# Project 1 - Classification, weight sharing, auxiliary losses

Emilio Fernández, Alejandro Bonell, Berk Mandiracioglu
*Deep Learning, EPFL, 2020*

***Abstract*—In the context of Image analysis, this paper explores different deep learning architectures for handwritten digit comparison. More precisely, given a pair of handwritten digits the task of our models is to determine which is the bigger digit. Digit Comparison is very related to the popular ML task of handwritten digit recognition. We explore impact of weight sharing and auxiliary losses on the performances of our network to solve digit comparison task.**

## I. Introduction

Handwritten digit recognition is a sub-area of pattern recognition that has been highly developed by deep learning techniques. This area is widely used in many domains such as automatic processing of bank cheques, and is also very useful to test NN architectures. In this project, the goal was to build a neural network that compared two handwritten digits of low resolution.
The following report is structured as follows: First, in section II , an exploratory data analysis is performed with the aim of better understanding the data. In section III, we describe the architecture of our principal models, and we show how weight sharing and auxiliary losses are implemented in our models and the tuning of hyperparameters in our models. Then we present our results in section IV where we compare the performance of each model. Finally we will discuss the improvements of implementing weight sharing and auxiliary losses in section V.

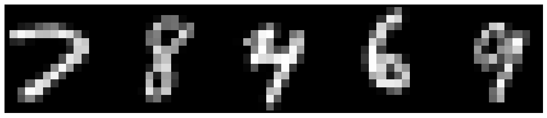

Figure 1: Handwritten digits of Mnist Database 14x14 pixels

## II. Exploratory Analysis

The data that we are studying are handwritten digits. This data is provided by the MNIST database which contains 70.000 examples of handwritten digits [1]. The data that we studied for this project contains 2000 pairs of images, half of this was used for training our model and the other half for testing. For this project we have skipped the data preprocessing part, since images from MNIST have no missing values and ”digits have been size-normalized and centered in a fixed-size image”. An image of our data set is represented in Pytorch [2] by a tensor of size 14x14 where each entry of the tensor represents the numerical value of the pixel (see Fig. 1).

## III. Models and Methods

In this section we present three different models to solve handwritten digit comparison. By comparing these models, we will assess the importance of auxilary losses and weight sharing. The architectures and hyperparameters of each model have been chosen carefully using cross validation. Besides the models that we present below, we have designed others that are presented on Table II.

### A. Basic Convolutional Neural Network

Convolutional Neural Networks are very popular models for image classification. They are robust to noise and are able to extract the important features about each class. The structure of our implementation of the CNN can be seen in detail on Table I.

Table I: CNN architecture

| **Layers** | **Parameters** |
|---|---|
| Input | 2 x 14 x 14 |
| Conv2D, ReLu | $channels = (2, 32)$ $kernel = (3, 3)$ |
| Conv2D, ReLu<br>Maxpool | $channels = (32, 64)$ $kernel = (3, 3)$<br>(2, 2) |
| Conv2D, ReLu<br>MaxPool | $channels = (64, 128)$ $kernel = (3, 3)$<br>(2, 2) |
| Linear , ReLu | Dimensions = ( 128, 100), p = 0.5 |
| Output | Dimensions = (100, 2) |

p: Dropout rate

### B. Convolutional Neural Network with Weight Sharing

In the previous model, the convolutional layers were treating each image as different objects. Nonetheless if we wanted to learn how to recognize digits it does not make sense to use different weights for each pair of images. To avoid this we implemented weight sharing to our model. Our model then has two parts: the first one where it learns digit recognition using weight sharing and the second one where it compares digits( see Figure 2). The weight sharing architecture is very similar to the architecture of Table 1 but for the first convolutional layer the number of input channels is 1 and at the end the output’s size is 10.

### C. Convolutional Neural Network with Weight Sharing and Auxiliary Losses

This last proposal is very similar to the previous one, the only amend is that the network has three outputs instead of

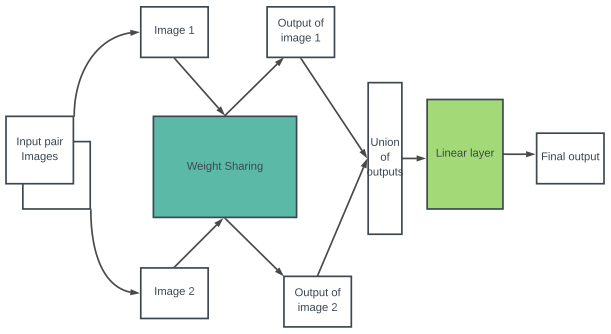

Figure 2: Diagram of a NN with Weight sharing

one. The first two outputs represent the class value of the handwritten digits and the third output represents the comparison value of input digits. We added two auxiliary losses to our training, that calculated the loss from comparison of the additional outputs of network with the real class values of each digit. Adding these auxiliary losses to our final loss allowed our Neural Network to learn more intrinsic features from the classes of digits.

