Deep Learning - Mini Project 1

Sylvain Lugeon, Tyler Benkley, Frédéric Berdoz
Deep Learning, EPFL, 2020

***Abstract*—In this project, we explore the effects of weight sharing and auxiliary losses over different neural network architectures. Each model is tasked with predicting the greater of two handwritten digits and is trained over downgraded images from the MNIST database. For training, we have access to the groundtruth of the task as well as the true classes of the two digits. For comparison, we evaluate the accuracy of each model. We witness beneficial effects with both weight sharing and auxiliary losses of varying magnitude depending on the architecture.**

I. Introduction

The goal of this project is to assess the impact of network architecture, weight sharing techniques and the use of auxiliary losses in a classification task where we predict the greater of two handwritten digits. To do so, we compare the performance of 4 architectures subject to different conditions.

To gauge the performance of a model, we train it 15 different times (rounds) over 200 epochs on 1000 samples. We average the model's prediction accuracy of a test set of 1000 samples over the 15 rounds to obtain an empirical mean and standard deviation of accuracy over the number of epochs. The train and test sets are randomly re-sampled between each round.

To understand the impacts of weight sharing, architecture and auxiliary loss, we assess the performance of each architecture under the various conditions of weight sharing, and auxiliary losses and observe the performance differences.

II. Data

The training data consists of $N = 1000$ samples. Each sample is comprised of two 14×14 images, each one being a different handwritten digit randomly selected from the MNIST handwritten database [1] but of lesser quality (14x14 instead of 28x28).

For every data point, there are two targets. The main target, referred to as 'binary target', is a binary value and is 0 if the second digit is strictly lower than the first and 1 otherwise. In expectation, 55% of the samples will have a positive target (two identical digits yield a positive target), which means that the data is relatively balanced. Thus, for the sake of clarity, model performance will be gauged using the percentage of correct predictions, i.e. the accuracy. Moreover, the secondary target, called 'digit target', contains the true classes of both digits of a given sample. The test set also consists of 1000 samples with their respective targets in the same format.

III. Architecture

Four different models were studied. We intentionally designed the first 3 models with roughly the same number of parameters ($\approx 80k$) such that they be comparable. Additionally, all the models have at some point a hidden layer with 20 nodes so that digit prediction can be enforced using auxiliary losses (see Fig 1). Apart from the output layer, only the rectified linear unit (ReLU) has been used as an activation function. For the main (and auxiliary) losses, cross entropy is preferred as it is known to perform well in classification tasks.

The first model, 'FCNet', is comprised only of linear (fully connected) layers. We intentionally designed this model with little forethought, and its performance serves as a benchmark for what a classic multilayer perceptron can achieve.

The second, 'ConvNet', has two convolutional layers and a max pooling layer in sequence that are followed by three linear layers. Its structure is partly inspired by LeNet 5 [1], a net that performs relatively well in digit recognition.

The third, 'ConvSepNet', possesses roughly the same convolutional layers as ConvNet. However, both channels are separated and then convolved by the same filters (see Fig. 1), before being recombined using two fully connected layers. As weights are shared between the two channels, a second level of weight-sharing is introduced (in addition to the convolution).

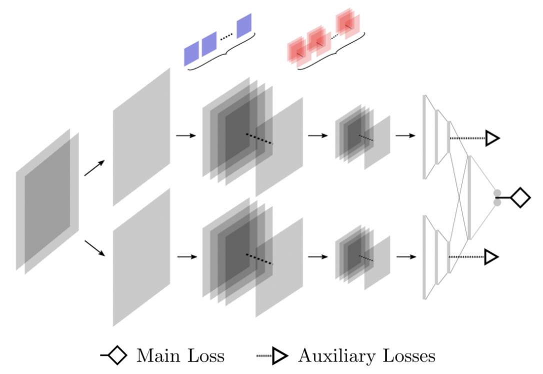

Figure 1. ConvSepNet architecture.

The final model, dubbed 'FinalNet', is larger than the others. It possesses approximately $360k$ parameters and is meant to pursue the limits of accuracy. It is identical to ConvSepNet with the exception of an additional linear layer.

The table below contains sums up the nets' global characteristics.

Table I
MODEL CHARACTERISTICS

Model	Parameters	Layers Type	Weight Sharing
FCNet	82,662	Linear	No
ConvNet	85,878	Conv+MaxPool+Linear	Yes
ConvSepNet	84,950	Conv+MaxPool+Linear	Yes
FinalNet	365,350	Conv+MaxPool+Linear	Yes

More complete information on each model can be found in the appendix.

IV. WEIGHT SHARING

Two levels of weight sharing are explored in this report. The first one is the use of convolution kernels, particularly in the ConvNet model, which constitute the most elementary weight-sharing technique. Additionally, since each channel consists of the same data type, i.e. a handwritten digit, it is intuitive to treat the channels separately in a first phase and share the weights across them. The effects of this are investigated in ConvSepNet.

