NEURAL NETWORKS AND DEEP LEARNING
Homework I - Supervised Deep Learning

Accomplished by Ali Bavarchee(1219425)

January 4, 2022

1 Introduction

Deep learning is a subclass of machine learning which uses multiple (hidden) layers of neural networks- based on human brain to perform in processing data and investigating the features and behaviour of data. On the other hand, machine learning framework could be classified based on the formalism of input data and methods of training a model, to supervised learning, unsupervised learning, semi supervised learning etc. This exercise aims to flourish the supervised learning. Supervised learning is defined as a class of machine learning which using the "labelled" input data to train algorithms by getting assistance with an external teaching signal so called supervision. Supervised learning can be exploited for different tasks such as classification, categorization, regression, function approximation ect. The first part of this exercise is allocated for regression and the second part practices classification. All of the assignments are promised to accomplish by using pytorch. PyTorch is a Python package that provides two high-level features:

- Tensor computation with strong GPU acceleration;
- Deep neural networks built on a tape-based autograd system.

1.1 Regression task

The mission of regression algorithm is to map input values with the continuous outputs. In other words, the goal is fitting the best curvature and finding the descent function.

$$\text{f:R}\rightarrow R,\ \ \text{x=f(x)},\quad \hat{y} = f(x) + noise$$

The provided raw data contains divides to two sets(training set and test set) and each of them has two sets of a 100 scaler numbers, one to one, which could be represented as x and $y = f(x)$. The regression task and finding the suite function for the present data is a quite difficult, mostly because of the size of

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   input   100 non-null    float64
 1   label   100 non-null    float64
dtypes: float64(2)
memory usage: 1.7 KB
======max values of train set======
input    4.977516
label    7.199304
dtype: float64
======min values of train set======
input   -4.915863
label   -3.742970
dtype: float64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   input   100 non-null    float64
 1   label   100 non-null    float64
dtypes: float64(2)
memory usage: 1.7 KB
======max values of train set======
input    4.977516
label    7.199304
dtype: float64
======min values of train set======
input   -4.915863
label   -3.742970
dtype: float64
```

Figure 1: Info of train and test datasets

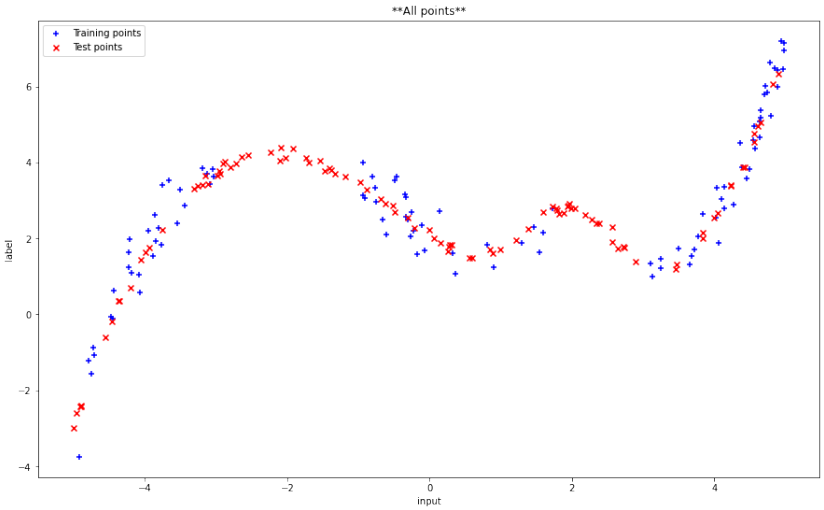

Figure 2: Plot of the all data

the input which is not some how sufficient. This problematic shows itself up in the final outcome. Both test and train is illustrated in fig.1 and fig.2.