### *D. Fully Connected Network with Auxiliary loss*

This network has 3 output layers. One of them represent the comparison values of the digits, the other output layers represent the class value of the input images.

### *E. Cross Validation For Hyperparameter Tuning*

Cross validation was performed in order to estimate the performance of the models on unseen data and how well they generalize to the underlying probability distribution of handwritten digits.
It was also performed to choose the optimal set of hyperparameters. A 10-fold cross validation was performed on the training set for every different value and combination of hyperparameters in the search space. Hyperparameter combinations with minimum error rate were selected. Table II shows results of our hyperparameter selection. The computations for the aforementioned search space lasted 6 hours on Google Colab GPUs [3].

Table II: Tuned hyperparameters for each model

| | **Parameters** | | |
|---|---|---|---|
| **Models** | $\lambda$ | Batch Size | Hidden Layers |
| Basic CNN | 0.1 | 20 | Not tuned |
| CNN weight sharing | 0.1 | 20 | Not tuned |
| CNN weight sharing & Aux | 0.1 | 20 | Not tuned |
| FC Network | 0.005 | 20 | (150, 50, 20, 10) |
| FC Network with Aux | 0.01 | 20 | (150, 50, 20, 10) |

It is clear that CNN networks performs more optimal with higher $\lambda$ value $(0.1)$ compared to FC network's optimal $\lambda$ which is significantly smaller. Moreover, all the models performed better in smaller batch size $(20)$.

## IV. RESULTS

After the hyperparameter tuning and determining the optimal hyperparameters, we have trained our models on training data until convergence using those hyperparamters. We have trained our models on Virtual Machine provided by the course [4]. That machine only utilises cpu and it took 1.5 hours to obtain these results. The stopping criteria we selected for convergence in SGD algorithm was the absolute value of difference between two consecutive losses of epochs should be less than $0.001$. We have picked 10 seeds for $randomseed$ in the range $(1, 11)$. For each seed value, we calculated the error rates on the validation set consisting of 1000 images. Then, calculated their mean and standard deviation (STD) to get a reliable result. Moreover, for each seed value, we have taken the output of the models which are the probability predictions for class values. We have averaged over these predictions for each seed and used them to plot ROC and Precision-Recall curves for each model.

### *A. Error Rates*

Table III depicts the mean and STD of error rates. It is clear that adding an auxiliary loss significantly decreases the STD of errors and more importantly the mean of errors in both CNN and FC network architectures. For example, FC network with auxiliary loss performs $4\%$ better than basic FC network in terms of mean accuracy. Furthermore, weight sharing had a similar affect and it decreased the mean and STD of errors in networks. For example, CNN with weight sharing performs $6\%$ better than Basic CNN in terms of mean accuracy. On the other hand, adding weight sharing and auxiliary losses increased the average number of epochs until convergence in both FC and CNN models.

Table III: Error rates over the 10 experiments

| **Models** | **Epochs** | **Mean Error** | **STD Error** |
|---|---|---|---|
| Basic CNN | 21.4 | 0.1808 | 0.007786 |
| CNN weight sharing | 23.0 | 0.1286 | 0.005168 |
| CNN weight sharing & Aux | 44.5 | 0.1016 | 0.007763 |
| FC Network | 51.6 | 0.2583 | 0.01721885 |
| FC Network with Aux | 95.5 | 0.2135 | 0.092483 |

### *B. ROC Curve*

The receiver operating characteristic curve (ROC curve) is used in order to compare the performances of each model with respect to different values of thresholds. The corresponding ROC curve of the models is depicted in the following Figure 3.

### *C. Precision-Recall Curve*

Precision-recall curve is used to compare how well the precision and recall are balanced over different values of thresholds and shows the performances of the models. The corresponding Precision-Recall curve of the models is depicted in the following Figure 4.

