Task 1

Dimitri Coukos, Paola Malsot, and Niels Mortenson

Mai 2019

1 Introduction

The aim of this project is to compare alternative network architectures on a classification task. Nets are fed with two 14x14 downscaled versions of MNIST images and then classify the images based on which represents the greater digit.

We explore the use of auxiliary losses, weight-sharing and noise removal in variants of a principal architecture (described in the following section).

2 Main Architecture

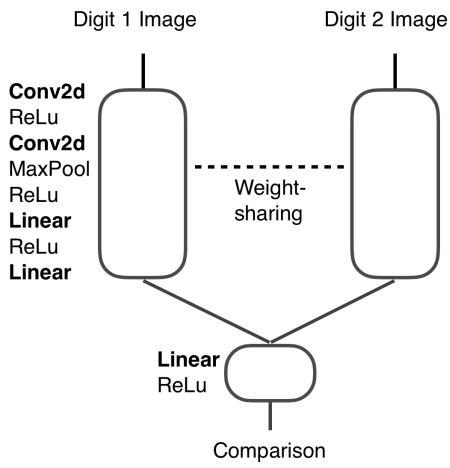

Figure 1: Architecture

The main architecture is summarized on figure 1. Each 14x14 image is fed to the same convolutional network outputting a 10 vector. The latter are then combined and processed through a fully-connected net (the comparison net).

The digit recognition part was inspired from the net shown in the course [1] designed to treat MNIST 28x28 images, but with kernel sizes adapted to the downscaled images. We adapted it by changing mainly the first layers, because after several stages, the same information should be carried in the net as in the 28x28 case. As an example, the kernel size of the first convolutional filter was changed from 5 to 3 so it scans roughly the same area on the digits. The maxpool after the first layer was removed, as downscaling the images is a similar operation to pooling.

Concerning the value comparison net, we investigated several methods, including a 'Cold' one which was to take as input a doublet of single values, the values of the recognized digits, and then a very small linear layer to compare them, but this was not an efficious implementation.

The complete description of the net are included below:

```
Digit recognition part (two instances in parallel) :
    Convolution1 : from 1 channel to 32 channels,
    kernel_size=(3, 3), stride=(1, 1)
    ReLu activation function
    Convolution2 : from 32 channels to 64 channels,
    kernel_size=(5, 5), stride=(1, 1)
    max_pool2d   : kernel_size=4, stride=4, dilation = 1
    ReLu activation function
    Linear1      : from 64 channels to 1 channel, Linear(256,200)
    ReLu activation function
    Linear2      : from 1 channel to 1 channel, Linear(200, 10)
Comparison part :
    Linear3      : 1 channel Linear(20,2)
    ReLu activation function
Total number of parameters : 210,030
```

2.1 Auxiliary Loss

Auxiliary loss is incorporated in the network in the form of cross-entropy loss between the 10-digit output of the digit-recognition part and the correct digit, and added to the normal loss (cross-entropy loss between the final output and the correct class).

The form of the total loss :

$$L = c_{\text{aux}} L_{\text{digit 1}} + c_{\text{aux}} L_{\text{digit 2}} + (1 - 2c_{\text{aux}} L_{\text{comparison}}),\ c_{\text{aux}} \in [0, 0.5]$$

This equation contains a weighting coefficient (c_{aux}), balancing the importance of the error on the digit's identification with that of the final comparison result. Clearly, different (c_{aux}) affect the gradient's back propagation, and so will influence the optimal learning rate.

2.2 Variants on Main Architecture

We explore following variants on the main architecture:

- No weight-sharing between the digit-recognition parts
- Separate training: We first train the digit-recognition part, and then the Fully-Connected part on the output of the pre-trained digit-recognition part.
- Noise Removal: The output of the digit recognition architecture is converted to a one-hot-vector and fed to the comparison architecture.

As a baseline for comparison, we use a standard fully connected net with the same number of parameters.

3 Training & Results

The training dataset was split into train and validation in order to avoid hyper-parameter tuning on test dataset. Each net was trained during 50 epochs with Stochastic Gradient Descent with batch size of 100, with optimal hyper-parameters (learning rate, and auxiliary loss for the main-architecture), obtained through grid search.