1.2 Method

For the regression task of this exercise, and by considering the size of data, it has been decided to exploit fully connect neural networks (with, 1, 2, 3 and 4 hidden layers). This is possible by torch.nn a sub-library of Pytorch. The

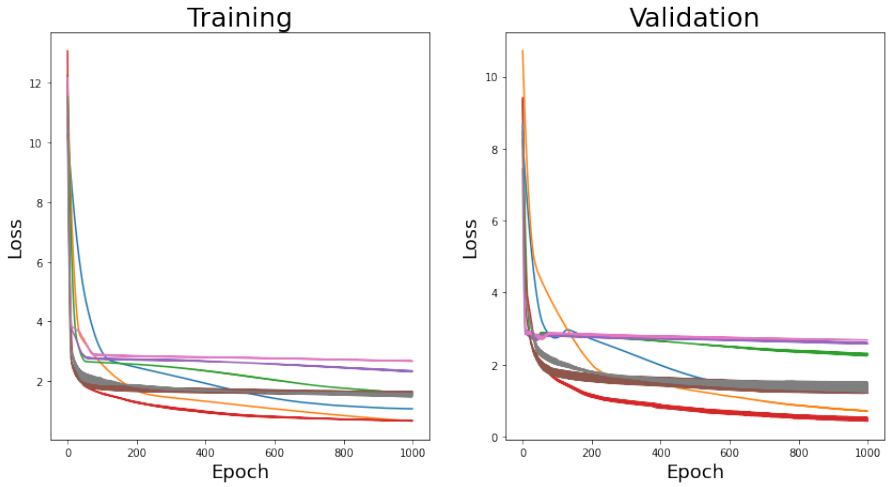

Figure 3: Train and val Loss of the different models

hyperparameters could be adjusted by grid search and random search. Generally the dataset should be split to train and test subset, while the train dataset also is divided to train and validation subsets. To do so, in this code, after transforming the provided raw data to tensor form, first uses the old fashioned method to evaluate the model by cross-validation method during the training, and also K-fold tactic to test the performance of the model. The K-fold tactic grantees the cross checking of the model thought the whole dataset. The selected loss function is the mean squared Error and the optimizer is Adam. As mentioned before, at first, 4 different networks are defined with 1, 2, 3 and 4 hidden layers and each of them with either "Sigmoid" and "ReLU" as their activation function. To avoid the overfitting, a L2 Regulation method (weight decay) is exploited. At the second try, the model with the least loss selected to maximize the performance. Therefore, the number of layers and activation function is fixed, but the number of neurons per layer, number of epochs, weights, learning rate and batch size get iteratively change to reach the best parameters.

1.3 Results

The best model to predict (and fit) the data for regression task is as follow:

By looking at the table, it is revealed that the best and optimized model through the others is net 4 with 2 hidden layers and ReLU as activation function.

