Ecole Polytechnique Fédérale de Lausanne

Deep Learning

EE-559

# Project 1 Classification, weight sharing, auxiliary losses

*Authors:*
Ertuğrul Mert
De Lima Carvalho Luis
Gaiffe Raphaël

May 22, 2020

# 1 Introduction

The aim of this project is to design neural network architectures to compare two digits present in the two 14x14 channels of the input image. Specifically, the effects of weight sharing and the introduction of an auxiliary loss are evaluated. Architectures are implemented with Pytorch.

# 2 Architectures

For the design of architectures, an overall top down approach was followed, as we had already used weight sharing and auxiliary loss extensively for the design of our initially proposed architecture. We proposed an initial architecture, optimized its hyperparameters, made changes to hidden layer sizes and depth as well as working to reduce its overfitting and afterwards, we removed or reduced the effect of key features we intended to observe the impact of. By changing the weight/impact of one feature at a time, we were able to demonstrate its effect on performance. Given the structure of the classification task at hand, it was intuitive to view it as the combination of two steps, the first one being the recognition of the digits given as input and the second step being their comparison. Every architecture we implemented stemmed from variations of this idea. Three main architectures were tested: Siamese Model, Reduced Weight Sharing Model and Siamese MLP Model. Each model has versions with and without auxiliary loss, when this is taken into account, they can be considered as 6 models. Our aim was to observe the effects of:

- Complete vs. partial weight sharing
- With vs. without (or varying amounts of) auxiliary loss
- With vs. without convolutional layers

Thus, all architectures were selected to inspect weight sharing and auxiliary loss, when convolutional layers are also thought of as a form of weight sharing.

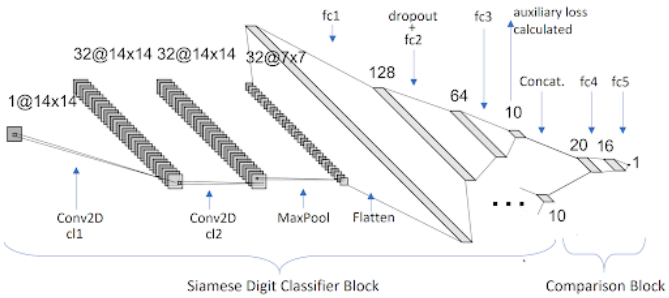



Figure 1: Siamese model

## 2.1 Siamese Model

The initial architecture we implemented was composed of a siamese (full weight sharing) block followed by fully connected layers without weight-sharing. The siamese block consists of two convolutional layers for feature extraction followed by max pooling and fully connected layers. The output of the siamese block is used to obtain an auxiliary loss based on the original digit labels of the images.

- Each 14x14 digit in the pair of digits is fed to the same block, effectively making the block siamese. Then their outputs are concatenated.
- Throughout the model, ReLU activation function was used. Only the final layer ends with a sigmoid activation.
- One layer of dropout was introduced to reduce overfitting and its parameter was optimized.
- The auxiliary loss was calculated with Cross Entropy Loss on 10-wide output of the siamese block using digit class labels.

- For the final output, one output was used, alternatively the binary classification can be in one-hot encoded format too, which would not change the result. For the loss criteria, since a single binary output is used, BCELoss was preferred. BCELoss preceded by sigmoid activation produces the same results (for single output) as Cross Entropy Loss (for one-hot encoded output).
- The final classification loss was combined with auxiliary loss obtained from each digit weighted by the parameter "loss_ratio". This parameter determines how important digit classification is to the overall classification. It was optimized during the grid search for parameters.
- By default the model is set to run with 25 epochs, 0.005 learning rate and 0.7 loss_ratio. Loss ratio and learning rate are not independent variables, therefore changing one may result in vanishing or overshooting of the gradient.
- Adam optimizer was used as it was found to work better than SGD for our model.

### 2.2 Reduced Weight Sharing (RWS) Model

This model differs from the Siamese Model only in two layers. The second convolutional layer cl2 and the second fully connected layer fc2 were duplicated for each digit of the pair. The aim was to produce the version of the siamese model with less weight sharing to see the impact of weight sharing on performance.

The Sequential module allows us to create a model by using the Linear module to create the layers in combination with one of the two activation functions. The following sequence creates one layer of N inputs, one of M outputs and a hidden layer with P units with their respective chosen activation functions:

### 2.3 MLP Model

This model was expected to be inefficient and it was implemented to demonstrate the jump in computational efficiency when convolutional layers are introduced. The sizes of the layers were selected to mirror the sizes of the siamese model, so the output of each layer is the same size as the convolutional layers' outputs after flattening. The resulting number of parameters was remarkably higher.

