Exercise 01 - Supervised Deep Learning

University of Padua

Name:	Daniele Mellino
Student Number:	2013373
Course of study :	Physics of Data
Date:	February 13, 2022

1 Project Introduction

The goal of this homework is the implementation of simple neural network models for solving regression and classification task, testing in both case different advanced optimizers and regularization methods. In the regression section a Fully Connected Network(FCN) composed of 2 hidden layers is implemented to approximate the behaviour of an unknown scalar function given some not uniform noisy measures.

For the classification task, a simple CNN is built to correctly classify images from the Fashion MNIST dataset.

2 Regression

In the regression task the training data is composed of 100 noisy measures from an unknown function :

$$\begin{aligned} f\colon \mathbb{R} &\to \mathbb{R} \\ x &\to f(x) \end{aligned}$$

The data is not uniform since there is zone in the domain in which measures lack (for instance $x \in [-3, -1]$ and $x \in [2, 3]$).

2.1 Methods

To tackle this problem a FCN is defined in the "Net" class. It has the following features :

- An input layer composed of $N_i = 1$ neuron;
- A first hidden layer of N_{h1} neurons;
- A second hidden layer of $N_{h2} = 2 * N_{h1}$ neurons;
- A final output layer $N_o = 1$

The loss function chosen is the Mean Squared Error(MSE), that measures the squared L2 norm between the elements of the input and target tensors.

To avoid the so called "vanishing gradient problem" Rectified Linear Unit(ReLu) is preferred to the Sigmoid activation function. This problem refers to the significant increase in the error backpropagation phase due to the magnitude of the gradient of the activation function.

In this experience, three different optimizer are tested:

- The stochastic gradient descent(SGD) with momentum $m = 0.9$, which is the basic algorithm for optimization. The addition of the momentum (which take cares of the memory of the previous steps) helps go out from the local minima and speeds up convergence.
- Adam optimiser, which use the estimation of first and second moment of the gradient to adapt the learning weight of each weight of the network.
- The RMSprop optimiser which divide the learning rate of a certain weight by a running average of the magnitudes of recent gradients for that weight.

For each of the above different learning rate is examined.

Since we have few data for training, regularization technique such as dropout and a penalty in the L2 loss function. The dropout is tested for the two middle layers and consists of turning off a percentage dp of the layer's nodes during training. The penalty in the L2 loss is specified in the optimiser choice using the weight_decay keyword.

To fully exploit the short number of data at disposal, a k cross-validation technique is used. The method consists in dividing the train dataset in $k = 4$ chunks. The model is then trained on the union of $k - 1$ sets, computing the validation loss on the remaining one. This procedure is repeated k times alternating each chunk for validation. The final estimation of the model's validation loss is obtained as the average of the k-fold obtained values.

The cross-validation algorithm is included in a random search approach. Therefore the parameters of a model are randomly sampled, afterwards that model goes into the cross-validation procedure. This random choice has been repeated for $N = 100$ set of parameters.

The parameters have been chosen among the ranges values in Table 2.1. Notice that * refers to a log uniform distribution, instead ** to a uniform one. Finally, to get more insight about the learning process histogram's

parameter	ranges
dropout**	(0.0,0.5)
$\#epochs$	[150, 200, 250, 300, 350]
learning rate*	(5e-5,1e-2)
optimiser	[SGD, Adam, RMSprop]
N_{h1}	[40, 50, 60, 75, 85]
regularization*	(1e-5,1e-3)

Table 1: Tables of ranges.

weight and activation profile are visualized.