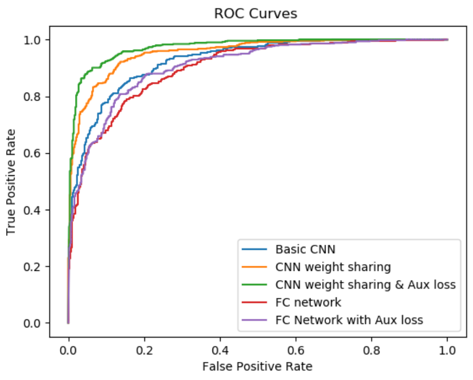

Figure 3: ROC Curves of the models

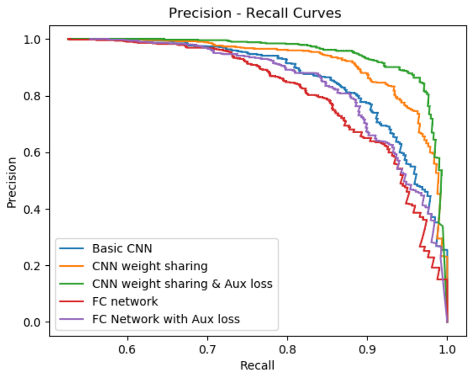

Figure 4: Precision-Recall Curves of the models

The analysis from the ROC and Precision-Recall curves show clearly that CNN with weight sharing and auxiliary loss performs the best over all the models. The effect of weight sharing is significant in the performance of models and this can be observed by comparing CNN with and without the weight sharing on the plots. Moreover, FC network with auxiliary loss performs significantly better than basic FC network and it can almost reach to performance of Basic CNN on the plots. This clearly shows the importance of auxiliary loss to increase the model performance. The FC network without the auxiliary loss performed the worst as expected.

### D. Experimenting with Auxiliary Loss Weight

Until this section, during training, we have calculated the total loss of the networks that utilised auxiliary losses (both CNN and FC) as follows: $totalloss = 0.7 * actualloss + 0.3 * auxiliaryloss$. We decided to change the impact $(0.3)$ of auxiliary loss on the total loss and observe the affects on mean error. We tried the impact values $[0.4, 0.3, 0.2, 0.1]$ on the networks with auxiliary loss and again averaged over 10 different fixed random seeds to get more reliable estimations. The Table IV depicts the affect of changing auxiliary impact on the networks.

Table IV: Changing Aux impact and effect on mean error

| Models | Aux Impact | Epochs | Mean Error |
|---|---|---|---|
| CNN weight sharing & Aux | 0.4 | 42.8 | 0.0996 |
| | 0.3 | 44.5 | 0.1016 |
| | 0.2 | 49.0 | 0.1012 |
| | 0.1 | 57.7 | 0.1064 |
| FC Network with Aux | 0.4 | 98.1 | 0.1745 |
| | 0.3 | 95.5 | 0.2135 |
| | 0.2 | 131.4 | 0.2144 |
| | 0.1 | 173.9 | 0.222 |

It is clear that both FC and CNN with auxiliary losses performs better with auxiliary loss impact of $0.4$ on the total loss. One important finding of this result is FC network with auxiliary loss with $0.4$ impact performs $(0.1745$ error rate) better than Basic CNN $(0.1808$ error rate) in terms of accuracy. This ultimately proves that auxiliary loss has great impact on performance and it can enable FC Networks to perform better than basic CNN networks without any auxiliary loss.

## V. Discussion & Conclusion

Our results state that the Convolutional Neural Network using weight sharing and auxiliary losses is the most effective model. It is not surprising since it is the model that best exploits data. Indeed, Adding the auxiliary loss introduces the auxiliary task of learning hand written digits class values besides their comparison, hence it extracts more information from the images. As a result it is the model that best generalizes to the data; the improvement made by the auxiliary losses is also present for the FCN. We observe that the implementation of weight sharing also guarantees an improvement with respect to the CNN. Indeed by sharing a part of the weights, weights were using more data for the training and since they were extracting the same information about each image they did not loose flexibility. Finally, combining the auxiliary losses with weight sharing can increase the performance by up to $8\%$ in terms of accuracy. This extend of performance increase is very crucial in image classification tasks and can really play important role in achieving state-of-the-art results.

## References

[1] Y. LeCun, C. Cortes, and C. Burges, “The mnist database.” [Online]. Available: http://yann.lecun.com/exdb/mnist/

[2] “Pytorch.” [Online]. Available: https://pytorch.org/

[3] “Google colaboratory.” [Online]. Available: https://colab.research.google.com/notebooks/intro.ipynb

[4] “Virtual machine.” [Online]. Available: https://fleuret.org/ee559/