EPFL

Project 1 – Classification, weight sharing, auxiliary losses

Martin Everaert, Antony Doukhan

May 22, 2020

Contents

This report is 3 pages long without this cover page and without the annexes with some additional works.

1 Introduction

The objective of this project was to study binary classification using Pytorch neural network modules. Given the MNIST data sets, the task was to predict whether the first digit is less or greater than the second one and study the performance of the performance when using weight sharing and/or auxiliary loss. The task was performed on the MNIST datasets.

2 Models

2.1 Architecture

We first define a module called `DigitNet` that has as input a 14×14 image and outputs a tensor of size 10. This module is composed of 3 convolution layers producing 16 6×6 feature maps and 3 fully-connected layers transforming those 16 6×6 feature maps into the tensor of size 10. Each layer is also composed of a batch normalization (2D) and an activation (SoftMax for the last fully-connected layer, parametric ReLU for all the other layers). Intuitively, the `DigitNet` module can be thought as a module that take an image of a digit and compute for each digit i the confidence of the module for the image to represent digit i. We use that for our auxiliary loss.
Our model, that takes as input 2 images and predicts whether the first digit is less or greater than the second one, processes each of the 2 images in 2 `DigitNet`s (possibly sharing weights). By multiplying the 2 computed tensors of size 10, it obtains a tensor of size 10x10 that is passed through a final fully connected layer to output one number. Intuitively, the tensor of size 10x10 contains, at coordinates (i, j), the confidence of the model for the images to represent digits (i, j). Therefore a single fully-connected layer is enough since "it only has to learn $1_{i \leq j}$ as weights" to compute the confidence for the first image to represent a lesser or equal digit than the second image. Figure 1 represents our model we called `PairSetsModel`.
Our main objective loss is the MSE between the output of the network and the class to predict (1 if the first image represents a lesser or equal digit than the second image, 0 otherwise).

