Mini-project 1: Classification, weight sharing, auxiliary losses.

Fares Ahmed, Andrej Janchevski, Nicolò Macellari
Deep Learning EE-559

***Abstract*—Understanding and being able to use techniques such as auxiliary losses and weight sharing as well as batch normalization and skip connections is a fundamental part of the developing of deep neural networks. In this report we explore the effect of these techniques in a subset of images coming from the MNIST database using a basic ConvNet.**

I. Introduction

The increase in computational power gives developers the ability to train deeper neural networks leading to unforeseen performances and increased scientific interest. For example a deep network called AlexNet [1] outperforms the precedent algorithms for classification of ImageNet database, reaching 17% error rate versus 25.7% of the previous best solution [2]. Such astonishing performances highlight the power of deep networks thus stimulating their study and developments.

Nevertheless, training deep networks is not immediate but has to face problems related to computational time, training complexity, overfitting, gradient backpropagation and vanishing and others. Many techniques have been introduced over the years to face these problems. Among them, weight sharing is a powerful tool, proposed for the first time already in 1985 [3], that reduces the amount of free parameters of the model by making several connection (links) controlled by the same parameter (weight) [4]. Thus, reducing the degree of freedom of the model and the training complexity.

The use of auxiliary losses instead helps back-propagating the gradient signal by adding small classifiers to early stage modules of a deep networks [5]. Their weighted losses is then summed to the loss of the main model.

For early stage developers in the field of deep learning, it is of the greatest importance to understand the mechanism of these solutions. Thus, we decide to investigate their effects when approaching a simple classification task of the MNIST database. We use a basic ConvNet to recognize the biggest out of two handwritten digits in the form of 14x14 images. Moreover, we investigate the limit case of siamese models sharing the same weights and architecture for both digits. We then evaluate the performances of the complete siamese model with auxiliary losses (called siamese 2) with the performances obtained by using the branches to singularly classify the digits and performing a maximum operation on the chosen label to get the output (siamese 10).

Finally, we study the effects of batch normalization [6] and skip connections in such a basic ConvNet and cross-validate the results for a different number of hidden units and dimensions of the convolutional layers.

II. Models and methods

MNIST database of handwritten digits [7] has been used over time as a platform for the development and testing of new machine learning and deep learning algorithms [8]. It is composed of a training set and a separate test set of respectively 60000 and 10000 labeled images of handwritten digits and is a subset of the larger NIST database [9].

The input to the models is composed of a set of 2x14x14 tensor corresponding to pairs of 14x14 gray scale images. These images correspond to handwritten digits from 0 to 9. The desired output should be the biggest of the two digits based on a binary [0,1] classification. A training set and a test set of respectively 1000 pairs were randomly generated and normalized according to the mean and standard deviation of the training set.

The basic ConvNet model used is composed of first a convolutional layer of 2 channels as input and kernel size 3, a maxpooling layer of kernel 2 and a ReLU non linearity, a second convolutional layer of kernel 6 and two fully connected layers with two output units. Each convolutional block can be followed by batch normalization and/or skip connections from the input signal.

The 2 skip connections pass the input signals through a fully connected layer and a ReLU non linearity in order to respect dimensionalities and then sum it to the processed signal after respectively the two convolutions. Batch normalization is done after each convolution (and maxpooling) block and before the following skip connection.

The same model has been used as the architecture of the siamese structure. Two branches with the same backbone have been linked with a final fully connected layer with 2 outputs and 20 inputs. The two branches have been structured with the same convolutional layers as the single-side model followed by batch normalization and skip connections, and two fully connected layers with 10 output units.

Stochastic gradient descent (SGD) has been implemented as the method for optimization and the model loss on the training set has been computed using cross-entropy. For the siamese model the loss has been computed as a weighted sum of the loss at the last fully connected layer and an auxiliary loss obtained with the outputs of the two parallel branches and the 10 classes (0 to 9) of the input digits.

