EE-559 Deep Learning

Spring Semester 2019

Development of a deep neural network comparing hand written digits

September 22, 2019

Project 1

Sacha Haidinger *sacha.haidinger@epfl.ch*
Nils Olsen *nils.olsen@epfl.ch*
Antoine Spahr *antoine.spahr@epfl.ch*

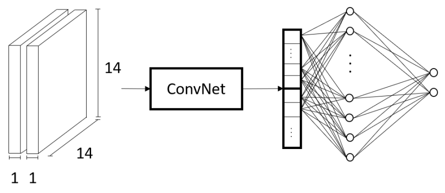
Contents

1 Introduction

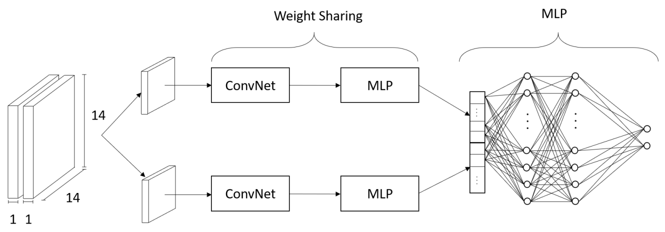
The scope of this project is to design different network architectures and assess their performance in a specific task of classification. The task itself requires to compare two numbers from the MNIST data set and determine whether the number on the first image is smaller or equal than the number on the second image. Each model has a different architecture implemented with the base class *Modules* from the Pytorch library. Two features are implemented in the different architectures: *weight sharing* and *auxiliary loss*. The performances of the models using such features are compared to a base model which is composed of convolutional layers and a Multi Layer Perceptron (MLP). The data set consists of 1000 train and 1000 test samples of dimension 14x14x2. Indeed, each sample consists of two images placed one after the other.

2 Architectures

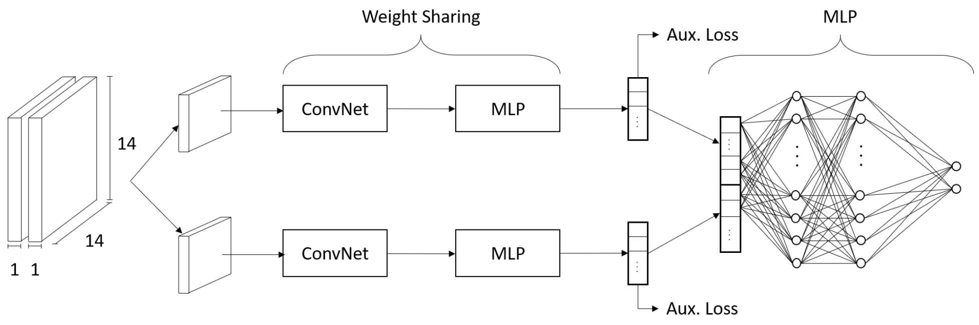
In order to assess the performance gain with the use of *weight sharing* and/or *auxiliary loss*, three different architectures are implemented.
The first one is a simple convolutional neural network (CNN) used as a reference (see figure 1a). It takes as input the set of two MNIST images. This 14x14x2 input goes first through convolutional layers and then through a MLP, whose output layer contains two units, representing the two classes of the main objective (1 if the first image is smaller than the second, 0 otherwise).
The second architecture implements the weight sharing (see figure 1b). The 14x14x2 input is first split into two 14x14x1 images. Each image goes through a convolutional layers and a first MLP that are sharing the same parameters for both images. The outputs of this common net are two 10x1 vectors. They are then concatenated into a 20x1 vector which goes through another MLP outputting the class (0 or 1).
The last architecture implements the weight sharing together with the auxiliary loss (see figure 1c). The architecture is thus similar to the one with weight sharing. The difference lays in two additional losses : the outputs of the first common net (10x1 vectors) are compared to the digit (as hot labels) present on the image. The network is thus enforced to learn the digits.



(a) ConvNet



(b) WeightSharing



(c) AuxiliaryLoss

Figure 1: Models architectures. Architectures are presented as block schemes. The detail of the parameters used is shown in table 1.

