EE-559 Deep Learning course : Project 1 report

Classification, weight sharing, auxiliary losses

Samuel Dubuis (259157)
IC - EPFL
s.dubuis@epfl.ch

Yann Gabbud (260036)
IC - EPFL
yann.gabbud@epfl.ch

Vincent Gürtler (263880)
IC - EPFL
vincent.gurtler@epfl.ch

I. Introduction

The goal of this project is to create a convolutional network capable of telling if the number on an image is smaller than the number on another image and see the impact of using weight sharing and auxiliary loss. To do this we are given a dataset of a thousand pairs of images.

As a first step, we describe the convolutional network that we use. As a second step, we show how auxiliary loss and weight sharing impact the accuracy of the result. In order to further improve the precision, we finally add dropout.

II. Models

A. Architecture

In this section, we detail the architecture of the network. The input of the model is a pair of grayscale images of dimension 14x14. These two images go into a layer of two convolutions followed by maxpoolings with strides of 2. If weight sharing is used, then both images go into the same layer. Otherwise, the images go through two distinct pipelines. These layers output feature vectors of size 128. Then, they go through two fully connected layers for the classification. We are left with two vectors of size 10 that we call class prediction vectors. As for the convolutions, if the weight sharing is activated, both vectors go through the same fully connected layers otherwise they go into distinct ones. Meanwhile, the output of the two convolutions are concatenated and also go through two fully connected layers that outputs a feature vector of size 32. This output is now concatenated with the two class prediction vectors to gather all the features together and the whole goes through two more fully connected layers that outputs a vector of size two for the binary target. Finally the network outputs the binary prediction target vector and the two class prediction vectors. The figure 1 illustrates the complete network.

B. Hyperparameters and training

Here, we explain our choice of hyperparameters and how we train our models. We chose the Adam optimizer with a learning rate of 0.001 and the cross entropy loss. We train each model ten times. Each training goes through 80 epochs. To improve the accuracy, we use a scheduler that adapts the learning rate during training. When the validation loss doesn't improve for seven consecutive epochs, the learning rate is reduced by a factor 0.2. This helps to converge towards the minima since we avoid oscillating around it. We also define a number of epochs whereupon the training stops if no progress is made. Early stopping is used to interrupt the training before the learner begins to overfit. We set the threshold at ten consecutive epochs without improvement. Finally, we train our model with several different auxiliary loss weights in order to find the one that gives the best accuracy.

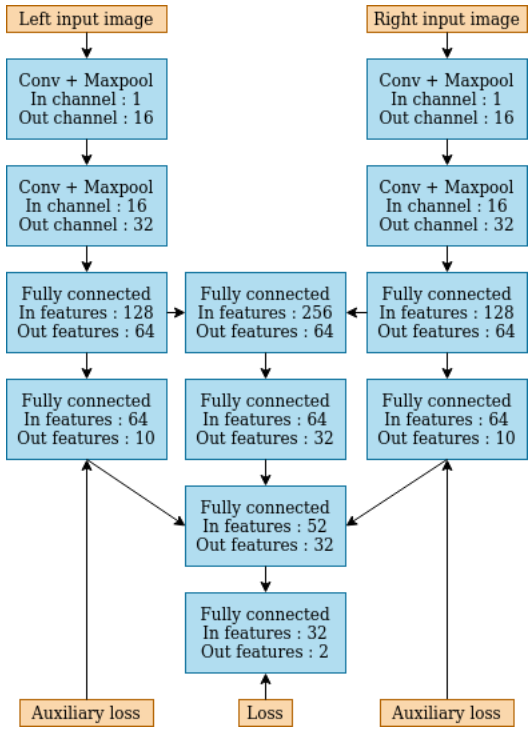

Fig. 1: Architecture of the neural network

III. Results

In this section, we present the result of the different models and their effectiveness. Basically, we searched for the best weight of the auxiliary loss, with and without weight sharing, and dropout. The auxilary loss weight is used to give more or less importance to the different losses. If the weight is zero, then we have no auxiliary loss and the class labels are not used. On the contrary, if the weight is a lot bigger than one, it will start to shadow the loss given by the binary label. We test for the following set of weights: $[0, 0.1, 0.5, 1, 3, 10, 100, 1000]$. In figure 2, we can see how the accuracy evolves for different weights of the auxiliary loss after 80 epochs averaged over

10 rounds. In figure 3, we see the same evolution of the accuracy but this time with weight sharing activated. We can draw several conclusions. First, we notice that even applied separately, both weight sharing and auxiliary loss improve the accuracy. Weight sharing improves the accuracy by about 3.2% and auxiliary loss by about 4.4%. As we can expect, auxiliary loss provides good results because training the model to recognise the digits is directly related to its main objective. Indeed, if the model performs at 100% accuracy on the class labels, then it will also perform at 100% on the binary labels. Weight sharing also helps because features detection is the same for the left and right images. Furthermore, it reduces the number of parameters to train for the features extraction. Therefore, it does not make a lot of sense to use different pipelines. However, the best results are obtained when both weight sharing and auxiliary loss are used together. With a weight of 10 for the auxiliary loss and weight sharing activated, we obtain an improvement of 5.3% of the accuracy compared to the case when both were deactivated. It means that we give more importance to the class label loss than the global binary loss. This make us reach the goal of at least 85% accuracy.

