Convolutional Neural Networks Performance Assessment to Classify Grayscale Pictures of Two Digits

Nicolas Jomeau, Ella Rajaonson, Sofia Dandjee

Abstract— **This paper addresses convolutional neural networks (CNN) architecture performance in the case of recognizing 2 MNIST digits and predicting which one is the greatest. To solve this problem, we examined the impact of sequential and parallel weight sharing and the use of an auxiliary loss (training on the digit classes) on the performance of a CNN. We also trained a model relying solely on the digit classes for the final prediction. Training with the mean squared error loss and the Adam optimization algorithm, we found out that the auxiliary loss with a weight of 40 and the digit model are the most performing networks by classifying around 97 % of the test pairs correctly.**

I. INTRODUCTION

CNN are commonly used in image or video classification, time series or natural language processing as they easily capture a context of an input (ie: neighbours of a pixel). One of the most known CNN task is the MNIST dataset that consist in grayscale pictures of hand-written numbers. This paper's dataset is composed of pairs of MNIST digits and the task of the model is to determine which digit is the greatest. Multiple models were tested to see how specific paradigms such as weight sharing and auxiliary loss may impact the quality of prediction. The use of an auxiliary classifier was notably used in the GoogLeNet paper [1] as well as in the PSPNet paper [2]. In this paper, the auxiliary classifier also aims at determining the class of single digits in a pair, in addition to the main objective. The proposed models are a simple CNN, a sequential and a parallel weight sharing CNN to allow more layers for the same amount of trainable parameters, a CNN using an auxiliary loss on each single digit class and finally a CNN that would strictly train on single digits. The models' performance were evaluated according to their label prediction error. We used the PyTorch package to create, train and evaluate our models.

II. DATASET

The training and test set contain 1000 pairs each. Each pair is composed of 2 channels (one for each digit of the pair) of 14x14 grayscale pixels. A pair is labelled as 1 if the first digit is smaller or equal to the second, and as 0 otherwise. Each digit is also assigned to a class between 0 and 9. The training and test set were normalized with respect to the training mean and standard deviation.

III. MODELS

In order to provide a fair comparison of our models, the 5 different architectures were designed to have around 70000 parameters, thus having comparable complexities. Note that same number of parameters does not mean same running time: some implementations may run way faster than others, especially deep neural nets with small layers versus shallow neural nets with large layers.

- Simple CNN: 3 convolutional layers (16, 32 and 32 filters with a kernel size of 2 for each), a 2d max pooling with kernel size of 2, 3 linear layers (800 input units, 3 hidden layers of 75 units and 1 output unit).

- Weight-sharing CNN 1 (CNNws1): 3 convolutional layers (same as above), a 2d max pooling with kernel size of 2, one linear layer (800 input units), one linear layer repeated 3 times (75 input and output units) and one final layer to output the prediction (75 input units and 1 output unit). The weight sharing here is done sequentially.

- Weight-sharing CNN 2 (CNNws2): The pairs are first split into single channels. These digits go through 3 convolutional layers (same as above) and 3 linear layers (800 input units, 2 hidden layers of 50 units and 10 output units) before being concatenated back together. Then the pairs go through 4 small linear layers (20 input units, 3 hidden layers of 50 units and 1 output unit) before outputting the result. Here, the weight sharing is done in parallel.

- Auxiliary loss CNN (CNNaux): The same architecture as the CNNws2 is used (see Figure 1). The main difference is that we use the output of the single channels before they are reconcatenated as an auxiliary output in order to compute the auxiliary loss. This loss will then be added to the main loss to help optimize the learning process and avoid the vanishing gradient problem. The weight of the auxiliary loss in the final loss computation can be tweaked to give it more or less importance and its influence is studied in the next section.

- Digit CNN (CNNd): Finally, we wanted to see how a CNN would perform if we only trained it to recognize digits. This CNN has 2 convolutional layers (with 8 and 16 filters with a kernel size of 3) with a 2d max pooling with kernel size of 2 in between, followed by 3 linear layers (256 input units, 3 hidden layers of 100, 200 and 100 units respectively and 10 output units). The i-th component of the output is the likeliness of the digit being equal to i. The prediction of a pair is then simply the comparison between the predicted classes of its digits.

