Project 1 – Classification, Weight Sharing and Auxiliary Losses

Niels Poulsen
SCIPER - 270494

Jean Chambras
SCIPER - 271630

1 Introduction

This project aims to develop neural network architectures capable of determining, when given as input two grayscale images taken from the MNIST dataset, which one represents the smaller digit. We experiment in particular the impact of weight sharing, and of the use of an auxiliary loss to help the training of the main objective.

2 Models

As our problem is a modification of the MNIST classification task, we decided to base our models on an architecture proven to perform well on it. We adopted a LeNet-5 model architecture, tweaking it to suit the demands of our particular problem. It has the advantage of being quite small, and can obtain near-perfect test results on the original MNIST task.

Having such a model as the core processing component sets a solid baseline for processing the input images. Our problem is not so different from the original classification task: our models should ideally learn to recognize the digit in each input image, and then compare its prediction for both images to determine which number is larger (which is very easy, as it can be done by simply comparing which digit is most likely in each image). To perform this final classification, we use a mutlilayer perceptron (MLP) with a single hidden layer.

2.1 Building Blocks

2.1.1 Customized LeNet-5 (CLeNet)

Our customized LeNet takes either one or two channels as input, so that it can be trained both with and without weight sharing. It can output an arbitrary number of values, which allows it to be trained to directly output logits representing the comparison of images, or logits representing the 10 classes of MNIST.

As our images are half the size of the normal images (14 by 14 instead of 28 by 28), we decrease the kernel size to 3 (so that it contains about the same information as in the original LeNet). We doubled the number of channels in the convolutional layers to give the model a bit more power, as well as to compensate for the fact that our kernels are smaller. We introduced padding and change the max-pooling stride to 1 for all layers, as our images are very small to start with. We use ReLU as an activation function.

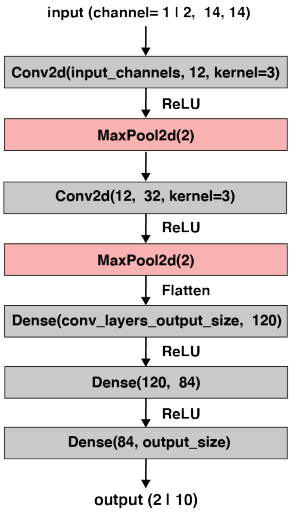

Figure 1: Customized LeNet (CLeNet)

2.1.2 MLP Classifier

We create an MLP with a single hidden layer (with a variable number of hidden units) to classify the output of our CLeNet. It also uses the ReLU activation function.

2.2 Baseline Models (B1 and B2)

Our first baseline model (B1, Fig 2, left) takes as input a 2x14x14 tensor, viewing the two images as simply being different channels, ignoring the structure of the data. It is passed through our modified LeNet, which outputs two logits.

As the weight sharing models benefit from an additional 1-hidden layer MLP to classify the output of the CLeNet, we create a second baseline (B2, Fig 2, right) with the same classifier.

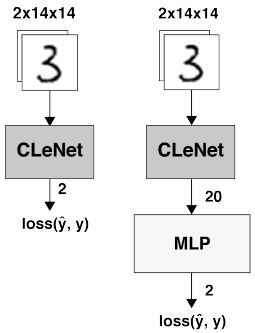

Figure 2: Baseline Models B1 (left) & B2 (right)

2.3 Weight Sharing Model (WS)

The second network uses weight sharing to process both input images the same way. The input vector x (2x14x14) is split into two vectors x_1 and x_2 (both 1x14x14). Each image is passed through the CLeNet, which outputs 10 logits. These outputs are then concatenated and passed through the MLP, which outputs two logits.

This model should perform better than the baselines, as both images (which represent the same information and come from the same distribution) are processed in the same way.