2.2 Results

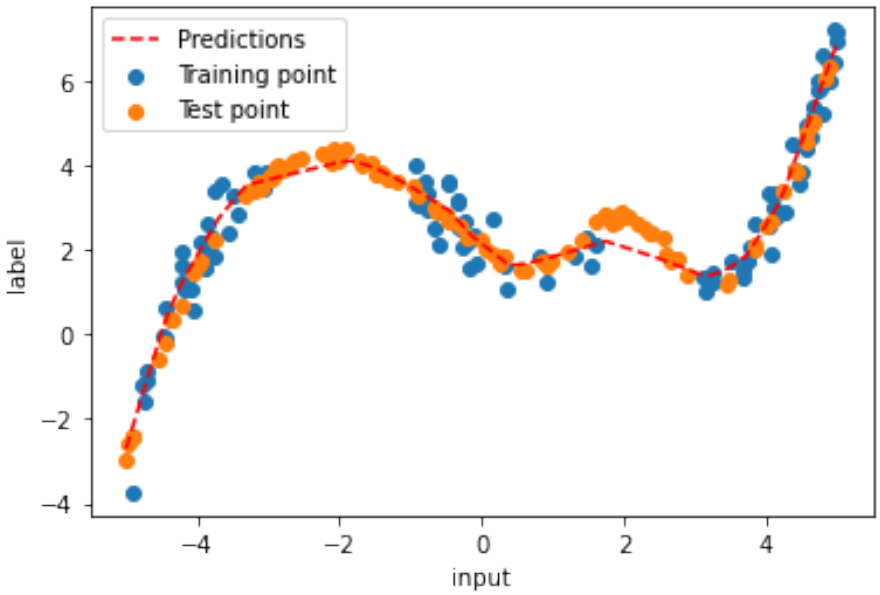


Figure 1: Regression result. Train data in blue, test data in orange and network prediction in red.

The minimization of the average loss functions from the random search procedure lead to the parameters reported in Table 2.

The results of the training with the best parameters on the whole training data is shown in Figure 1.

As one can observe, the network does not generalise well the data tendency in the training set's hole. In particular the model does not predict the local maximum of the function around $x = 2$. In Figures 4 and 4 one can see the two layers activation profile and the weight histograms. One can notice that the network's node have a discrete activation, even though in the last layer there is some neuron that does not seem to fire.

dropout	num_epochs	learning_rate	optimiser	N_{h1}	regularization
0.0144	350	0.00292	Adam	75	$5.5 \cdot 10^{-4}$

Table 2: Regression's best parameters

3 Classification

The aim of this section is to perform a supervised classification task on the FashionMNIST dataset, which contain 10 different classes. The dataset contains 60 000 images 28x28 for training and 10000 images for testing.

3.1 Methods

In the work the classification method is carried out by a simple Convolutional Neural Network(CNN) composed as follow :

- A first convolution layer that have in input one channel(grayscale images) and n_feat in output, with kernel size of 3 and padding equal to 1 to keep the initial dimension(28x28 pixels);
- A max pooling layer which halves the picture dimension size (14x14 pixels)
- A second convolutional layer that double the number of features and have kernel size of 5 and no padding (10x10 pixels);
- A second max pooling layer equal to the first (5x5 pixels);
- One fully connected layer with (5x5x2xn_feat inputs and N_f outputs)
- A final fully connected layer with 10 final nodes (as the number of classes) .

After the first pooling layer and after the first fully connected layer is inserted dropout with percentage respectively dp1D and dp2D. As in the previous section RelU activation function is used. Differently from before, this time cross validation is not implemented since the data is considered exhaustive and homogeneous. Also this time we use random search to tune the network. To do that, differently from the previous section we use Optuna. To save computational time since the dataset contains a lot of images pruning is adopted. In particular 100 different parameter sets are tested, for 10 epochs each. However if after 3 iteration the validation loss is higher than the median value of the previous models the train stop and the model discarded. The parameter range tested is in Table 3.1 or in Figure 6.

parameter	ranges
dp1D	[0, 0.05,0.10,...,0.3]
dp2D	[0, 0.05,0.10,...,0.3]
learning rate*	(1e-4,1e-2)
optimiser	[SGD, Adam]
n_feat	[3,4,5..,11]
N_f	[200, 250, 300,...,500]
regularization	[0.0,0.05,..., 0.20, 0.25]

Once obtained the best parameters the model is trained from scratch. Also here to speed up the training early stopping is adopted. In particular if the validation loss does not increase for more than 4 consecutive epochs, the train stop.