In each architecture, the layers were followed by the ReLu activation function.

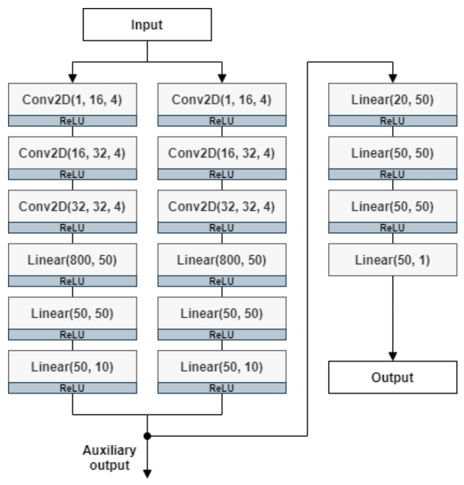

Fig. 1: Architecture of the CNNaux model

IV. EXPERIMENT

The models were trained using the Adam algorithm [3] for 25 epochs and a mini batch size of 100.

A. *Learning rate optimization*

The optimal learning rate for each model was tuned with a 4-fold cross validation in the range of $[10^{-4}, 10^{-2}]$. The results are shown in Table I.

Architecture	Learning Rate	Mean validation acc (%)
Simple	1.10^{-3}	80.6
Shared Weights 1	1.10^{-3}	76.6
Shared Weights 2	5.10^{-4}	84.0
Auxiliary @ 0.5	5.10^{-3}	86.8
Auxiliary @ 2	1.10^{-3}	94.4
Auxiliary @ 10	5.10^{-3}	97.1
Auxiliary @ 40	1.10^{-3}	96.8
Digit	5.10^{-4}	96.6

TABLE I: Optimal learning rate for each architecture

B. *Influence of the weight of the auxiliary loss*

We also investigated the influence of the weight α of the auxiliary loss on the performance of the auxiliary network with a 4-fold cross validation. The mean validation accuracy and its standard deviation were plotted as a function of α (Figure 2). The siamese network and the digit CNN mean validation accuracies were plotted as well. The siamese network corresponds to the auxiliary network with $\alpha = 0$. On the contrary, the more α increases, the more the network relies on the digit individual classes. Indeed, as the auxiliary loss weight increases, the accuracy of the network becomes comparable to the one of the digit CNN, where only the digit classes are used. For the rest of the paper, we keep $\alpha \in \{0.5; 2; 10; 40\}$ as the accuracy does not improve with a superior weight.

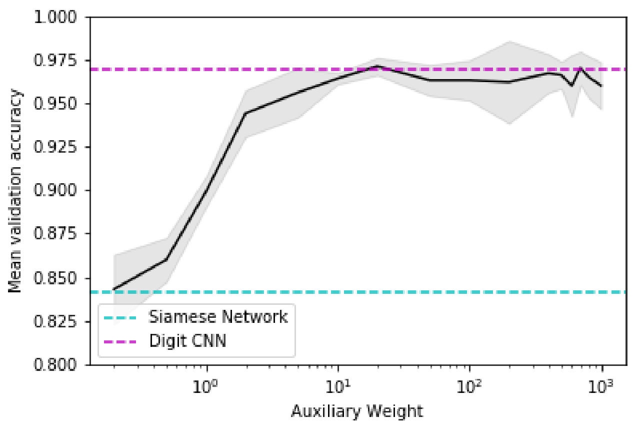

Fig. 2: Influence of the auxiliary weight α on the performance of the auxiliary network

V. EVALUATION

A. *Protocol*

To assess the models' performances, we plotted the average training error over the epochs (Figure 3) over 10 runs. We also collected the average training and test accuracy over the 10 runs as well as their standard deviation (Table II). Each run consisted in training the model over 25 epochs with its optimal learning rate, randomized initial parameters and 1000 randomized training and test samples. All runs were performed on a i7-6560U with 8GB of RAM and took from 5s (CNNd) to up to 15s (CNNaux) each.