V. AUXILIARY LOSSES

Typically, one would optimize a model with respect to only one loss. However, our task is particular because we have access not only to the binary targets but also to the digit targets. A natural idea is for the network to have two auxiliary losses in addition to the main loss, where each one quantifies the error in predicting its digit. Hence, we can compel one part of the network to learn to recognize digits before comparing them. This would mimic the way a human performs the task.

Moreover, the weighting of the different losses is left to choice, depending on which part of the network we would like to emphasize the training. We can even alter the loss weighting dynamically during training. In the scope of this report, however, the weighting of the losses will be constant during training and will obey the following relation (for a given $f \in [0, 1]$):

$$Loss = (1 - f)Loss_{main} + \frac{f}{2}(Loss_{aux1} + Loss_{aux2}). \quad (1)$$

VI. RESULTS

The following data emanates from 15 rounds of training over 1000 sample points during 200 epochs, using an Adam optimizer with batch size 25 and with learning rate 10^{-3} in the first 100 epochs and 10^{-4} in the last 100. The mean values (μ) and standard deviations (σ) are all computed over these rounds of training.

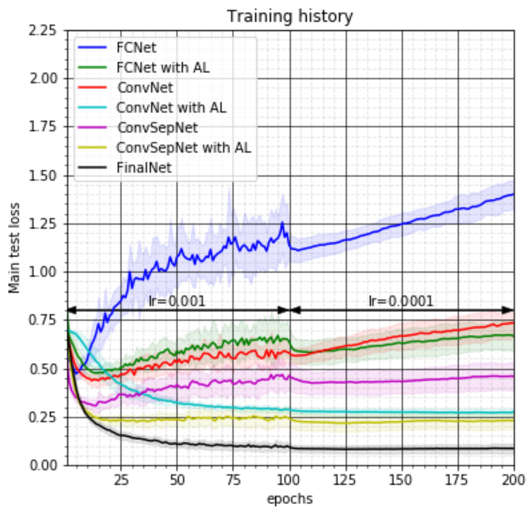

Figure 2. Evolution of the cross entropy loss of the main binary prediction. The shaded area represents the empirical standard deviation on both sides of the mean.

We start off with a graph that displays the main test loss with respect to the number of epochs of training. The test loss curves help infer whether there is overfitting.

Next, follows a table filled with numerical results of each model trained using the training procedure mentioned previously. Note that μ and σ denote the empirical mean and standard deviation, respectively. The performance measurements shown here are the test accuracy and the digit prediction test accuracy when auxiliary losses (AL) are used to enforce digit recognition. As for the train accuracy, it reached 100% for every model due to the simplicity of the task. In this table, when ALs are used, the total loss is computed using equation (1) with $f = 0.5$.

Table II
MODEL PERFORMANCE OVER 15 TRAINING ROUNDS

Model	Test Accuracy		Test Digit Accuracy	
	μ [%]	σ [%]	μ [%]	σ [%]
FCNet	81.90	0.89	-	-
FCNet + AL	85.31	1.30	81.74	0.79
ConvNet	82.57	1.60	-	-
ConvNet + AL	89.44	0.68	89.75	0.97
ConvSepNet	90.99	1.3	-	-
ConvSepNet + AL	94.65	0.96	97.68	0.35
FinalNet	98.08	0.43	97.90	0.35

Finally, Figure 3 illustrates a plot of the test accuracy improvement with respect to the weight of auxiliary losses in the total loss, this time averaged over 10 rounds per coefficient $f = \frac{i}{20}$, $i = 0, 1, ..., 20$. The raw data corresponds to the points, and the dashed curves correspond to a spline fit of the data.

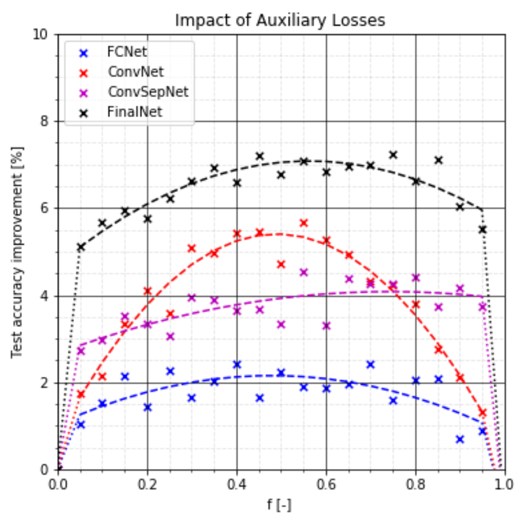

Figure 3. Effect of auxiliary losses on the final test accuracy.

Furthermore, we also tested dynamic weighting of auxiliary losses but none of the results showed significant improvement compared to constant weighting.

VII. Discussion

Now, we interpret the impact of the various techniques which are implemented.

Weight Sharing

As mentioned previously, weight sharing occurs at two levels: the use of convolutional layers, and the separation of channels.

In the case of convolutional layers, moving filters are used across channels, making use of the same weights. This helps in detecting patterns in the data with consistency. The most basic example is edge detection and such a task is easily seen to be useful in digit recognition. From Fig. 3, one can see that this first type of weight sharing reduces overfitting but Table II does not boast a significant increase in accuracy. This is because our model's overall capacity is not readily increased.