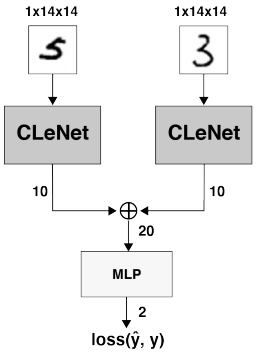

Figure 3: Weight Sharing Model (WS)

2.4 Weight Sharing and Auxiliary loss (WSAL)

Our last network introduces an auxiliary loss during training. It has exactly the same architecture as WS, but keeps the outputs of the CLeNet ($\hat{c_1}, \hat{c_2} \in \{0, 1, ..., 9\}$) to use as an auxiliary loss. Our final loss is a weighted sum of the loss of the CLeNet and the loss of the MLP (main loss). With c_1, c_2 the true classes of the two images:

$$loss = w(\ell_1(\hat{c_1}, c_1) + \ell_1(\hat{c_2}, c_2)) + \ell_2(\hat{y}, y)$$

Training with an auxiliary loss should bring one major advantage: instead of having to rely on the final loss (only being given the information indicating whether the first digit was greater than the second one), the CLeNet is explicitly trained to recognize the digits in the image.

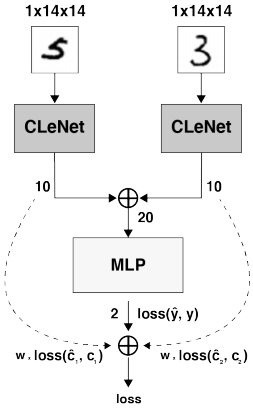

Figure 4: Weight Sharing and Auxiliary Loss Model (WSAL)

2.5 Loss Functions

We use the Cross Entropy loss function for all of our models. It is used to compute the loss for the final classification (digit comparison) but also for the auxiliary losses (digit prediction for each image).

3 Results

3.1 Implementation

We train our model using the Adam optimizer, which performed better in our tests than SGD with momentum and weight decay. Our hyperparameters (mini-batch size, learning rate, weight decay, auxiliary loss weights, hidden layer units in the classifier) are selected using cross-validation on 10 different generated training samples containing 1000 images. We try mini-batch sizes of 1, 10, 25 and 50, learning rates of 1e-5, 1e-4, 1e-3 and 1e-2, weight decays of 0, 0.001, 0.01 and 0.1, hidden layers with 10, 50 and 100 units and auxiliary loss weights of 0.5, 1.0, 2.0, 5.0 and 10.0.

We fix the Pytorch random seed for reproducibility before starting each evaluation run, as well as before performing hyperparameter selection.

Model		Hyperparameters					Error Mean [%]		Error STD [%]	
Name	Parameters	BS	LR	WD	HL	ALW	Train	Test	Train	Test
B1	48 730	25	1e-4	0.1	–	–	3.19	21.35	1.54	1.41
B2	51 412	25	1e-3	0.1	50	–	1.20	19.82	1.62	2.23
WS	50 454	25	1e-4	0.1	50	–	3.19	15.26	1.26	1.06
WSAL	51 604	10	1e-3	0.0	100	10.0	0.00	6.06	0.00	1.28

Table 1: Results over 10 rounds of training. Abbreviations: BS = Batch size, LR = Learning rate, WD = Weight decay, HL = Hidden layer units, ALW = Auxiliary loss weight.

3.2 Results

The models were trained 10 times for 30 epochs. We compute the mean and standard deviation of the error rates for each model over the 10 rounds. The performances of our models are displayed in Table 1. In figure 5, we show box plots of the of the error rate on the test set for each of our models. The plots of the learning rates per epoch of each model are presented in the appendix section below.

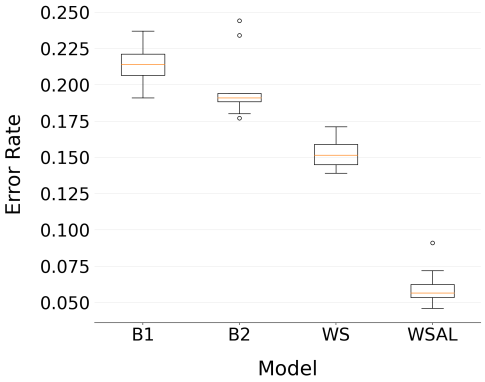

Figure 5: Box plot showing the distribution of the test error rates of each model.

