Deep Learning – Mini project 1

Alexandre Di Piazza, Elodie Adam, Clémentine Lévy-Fidel

***Abstract*—Image classification is a fundamental task in Machine Learning. Neural networks are a preferred choice when it comes to computer vision tasks. In this paper, we will compare the performance of different architectures on a particular task. The aim of the task is to compare two handwritten digits and determine their order of magnitude.**

I. Introduction

Deep Neural Networks have been very present and particularly appropriate in computer vision in the recent years. However, training a Neural Network requires the choice of many hyper-parameters, and many architectures exist. These different choices directly impact the global performance of the Neural Network. Many problems can appear when decisions regarding the choice of Network are not appropriate, going from the Computational Cost to overfitting. Some architectures are more appropriate for some specific tasks, and it is thus crucial to understand and study the differences of performance of different models.
In this project, the goal of the task is to compare two digits and asses the biggest of the two. We compare 4 different architectures, a multi-layer perceptron (MLP), the Siamese version of the MLP, a convolutional neural network (CNN), and the Siamese version of the CNN.

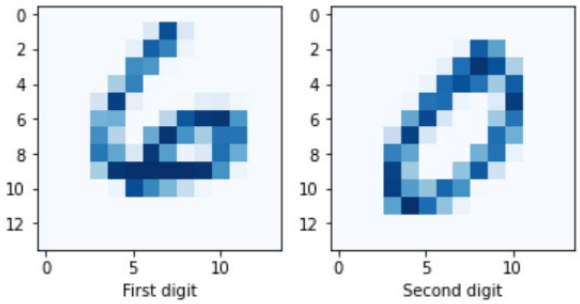

Fig. 1: Example of a pair of digits extracted from the MNIST dataset

II. Data

The dataset is composed of 2000 pairs of 14x14 grayscale images. The images are handwritten digits extracted from the MNIST database. The training and the testing set are each composed of 1000 pairs. In addition to the pair of digits, there is also the corresponding binary number indicating which digit is bigger, as well as the class of each digit of the pair. Table 1 summarizes the dataset, with N = 1000 in our case.

Name	Dimmension	Content
train_input	N*2*14*14	Pairs of digits
train_target	N	Class to predict $\in 0, 1$
train_classes	N * 2	Digit classes $\in 0, ..., 9$
test_input	N*2*14*14	Pairs of digits
test_target	N	Class to predict $\in 0, 1$
test_classes	N*2	Digit classes $\in 0, ..., 9$

TABLE I: Dataset description

III. Architectures

We compare the 4 following architectures :

- Multi-Layer Perceptron (MLP)
- Siamese MLP
- Convolutional Neural Network (CNN)
- Siamese CNN

A MLP is organized in fully connected layers within which information flows from the input layer to the output layer only. The MLP if composed of 4 fully connected layers in our case. A CNN is another class of neural network, whose specificity comes from the convolutional layers, using a kernel to recognize patterns of an image. Our CNN is composed of 3 convolutional layers and 4 fully connected layers. The precise description of the architectures (depth, activation, etc) is shown in the annexe.
For the Siamese version of the network, each digit of the input pair is fed into the same subnetwork (they thus share the same weights), which outputs the class of the digit. At this step we use an auxiliary loss to compare this prediction with the real class. We then merge the two digits together and output a binary number to tell which number is bigger, computing the loss with the real result. The final loss is then a weighted sum of the 3 losses. Figure 2 shows the description of the Siamese version of the CNN and MLP.