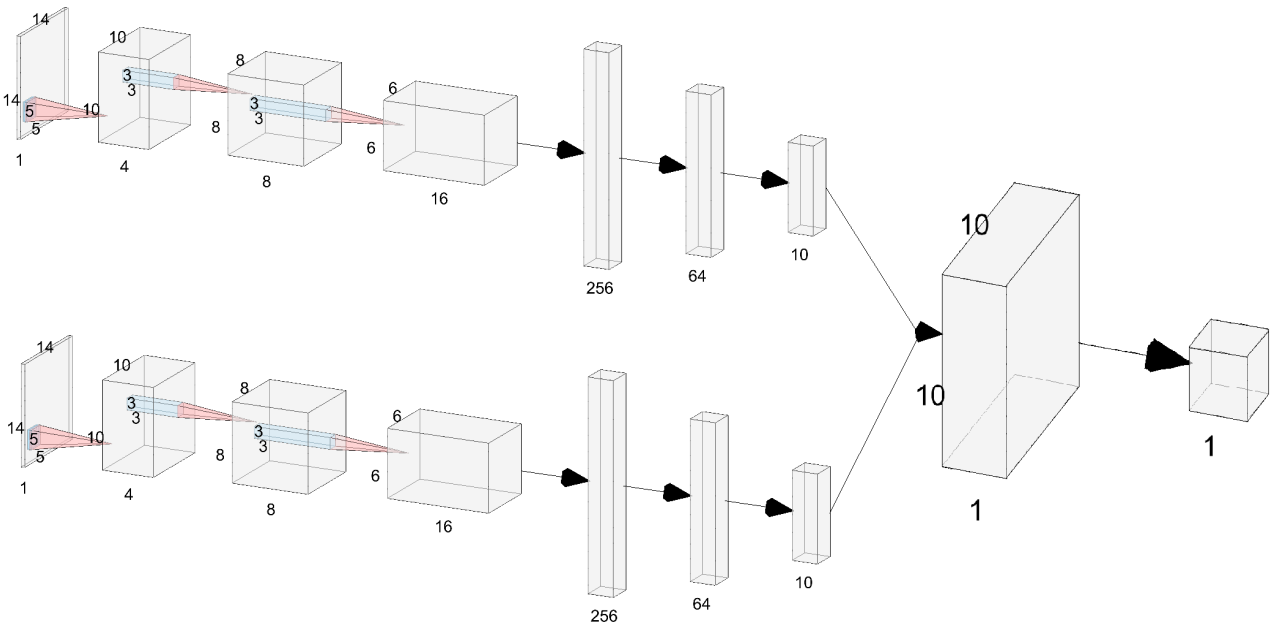

Figure 1: our `PairSetsModel` architecture

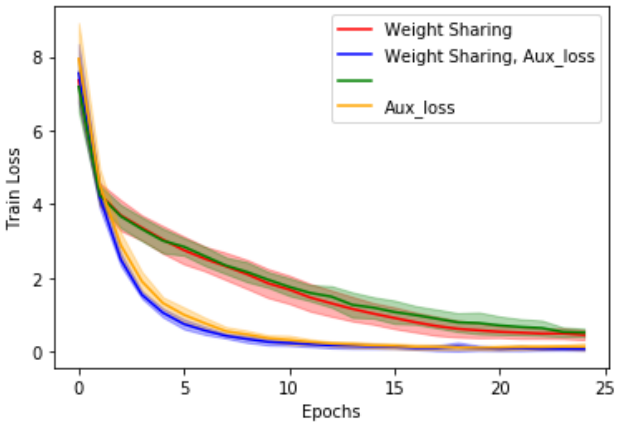

Figure 2: Evolution of train loss during 25 epochs

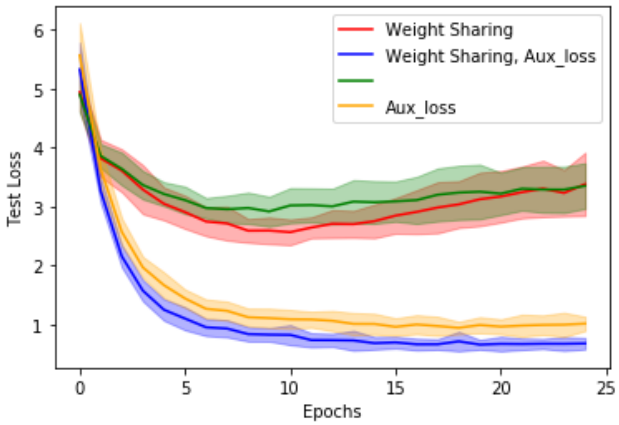

Figure 3: Evolution of test loss during 25 epochs

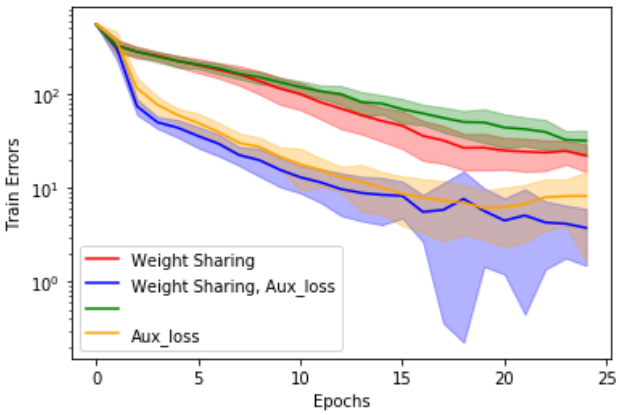

Figure 4: Evolution of train errors during 25 epochs

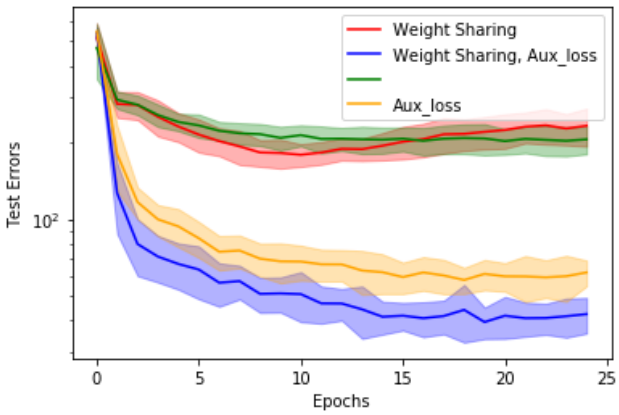

Figure 5: Evolution of test errors during 25 epochs

2.2 Auxiliary loss

Our approach for auxiliary loss consists in adding the MSE between the outputs of the `DigitNets` and the (one-hot encoded) digits represented in the images. In order to compare models with and without auxiliary loss, the train/test loss plots are only the main objective loss.

2.3 Weight sharing

For weight sharing models, the two images are processed throught the same `DigitNet`. This can be seen as 2 `DigitNets` sharing weights. In practice, our class `PairSetsModel` includes a boolean hyper-parameter for this weight sharing technique.

3 Results

All the experiments have be done with 1,000 pairs for training and test. Provided estimations have been made through 15 rounds for each architecture, where both data and weight initialization are randomized. For a fixed architecture and a fixed epoch, we compute the mean (*mean*) and the standard variation (*std*) of the 15 numbers we got. The plots show the mean value *mean* and the area from $mean - std$ to $mean + std$ is filled.

3.1 Impact of auxiliary loss

Auxiliary loss usage is the main performance improvement. In all plots, the 2 models with auxiliary loss outperform the 2 models without. Indeed the classes of the digits represented in the images is a key element to know whether the first represented digit is smaller than the second. It is harder to learn those 10 classes with only the main objective loss. In addition, you may note that models without auxiliary