Pattern Recognition Report - Serie 2c - CNN

JungleSpeed

Optimization of learning rate

Then we try to optimize the learning rate parameter. We fixed :

- epochs = 5 (default settings)

For a learning rate in the range [0.001, 0.1] we obtain the following results :

number of neurons	loss value	accuracy (%)
0.001	0.2993	91.66
0.005	0.1499	95.99
0.01	0.0936	97.37
0.05	0.0576	98.33
0.1	0.0542	98.39

TABLE 1 – Learning rate optimization

We obtained a better couple accuracy-loss values with a learning rate of 0.1.

Optimization of number of iterations

Finally we optimized the number of iterations. We fixed :

- learning rate = 0.1

For a number of iterations in the range [10, 100] we obtain the following results :

As we can see, values start to stabilize between 20 and 40 epochs. So there is no need to run the model with more than 40 epochs.

epochs	loss value	accuracy (%)
5	0.0542	98.39
10	0.0512	98.40
15	0.0482	98.56
20	0.0535	98.63
25	0.0639	98.55
30	0.0701	98.48
40	0.0669	98.57
50	0.0842	98.36
75	0.0596	98.44
100	0.0685	98.63

TABLE 2 – Number of iterations optimization

Random initialization

According to the previous results, we perform random initialization to refine our parameters and choose the model performing the best. Here our random initialization features :

- epochs = random.choice([20, 25, 30, 35, 40])
- learning rate = random.choice([0.05, 0.075, 0.1])

We obtain the following results for 8 steps :

epochs	lr	loss value	accuracy
30	0.075	0.0642	0.9852
20	0.075	0.0526	0.9842
35	0.1	0.0699	0.9863
35	0.05	0.0547	0.9852
40	0.075	0.0725	0.9840
20	0.05	0.0505	0.9854
25	0.075	0.0593	0.9851
25	0.005	0.0593	0.9846

TABLE 3 – random initialization

Best results are with the model :

- learning rate = 0.1
- epochs = 35

Here the plots showing the accuracy and loss on the training and the validation set with respect to the training epochs using tensorboard.

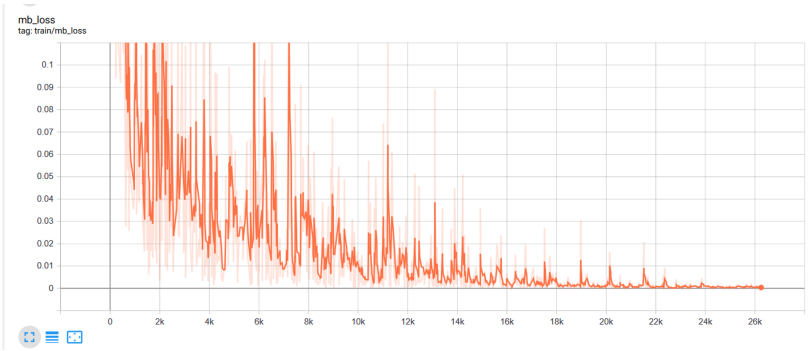

FIGURE 1 – Loss function - trainset

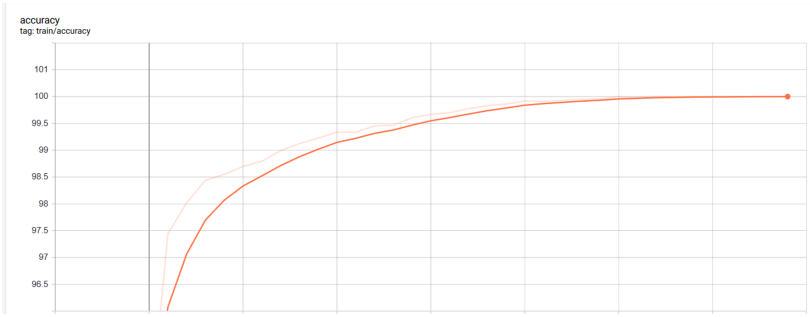

FIGURE 2 – Accuracy function - trainset

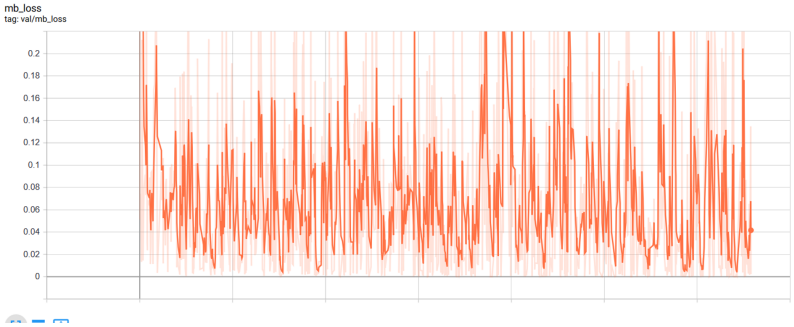

FIGURE 3 – Loss function - validation

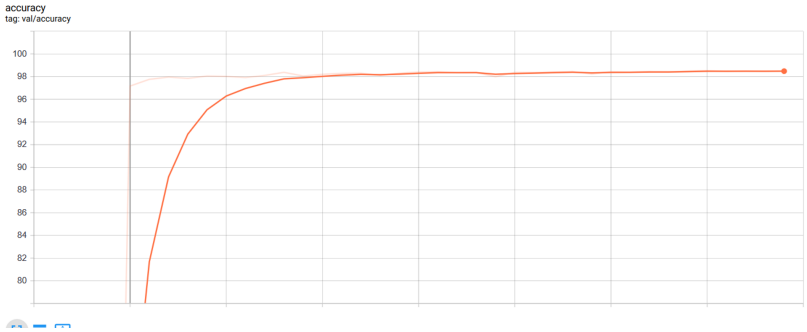

FIGURE 4 – Accuracy function - validation

Notes

For all our models, we used the *DeepDiva* framework and *TensorFlow*. We also choose to use SGD as default optimizer. We are aware that we can also change the optimizer to optimize our model.