Simple convolutional layers combine the different filter outputs of the various channels. For an RGB image, this is useful as it is the weighted combination of all colors that helps distinguish patterns. In our case, the two channels are two different images. It makes more sense to filter them independently with the same weights. When the channels are separated and their weights are shared we divide the number of parameters by two. We can now increase the number of layers to maintain the same number of parameters and our model's capacity drastically benefits from this. This explains why we see a significant increase in test accuracy going from ConvNet to ConvSepNet (see Table II).

Auxiliary Loss

As shown in Table II, and as expected, the use of auxiliary losses has an overall positive impact on the main performance of each model. This stems from the fact that we are giving more information about what the model must learn in order to make accurate predictions, i.e. the digits of each channel.

However, the magnitude of this improvement varies from one model to another. As illustrated in Fig. 3, the ConvNet model is particularly sensitive to the auxiliary loss weighting compared to other models. In fact, with a fine-tuning of the parameter f, one can improve its test performance by almost 6%. The intuition behind this variation of sensitivity is the following: ConvNet has an architecture that is efficient to predict digits (because it is inspired by LeNet 5 [1]), but it has no way of learning these digits without auxiliary losses. On the other hand, FCNet's architecture is ineffective for digit recognition, which mitigates the impact of auxiliary losses.

Finally, as shown with FinalNet, one sees that augmenting the complexity of the model with one fully connected layer greatly improves the performance and diminishes the variance of the predictions.

VIII. Summary

All in all, if a data set allows for natural weight sharing across channels, the performance can be readily improved. Moreover, access to additional data relevant to the main task can efficiently be used in auxiliary losses. An intelligent architecture and good understanding of the data is what enables and renders these techniques interesting.

It is important to mention that the complexity of a model, i.e. its number of parameters, is always a limiting factor in its performance.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

IX. Appendix

Table III
FCNet Architecture

FCNet	
Layers	**Parameters**
Input	$N \times 392$
Fully Connected Relu Dropout	$392 \rightarrow 200$ nodes $p = 0.25$
Fully Connected Relu	$200 \rightarrow 20$ nodes
Fully Connected	$20 \rightarrow 2$ nodes

Table IV
ConvNet Architecture

ConvNet	
Layers	**Parameters**
Input	$N \times 2 \times 14 \times 14$
Conv2D Relu	$2 \rightarrow 12$ channels, $k = (3, 3)$
Conv2D Relu	$12 \rightarrow 32$ channels $k = (3, 3)$
MaxPool Dropout	$k = (2, 2)$, $s = 2$ $p = 0.5$
Fully Connected Relu Dropout	$800 \rightarrow 100$ nodes $p = 0.5$
Fully Connected Relu	$100 \rightarrow 20$ nodes
Fully Connected	$20 \rightarrow 2$ nodes

Table V
ConvSepNet Architecture

DigitNet	
Layers	**Parameters**
Input	$N \times 1 \times 14 \times 14$
Conv2D Relu	$1 \rightarrow 12$ channels, $k = (3, 3)$
Conv2D Relu	$12 \rightarrow 32$ channels $k = (3, 3)$
MaxPool Dropout	$k = (2, 2)$, $s = 2$ $p = 0.5$
Fully Connected Relu Dropout	$800 \rightarrow 100$ nodes $p = 0.5$
Fully Connected	$100 \rightarrow 10$ nodes

ConvSepNet	
Layers	**Parameters**
Input	$N \times 2 \times 14 \times 14$
First Digit 1 Second Digit Concatenate	$x_1 \leftarrow$ DigitNet(Input[:,0,:,:]) $x_2 \leftarrow$ DigitNet(Input[:,1,:,:]) $[x_1, x_2]$
Fully Connected Relu	$20 \rightarrow 10$ nodes
Fully Connected	$10 \rightarrow 2$ nodes

Table VI
FinalNet Architecture

FinalDigitNet	
Layers	**Parameters**
Input	$N \times 1 \times 14 \times 14$
Conv2D Relu	$1 \rightarrow 12$ channels, $k = (3, 3)$
Conv2D Relu	$12 \rightarrow 32$ channels $k = (3, 3)$
MaxPool Dropout	$k = (2, 2)$, $s = 2$ $p = 0.5$
Fully Connected Relu Dropout	$800 \rightarrow 400$ nodes $p = 0.5$
Fully Connected Relu Dropout	$400 \rightarrow 100$ nodes $p = 0.5$
Fully Connected	$100 \rightarrow 10$ nodes

FinalNet	
Layers	**Parameters**
Input	$N \times 2 \times 14 \times 14$
First Digit 1 Second Digit Concatenate	$x_1 \leftarrow$ FinalDigitNet(Input[:,0,:,:]) $x_2 \leftarrow$ FinalDigitNet(Input[:,1,:,:]) $[x_1, x_2]$
Fully Connected Relu	$20 \rightarrow 10$ nodes
Fully Connected	$10 \rightarrow 2$ nodes

Notation:

- k: kernel size
- s: stride
- p: dropout rate