Image Classification: The Impact of Weight Sharing and Auxiliary Losses

Virginia Bordignon
EE-559 Deep Learning, EPFL, Switzerland

***Abstract*—In this report, we study the impact of weight sharing and using auxiliary losses to regularize the image classification problem. We focus on the following classification task: given two images of handwritten digits, the task is to classify whether the first digit is smaller or equal than the second. Different architectures of CNNs are proposed, and their performance evaluated. They show that there is a substantial gain in generalization performance when auxiliary losses are added to the main cost.**

I. Introduction

Training deep models has allowed us to approximate increasingly complex data distributions. The drawback of having models with millions of parameters, is that these, however powerful, are more prone to overfitting [1]. If training data is limited, some of the tools we might use to compensate overfitting is to include regularizer functionals in the formulation of the cost.

Another form of regularization is to use *parameter sharing*. Whenever we expect tasks to be similar across different modules, it is meaningful to explore the concept of parameter sharing, in which the weights of these modules are forced to be the same [2]. This idea is in the heart of the success of CNNs for image applications.

Another way of improving generalization is to use some form of multitask learning [3]. In a typical form of multitask learning, whenever tasks are related in some fundamental sense, parts of the model can be shared across different tasks. The intuition is that solving multiple related tasks, the model translates more sensible associations yielding a better generalization performance.

II. Data and Models

A. Digits Dataset

In this work, we consider a dataset with pairs of grayscale images of dimension 14×14 consisting of 1000 training and 1000 test pairs. To each training pair we are given the binary target (label 0: digit 1 is smaller or equal than digit 2, label 1: digit 1 is greater than digit 2) as well as the individual digits labels (from 0 to 9). We split the training data into a validation set of 100 image pairs, and perform the training using the remaining 900 pairs.

B. Proposed Models

For the binary classification task, we minimize the loss:

$$\mathcal{L}(w) = \mathbb{H}(q(\mathbf{x}; w)) \tag{1}$$

where $\mathbb{H}$ represents the cross-entropy between the data distribution and the model output $q(\mathbf{x}; w)$.

We propose to exploit the following three classifier structures. First, we consider two sets of convolutional layers dedicated to each of the digits (refer to Fig. 1). The output of these layers is flattened, concatenated and serves as input to a final linear layer.

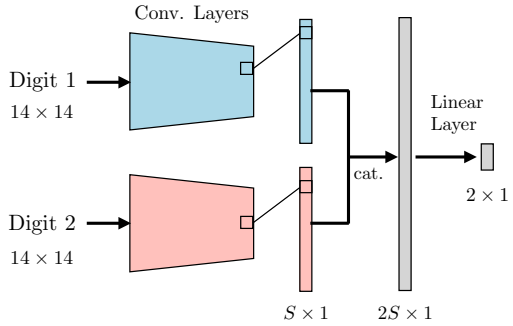

Figure 1. Independent CNN architecture.

The sets of convolutional layers can have one among 6 possible channels architecture: architectures A1, A2, A3, A4, A5, A6. These can be seen in Table I. For example, A4 corresponds to having two convolutional layers with 32 and 64 channels.

Table I
A: Architecture, C: Number of channels per CNN layer.

A	A1	A2	A3	A4	A5	A6
C	[16]	[32]	[16,32]	[32,64]	[16,32,64]	[32,64,128]

In the second structure, we consider that both sets of convolutional layers share weights (refer to Fig. 2). We consider here the same possible architectures in Table I.

Finally, in the third proposed structure, we consider that the output of each of the sets of convolutional layers is fed through a linear layer resulting in a final layer of dimension 10×1 (refer to Fig. 3). The idea is to use these intermediate outputs in the auxiliary losses to be optimized. Under this structure, we seek to minimize the loss is given by:

$$\mathcal{L}(w) = \mathbb{H}(q(\mathbf{x}; w)) + \mathbb{H}_1(q_1(\mathbf{x}_1; w_1)) + \mathbb{H}_2(q_2(\mathbf{x}_2; w_2)) \tag{2}$$

where $q_1(\mathbf{x}_1; w_1)$ and $q_2(\mathbf{x}_2; w_2)$ corresponds to the intermediate outputs given digits 1 and 2, respectively. The

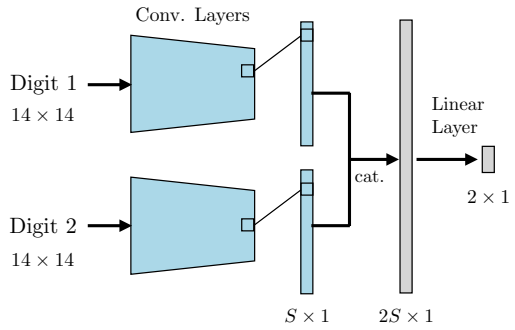

Figure 2. CNN architecture with weight sharing.