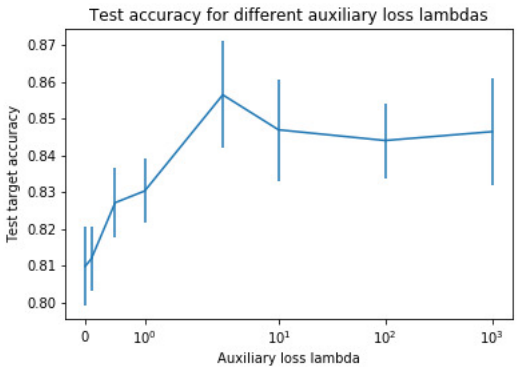

Fig. 2: Evolution of the accuracy with different auxiliary loss weights without weight sharing

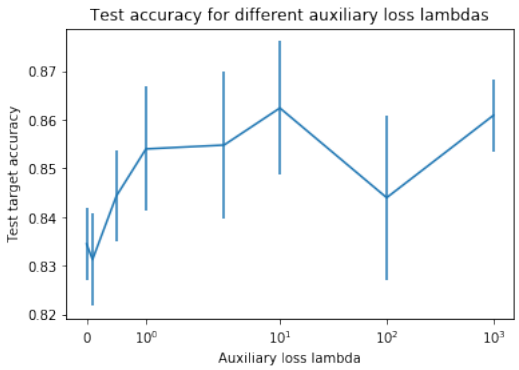

Fig. 3: Evolution of the accuracy with different auxiliary loss weights with weight sharing

In a second time, we decide to use dropout to improve the accuracy. Dropout, which is a regularization technique that performs model averaging, has the advantage to often improve the accuracy without having to modify the architecture of the network. The figures 4 and 5 show the best results obtained respectively without and with dropout activated. As we can see, we manage to gain an extra 5% accuracy with dropout.

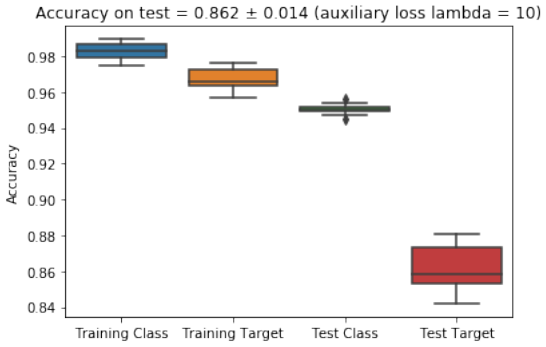

Fig. 4: Best accuracy without dropout

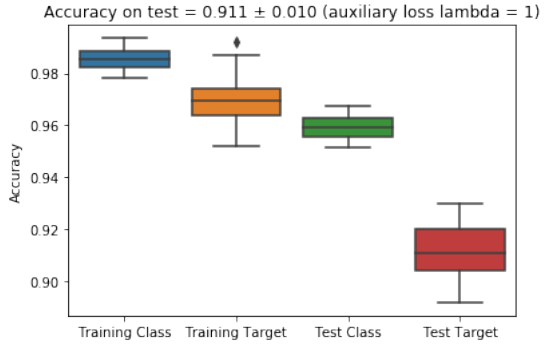

Fig. 5: Best accuracy with dropout

Although we reach the goal of 85% accuracy, we want to point out that with a different architecture, we can obtain much better results. Indeed, with a model that directly learns to classify the digits, we managed to get 95.5% accuracy on the class labels. Once the model is trained, we predict the digits for all the pairs and perform a simple comparison between them to predict the binary label. With this method, which is a lot simpler and more intuitive, we obtain an accuracy of 97%. That is by far better than the 86% we reached with the model exposed in this project.

Finally, we can find in the following table I a summary of the accuracies obtained with the different configurations we tried.

	Weight sharing off	Weight sharing on
Auxiliary loss off	80.7 ±1.1%	83.9 ±0.6%
Auxiliary loss on	85.1 ±1.8%	86.0 ±1.5%
Auxiliary loss on + dropout	89.2 ±1.0%	91.2 ±1.1%

TABLE I: Summary of the accuracy per configurations

IV. Conclusion

This mini-project highlights how the auxiliary loss can greatly improve the accuracy by simply giving additional information

about the class of the digits. We have seen that weight sharing also helps to improve the accuracy because the same features need to be extracted from both left and right images. We also figured out that we can greatly improve accuracy by using dropout without changing the architecture. Finally, we conclude that we can reach much higher accuracy by using a different model. However, that was not the main purpose of this project.