The performance of the model after using K-fold validation trick is reported in fig.7.

Also the output of the model and the test (and train) data is shown in the canvas below:

By looking at the fig.10, it is revealed that the model has some deficits, especially to predict the maxima of the data. To get the better performance, it

```
______net1______
AVERAGE TEST LOSS: 1.2861871719360352
______net2______
AVERAGE TEST LOSS: 0.7071226239204407
______net3______
AVERAGE TEST LOSS: 2.311004400253296
______net4______
AVERAGE TEST LOSS: 0.4353371560573578
______net5______
AVERAGE TEST LOSS: 2.580313205718994
______net6______
AVERAGE TEST LOSS: 1.2109088897705078
______net7______
AVERAGE TEST LOSS: 2.6839864253997803
______net8______
AVERAGE TEST LOSS: 1.2538753747940063
```

Figure 4: Average loss on each model separetly

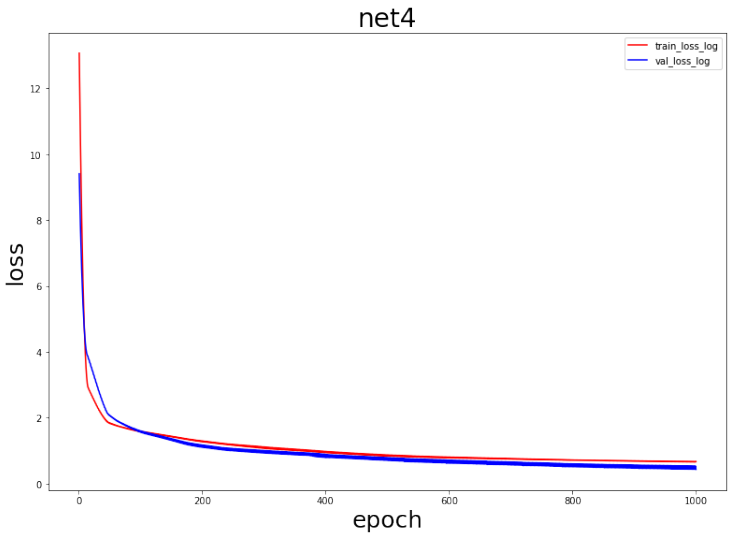

Figure 5: train and validation loss for net4

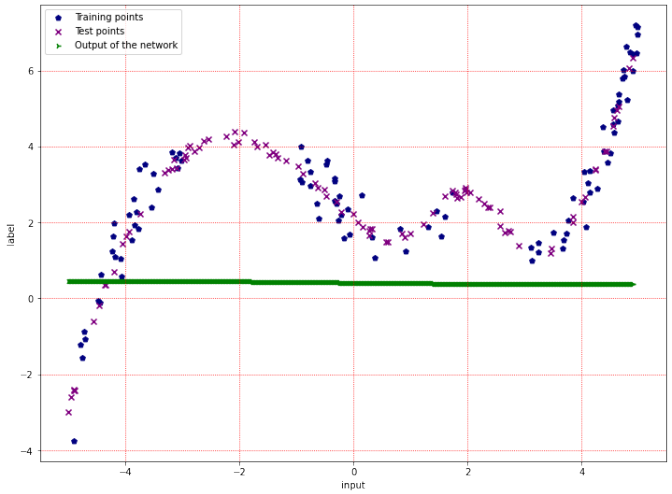

Figure 6: Performance of the net4

	0	1	2	3	val_kfold_avg
13	8	0	32	256	0.455567
18	8	1	64	128	0.515329
16	8	1	32	128	0.525820
32	16	2	32	128	0.552580
17	8	1	32	256	0.581040
29	16	1	32	256	0.605876
33	16	2	32	256	0.632759

Figure 7: Train loss and validation loss after k-fold cross validation

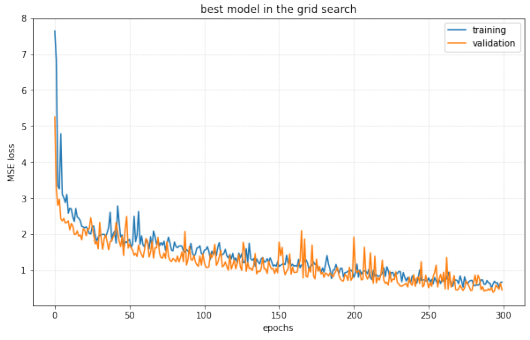

Figure 8: Train loss and validation loss after grid search

Figure 9: Train loss and validation loss after grid search for the best model for test set over whole the data

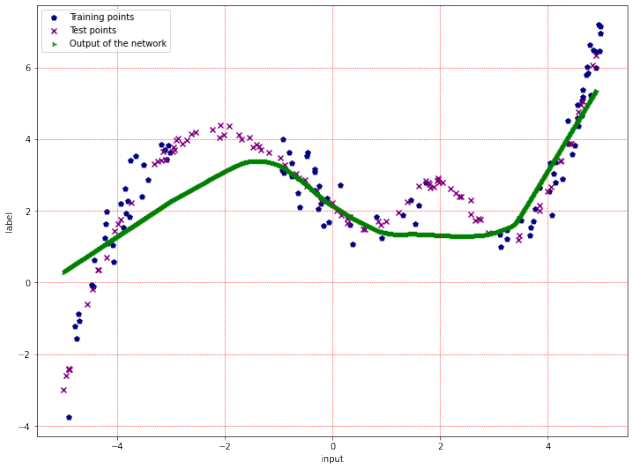

Figure 10: The final performance of the best model

seems that using the Dropout technique could be efficient.