3.1 Effect of auxiliary loss coefficient on performance of the main architecture

The grid search was first performed on the main net architecture, for 5 values of learning rates logarithmically spaced between 0.0001 and 0.1. The minimal mean error rate achieved on all the learning rates tested is then displayed (on figure 2), along with its empirical uncertainties, and this for each auxiliary loss coefficients tested. The latter were linearly chosen between 0 and 0.5.

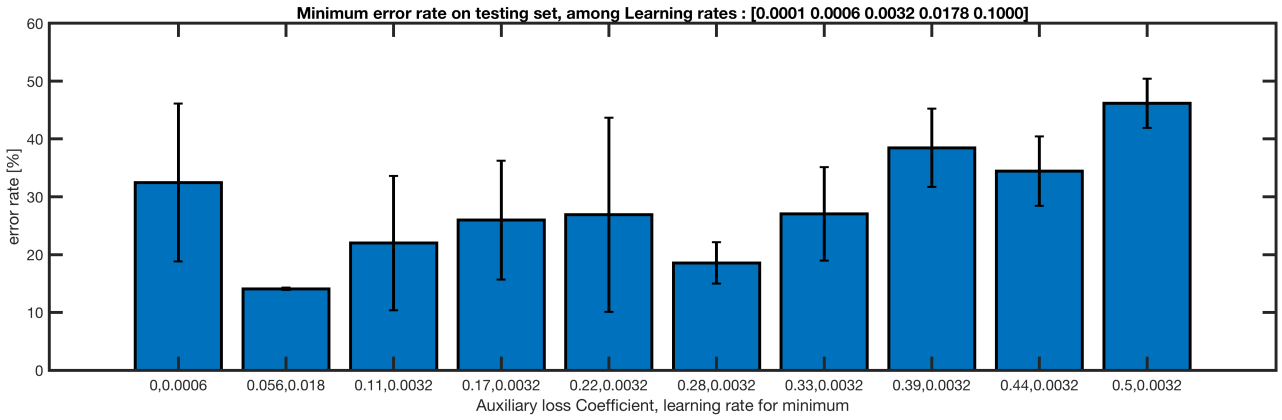

Figure 2: Finding the best hyper-parameters : learning-rate and auxiliary loss coefficient

Let's start by analyzing the extreme values of the auxiliary loss. It appears that without auxiliary loss, the net's best performance is rather bad and variable. This is explainable by the fact that without this loss, the net doesn't necessarily find a way to classify the training digits by itself. This is confirmed on figure 3, where the corresponding digit recognition is shown to be bad. In the case of an extreme auxiliary loss, the performance on comparison is the worse, and confirms intuition.

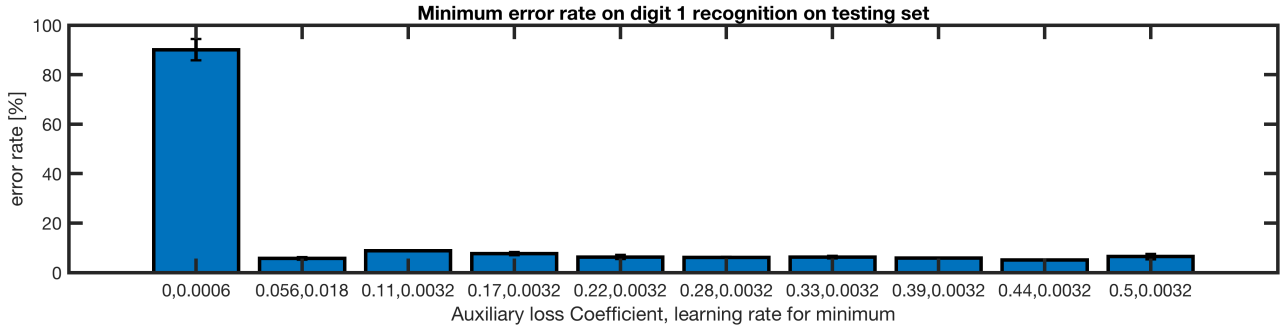

Figure 3: Error being made on digit 1, corresponding to the results of figure 2.

