Deep Learning - Project 1

Bassel Belhadj, Charles Gallay, Gregoire Clement

I. Abstract

In this project, we implement different models in order to compare the impact of their architecture on a given task. Using the MNIST dataset [1], the task is to decide between two images if one is lesser or equal to the other. We show the beneficial impact of weight sharing, in particular through the use of CNN [2]. Also, we demonstrate how the use of an auxiliary loss can help the training of the main objective.

II. Introduction

A. Data

The data comes from the MNIST dataset. Instead of the original resolution of 28x28, we work with a computationally less expensive resolution of 14x14 in order to run on CPU more easily.

The dataset we are using consists of randomly sampled pair of images, thus data of shape 2x14x14. It also includes a label which is a boolean value and the respective classes of the images which will be used to construct an auxiliary loss.

B. Task

Given two images representing digits, the task is to predict a boolean value which represent whether the first digit is lesser or equal to the second. In other words, it is a binary classification task.

III. Implementation

In order to solve this rather simple task, two main models are implemented. This first one is a straightforward fully connected network and the second one makes use of convolutional layers. Here, the goal is to analyse the impact of the convolutional layers by comparing the performance of each model.

These architectures are more detailed below. Note that for each model, the criterion used to compute the loss is the binary cross entropy. The entire pipeline is partially summarized in Fig. 1.

A. Models

1) Fully Connected (FC): This architecture will act as a baseline as it is rather straightforward. It consists of three separate fully connected layers. Each image are first fed individually into two successive fully connected layers, then their respective outputs are concatenated and finally fed into the last fully connected layer to predict a single binary output. After each layer, the activation function used is ReLU, except the last one where a sigmoid is used for the binary classification.

2) Convolutional (CNN): This second architecture is quite similar, except at the beginning where it contains two additional convolutional layers. Note that in order to roughly keep the same amount of trainable parameter than Fully Connected variant, we make the fully connected layers thinner. The two convolutional layers together with two fully connected layers extract features for each image which are then concatenated. Finally, there is a fully connected layer with a single output. Again, after each layer, the ReLU activation function is used, except for last one where the sigmoid is preferred.

B. Auxiliary Loss

For each model explained above, an auxiliary loss is added to the original. This loss is calculated with the output of the one but last fully connected layer of width 10 which is meant to represent, after the softmax activation function is used, the probability distribution of the digits. For this reason, the criterion used is the cross entropy loss. A parameter α is added to control the influence of the auxiliary loss.

$$L = L_1 + \alpha L_2$$

Hence, one should note that setting the parameter $\alpha = 0$ means the auxiliary loss is not used.

C. Hyperparameters

Few hyperparameters were tuned using a simple grid search algorithm. Especially the auxiliary loss factor (α) such that the attention of the model is shared optimally between the two tasks. After some iterations, we fixed the hyperparameter $\alpha = 0.5$

D. Architectures comparison

We have two different models and each of them can make use of the auxiliary labels to compute a more sophisticated loss. In order to be able to compare them, they need to have a similar amount of trainable parameters. This results in the following four different models:

Architecture	# parameters	α	weight sharing
FC	2101	0	No
FC$_{aux}$	2101	0.5	No
CNN	1781	0	Yes
CNN$_{aux}$	1781	0.5	Yes

IV. Results

In order to assess the performance of each model, we need to construct confident estimates. For this reason, we start the training of each model with the same seed and train it for a number of 10 different rounds. Therefor each model are evaluated on the exact same validation set. Also, at the start

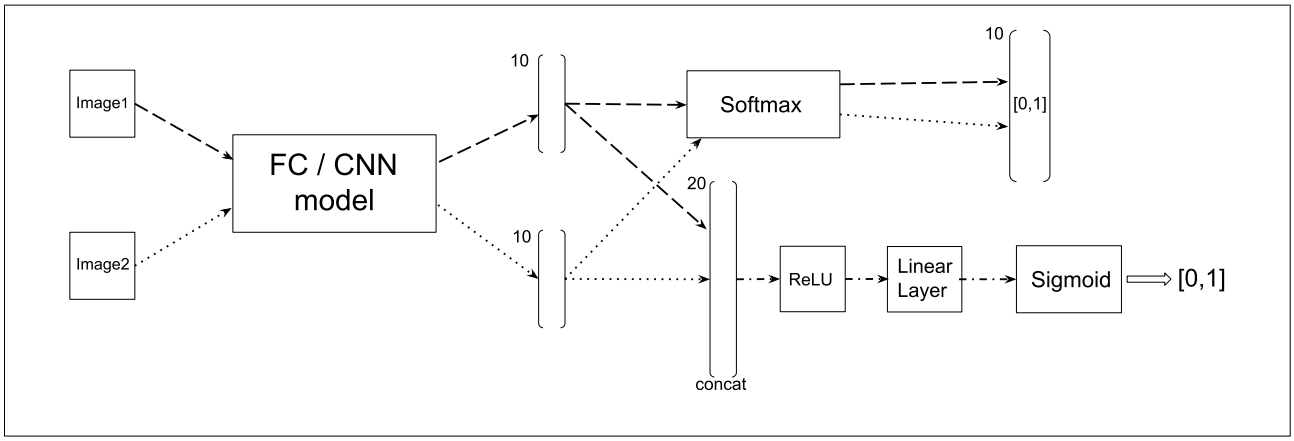

Fig. 1. Model architecture (above part is used for the auxiliary loss)

of each round, we initialise the model weights and reset the optimizer. Each round consists of a number of 150 epochs, with a batch size of 32. In this setting, we obtain the following results:

Architecture	Accuracy	Standard deviation
FC	0.7941	0.0190
FC_{aux}	0.8143	0.0344
CNN	0.8428	0.0106
CNN_{aux}	0.8759	0.0162

As a matter of fact, together with Fig. 2 and Fig. 3 one can make the following observations:

First, a well adjusted auxiliary loss help the model to generalise better in the long run. It enables the model to better extract features in both models. Even though, one should notice that in the short run, as it shifts part of the attention of the model, it performs slightly worse for both models.