Table 1: Architecture parameters. The details of the parameters used to build the three models are presented here. Note that there is no MLP shared in the simple CNN. Note also, that the common parts are kept as similar as possible to provide a better comparison between the performance of each model.

	Simple CNN	Weight Sharing	Auxiliary Loss & Weight Sharing

Convolutional Layers	Conv2d(2, 32, kernel,= 5)
	MaxPool2d(kernel = 2)
	Selu()
	Conv2d(32, 64, kernel = 3)
	MaxPool2d(kernel = 3)
		Conv2d(1, 32, kernel,= 5)
		MaxPool2d(kernel = 2)
		Selu()
		Conv2d(32, 64, kernel = 3)
		MaxPool2d(kernel = 3)
			Conv2d(1, 32, kernel,= 5)
			MaxPool2d(kernel = 2)
			Selu()
			Conv2d(32, 64, kernel = 3)
			MaxPool2d(kernel = 3)

MLP shared	-
	-
	-
		Linear(64, 120)
		Selu()
		Linear(120,10)
			Linear(64, 120)
			Selu()
			Linear(120,10)

MLP	Linear(64,230)
	Selu()
	Linear(230,100)
	Selu()
	Linear(100,2)
		Linear(20,230)
		Selu()
		Linear(230,100)
		Selu()
		Linear(100,2)
			Linear(20,230)
			Selu()
			Linear(230,100)
			Selu()
			Linear(100,2)

3 Training and testing procedures

Final performances of all three models were obtained following the most possible similar training and testing procedures. As required, a training set of exactly 1000 pairs of downsampled and scaled to 0-1 MNIST images data was first generated, but enlarged to 18'000 samples in a second time throughout a data augmentation process. Although the resolution is too low to enable a convenient rotation, new samples were generated by shifting original samples by one pixel in eight different directions and by flipping the order of the two images.
Models were trained over 40 epochs, and data was presented by random batches of 100 samples (samples were randomly distributed in batches at each epoch). To speed up the learning process, Adaptive Moment Estimation (Adam) optimizer was used with a given stable learning rate ($\eta = 3 \cdot 10^{-3}$) and L2 regularization strength (weight decay, $\gamma = 2.5 \cdot 10^{-5}$). In addition, overfitting management was enhanced by a dropout regularization at a fixed level of 0.15 in all MLP layers. Cross Entropy criterion was used as loss function in all models.
In the architecture using auxiliary losses, the overall loss was a weighted sum of both auxiliary losses associated with the digit recognition task (*weight coefficient of 1*) and the main loss associated with the two digits comparison task (*weight coefficient of 0.5*).
Models were eventually tested with a test set of 1000 newly generated samples and scaled to 0-1 values. All final performances shown in Figure 2 were estimated throughout 20 rounds for each architecture, and are presented alongside their standard deviation to assess models stability.

All the hyperparameters values given above were obtained by grid-search optimizations over a broad range of values with low resolution and assuming hyperparameters are inter-independent to one another. We are totally aware that in order to reach the best performances possible, each model should have been optimized separately and dependencies between hyperparameters should have been tested. Finding respective global optima of all three models would have been the best way to compare their performances. However, as it is time-consuming and out of the scope of this report, we decided to restrain the optimization to a broadly tuned set of hyperparameters values common for all three architectures. Indeed, as all architectures share similar core network blocks, finding a common local optima should be sufficient to compare performances across models.

