Università degli Studi di Padova

Neural Networks and Deep Learning

Corso di Laurea Magistrale in Information and Communication Technologies

Homework 1: Supervised Deep Learning

Candidato:
Filippo Dalla Zuanna, filippo.dallazuanna@studenti.unipd.it
Matricola 1238613

Anno Accademico 2020-2021

Contents

1 Introduction

This homework aims to explore and study the supervised learning paradigm applied to Deep Learning techniques. This paradigm consists in learning tasks using a multitude of note examples. Them will guide the model to acquire and learn the properties of a certain function described by these examples. There are two main classes of supervised learning tasks, regression and classification. Regression consists in learning a continuous function with real numbers as output while classification consists in assigning a category in a limited set to each input. Each of these tasks is handled in this homework.

2 Regression task

2.1 Workflow

I have studied some configurations of a two layer feed forward neural network to find the best hyper-parameters of this model for the function estimation. The parameters that I have investigated are :
-Learning rate: 0.1, 0.01 and 0.001.
-Activation function: sigmoid, hyperbolic tangent, rectified linear unit and exponential linear unit.
-Neurons of first hidden layer: 64 and 128.
-Neurons of second hidden layer: 64, 128 and 256.
The combinations of all these parameters gave 72 configurations that have been trained with cross-validation. Cross-validation, in this case, was necessary since the dataset is small and so a single validation set can not give a good approximation of the dataset. With all these trained models we can evaluate and compare the dependence of the performance with these parameters. Then we trained some selected models again using the whole dataset as a training set without validation and use them for testing. Finally, I performed a dataset augmentation trying to improve the function estimation.

2.2 Dataset

The dataset is composed of 100 training samples and 100 test samples representing a polynomial function. We can see that 2 regions don't have training data so that part is difficult for the function estimation because of the absence of information.

2.3 Parameters choice and properties

2.3.1 Learning rate

The learning rate is a parameter that determines the amount of the variation of the weights in the training procedure and it is very important for the effectiveness of the learning process and so for the final performance. If it is too high can cause unstable changes in the weights that could not lead to convergence. If it is too small the overfitting problem and so bad task generalization can occur.
The results using the mean of the 6 best models of the corresponding parameter are:
-0.1 Learning Rate: 5.21 Mean Validation Error.
-0.01 Learning Rate: 0.50 Mean Validation Error.
-0.001 Learning Rate: 0.41 Mean Validation Error.
In this case, we can state that the highest learning rate doesn't work because of stability problems while the lowest one doesn't incur overfitting since both the train and validation errors are smaller.

2.3.2 Activation functions

The neuron processing function is another characteristic that might improve the learning of a task. For this reason, different types of functions were chosen and tested in a cross-validation procedure. The Sigmoid, ReLU, ELU and Tanh functions permit to study if there is one that works better with this specific task.
The results using the mean of the 6 best models of the corresponding parameter are:
-Sigmoid Function: 0.437 Mean Validation Error.
-Tanh Function: 0.471 Mean Validation Error.
-ReLU Function: 0.591 Mean Validation Error.
-ELU Function: 0.514 Mean Validation Error.
In this task, the activation function doesn't impact so much in the results. However, we can state that there is a better and a worse activation function for this task.

2.3.3 Number of hidden neurons

The number of hidden neurons, with also the number of layers, determine the complexity of the network. His structure can't be the same for every problem and has to be tuned since the complexity is different among them. This is the reason why we add also this hyper-parameter in the study.

The results using the mean of the 6 best models of the corresponding parameter are:
-64 first hidden units: 0.457 Mean Validation Error.
-128 first hidden units: 0.425 Mean Validation Error.
The difference, in this case, is not so visible probably because the two parameters are too close and so the complexity of the network is similar. Having chosen different parameters may have provided more hints about this aspect.

2.4 Selected models training and testing

Considering the cross-validation error, the best model for each activation function, the model in the middle and the worst model have been selected for performing a training procedure over the entire training set and the test procedure. The training epochs for each model is the epoch where the cross-validation error was at a minimum.
The test results are:
-Sigmoid Function: 0.204 MSE Test Error.
-Tanh Function: 0.178 MSE Test Error.
-ReLU Function: 0.312 MSE Test Error.
-ELU Function: 0.064 MSE Test Error.
-The Middle: 0.168 MSE Test Error.
-The Worse: 4.208 MSE Test Error.
These results confirmed the good operation of the parameters selection done since the test results of all the first models were good. However there are some inconsistencies since the activation function that was supposed to be better is not and, also, the middle model is performing as good as the best ones highlighting how it is difficult that the validation performances match the test ones. The extreme difficulty is to generalize the problem in the space where there isn't any information and this increases the disparity between the validation and the test results.

