Comparison of different network architectures on the classification and comparison of handwritten digits

Hugo Michel
hugo.michel@epfl.ch

Daniel Tadros
daniel.tadros@epfl.ch

Gianni Giusto
gianni.giusto@epfl.ch

Abstract

The present project proposes a training protocol for the classification and the comparison of handwritten digit images from the MNIST dataset. Methods such as weight sharing as well as parameters optimization were implemented in several models through different architectures to assess the overall performance in terms of error and computation time for the given task. A testing error of 2.88% was finally achieved with the best performing model.

1 Introduction

Artificial neural networks have become increasingly important in the context of deep learning and pattern recognition. These powerful processing paradigms consist of dense interconnected layers and aim to solve specific tasks. The architecture varies from application to application and not surprisingly affects the outcome and the overall performance of the process.

The main concern of this study is to compare different architectures and train a network to discriminate between two handwritten digits. Indeed, given a set of two images from the MNIST database as input, the model aims to determine whether the first one is smaller or equal to the second. To do so, parameters optimization through grid search was conducted in parallel to commonly used techniques such as weight sharing. A final model was proposed with the optimal parameters and network structure leading to a minimal error. The different models were all implemented using the `PyTorch` framework.

2 Models and methods

The given training and testing data consist each of 1000 pairs of 14×14 grayscale images from the MNIST dataset. Each image in a pair is associated to a digit class ranging from 0 to 9 and each pair is labelled with a 0 if the first digit is smaller or equal to the second or 1 in the other case.

A wide variety of architectures can be used to solve this task. Nonetheless, two main pipelines were tested (hereafter referred as *baseline* or *reference* and *2-networks architecture*): the first one is a single network that performs only the classification of each digit and compare them "manually", whereas the second architecture trains a second network to compare them.

2.1 Baseline

For the classification alone, the network was inspired by architecture already used for similar task from the literature. The one used consists of 2 convolutional layers with a kernel size of 5. After each convolution, a max pooling is performed with a kernel size and stride of 2. At the output of these CNN, a fully connected layer flattens the image into a 64×1 dimension vector. Two hidden layers will rise the dimension up to 256 and the final output is a 10×1 tensor containing the power for each class. The activation function used is the Rectified Linear Units (ReLU) and a 2-dimensional batch normalization is performed between the 2 hidden layers. Comparison is performed by selecting the highest probability from the digit classification (*i.e.* by performing an *argmax*) and compare the pairs systematically.

2.2 *2-networks* architecture

The *classification* part of the second architecture is the same as in the baseline. The output of this first network is a 10×1 tensor corresponding to the probability of a digit to be belong to a certain class (from 0 to 9). The loss obtained from the classification, referred as auxiliary loss, is used together with the loss from the comparison model. The total loss is expressed as a weighted sum of the 2 losses and the backpropagation is performed along

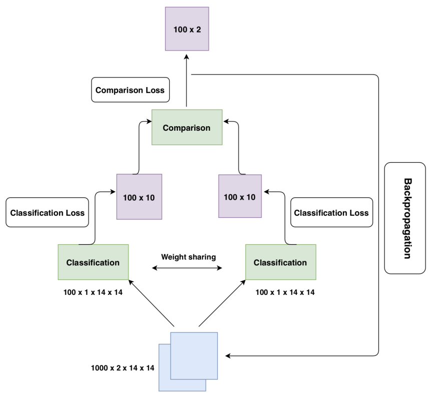

Figure 1: Training pipeline followed for the *2-networks* model. The input consists of two 14×14 images (*blue*) each of which are passed successively into the classification and the comparison networks (*green*). The weight-sharing is done during the classification task. Output tensors are represented in *purple*.

all the architecture at once (Fig. 1). The effect of weight sharing is also tested in this model to assess performance and computation time associated to a reduction in the number of parameters. For the non-weight sharing condition, the backward pass and the optimisation steps are performed for both images individually.

2.3 Protocol

Tests were first carried out on the *baseline* to make the classification as efficient as possible prior comparing the digits. Parameters and architecture modalities were then tuned and optimised using grid search to minimise the loss on the *2-networks* architecture. The entire dataset was passed 25 times to the networks and the training was repeated 10 times.

3 Results

3.1 Baseline

The model trained with the architecture mentioned in section 2.1 together with a learning rate of 10^{-3} and an Adam optimizer which adapts individual learning rates for each parameters led to an error of 2.88% (CNN + *argmax* in Table 1). The *cross entropy loss* criterion is more suitable for classification task and was hence chosen here.

