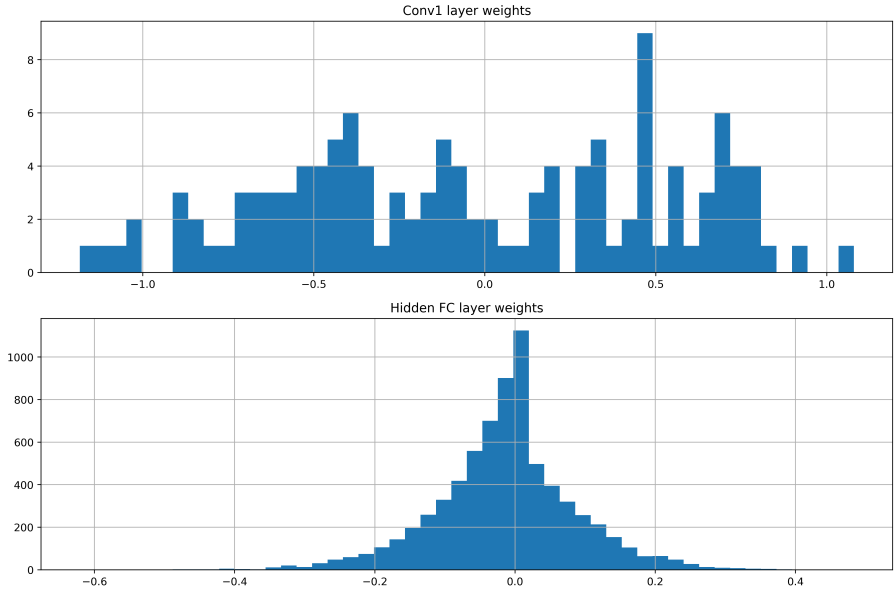


Figure 7: weights histo

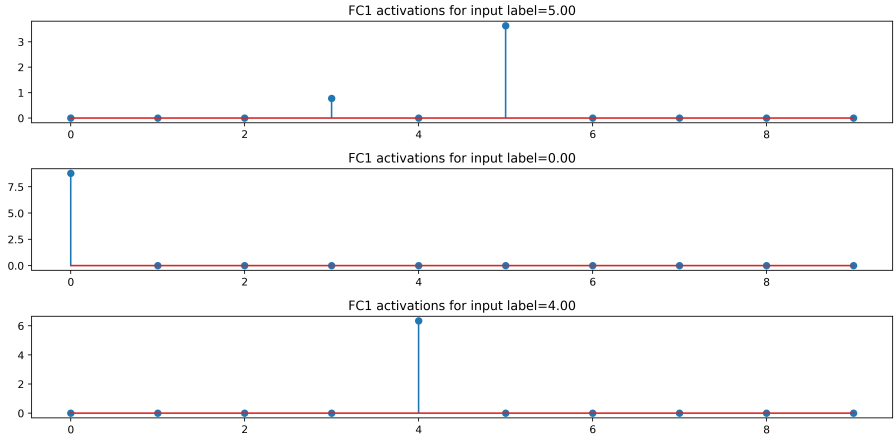
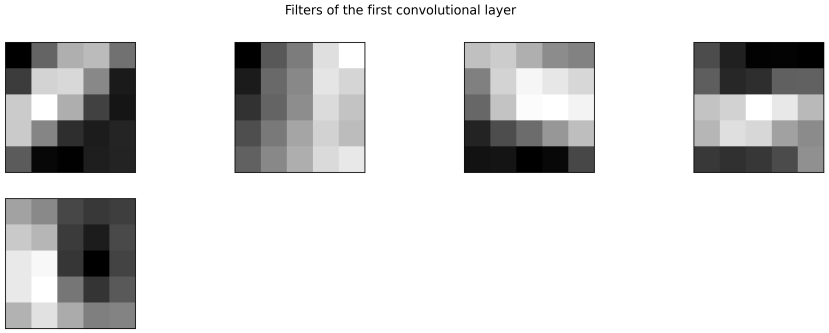
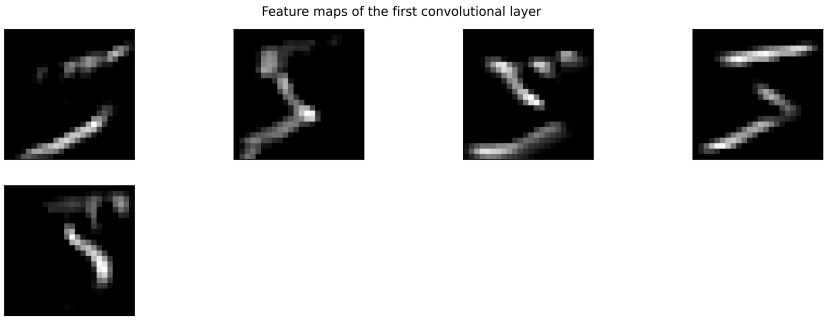


Figure 8: last layer activation




Feature maps of the first convolutional layer