2.5 Dataset augmentation

With the intent of overcoming the last problem, I tried to fill the part of the dataset without info with randomly generated data based on the original one and the centroids that forms a "skeleton" of the function. This didn't give an improvement, probably because of too much noise or simply the high importance given to the original dataset that had to be done, thus presenting the usual problem.

3 Classification task

3.1 Workflow

I have studied some configurations of a Convolutional Neural Network to find the best hyper-parameters of the model among those inspected for the classification of handwritten digits. The parameters that I have investigated are:
-Optimizers: Stochastic gradient descent with and without Momentum and Adaptive Moment Estimation(Adam) (0.01 and 0.001 learning rates).
-Regularization techniques: L2 regularization (5e-4 weight decay) and dropout (0.3 dropout probability).
-Numbers of convolutional layers: from 1 to 3.
The fixed parameters of the net are the type of convolutional layer(32 filters with kernel size 3 and max pooling) and the neurons of the fully connected layer(200). In the classification part, the fully connected structure was not used since I preferred focusing on CNN analysis because the changing of the model for classification purposes would have been straightforward(changing the last layer) but a decent analysis would have taken a lot of time, leading me to choose to deepen only the CNN structure. The combinations of all these parameters gave 72 configurations that have been trained with simple validation. Cross-validation, in this case, was not necessary since the dataset is big and so the validation set is a good approximation of the information of the overall dataset. Using cross-validation also would have taken a lot of time in addiction. Early stopping was implemented in the training that was stopped if a certain average of the validation loss is increasing. With all these trained models we can evaluate and compare the dependence of the performance with these parameters, trying to understand better the influence of more sophisticated hyper-parameters such as optimizer and regularization. Finally, all these models have been tested and the filters and the feature maps of the best model have been visualized.

3.2 Dataset

The MNIST dataset used in this part is composed of 60000 training samples and 10000 test samples containing a multitude of handwritten digits examples. These samples are images 28X28 in black and white.

3.3 Parameters choice and properties

3.3.1 Optimizers

An optimizer is a tool that determines how the weights of the network should be updated during the training phase. This usually is based on the gradient descent technique that locates the direction of the parameters' update basing on the errors made by the network compared to the desired output. The various criteria to select this direction based on the samples processed drastically affect the training procedure and, so, it is relevant for the effectiveness of the learning process.
The results using the mean of the 6 best models of the corresponding parameter are:
-SGD Optimizer: 0.0473 Mean Validation Error, 0.985 Mean Test Accuracy.
-SGD Optimizer with Momentum: 0.0467 Mean Validation Error, 0.986 Mean Test Accuracy.
-ADAM Optimizer: 0.0364 Mean Validation Error, 0.936 Mean Test Accuracy.
In this case, we can state that the momentum is increasing the performance of the standard stochastic gradient descent since the validation error is smaller and also the generalization capability is better, measured by an increase of the test performance. The Adam optimizer instead is generalizing less since the test performance is worse even if the validation error is smaller. This kind of behaviour is difficult to understand and points out how the choice of the optimizer is a critical and difficult aspect that can be addressed only with direct tests on the problem.

3.3.2 Regularization techniques

Regularization techniques are used to improve the generalization capabilities of the network giving some limits on its parameters and complexity. This is for avoiding the network to easily and perfectly adapt to the training data that will reach out to overfitting problems. One example is L2 regularization where there is a penalty on the loss function if the parameters of the network grow so much. In this manner they are kept low and uniform, avoiding some parameters to prevail in importance among the others. Another example is dropout where, during training, each neuron could be deactivated with a certain probability. This force each neuron to learn during training and prevents the presence of dead and prevalent neurons.
The results using the mean of the 6 best models of the corresponding parameter are:
-No L2 regularization and no dropout: 0.0470 Mean Validation Error, 0.974

Mean Test Accuracy.
-L2 regularization: 0.0465 Mean Validation Error, 0.979 Mean Test Accuracy.
-Dropout: 0.0458 Mean Validation Error, 0.971 Mean Test Accuracy.
-L2 regularization and dropout: 0.0422 Mean Validation Error, 0.953 Mean Test Accuracy.
The regularization effect is evident seeing the validation error that goes down with more regularization, showing that generalization is improving in the validation set. However, this statement doesn't find a match in the test performances because of the non-perfect relation between validation and test data.