B. *Results*

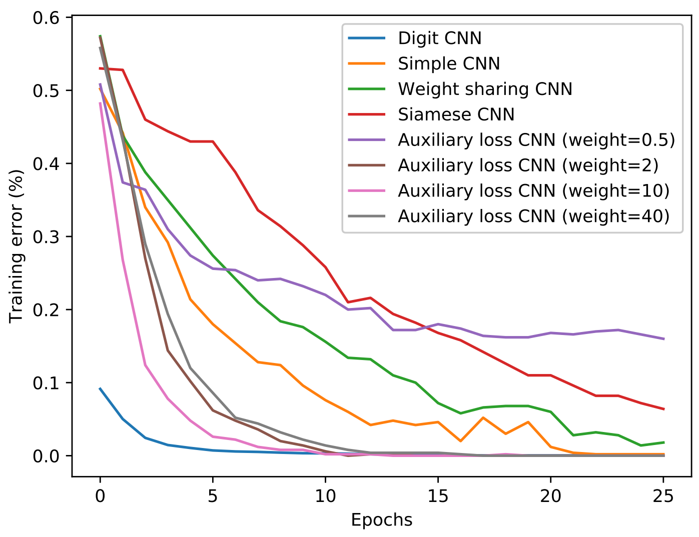

Fig. 3: Models' average training error over 25 rounds of training

Architecture	Test accuracy (%)	Training accuracy (%)
Simple	80.2 ± 1.9	99 ± 1.4
Shared Weights 1	79.6 ± 2.0	95.8 ± 2.7
Shared Weights 2	83.5 ± 1.1	92.9 ± 1.2
Auxiliary @ 0.5	84.6 ± 11	84.9 ± 22.3
Auxiliary @ 2	96.4 ± 0.9	100 ± 0
Auxiliary @ 10	97.2 ± 1.2	100 ± 0
Auxiliary @ 40	97.6 ± 0.3	100 ± 0
Digit	97.8 ± 0.2	100 ± 0

TABLE II: Models' training and test accuracy after 10 rounds of training

C. Discussion

We can observe on Figure 3 that the digit model, the simple CNN and the auxiliary loss models converge more rapidly and reach a 0 % training error after 25 epochs. On the contrary, the CNNws1 and the CNNws2 have not yet converged and only manage to reach below 10 % of error at the end of the training. Note that the CNNaux with a $\alpha = 0.5$ has a very high variance, which we could not justify. Therefore, it will not be taken account in the discussion. Table II shows that the simple CNN and the CNNws1 are close in terms of training and test accuracy. This is surprising as the CNN with weight-sharing should have more generalization power because of the higher complexity of its layers. We can observe that both models overfit the data.

The siamese network brings some improvements on the test data and is less prone to overfitting. However, it is nothing compared to the outperforming models: CNNd and CNNaux. This can be explained by the easier problem they have to deal with. Indeed, instead of using 1000 samples with around 100 possible combinations (from 0-0, 0-1, ... to 9-9), they have access to twice the amount of samples (2 digits per pair) for ten times less labels (from 0 to 9). Furthermore, the auxiliary loss CNN can be tweaked to behave like the siamese CNN (in which the auxiliary loss is ignored) or like the CNNd (in which the auxiliary loss weights much more than main loss). The auxiliary network's performance with a weight $\alpha = 10$ or $\alpha = 40$ have comparable performances with that of the digit CNN, achieving a test error of less than 2 %. This shows that in order to predict the greatest of two numbers, it relies heavily on identifying the class of the single digits (between 0 and 9).

VI. CONCLUSIONS

The auxiliary loss CNN and the digit CNN proved to be the best ways to determine the biggest of two grayscale numbers with a test accuracy of around 97 % although they require more annotated data (in this case each digit's class) and, in the case of the auxiliary loss, run much slower than simplistic CNNs. Regarding the weight-sharing CNN, we haven't found any interesting result with the sequential weight sharing but the parallel weight sharing improved the accuracy by an additional 3.3%.

VII. GOING FURTHER

One more test that could have been tried but was unrelated to the main problem was data augmentation. The technique consists in creating the mirror pair and opposite label of each sample to effectively double the training dataset. We could expect from this increased model's quality but we are unsure about the CNNaux and CNNd as they would surely overfit from seeing the same single digit sample multiple times. Also, even more new pairs can be generated thanks to the digit's class by mixing samples between each others.

REFERENCES

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions, corr abs/1409.4842," *URL http://arxiv.org/abs/1409.4842*, 2014.

[2] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," 2016.

[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.