Secondly, the CNN model, with even slightly less parameters, also generalises better. This can be explained by the weight sharing property of its architecture which is more suited for pictures where locality matters. Extremely far pixels do not contain much information together. It can also be explained similarly by its translation equivariance property. In fact, a filter from the convolutional layers react to a particular pattern regardless of its location in the image and need to be learned only once.

V. Conclusion

In this paper, we presented an implementation which quantifies, using two different models, both this impact of weight sharing and the use of an auxiliary loss in the provided data training. We conclude by showing that both weight sharing and an auxiliary loss can help the model to generalise better as long as they are used correctly. In fact, an auxiliary loss, if it contains additional information, can always be plugged into a model helping it generalise on a given task. Finally, it is now common knowledge [3] that CNNs have increased tremendously the performance on almost all computer vision related tasks.

VI. Future Improvement

One way to improve on the accuracy could have been to introduce a new architecture that make use of CNN layers only. Nowadays, one of the common practice for such classification task is to use a Global Average Pooling at the very end of the convolutional layers. Doing so the network becomes globally invariant to translation of which might be a desire property. This allows for even more weight sharing.

References

[1] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits, 1998," *URL http://yann. lecun. com/exdb/mnist*, vol. 10, p. 34, 1998.

[2] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[3] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

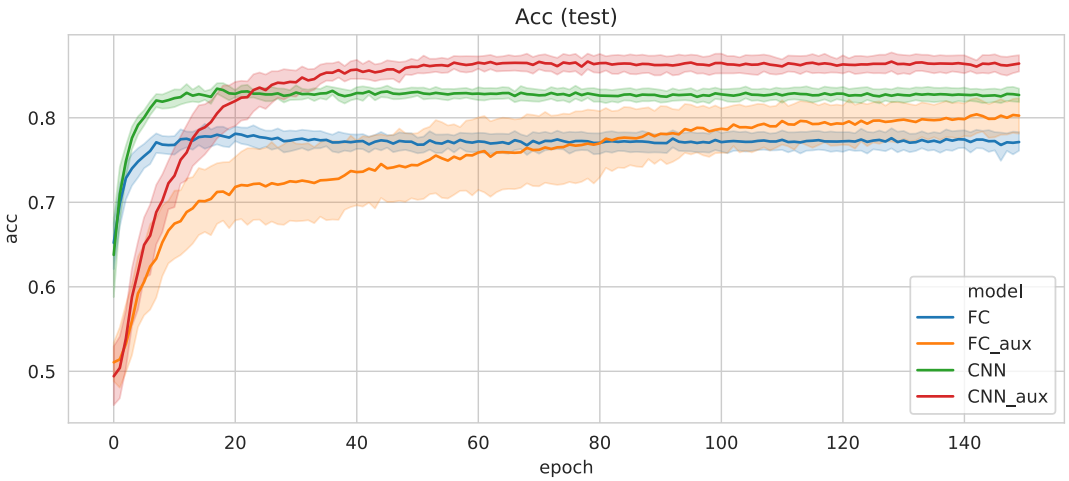

Fig. 2. Accuracy on the test data

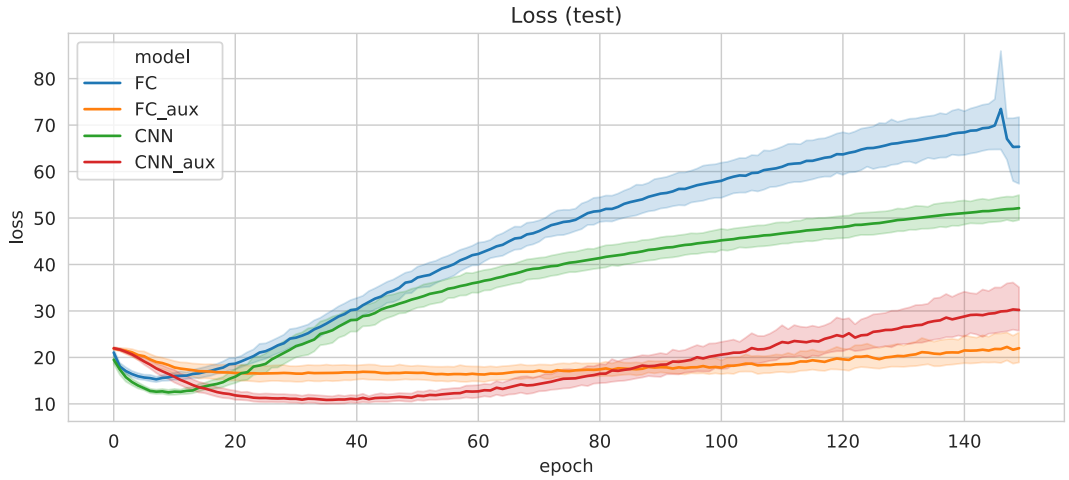

Fig. 3. Loss (without auxiliary) on the test data