Mini-Project 1 · Report
Deep Learning · EE-559

Tomas Larraz Giro ◇ Daniele Rocchi ◇ Giulio Trichilo

May 17, 2019

The purpose of this project is to solve the problem of comparing two digits of the MNIST dataset to see which one is greater. Hence, first we compare different architectures and their accuracy. Then, we implement weight sharing and an auxiliary loss and we find that, together, increase the accuracy of the network, on average.

1 Implementation of the architectures

Architecture presentation To solve the problem, we implemented three types of architectures. The three models have the same structure. As we can see in Figure 1, the images are fist recognized separately: they go through the "architecture" layer. This layer is the variable layer across models. Indeed, in the "deep" implementation, it is composed of two convolutional layers followed by three linear layers, all with ReLU activation functions. Then the "deep with sigmoids" implementation is similar to the "deep" implementation but with sigmoid activation functions instead of ReLU. Finally, the third architecture is the "fully connected", which has four fully connected linear layers with ReLU activation functions. Once the images have gone through the "architecture" layer, they are flattened and concatenated in one tensor. Then, they go through one last linear layer which is common across different implementations. As we can see in Figure 1, there are 2 outputs to the models, an "auxiliary output" and the final "output". These will be useful for the second part. Finally, we recall that to make our comparison of different models fair, we made sure to let them have the same number of parameters (around 95'000).

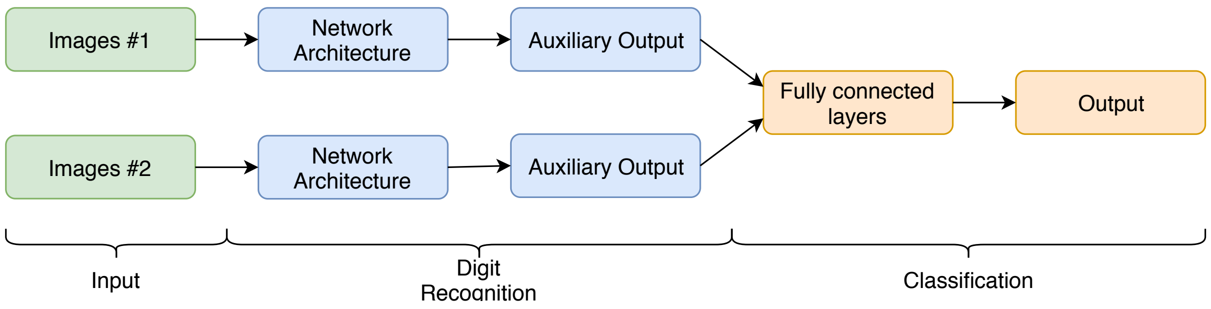

Figure 1: Diagram of the architectures of the models.

Results We trained each architecture 10 times (with 30 epochs each time) and took the average of the maximum accuracy. We also computed the standard deviation of the accuracy for each architecture. The results can be seen in Table 1. Furthermore, as we can see in Figure 2, the most accurate model is the "deep" model, followed by the "deep with sigmoids" and, finally, the "fully connected".

Architecture	Average accuracy	Std. of accuracy
Deep	78.83%	0.72%
Deep with sigmoids	75.56%	1.04%
Fully connected	74.58%	0.49%

Table 1: Average accuracy after training the model with 10 times and without using weight sharing or auxiliary loss function.

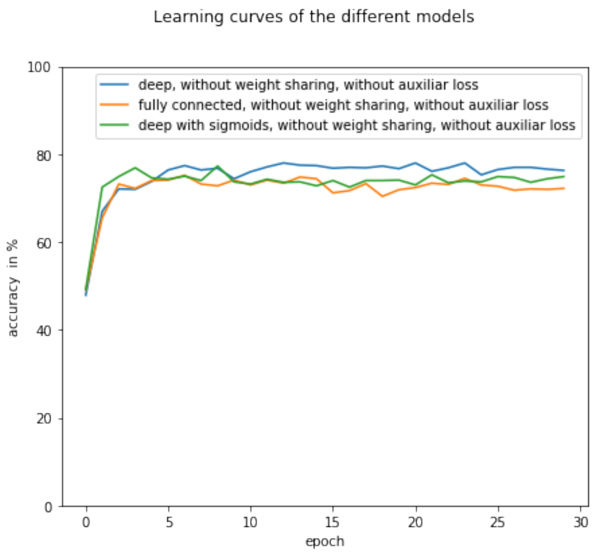

Figure 2: Comparison of the learning curves without weight sharing or auxiliary loss.