3.2 Results

The best network's parameters found during the random search are displayed in Table 3.

dp1D	dp2D	learning_rate	optimiser	n_feat	N_f	regularization
0.0	0.05	0.0028	Adam	10	500	0.0

Table 3: Classification's best parameters

class	accuracy
T-shirt	91.5%
Trouser	98.0%
Pullover	89.8%
Dress	87.0%
Coat	89.2%
Sandal	98.5%
Shirt	66.9%
Sneaker	96.3%
Bag	97.7%
Ankle Boot	96.9%

Table 4: Accuracy per class

The model tested on the test data give 91.18% accuracy. The accuracy for each classes is showed in Table4. As one can see from the table the model works well for the majority of the classes, except for the Shirt which is easly mistaken with others. This is reasonable since the difference in shape between a Shirt and a Coat or a Pullover is subtle. Finally in Figure 3 and 8 is shown the filters of the two convolutional layers. One can recognize some pattern for edge detection mixed with some smoothing filters, at least for the first layer. In the second layer they became case-specific and less interpretable. Finally in Figure 2 and 7 shows the activation profile of the convolutional layers that confirms the above considerations.

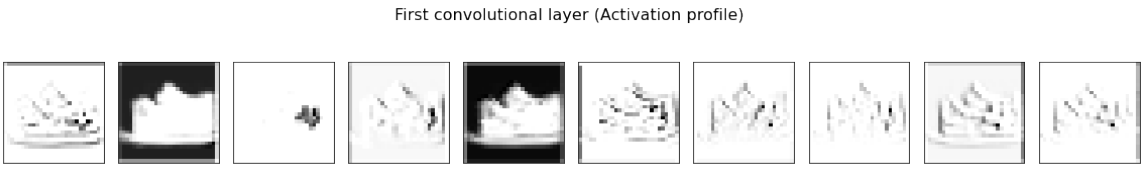


Figure 2: Classification's activation profile of a sandal (1 layer)

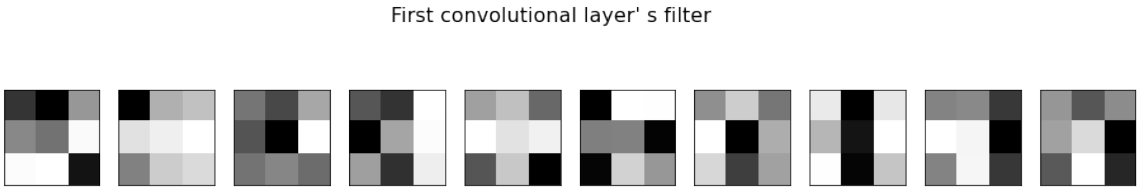


Figure 3: Classification's learned filters (1 layer)