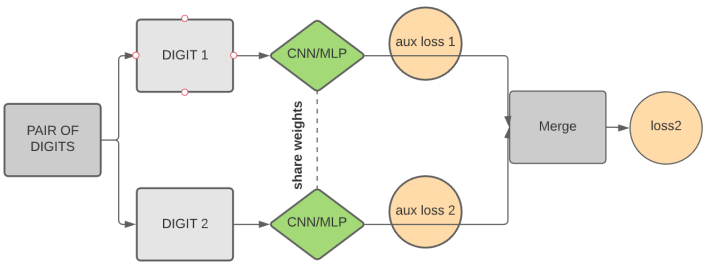

Fig. 2: Description of the Siamese Network

The goal of this project is to have the best fair comparison possible. In order to do that, we decided to have the same fully connected layers for all the models. The only small differences being the dimension of the first hidden layer between the MLP and the CNN (because the convolutional layers change the dimensions), as well as the first dimension of the last connected layer between a model and its Siamese version (because we need to merge the two images in the Siamese version). This way, we can clearly measure the impact of the convolutional layers by comparing the results of the MLP and the CNN. Similarly, we can clearly see the impact of weight sharing and the use of auxiliary loss by comparing each model with its Siamese version.
We used Adam optimization algorithm with learning rate 0.0001, Binary cross entropy for the loss function and Cross Entropy for the auxiliary loss function used in the Siamese version. For the same purpose of a fair comparison, we use the same dropout regularization for each model. We also use the same hyper parameters (epochs, activation function, learning rate, batchsize etc..) for each model. The Siamese version has one extra hyper parameter which is the coefficient for the second auxiliary loss function before computing the weighted sum. We decided to optimize it through a Cross Validation by choosing between 5 values.

IV. Results

For each model, we trained 10 times, each time randomizing the data as well as weight initialization, with 30 epochs for each run. We first plot the results on one run, where we can see evolution of the training loss and test accuracy through the epochs. We then provide the mean of the final test Accuracies, Precision and Recall after 10 rounds, with an estimate of the standard deviation for each metric.

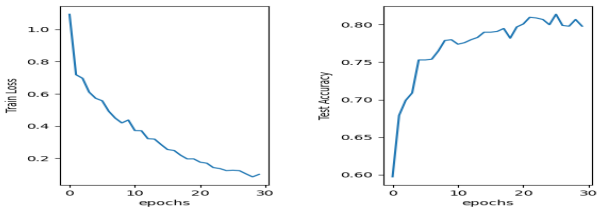

Fig. 3: Training loss and test accuracies versus epochs for the MLP

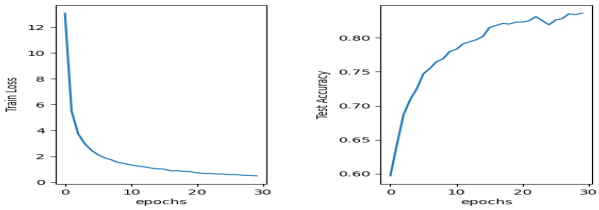

Fig. 4: Training loss and test accuracies versus epochs for the Siamese MLP

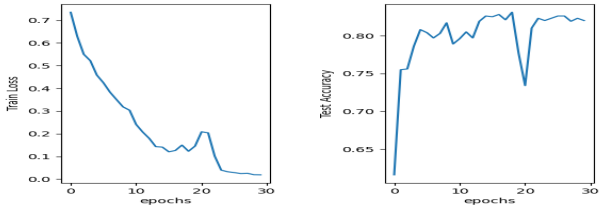

Fig. 5: Training loss and test accuracies versus epochs for the CNN

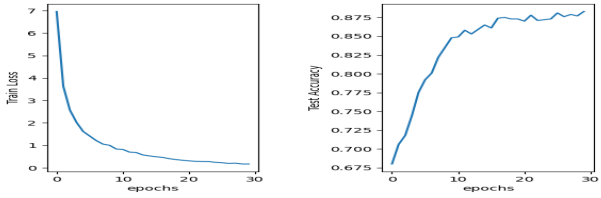

Fig. 6: Training loss and test accuracies versus epochs for the Siamese CNN

Models	Accuracy $\pm\sigma$	Recall $\pm\sigma$	Specificity $\pm\sigma$
MLP	0.79±0.01	0.79±0.05	0.78±0.05
Siamese MLP	0.81±0.02	0.84±0.04	0.79±0.02
CNN	0.81±0.03	0.83±0.06	0.79±0.05
Siamese CNN	0.87±0.01	0.89±0.02	0.86±0.02

TABLE II: Results for the 4 architectures for 10 runs

Having estimations for the standard deviation, the mean and the number of simulations, we can compute confidence intervals. We provide here 95 % confidence intervals with the following formula, based on the law of large numbers and the CI for normal distributions : $[\widehat{\mu} \pm \frac{1.96\widehat{\sigma}}{\sqrt{(N)}}]$, where N is the number of simulations, 10 in our case. We note that this formula is an asymptotic confidence interval, meaning that the probability that the real mean belongs to this interval tends to 1 as N tends to infinity.