3.2 *2-networks* architecture

Testing errors obtained from the parameter optimization step are summarized in the grid search in Fig. 2. A learning rate of 10^{-3} for the Adam optimizer turned out to be optimal when associated to a ReLU activation function. Convergence does not occur for learning rate higher than 10^{-1} and sigmoid activation function systematically led to higher error rates. Little difference was observed between ReLU and leaky-ReLU.

A simple architecture of three fully connected layers with hidden layers of size 40 and 80 associated to ReLU led to an test error of 2.92% for the weight-sharing condition and 3.56% without (both referred as CNN + fully-connected small in Table 1).

A second larger architecture consisting of 5 fully-connected layers (hidden layers of size 40, 80, 140, 80) for the comparison model led to an error of 3.17% with weight-sharing and 3.64% without (CNN + fully-connected large in Table 1). No sign of overfitting was observed.

The number of parameters were significantly reduced by using weight-sharing and small fully-connected layers. Hence, the computation process for the training was decreased by 28% to 37% depending on the computer. Note that learning rates, optimizers as well as activation functions were kept constant across the different models.

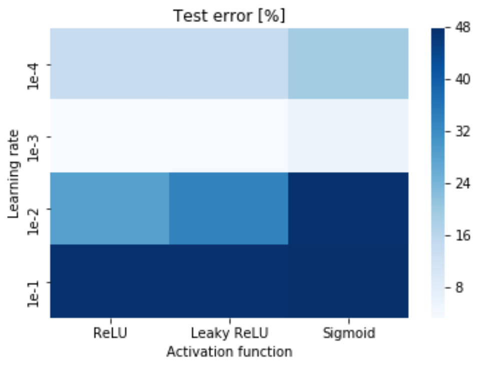

Figure 2: Test errors (in %) associated to the *classification* model for different learning rates and activation functions.

4 Discussion

Low error rates were obtained for both the *baseline* and the 2-networks architecture (systematically below 4%). However the first one surprisingly turned out to be more efficient.

The choice of the 2-model-based approach with a backpropagation operating on all the architecture seemed to be more appropriate for the *comparison*

Table 1: Performance comparison between different model architectures.

Models	w **sharing**	**Parameters**	**Training %**	**Testing %**
Baseline				
CNN + *argmax*	No	(192'020, -)	0.26 ± 0.089	2.88 ± 0.244
2-networks architecture				
CNN + fully-connected small	No	(192'020, 4'282)	0.21 ± 0.073	3.56 ± 0.43
CNN + fully-connected small	Yes	(96'010, 4'282)	0.22 ± 0.131	2.92 ± 0.45
CNN + fully-connected large	No	(192'020, 30'122)	0.89 ± 0.38	3.64 ± 0.42
CNN + fully-connected large	Yes	(96'010, 30'122)	0.29 ± 0.242	3.17 ± 0.921

step. Indeed, both networks are not totally independant and it is possible for a digit to be misclassified by the first one, hence biasing the comparison process. For example, imagine a case where the first digit is a 5 and the output of the classification for the second one is 0.15 for the first 4 digits and 0.4 for 9. The method used for the classification only pipeline will perform an *argmax* to choose the number with the highest power (9 in this case). However, there is a higher probability for the second digit to be smaller than 5, as the combined power of 0 to 3 is 0.6. Hence, a second network collecting the class power for each digits and trained to assess whether the first one is smaller/equal or greater than the second was thought to be more robust for the given task.

This difference between our initial expectations and the obtained results could be explained by the high efficiency and accuracy of the classification process. Indeed, it is known from the literature that digits analysis from the MNIST dataset is a well-known problem and simple models implemented in `PyTorch` can easily lead to low error rates. Increasing the complexity by adding a second network will lead to higher error rate in the present case.

However, this second approach is also characterized by the use of weight sharing, a commonly used method in machine learning to reduce the number of parameters and hence the overfitting. Indeed, the pair of images to compare can be passed in the model (one after the other) without updating the parameters in-between. Thus, the same parameters are used to classify a pair of digits (and not two different networks). The errors were systematically lower and the process was significantly speeded up because the number of parameters was reduced by half when using these shared weights.

Increasing the number of fully-connected layers in the *comparison* network did not show any improvement in the performance. Furthermore, the number of parameters was obviously higher as the architecture increases in complexity. Hence, the simplest architecture of 3 layers is preferred here in order to keep the model as simple as possible.

5 Conclusion

The baseline remains a powerful and appropriate model for the comparison of digits. However, weight sharing together with the weighted loss of the *2-networks* architecture increases the overall performance in terms of time while keeping a low testing error. This alternative may be more convenient for larger datasets and would be preferred for more complex tasks as the backpropagation is operated on the whole training pipeline.

References

Yann. LeCun, Yoshua. Bengio and Geoffrey. Hinton. 2015. *Deep Learning*. Nature, Vol. 521.