Also validation loss in the grid search and weight of the each layers are shown in fig.11 and fig.12.

The last word is this results indicates that the deeper network is not necessarily conduces to the reasonable outcomes.

2 Classification task

Generally (supervised) Classification algorithms aim to learn the machine to distinguish an observation as input and assign it to one of the proper pre-defined categories. If there are two categories, them it is called binary classification and

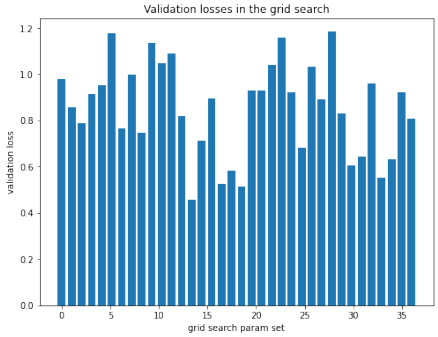

Figure 11: Val loss under grid search

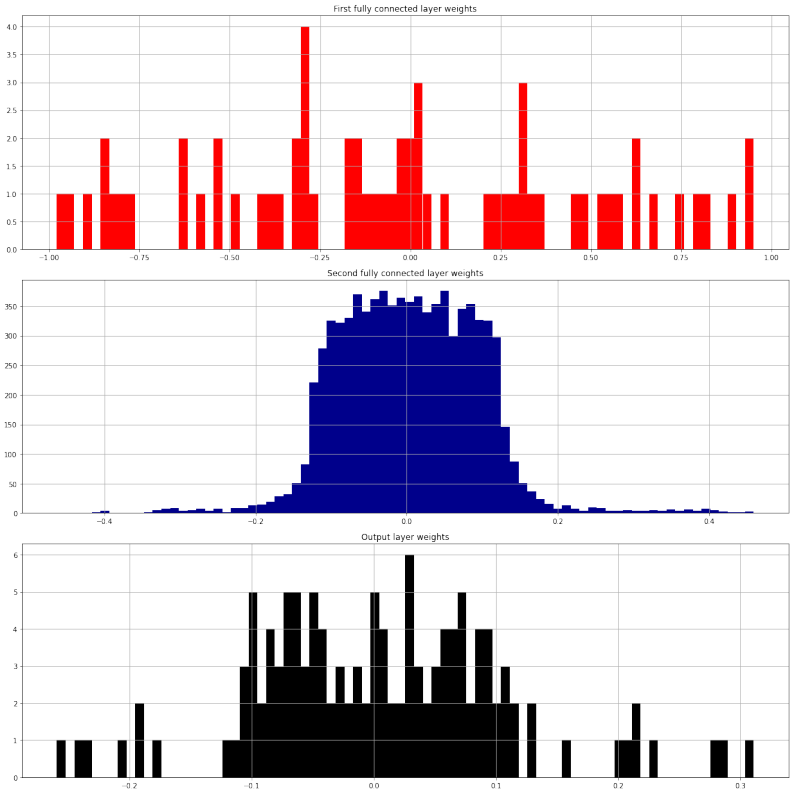

Figure 12: Weight of the each layer of the best model

Figure 13: Examples of the train set with their labels

if more than two classes label, it is called multi classification. In this part of the assignment, the input data is Fashion MNIST - Zalando's article image which is contained a training set with 60000 and test set with 10000 28×28 gray-scale images. The images labelled from 0 to 9 (which each class refers to a type of gatments). Hence, the task here is teaching this multi-classification to the machine.

The dataset is quite well known and simple and its size is adequate for implementing a convolutional neural networks.

2.1 Method

As it is promised to accomplish the assignments by using Pytorch, first it needs to transform the downloaded Fashion MNIST- by torchvision module, to tensor form and load them. It is feasible by using Dataset and DataLoader respectively from torch.utils.data module. For first try, train set is divided to training dataset with 48k samples and validation set with 12k samples. Then the CNN is defined with two hidden layers and one Max pooling and one Dropout which are posed in between the hidden layers and another Dropout between the later hidden layer and output layer. To set the best hyperparameters, it is exploited a function called random-params to reach the best model. Loss function is CrossEntropyLoss.

