

Project 1: Classification, Weight Sharing, Auxiliary Losses

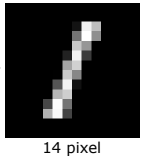
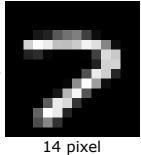
Johan Claudius Jean-Paul Félisaz, Lorenzo Robert Jacqueroud, Amir Rezaie, *EPFL Lausanne, Switzerland*

Abstract—This project aims at developing a deep neural network to compare pairs of MNIST digits and to predict for each pair if the first digit is lesser or equal to the second. To do so, different CNN architectures by changing systematically the number of filters were trained. Moreover, the effect of using an auxiliary loss function on the model performance was investigated.

I. PROBLEM DEFINITION

The goal of the project is, given a pair of gray-scale images of MNIST digits [1] stored as a tensor of the size $2 \times 14 \times 14$, to predict whether the first digit is smaller or equal to the second. Figure 1 shows an illustration of an example of the dataset input, the class of each image and the target. In this example, the first digit, 1, is smaller than the second digit, 7, thus the target is 1. For the cases where this condition does not hold, the target values are 0. Therefore, the original problem is a binary classification task.

Figure 1: Illustration of an example of the dataset

Input tensor		Class label	Target
First image	14 pixel  14 pixel	1	1
Second image	14 pixel  14 pixel	7	

II. METHODOLOGY

A. Network architecture

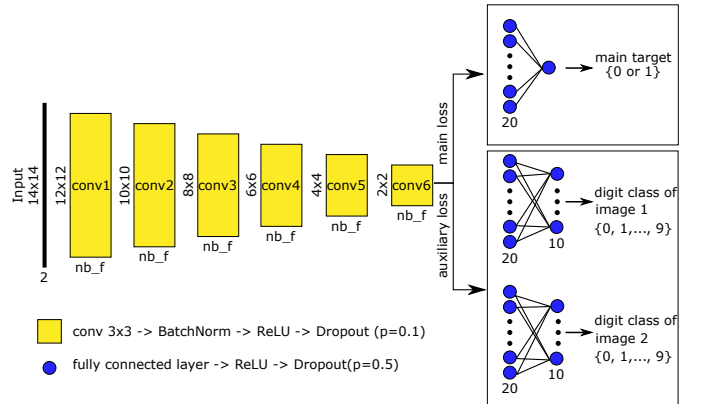
To solve the mentioned classification task, a convolutional neural network consisting of six successive blocks of convolutions followed by three parallel fully connected layers was designed (see Figure 2). Each convolution block includes, a convolution layer with “nb_f” number of filters of size 3×3 (without padding), a batch normalization layer, the ReLU activation function and a dropout layer with probability of 0.1. Using six such convolutional blocks, the feature map size reduces to $\text{nb_f} \times 2 \times 2$. The head of the network consists of three parallel fully connected layers each with 20 units. Same as the convolution layers, the ReLU was used as the activation function of fully connected layers. The dropout layer with probability of 0.5 was also used in the fully connected layers.

It must be mentioned that the number of filters (nb_filters) was set to 8, 16, 32 and 64 to investigate its effect on the performance of the model.

In the designed network, the path related to the “main loss” (see Figure 2), predicts whether the input belongs to the class 1 or 0.

The reason to use two other fully connected layers was to investigate the influence of using an auxiliary loss function on the model performance. To do so, in addition to the prediction of the main target, the class of each input image could be also predicted. Therefore, in the fully connected layers related to the “auxiliary loss” (see Figure 2), the output layer includes 10 units.

Figure 2: Network Architecture



B. Dataset

In the original MNIST dataset [1], there are 50000 and 10000 train and test images, respectively. In this project, however, 1000 samples are selected randomly for both train and test data.

Notice that the hyper-parameter search was performed using one train/test split; and to evaluate the final performance, each model was trained 10 times over randomly selected train/test data. The average accuracy and standard deviation are reported in section III.

C. Optimization Scheme

The loss function to optimize was the weighted average sum of the main loss and auxiliary loss, defined as:

$$Total_Loss = \frac{w_m \times Main_Loss + w_a \times Auxiliary_Loss}{w_m + w_a} \quad (1)$$

where, $Main_Loss$ is the binary cross entropy loss related to the prediction of the main target and $Auxiliary_Loss$ is the sum of two cross entropy losses related to the prediction of two class digits. The coefficients w_m and w_a are used to weigh the main and the auxiliary loss, respectively. The coefficient w_m was set to one, and the coefficient w_a was tuned using the grid search from the set $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0\}$.

Models were trained for 150 epochs using the Adam optimizer [2] with the parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e - 08$, and the weight decay of zero. The learning rate was selected from the set $\{1e - 2, 5e - 3, 1e - 3, 5e - 4, 1e - 4\}$ for each model separately performing a grid search. The batch size was also tuned and selected from the set $\{10, 20, 50, 100, 200\}$.

D. Evaluation metric

As our problem is a binary classification task and the dataset is not imbalanced, the metric “Accuracy” as defined below was utilized:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where, TP and TN are true positive and true negative predictions, respectively, and FP and FN are false positive and false negative predictions, respectively.

III. RESULTS AND DISCUSSIONS

Table I summarizes the selected hyper-parameters, and mean and standard deviation of the accuracy of predictions over 10 test data split for models with different nb_filters and with/without auxiliary loss (i.e. $w_a = 0$).

From Table I, it can be seen that when $w_a = 0$, increase of the nb_filters does not significantly change the mean accuracy, and the model with 32 filters has the highest average accuracy of 83.84%.

However, for the cases where $w_a > 0$ the performance of the model depends more on the nb_filters. For example, for the model with 8 filters for each convolution block, adding the auxiliary loss decreases the average performance from 81.69% to 80.81%. On the other hand, for models with the nb_filters of 16, 32 and 64 adding the auxiliary loss could improve the average accuracy by roughly 1% to 9%. This improvement is much more significant for models with nb_filters of 32 and 64 than the model with nb_filters of 16. Additionally, it appears that the rate of the increase of the average accuracy is not proportional to the nb_filters. Among all these models, the model with 64 filters per convolution block and w_a of 1.0 has the highest average accuracy 92.03% and the lowest standard deviation of 0.64%.

nb_f	lr	Batch size	w_a	Accuracy (%)
8	0.010	200	0	81.69 \pm 1.70
	0.005	200	0.7	80.81 \pm 3.38
16	0.005	100	0	83.05 \pm 0.97
	0.010	50	0.8	84.19 \pm 4.46
32	0.010	50	0	83.84 \pm 0.94
	0.005	50	1.0	90.49 \pm 1.76
64	0.005	100	0	83.10 \pm 0.90
	0.005	100	1.0	92.03 \pm 0.64

Table I: Hyper-parameters and performance of models over 10 test data splits

REFERENCES

- [1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [2] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.