4 Appendix

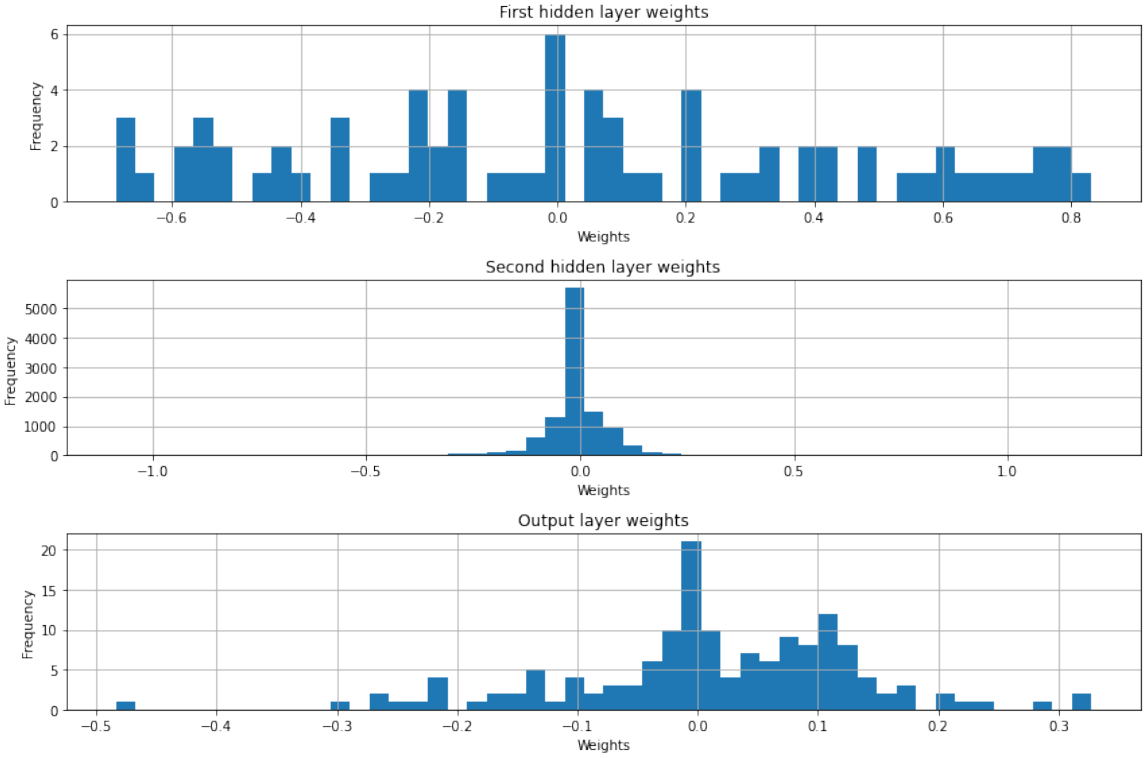
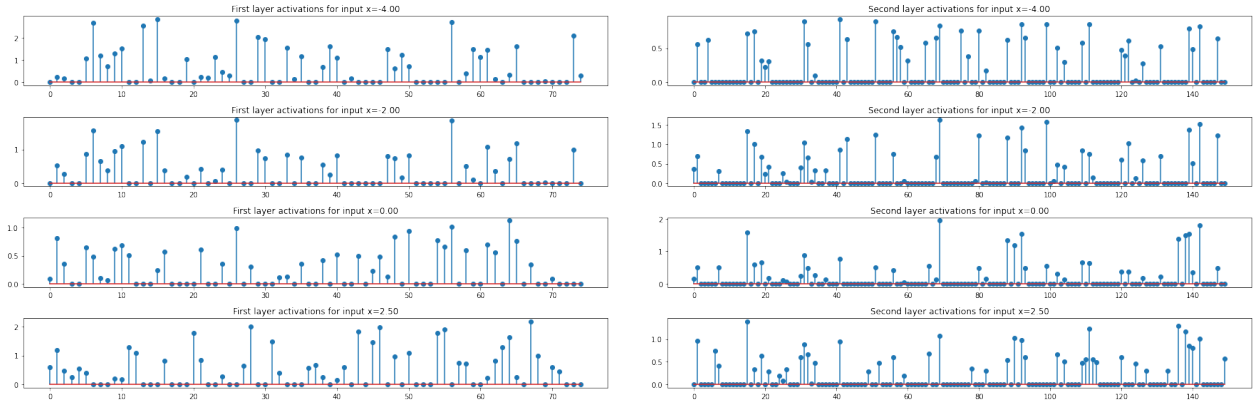