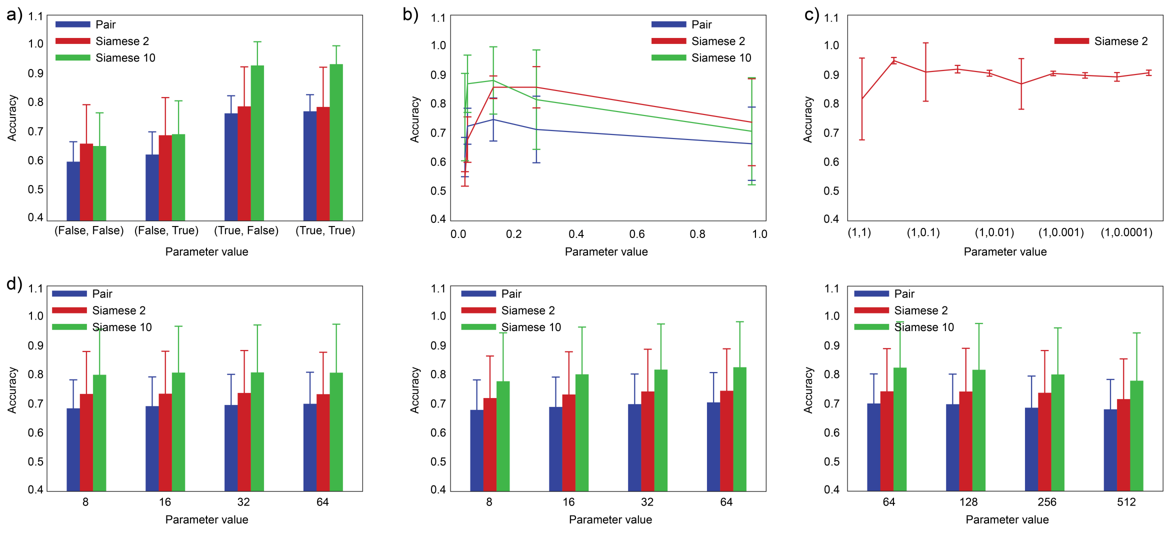

Figure 1: Cross-validation results for the three models. **a)** Effects of batch-normalization and skip connections on paired, siamese with two outputs and siamese with 10 outputs units for all channels and learning rate values. The first boolean value represents batch normalization, the second one skip connections. **b)** Learning rate effects on the three models for all parameter combinations. **c)** Accuracy of the siamese 2 model for different weights of the auxiliary loss. **d)** Effects of layers dimension on the performance accuracy, from left to right the first two convolutions and the last fully connected layer.

To explore the effect of auxiliary losses the siamese models were used first with (siamese 2) and then without (siamese 10) the last fully connected layer. In the second case the final comparison between the two digits has been done by choosing the biggest predicted class of the two inputs digits by a simple max operation.

Cross-validation has been used to estimate the number of channels of the two hidden layers (8, 16, 32, 64), the number of hidden units of the first fully connected layer (64, 128, 256, 512) and the learning rate of the optimizer (0.001, 0.01, 0.1, 0.25 and 1). The same cross-validation has been used to evaluate the effects of the absence-presence of batch normalization and skip connections. Once found the best combination of these parameters, cross-validation has been used to determine the best weight of the auxiliary loss.

Training for the three models has been done using 25 epochs and mini-batches of 100 training samples. Performances are estimated using the accuracy over the test sets in 10 rounds per architecture per combination of parameters. In each round training and test data are randomly selected and the weights randomly initialized.

III. Results

As we can see from Figure 1**a** the presence of batch normalization improves the performance in all the three cases, whereas the presence or not of skip connections is not strongly influencing them. For the siamese 2 the weights for the two losses were 1 and $10^{-0.5}$ respectively for the final layer loss and the auxiliary losses.

Values for learning rates have been cross-validated for the three models and for all the different number of channels (Figure 1**b**). All three models have higher performances for learning rate in the range between 0.1 and 0.25. Higher standard deviations with lower average accuracy for high values of learning rate may suggest bouncing around the minimum during training. Siamese 2 model records a drop in the accuracy for low learning rate that might suggest the learning rate was too small and underfitting was unavoidable.

In Figure 1**c** we cross-validate the weights of the two losses of the siamese 2 model. Best performances are acquired with (1, 0.1). However the most consistently best performer (highest mean and lowest standard deviation) was the loss weight combination we used previously $(1, 10^{-0.5})$. Indeed pushing exagerately with the second weight would improve the single digit recognition to the expense of the final classification.

Figure 1**d** shows the results of cross-validating the number of channels of the convolutional layers and the number of hidden units of the last convolutional layer of the architecture. It appears evident how the dimensions of the layers do not affect the performance of the models.