3.3.3 Number of convolutional layers

The number of convolutional layers determines how deep and complex are the features extracted from the images. However, the level of abstraction of them can suit well some problems but are unsuitable for others even if more complex features usually works better. Also how the layers are built influence the effectiveness of the model such as the dimension of the filters, the number of them, the kind of pooling and others. Anyhow, I choose to investigate only the layers quantity.
The results using the mean of the 6 best models of the corresponding parameter are:
-1 Convolutional layer: 0.0567 Mean Validation Error, 0.500 Mean Test Accuracy.
-2 Convolutional layers: 0.0427 Mean Validation Error, 0.931 Mean Test Accuracy.
-3 Convolutional layers: 0.0383 Mean Validation Error, 0.986 Mean Test Accuracy.
The improvement made by the deeper and more abstracted features is evident since the performance are improving with the increase of the convolutional layers.

3.4 Testing and feature maps

As seen in the various examples before, the testing was performed on every model and is clear that good models are performing well since the accuracy was superior to 98%. The best network considering the minimum validation error achieved 98.34% of accuracy. Finally, the extraction of the feature maps shows how the filters can discriminate characteristics of the input, such as the edges, the contours or a line segment, showing so complex information.

A Regression Appendix

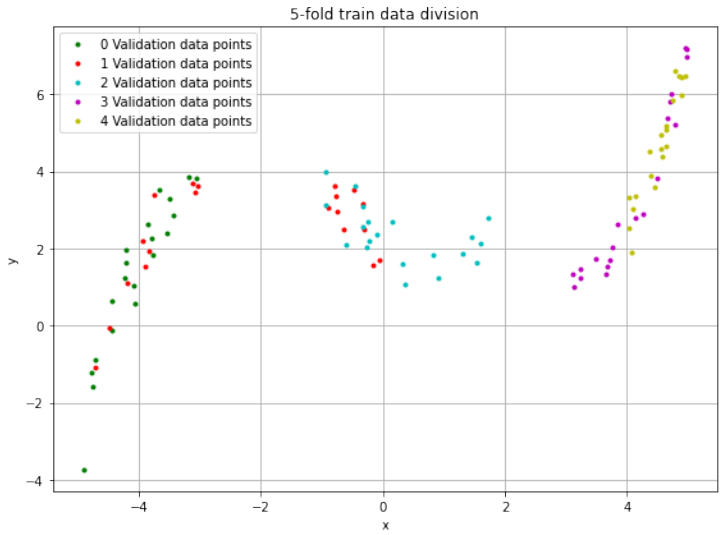

Figure 1: K-Fold division of training dataset

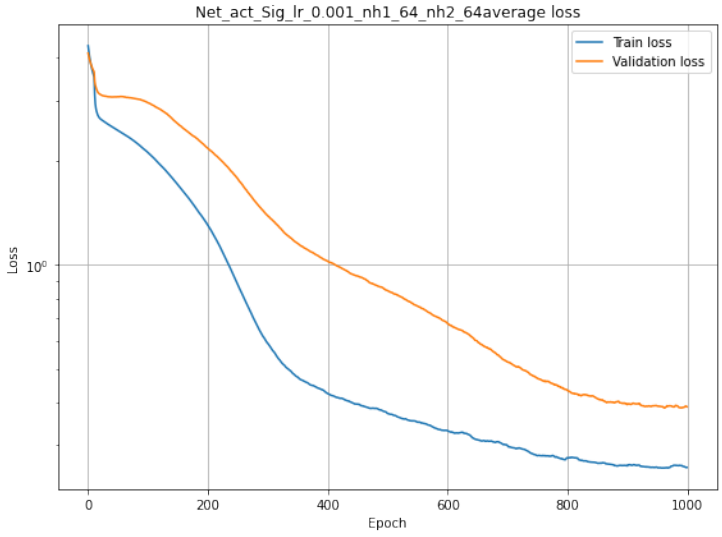

Figure 2: Training history of best model with cross validation

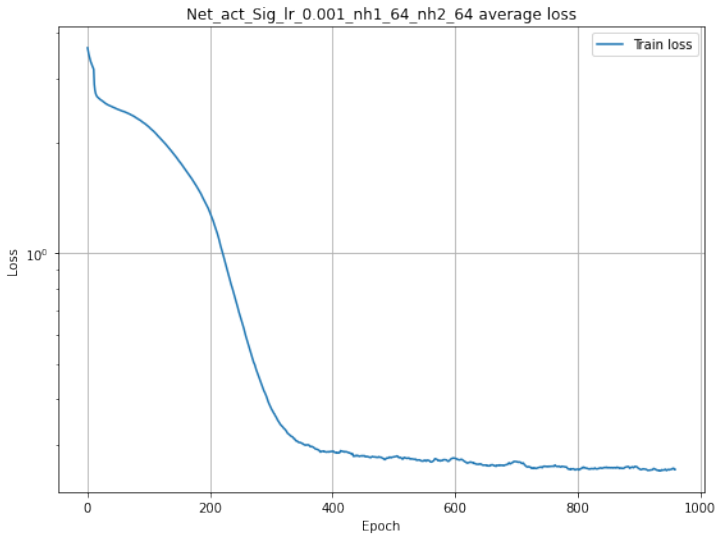

Figure 3: Training history of best model on all the training dataset

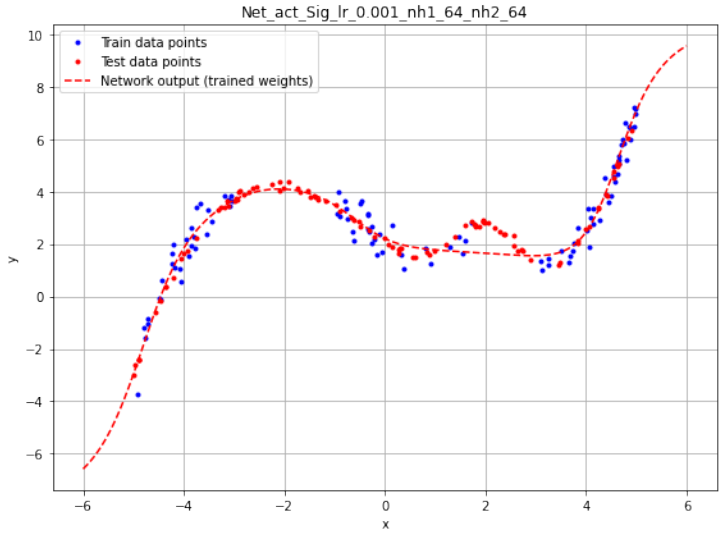

Figure 4: Network test

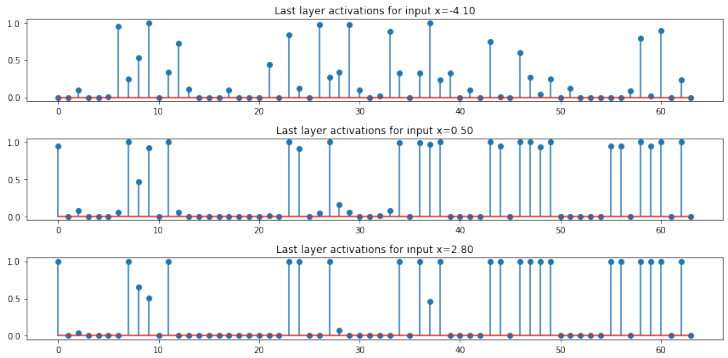