	Unnamed: 0	train_loss	validation_loss	parameters
8	8	0.416366	0.369050	[2, [32, 16], 0.001, 30, 0.0, 0.2]
37	37	0.319458	0.379771	[2, [16, 16], 0.001, 30, 0.0001, 0.0]
26	26	0.296369	0.381935	[2, [16, 32], 0.001, 40, 0.0001, 0.0]
27	27	0.535231	0.399417	[2, [32, 32], 0.001, 40, 0.0001, 0.4]
23	23	0.389887	0.404327	[2, [32, 32], 0.0001, 30, 0.0001, 0.0]

Figure 14: The 5 best hyperparams

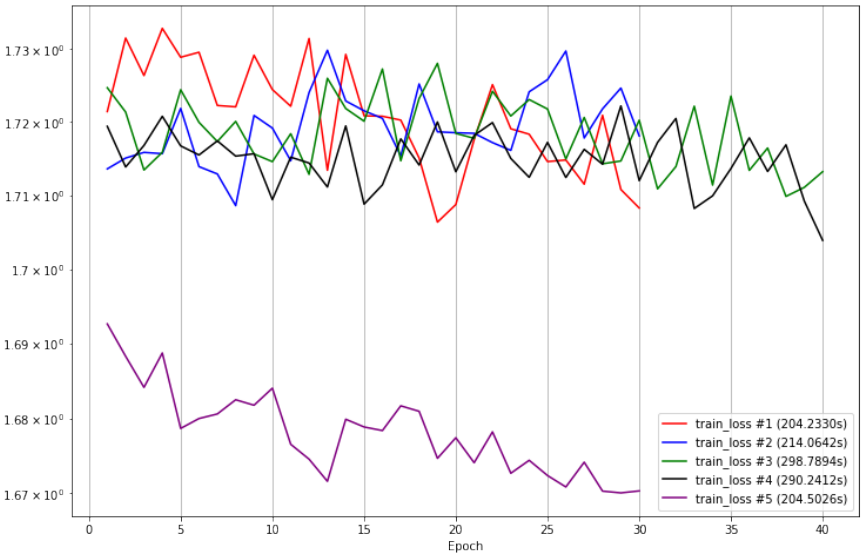

Figure 15: Train loss of the models

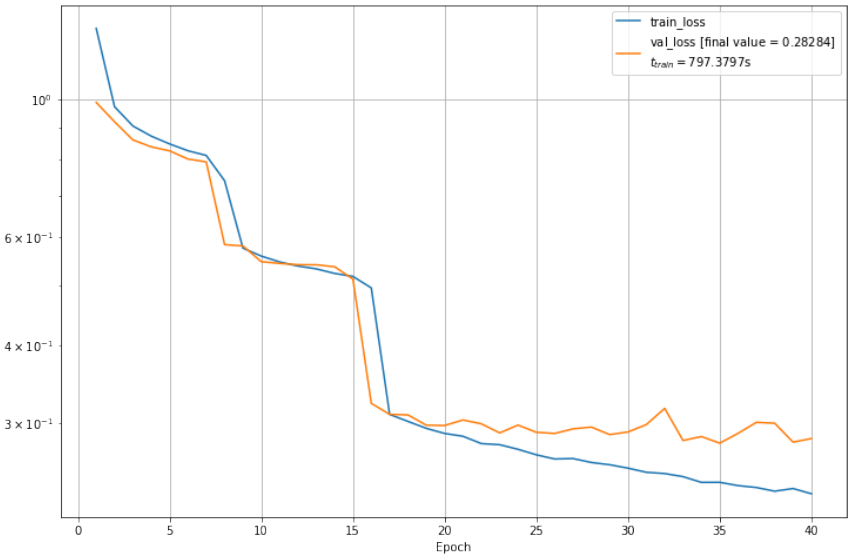

Figure 16: Train loss and val loss of the best model

The next try is to explore the different model(optimizer, act. function, network architecture, ...). It is decided to go with a fully connected convolutional neural network with 3 convolutional layers with ReLU activation function and 2 linear layers.