4 Evaluation and Discussions

The dataset didn't seem like the challenge in itself for this project. The test error on full MNIST can become arbitrarily low. Our main challenge was to train the network in a way that forced it to learn the "correct information". Of course, we won't reach scores as high as on MNIST, as our task is a bit more complicated, we only have 1000 training samples and the images we receive are compressed.

The difficulty to make the model learn the correct information can be seen in the performance of the baselines. As they don't use any information contained in the structure of the data, they quite violently overfit the training set. Without weight decay, we reached 0% error on the training set with all of our models. This shows us that the simple CLeNet chosen is powerful enough to fit the data, but when trained naively simply doesn't understand the underlying distribution.

When introducing weight sharing, we can see that the models starts performing better. Adding the auxiliary loss is what really boosts our results. While training, we noted that the performances improved when increasing the weight attached to the auxiliary loss. Our best performances came with an auxiliary weight of 10, which shows that it was in fact much more important to train the model to recognize the digits in the image than it was to minimize the final loss (as the latter arrived naturally with the former).

This is quite easy to understand intuitively: if the CLeNet outputs a good probability distribution for each number given as input, the MLP simply needs to compare these distributions to output a decision. Moreover, when using weight sharing, each input we have is used to train the CLeNet with 2 separate images, effectively increasing the size of the training set for it to 2000 images.

We believe that we could have achieved similar performances in another manner: instead of using an auxiliary loss, we could have simply pre-trained CLeNet to recognize the digits, frozen some (or all) CLeNet layers, and simply trained the classifier to choose which digit is likely to be larger.

5 Conclusion

We achieve a good error rate on the test set, with relatively small models (around 50'000 trainable parameters for all models). We were able to see that models should be trained in a smart way, instead of trying to brute-force a solution by deploying models with as many parameters as possible.

Appendix

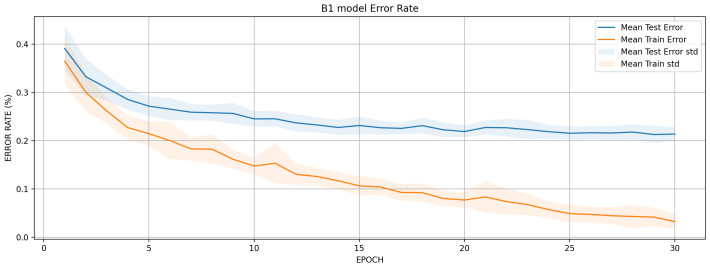

Figure 6: Error rates per epoch for the baseline model 1 (B1)

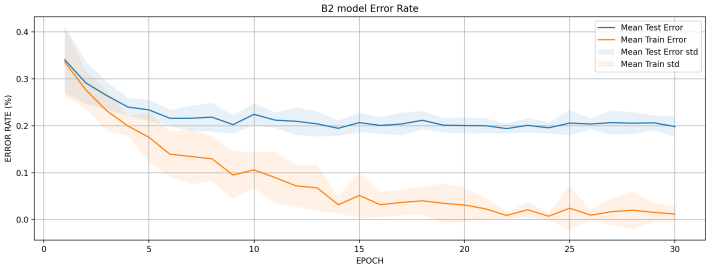

Figure 7: Error rates per epoch for the baseline model 2 (B2)

Figure 8: Error rates per epoch for the weight sharing model (WS)

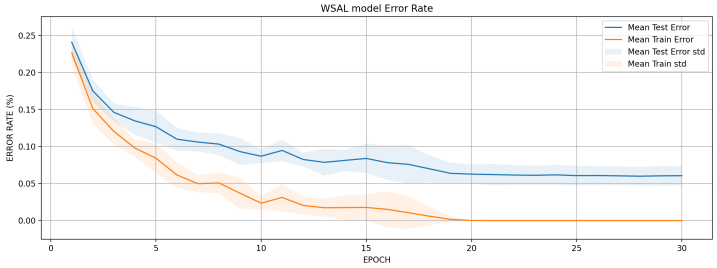

Figure 9: Error rates per epoch for the weight sharing and auxiliary loss model (WSAL)