Mini Project 1

Deep Learning (EE-559), Prof. François Fleuret

May 22^{nd}, 2020

Name: PILLONEL Ken SCIPER: 270852
Name: PLASSMANN Jérémy SCIPER: 273908

Contents

Introduction

The goal of this mini-project is to study the performance of different architectures when comparing two hand-written digits visible in a two-channel image. The effects of weight sharing and an auxiliary loss will be shown.

1 Architecture

1.1 Model 1

This first model uses no weight sharing, the two input images are passed through two separate convolutional networks. The output of these 2 networks is then concatenated and passed through 4 fully connected layers. There is a single output neuron indicating if the first digit is larger (output should be 1) or if the second one is (output should be 0).

1.2 Model 2

This second model uses weight sharing, the two images are passed through a single convolutional network. The outputs are then concatenated and passed through 4 fully connected layers. There is a single output neuron indicating if the first digit is larger (output should be 1) or if the second one is (output should be 0).

1.3 Model 3

This third model introduces a set of new outputs. The network must now also predict the digit of each image. This is used as an auxiliary loss. The two images are passed through a single convolutional network. The outputs are concatenated and passed through a series of fully connected layers. The network then has 3 outputs:

- A single neuron to predict which digit is the largest.
- Two softmax outputs two predict the labels of the 2 images. These two outputs use weight sharing.

1.4 Model 4

This fourth model adds batch normalization on top of the third model. The network then has 3 outputs:

- A single neuron to predict which digit is the largest.
- Two softmax outputs two predict the labels of the 2 images. These two outputs use weight sharing.

2 Results

Our experiments were done with 1000 pairs for training and testing. We trained them with 60 epochs so that the accuracy could saturate. 100 rounds were run over each model to compute accurate mean and standard deviation values for each epoch. Results are shown on Figure 1.

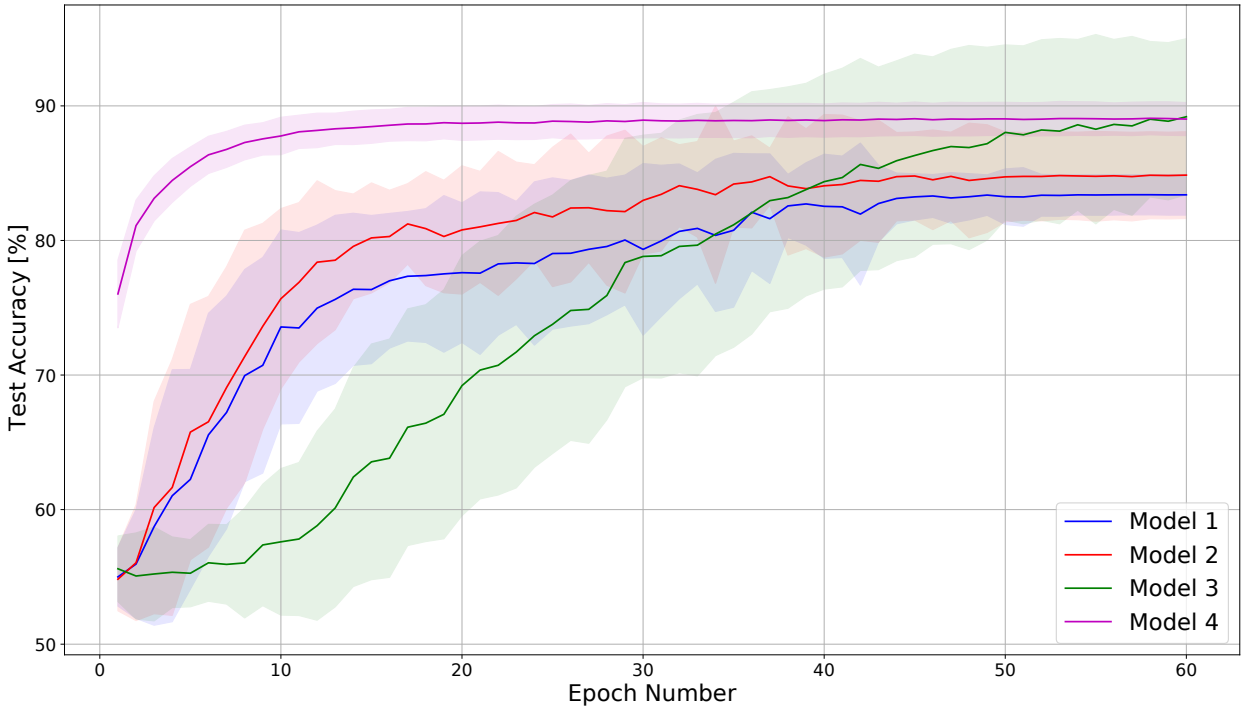

Figure 1: Mean and standard deviation for each epoch over 100 rounds

	End Test Accuracy [%]	**Total Time [min]**	**Average Round Time [s]**	**Normalised Time [%]**
Model 1	83.75 ± 0.95	27.06	27	100
Model 2	85.15 ± 2.41	29.33	29	107.4
Model 3	89.35 ± 4.30	30.75	31	114.8
Model 4	88.94 ± 0.91	56.76	57	211.1

Table 1: Performance over 100 rounds with 60 epochs

2.1 Discussion

When comparing the first two models that are similar in their architecture, we see that the introduction of weight sharing slightly improves the overall accuracy of the model while increasing computation time by 7%.

The training function was then adapted to be able to train Model 3 which introduced the prediction of the digit. To do so, an auxiliary loss has been added to the original loss. Compute time was again increased and accuracy improved yet the performance was highly unstable and unreproducible as variance was very high. This is shown in light green in Figure 1. With the introduction of batch normalization in Model 4 the results are a lot more stable and there is considerably less variance for each epoch. Accuracy is kept to a similar level compared to Model 3. The major downside to the introduction of batch normalization was that the compute time doubled which can be a problem if resources are a constraint.

A solution would be to decrease the number of epochs for this model as it can be seen that it tends to saturate already after roughly 30 epochs. This would divide the compute time by 2 and bring it in the same range of computing times as the other 3 models, making it the best model out of the lot for this problem.

3 Conclusion

In conclusion it was a great mini-project to help us visualize the effect of different deep learning techniques on our dataset. It was also a great opportunity to get some practical experience as an exercise with the PyTorch library and improve our group coding skills.