When the coefficient is in the middle, a recurrent minimum (over the 10 runs) seems to appear on the second bar, with very little auxiliary loss, but this trend doesn't expand to the other bars, and was achieved with a rather big learning rate. The other minimum can be identified as the intermediate 0.28 coefficient. This one is more consistent in performance than its neighbours, and can reach down to 15% comparison error rate, which will be our overall best performance.

It is to note that the higher coefficients tent to perform better with the specific learning rate of 0.0032, whereas the lower ones seem to prefer bigger ones.

We ultimately see that he digit recognition gets very accurate as soon as the auxiliary loss is introduced, regardless of its intensity.

3.2 Variants on the Main architecture

Performance results are shown below for 10 rounds of cross-validation, along with their empirical standard deviation.

Error Rate (%)	Train set: digit 1	Train set: digit 2	Train set: comparison	Test set: digit 1	Test set: digit 2	Test set: comparison
Main architecture	3.14 ± 1.56	3.80 ± 1.60	25.58 ± 9.41	7.72 ± 1.28	7.60 ± 1.87	30.45 ± 8.51
Fully Connected Net	-	-	44.62 ± 1.94	-	-	45.32 ± 1.45
Main architecture without Shared Weights	1.91 ± 0.92	1.75 ± 1.08	24.97 ± 6.77	10.12 ± 1.01	9.78 ± 1.14	29.94 ± 5.33
Main architecture with Noise Removal	2.05 ± 0.91	2.25 ± 1.37	40.04 ± 6.95	7.40 ± 1.24	6.73 ± 0.80	40.29 ± 6.73
Concatenation of already trained parts	2.22 ± 0.50	-	24.24 ± 4.00	10.06 ± 1.21	-	26.25 ± 5.26
(the above) with Noise Removal	1.62 ± 0.48	-	7.95 ± 1.39	10.33 ± 1.20	-	11.66 ± 2.24

Error is greater for comparison than for digit recognition in the main architecture and its variants (except in the separate training schemes). This is a clear manifestation of the lagging of the comparison network behind the digit recognition part, which is partly fixed by the separate training as we shall see hereafter. Fully-Connected net is performing poorly as expected. In fact architecture does not stem from any relevant considerations of the data at hand, (for example translation invariance in digit classification should indicate a fully convolutional structure in the first layers). Removing weight sharing does not significantly alter performance. However, note that the number of parameters to store is lower with weight sharing so might speed up calculations. Eliminating the noise in the main architecture seems to affect badly (but not in a very significant way) the performance. In fact, because the digit error remains quite high, it is plausible that the score on the less probable digits still contain relevant information for the comparison. The best performance is achieved when the digit-recognition part and the comparison part are trained subsequently. In fact, when the network is trained all at once, the comparison net is trained to adapt to the mistakes of the digit-recognition part, slowing down the learning process. Note that removing the noise (i.e feeding only the most probable class) to the comparison part yielded significantly better results in contrast to the all integrated architecture. In fact, in this scheme, the digit recognition is nearly perfect so the score on the digits other than the most probable one are irrelevant to the comparison.

Our overall experience shows a lot of variability of performance in the first epochs of the training. This could be probably avoided by implementing a more sophisticated gradient descent, and a dynamic auxiliary loss. Training the net during 100 epochs improved significantly the results but show the same trend among alternative architectures.

4 Conclusion

The different architectures tested here all seem to perform at a best of 15% of errors for this task, which is not a very satisfying result. It has been shown that assembling together pre-trained nets for their specific tasks, and using noise removal, did outperform our main Architecture, even when well trained, and this by one third of accuracy.

This outlines the gain of complexity in training a deep model, in which several very different tasks have to be performed. This unnatural architecture layout should therefore be avoided in future architecture plannings, and replaced by separately trained parts when possible.

References

[1] François Fleuret. *Current applications and success 1-2 Slide 23.* 2019.