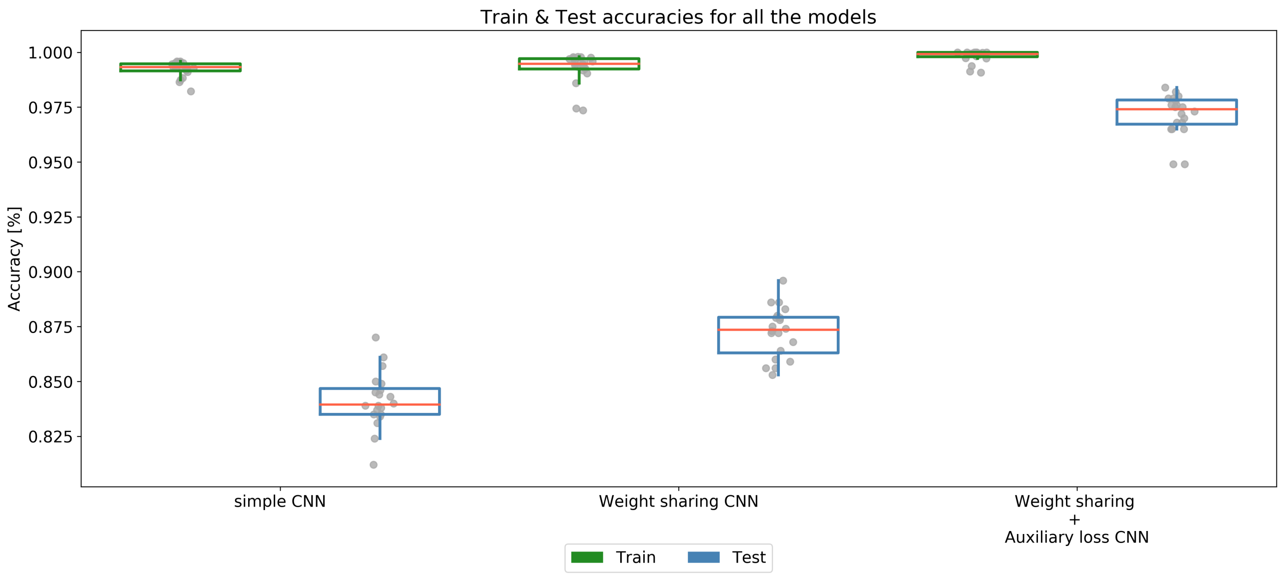


Figure 2: Models performances. The performances over 20 train procedures are presented as the accuracy on the train and test data for the three architectures. Simple model : Training 99.2% ± 0.4% Testing 84.1% ± 1.2%. Weight Sharing model : Training 99.3% ± 0.7% Testing 87.2% ± 1.1%. Weight Sharing and Auxiliary loss model : Training 99.8% ± 0.3% Testing 97.1% ± 0.9%

4 Discussion

The different model architectures used in this project were particularly chosen in order to assess the impact of weight sharing and the use of auxiliary losses on the performance of the main objective: compare two digit of the MNIST data. To stay consistent and end up with a meaningful comparison, a similar training procedure (cf. *Section* 3) and core CONV and MLP blocks (similar in their number of layers, number of units and kernel sizes, cf. *Table* 1) across the architectures were used. All models seemed to rapidly overfit and their train performances hit values near 100%. It might be due to the high number of parameters compared to the number of samples, or by a too prominent number of epochs; train data is presented so many times that models became too specific to it. As expected, the reference simple CNN model reach the lowest test performance (*84.1%*).

The Weight-Sharing CNN model reach 87.2% whilst adding the use of auxiliary losses gave the best test performance, namely 97.1%. The reference model took as input two-channels images and was tuned to represent through its two last units the two classes of the main objective. However, the model had obviously no idea about the task to perform and that input data were in fact two images to compare. An efficient way of performing the main task would be to first decode the digit present on each images and then compare them. By the use of weight sharing, a common network was fed and tuned with the two images separately. Model was therefore more prompt to learn features related to a single image and thus the digit associated to it; the model is in a way forced to compare the two images. Moreover, as the recognition task is the same for the first and second image, weight-sharing enabled to double the number of samples presented to the network's modules responsible for this task.

Lastly, the use of auxiliary losses forced the model the strongest way possible to solve the main objective in a efficient manner. Models first learned features related to digit representation, as the two newly added losses pushed the optimized model to decode those digits, and only then compare the images. It was therefore expected to obtain the best performances with that model. It is interesting to note that the optimal hyperparameters defining the overall loss presented in *Section* 3 gave more weight to the auxiliary losses rather than the main one. Digit recognition task is essential for the main objective, and once done accurately, the comparison task gets simpler.