Figure 4: Regression's weight histogram



(a) First layer activation fuction

(b) Second layer activation fuction

Figure 5: Regression activation profile

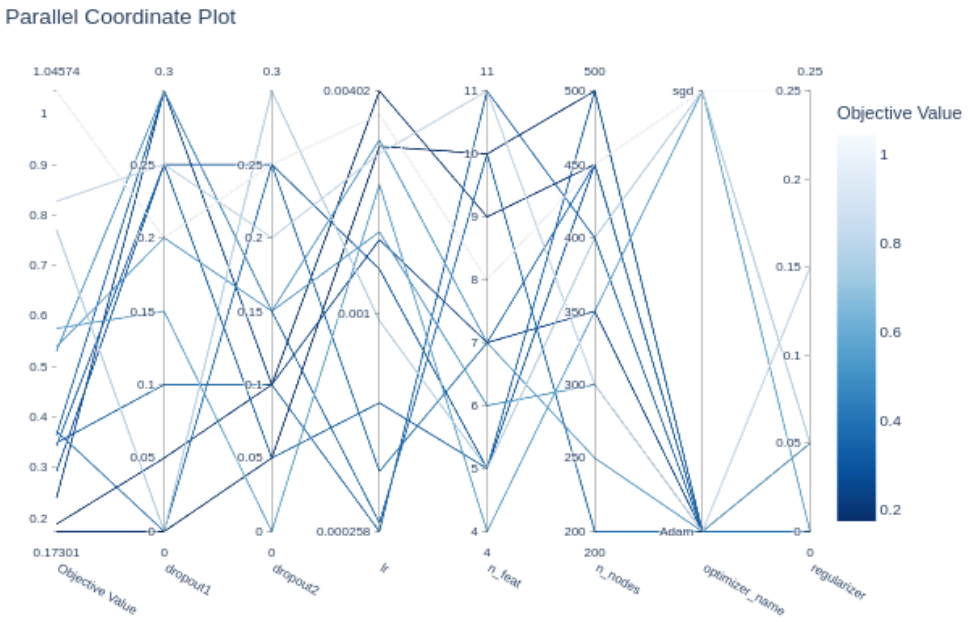


Figure 6: Hyperparameters search for the classification task. The color is proportional to the loss value reported in log-scale. The darker the color, the better the result. Notice that here probably too hight values of regularization has been used.

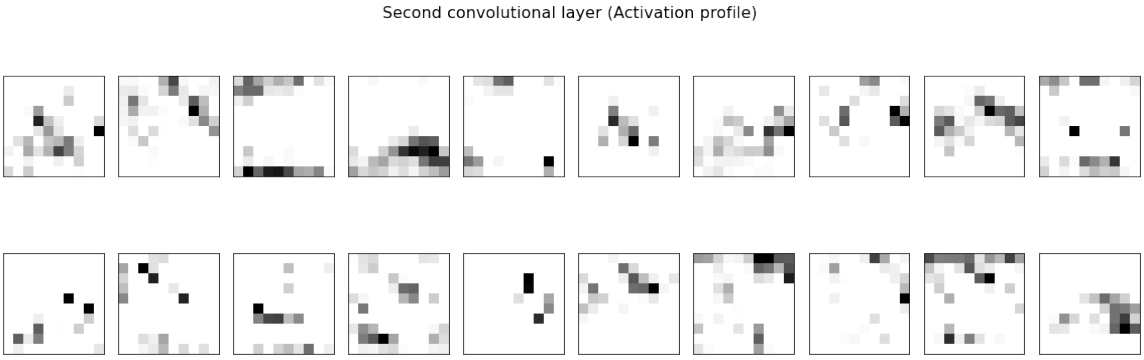


Figure 7: Classification's activation profile of a sandal (2 layer)

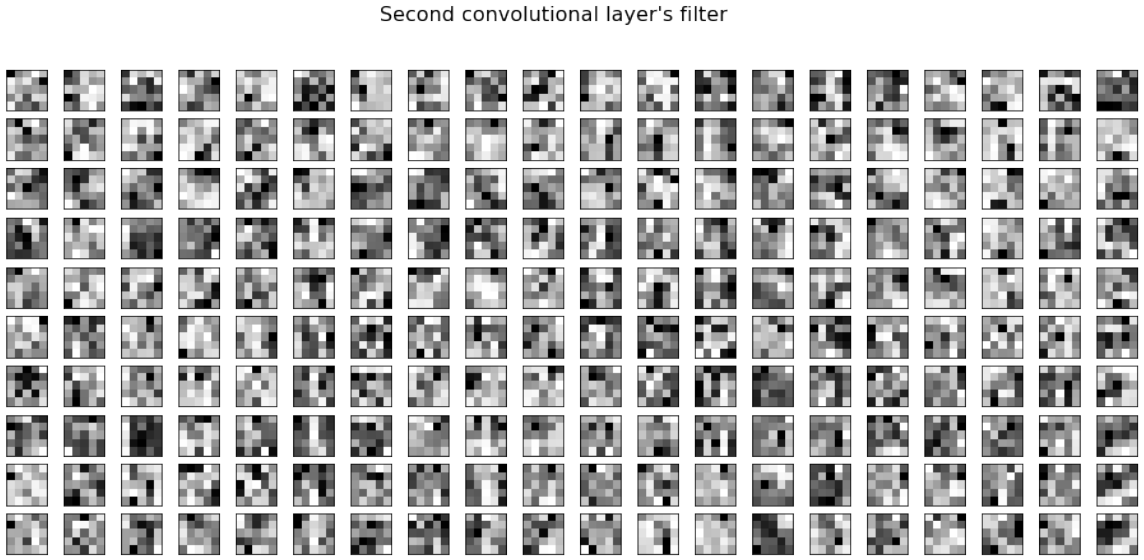


Figure 8: Classification's learned filters (2 layer)