Models	CI Accuracy	CI Recall	CI Specificity
MLP	[0.784,0.796]	[0.76,0.82]	[0.75,0.81]
Siamese MLP	[0.797,0.822]	[0.815,0.875]	[0.77,0.81]
CNN	[0.791,0.828]	[0.793,0.867]	[0.76,0.82]
Siamese CNN	[0.864,0.877]	[0.878,0.9]	[0.85,0.87]

TABLE III: 95 % Confidence intervals for the Models

V. Discussion

First of all, we can see that all the 4 models achieve a reasonable performance for the task. Indeed, the class being balanced (around 55% of 1 in the train target set, and 52% of 1 in the test target set), we can assess that around 80% accuracy with 30 epochs is a reasonable performance. We can observe on the plots of the training loss and test accuracy that for all the architectures, the test accuracy increases and the training loss decreases through the epochs. It is worth noting that we also first plotted the test loss versus epochs. However, the test loss first decreased trough the epochs, then started increasing even though the test accuracy continued to increase. This phenomenon comes from the fact that some wrong predictions on the test values kept getting worse, making the test loss greatly increase, even though those values didn't affect the test accuracy. Bad predictions are penalized more strongly than good predictions are rewarded. Thus, plotting the test loss wasn't meaningful to understand the performance of the models.

In our task of predicting the highest digit of the pairs, a False positive is not more important than a False negative. Similarly, a True negative is as important as a True positive. In addition, as noted earlier, the targets were balanced, and thus the training was not more influenced by a specific target. We thus decided that the most appropriate metrics were accuracy, recall (true positive rate) and specificity (true negative rate), which allow to estimate the performance of the models on predicting the right target. Recall measures the number of correct positive predictions out of all the positive predictions that could have been made. Similarly, specificity measures the number of correct negative predictions out of all the negative predictions that could have been made.

As discussed in the choice of architectures, the only main difference between the MLP and the CNN is the convolutional layers. The CNN achieves better results, which is not surprising as CNN have proved to be appropriate for visual imagery. The convolutional layers are thus appropriate in this task to achieve a better performance. It is also worth mentioning that we tested to add fully connected layers to the MLP in order to see if the difference of performance on the MLP and the CNN came from the fact that the CNN had more layers. However, it appeared that adding one or two layers to the MLP didn't affect its performance.

Comparing each model with its Siamese version, we can also clearly see the impact of weight sharing and the auxiliary loss function. We can see that the Siamese CNN and Siamese MLP both achieve a better performance than their simple version. It is also not surprising. The Siamese version uses an auxiliary loss and have access to additional information by knowing the class of each digit in the train set. It thus makes sense that additional train information helps the network to perform better on this particular task. In addition, the idea of the Siamese network seems more intuitive for this task; it is more similar to what a Human would do. It first classifies the two digits, and then compare them to say which one is bigger.

Finally, it is important to note that while the Siamese versions improve the models for both MLP and CNN, the improvement is much greater for the CNN. Indeed, the Siamese MLP increases the accuracy by 2% whereas the Siamese CNN increases the accuracy by 6%. We can thus think that the convolutional layers help classifying each digit, which allows the Siamese CNN to have a better prediction of each class, and finally be more accurate for the comparison.

VI. Annexe

Here are the two descriptions of the MLP and the CNN. Their Siamese version can be understood directly with figure 2

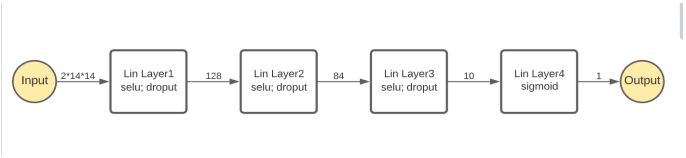

Fig. 7: Architecture of the MLP

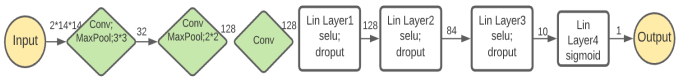

Fig. 8: Architecture of the CNN