Figure 5: Example of network activation

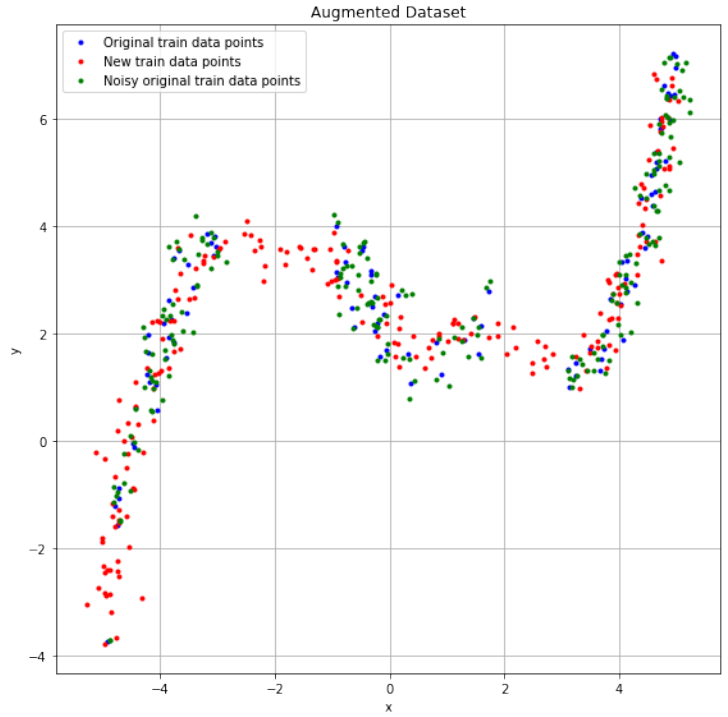

Figure 6: Augmented dataset

B Classification Appendix

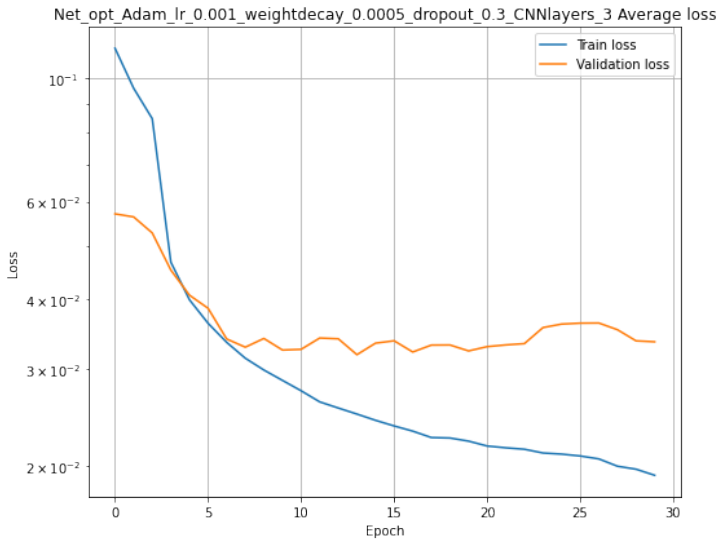

Figure 7: Best model: training history

```
[[ 967    0    3    0    0    0    1    2    7    0]
 [   0 1122    2    0    1    0    1    6    3    0]
 [   1    0 1026    0    0    0    0    4    1    0]
 [   0    0    5 1001    0    2    0    2    0    0]
 [   0    0    1    0  962    0    1    4   13    1]
 [   0    0    0    4    1  885    1    1    0    0]
 [   3    1    1    0    1    2  946    0    4    0]
 [   0    0    6    0    0    0    0 1019    1    2]
 [   1    0    3    0    0    1    0    1  967    1]
 [   1    0    0    0   13    7    0   12   37  939]]
```

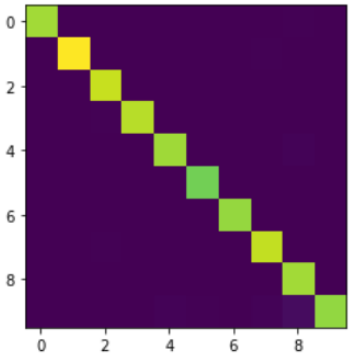

Figure 8: Best model: confusion matrix

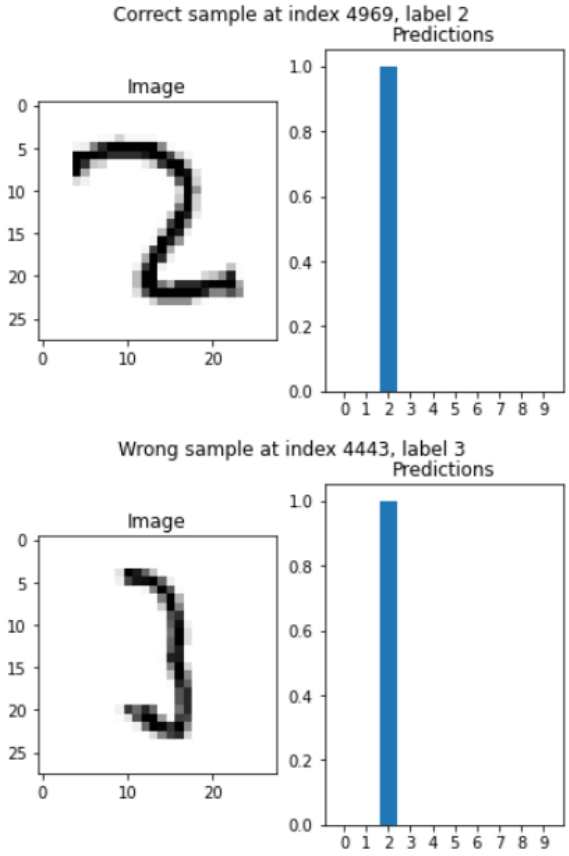

Figure 9: Examples of best model's predictions

Figure 10: Examples of feature maps