Figure 2 shows the effect of batch normalization on the gradients of the pair model (the effect was the same also for the siamese network) across training time after each mini-batch. Without batch normalization steps (Figure 2**a**), the gradients at any layer drop to 0 after two mini-batches. Normalization preserved the gradients and the ongoing of the learning during all the mini-batches. This phenomenon may be caused by dead neurons using ReLU activation [10]. The gradients in the first iterations may cause the weights to update in such a way that the neuron will never activate on any data-point again thus the gradient flowing through the unit will be 0 [11]. In Figure 2**b** the batch normalization mainly solves this problem by controlling the statistics of the inputs to the different layers and maintaining them in a range in which ReLU activation can work properly [6].

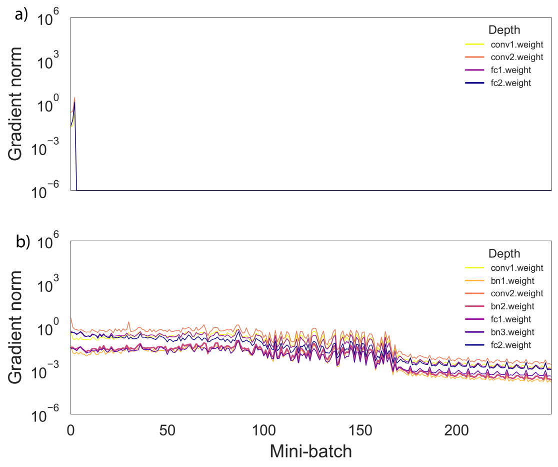

Figure 2: Effects of batch normalization in the pair model. **a)** With batch normalization. **b)** Without batch normalization.

IV. Discussion

We proposed a comparison of two different structures, a simple ConvNet with two convolutions and two fully connected layers with a siamese model obtained by duplicating the same structure for both input images. We compare their results when enriched with batch normalization and skip connections. We then compare their results with the capability of the same ConvNet to recognize each singular digit class. It seems evident from Figure 1**a** that batch normalization is playing an important role for the final accuracy of the two models. Nevertheless, skip connections do not change drastically the results. Indeed, the first technique helps us to solve problem related to dead neurons (Figure 2**a**) and moreover help us control the statistic of the inputs to each layer. Similar results may be obtained with a proper initialization of the weights, like Xavier initialization [12]. Skip connections are a useful tool to control training and test error in deeper networks as proved in [13] but are indeed not effective in a small model.

Siamese 2 appears to improve performance of the pair model but has still lower accuracy than the siamese 10 as seen in Figure 1. The results show the ConvNet is able to easily discriminate the class of the two digits separately but struggles to make the final decision. This can actually be due to the simple structure of the model that uses only a single fully connected layer to perform that choice. Moreover, the cross-validated weights of the losses in Figure 1**c** shows that even in the siamese 2 model the best performances are already better when increasing the weight of the auxiliary losses. Table I shows the best performance obtained with the three models and the combination of parameters used.

model	parameters	accuracy
pair	(64, 64, 64, True, False, 1)	0.83 $\pm$ 0.012
siamese 2	(64, 64, 64, True, False, 0.25)	0.93 $\pm$ 0.004
siamese 10	(64, 64, 64, True, False, 0.25)	0.98 $\pm$ 0.006

Table I: Best model's accuracy.

Finally, we experiment with the number of channels in each convolutional and fully connected layer. The invariance to these numbers may be related to the original small dimension of the images. These results prove that for small inputs and simple networks the actual number is not playing a key role for the accuracy of the model.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," pp. 1097–1105, 2012.

[2] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," pp. 1665–1672, 2011.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," 1985.

[4] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," pp. 1–9, 2015.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[7] 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.

[8] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with mnist and emnist," *Applied Sciences*, vol. 9, no. 15, p. 3169, 2019.

[9] C. I. Watson and C. L. Wilson, "Nist special database 4," *Fingerprint Database, National Institute of Standards and Technology*, vol. 17, no. 77, p. 5, 1992.

[10] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis, "Dying relu and initialization: Theory and numerical examples," *arXiv preprint arXiv:1903.06733*, 2019.

[11] F. Schilling, "The effect of batch normalization on deep convolutional neural networks," 2016.

[12] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," pp. 249–256, 2010.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," pp. 770–778, 2016.