2 Weight sharing and auxiliary loss

Weight Sharing The idea to improve the accuracy of the models was to use weight sharing and auxiliary loss. Specifically, with the expression "weight sharing" we mean that the weights for the two "architectures" in Figure 1 are the same. This should make the models more efficient. Indeed, when we train, we pass the first image through the "architecture" layers, then the second, and then we let autograd deal with the computation of the gradients. This approach seems, on average, to improve the accuracy of the models. As we can see in Table 4, almost all models with weight sharing did better than the models without weight sharing. Quantitatively, as indicated in Table 2, we see that the weight sharing accounts for almost 1% increase in the average maximum accuracy across models.

Weight sharing	Average accuracy	Std. of accuracy
True	77.37%	2.80%
False	76.31%	1.34%

Table 2: Average results of the tree models with and without using weight sharing.

Our findings indicate that weight sharing is a very useful technique. In this particular problem with scarce data it allows the "architecture" component to gain experience by learning from the dataset of the first image and the second. This explains why the weight sharing has a positive effect (+1% on average) on the accuracy.

Auxiliary loss The idea behind auxiliary loss is to use the true class prediction in order to train the model in a more efficient way. In order to test whether we could improve our accuracy using auxiliary loss, we trained all our models with the normal loss, namely "main loss", and with another loss, namely "sum loss". The "main loss" is the binary cross entropy from the output and the expected output. The "sum loss" is composed of two parts: a part proportional to "main loss" and a part proportional to "auxiliary loss". This last loss is the sum of the categorical cross entropy of the recognition of the first image (i.e. the auxiliary output in Figure 1) and the categorical cross entropy of the output of the second image. To implement "sum loss", we were careful not to simply sum the main loss and the auxiliary loss. This result would potentially over-weight one over the other. What we did was to normalize the two losses by their initial values and then add them.

Results In practice we found out that the models with auxiliary loss outperform the models trained without auxiliary loss. Although, for the "fully connected" architecture this is not the case. Quantitatively speaking, we found out that the accuracy in the "deep" and "deep with sigmoids" architectures was raised by 0.9% on average. The overall results are given in Table 3. Note that the confidence intervals to one standard deviation do not overlap. Indeed, the estimated standard deviation for the mean of one training example is around 1%. This gives a confidence interval of ±0.31% for one standard deviation around the mean.

Auxiliary loss	Average accuracy	Std. of accuracy
True	78.82%	0.96%
False	77.91%	0.82%

Table 3: Average results of the tree models with and without using auxiliary loss.

Architecture	Auxiliary loss	Weight sharing	Average accuracy	Std. of accuracy
Deep	True	True	81.91%	1.21%
Deep	False	True	80.62%	0.44%
Deep	True	False	79.43%	1.34%
Deep	False	False	78.84%	0.72%
Deep with sigmoids	True	True	78.03%	0.60%
Fully connected	False	True	76.71%	1.06%
Deep with sigmoids	False	True	76.64%	1.08%
Deep with sigmoids	True	False	75.91%	0.70%
Deep with sigmoids	False	False	75.56%	1.05%
Fully connected	False	False	74.58%	0.49%
Fully connected	True	False	73.55%	3.77%
Fully connected	True	True	70.31%	12.40%

Table 4: Ranking of all models by maximum average accuracy.

In conclusion, we have tried to solve the digit comparison problem with three different architectures. Hence, first we have seen that auxiliary loss can be useful for some architectures like convolutional layers. Then, we noticed that weight sharing proved to be very useful in this problem. To sum up, the best model in terms of average accuracy reached (out of 10 iterations) is the "deep" architecture with both auxiliary loss and weight sharing enabled as shown in Table 4. This architecture reaches and average accuracy of almost 82% with 1.21% of standard deviation.