## 3 Results

Two different types of comparison were made, first the performances of different architectures were compared and afterwards, the performance of the siamese model with different auxiliary loss weights was observed.

### 3.1 Performance Comparison of Architectures

Each model and their versions without auxiliary loss were trained and tested for 10 rounds each on 1000 training and test samples for 25 epochs. The performance results are given in the table below.

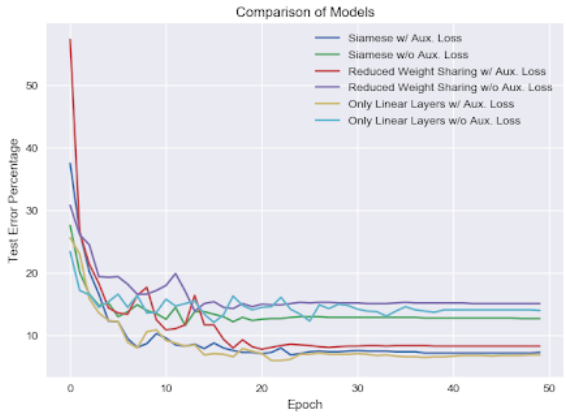
| Architecture | Avg. Test Error | Average Training Error | Std of Test Error | Nb. of Params | Training + Test Time (s) |
|---|---|---|---|---|---|
| Siamese w/ Aux. Loss | 4.56% | 0.40% | 0.970 | 115147 | 23 |
| Siamese w/o Aux. Loss | 12.64% | 0.22% | 1.776 | 115147 | 27 |
| RWS w/ Aux. Loss | 7.04% | 0.58% | 1.028 | 128555 | 23 |
| RWS w/o Aux. Loss | 16.07% | 0.53% | 1.303 | 128555 | 26 |
| MLP w/ Aux. Loss | 6.57% | 0.55% | 1.110 | 11177227 | 70 |

Table 1: Performance Results

The time measurements were taken over a single trial as computer performance will drop with multiple trials. They depend on the device used so they should be considered as an ordinal metric. Among all models, the Siamese model with auxiliary loss gives the smallest average test error, and compared to the version without auxiliary loss, it appears that overfitting is reduced dramatically. MLP comes second, but at the cost of a much longer training + test time, meaning that it is computationally inefficient which is

also seen from the number of parameters. The increase in error rate from Siamese to RWS demonstrates the incremental positive impact of weight sharing on performance. The standard deviations are affected by the inherent randomness of the dropout layer as well as weight initialisation and sample selection but overall, auxiliary loss decreases deviation in all models.

The plot on the right depicts the comparison of the models' learning over epochs in terms of test error percentage (as auxiliary loss effects loss calculation, error was preferred as the metric). This plot was obtained from only one iteration of each model. It parallels the results in the table above, although the MLP and Siamese errors change place. This can be attributed to the data being taken from 1 iteration. The versions with and without auxiliary loss are clearly separated.




### 3.2 Effect of Auxiliary Loss


The Siamese model was trained and tested for one round each for different values of the parameter "loss_ratio" which is the weight of the auxiliary loss. Overall, performance appears to improve with auxiliary loss weight, but the trend becomes unclear for weights above 0.5. Since we run our models for 25 epochs in our test script, we optimized this parameter differently than depicted here and we chose 0.7. Since this plot is made up of 1 round of data for each value, it should not be seen as a definite comparison but it demonstrates a general trend of improvement with auxiliary loss until a certain level before error rates plateau.

## 4 Conclusion

The architectures we used for this classification task used weight sharing and auxiliary loss extensively and demonstrated how an incremental change in their weight or amount is reflected in the performance of the classifier. Complete weight sharing (Siamese) was found to be optimal in the digit classifier block, while the optimal weight of the auxiliary loss depends on the number of epochs and is affected by learning rate. The Siamese Model with auxiliary loss was both the most accurate and cost-effective model.

# 5 Appendix

# References


[1] Fleuret, F., 2020. EPFL EE-559 – Deep Learning., *[online] Fleuret.org. Available at: <https://fleuret.org/ee559/> [Accessed 15 April 2020].*

[2] Unsupervised Thoughts, 2020. Learning through Auxiliary Tasks*[Blog] Available at: <https://becominghuman.ai/siamese-networks-algorithm-applications-and-pytorch-implementation-4ffa3304c18> [Accessed 23 April 2020].*

[3] Holländer, B., 2018. Siamese Networks: Algorithm, Applications And PyTorch Implementation. *Available at: <https://vivien000.github.io/blog/journal/learning-though-auxiliary_ tasks.html#fn:2> [Accessed 18 April 2020].*