loss begin over-fitting around the 10^{th} epoch whereas it is not clear whether models with have started over-fitting after 25 epochs. The out-performance is even clearer in the test set than in the train set. We interpret this as the fact auxiliary loss also provides the network a better generalization ability.

3.2 Impact of weight sharing

Models with weight sharing perform also slightly better than the ones without. The out-performance is not negligible since it has an order of magnitude of 1 or 2 times the standard deviation. Indeed, our architecture has been designed having in mind a first digit classification step (`DigitNets`). This intuition leads us to think the 2 images have to be processed the same way in this classification step. You may note the model without weights sharing (no auxiliary loss) seems to catch up the one with. This could be explained by the fact the `DigitNets` 'get aware' of the main objective.

4 Conclusion

Our experiments show that the use of auxiliary loss, in particular digit classes which is undoubtedly useful for digits comparison inequality, is a very significant way to improve main objective performance and generalization ability. This also shows the importance of loss tuning in neural networks design. In a less importantly manner, in case it is meaningful to pre-process the same way several areas of the input (the 2 digit images in our case), weight sharing also provides an additional performance improvement. Furthermore, it is beyond doubt that weight sharing can only reduce model size, over-fitting effect and training duration thanks to less parameters to train.

References

[1] *Course EE-559 "Deep Learning", by* François Fleuret, *EPFL, Switzerland.*

5 Annexes with some additional works

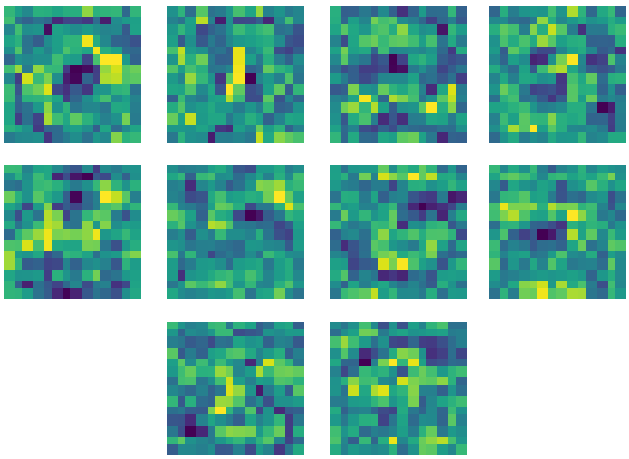

Figure 6: Weights learned by a model with only 10 convolutional 14×14 kernels with auxiliary loss after 1 epoch