This time, the outcomes are more reliable.

2.2 Results

The evaluation and the results could have been more accurate if some methods such as Optuna and K-fold cross validation had been implemented. Optuna one of the tuner modules which is meant to optimize the model by adjusting the hyperparameters. Nevertheless, the performance of the models are reported. The last CNN is reached to the best model based on the results. The results -as usual for classification, are represented in the 10×10 confusion matrix as the number of class is 10 and therefore true positive is the main diagonal of the matrix.

The results of the later defined network with fully connected nodes are reported below.

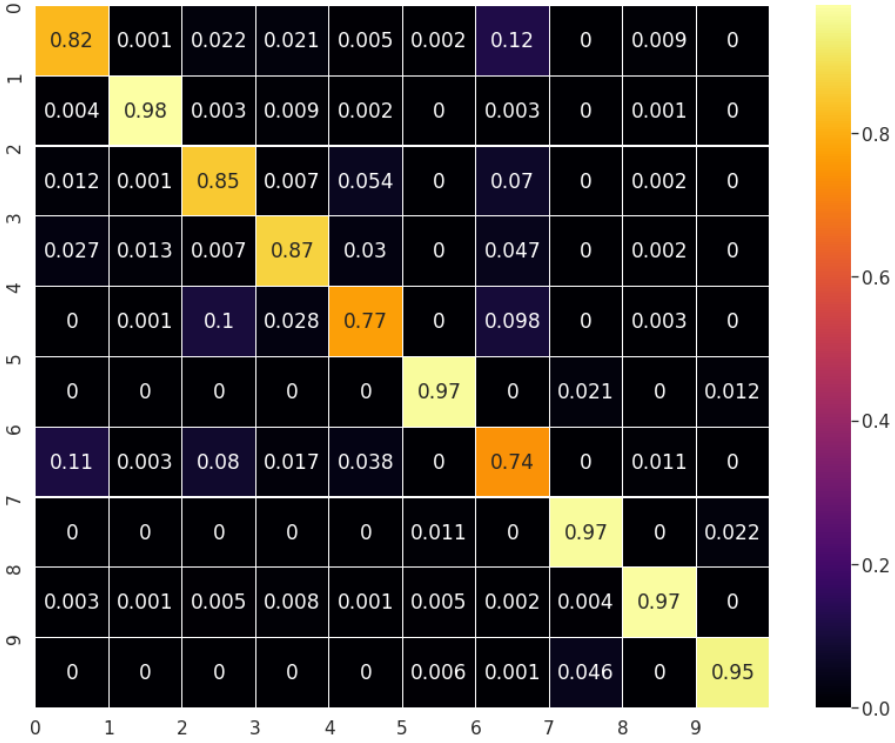

Figure 17: Confusion matrix of the first best model

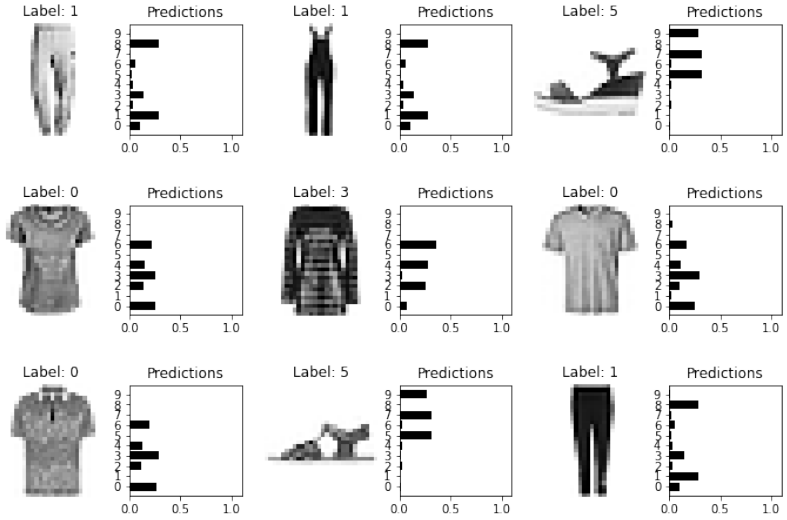

Figure 18: Performance of the best model1

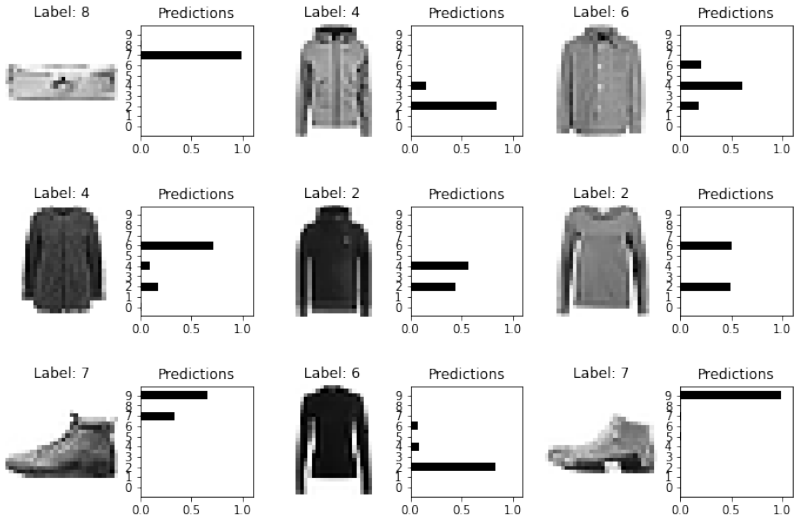

Figure 19: Performance of the best model2

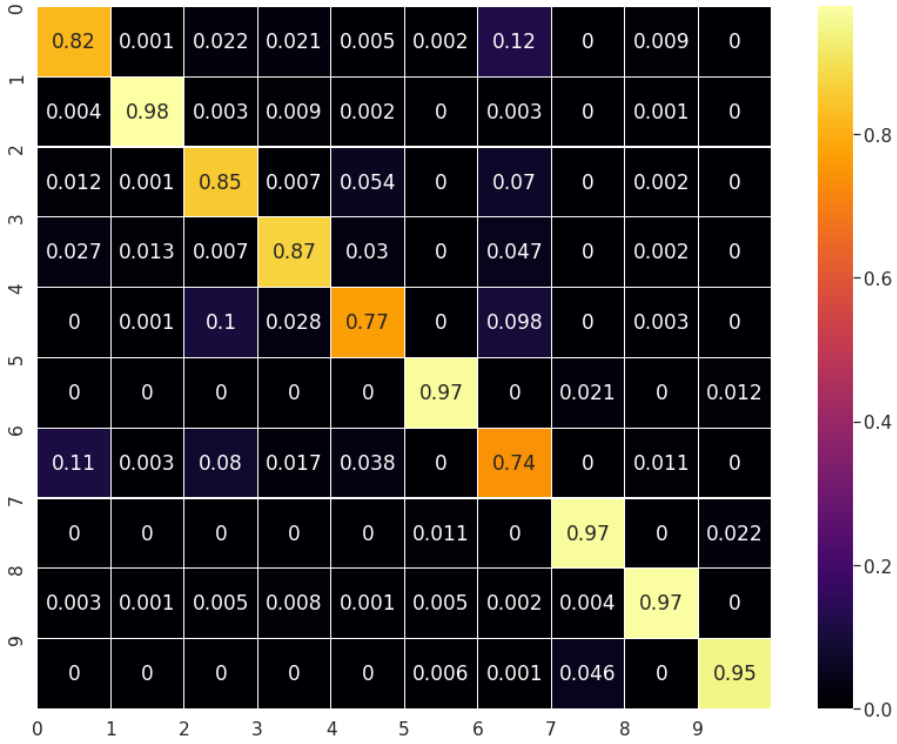

Figure 20: Confusion matrix of the best model2