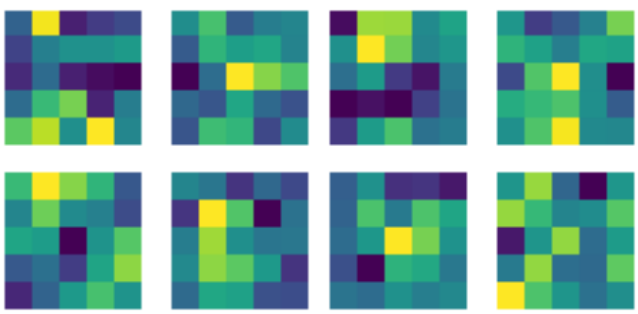

Figure 7: Weight-sharing, no auxiliary loss

Figure 8: Weight-sharing, auxiliary loss

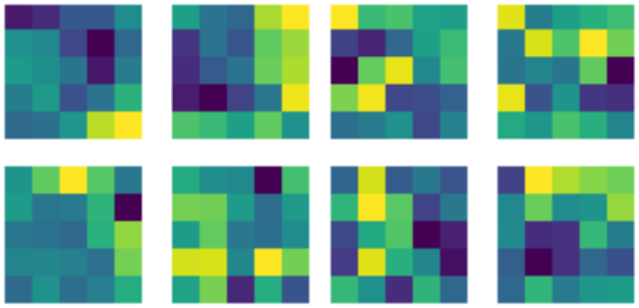

Figure 9: No weight-sharing, no auxiliary loss

Figure 10: No weight-sharing, auxiliary loss

Figure 11: Weights of the 4 first-layer convolutional (`Conv2D`) 5×5 kernels after 25 epochs without dropouts *(4 filters for each of the 4 models)*

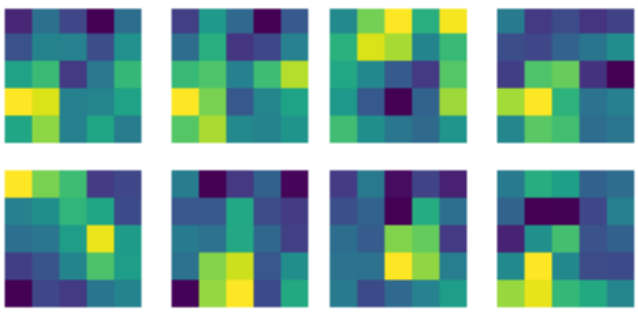

Figure 12: Weight-sharing, no auxiliary loss

Figure 13: Weight-sharing, auxiliary loss

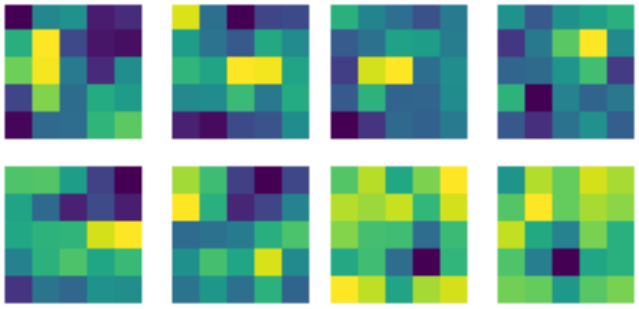

Figure 14: No weight-sharing, no auxiliary loss

Figure 15: No weight-sharing, auxiliary loss

Figure 16: Weights of the 4 first-layer convolutional (`Conv2D`) 5×5 kernels after 25 epochs with dropouts (`Dropout2D`s and `Dropout`s) *(4 filters for each of the 4 models)*