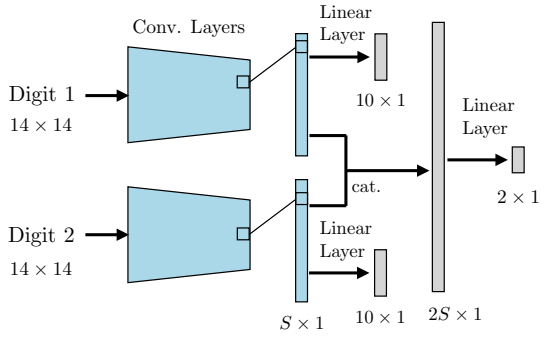

Figure 3. CNN architecture with weight sharing and auxiliary loss.

weights w, w_1, w_2 are partially shared according to the diagram in Fig. 3.

III. Results

We have trained the three aforementioned models using the six architectures described in Table I, each for two different sizes of convolution kernels: 3×3 and 5×5. We used 30 epochs in each case, with initial learning rate 0.01, and 0.001 after 15 epochs. We repeat the training 10 times for each setup.

The training and validation performance for the first structure, with independent convolutional layers, can be seen in Table II.

Table II
Training performance using two independent CNNs. A: CNN Architecture, K: Convolution Kernel

A	K	Train Loss	Val. Loss	Val. Acc. (%)
A1	3	0.0895 ± 0.0348	0.4764 ± 0.1112	80.20 ± 3.74
	5	0.0747 ± 0.0634	$\mathbf{0.4460 \pm 0.0830}$	80.90 ± 3.14
A2	3	0.0578 ± 0.0351	0.5039 ± 0.0948	79.50 ± 4.40
	5	0.0408 ± 0.0250	0.5078 ± 0.1016	79.90 ± 2.85
A3	3	0.0495 ± 0.0255	0.5924 ± 0.0917	78.90 ± 2.51
	5	0.0233 ± 0.0115	0.5283 ± 0.1093	81.00 ± 2.94
A4	3	0.0121 ± 0.0090	0.6108 ± 0.0906	81.20 ± 2.90
	5	0.0034 ± 0.0025	0.5988 ± 0.1069	$\mathbf{82.90 \pm 3.28}$
A5	3	0.0127 ± 0.0144	0.7017 ± 0.1942	80.80 ± 4.39
	5	0.0039 ± 0.0040	0.7473 ± 0.2204	82.10 ± 2.47
A6	3	0.0009 ± 0.0009	0.9548 ± 0.3045	81.50 ± 4.99
	5	0.0006 ± 0.0007	0.8824 ± 0.2810	80.90 ± 4.25

The results are reported as the average and standard deviation of the performance indicators over 10 training repetitions with randomly initialized weights and data.

In Fig. 4, we see the evolution of the validation and training losses and the validation accuracy over training epochs for the setup with best final validation accuracy.

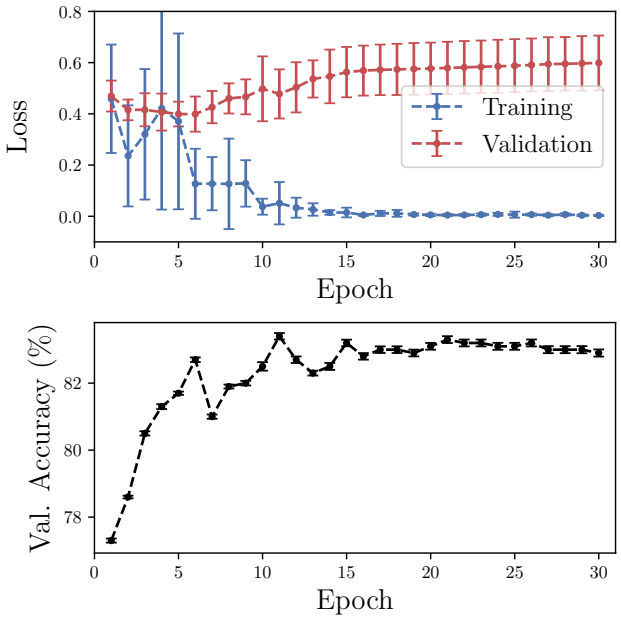

Figure 4. Best training performance with independent CNNs: evolution of training and validation losses and validation accuracy over training epochs.

The training and validation performance for the second structure, with weight sharing across the sets of convolutional layers, can be seen in Table III.

Table III
Training performance using weight sharing. A: CNN Architecture, K: Convolution Kernel

A	K	Train Loss	Val. Loss	Val. Acc. (%)
A1	3	0.1481 ± 0.1239	0.5176 ± 0.1242	79.70 ± 3.13
	5	0.1058 ± 0.1176	$\mathbf{0.4704 \pm 0.0790}$	80.60 ± 2.91
A2	3	0.0967 ± 0.0814	0.5579 ± 0.1200	77.50 ± 4.33
	5	0.0499 ± 0.0330	0.5261 ± 0.1293	79.90 ± 4.18
A3	3	0.0274 ± 0.0147	0.6215 ± 0.1692	82.30 ± 4.52
	5	0.0260 ± 0.0190	0.5918 ± 0.1261	80.50 ± 3.60
A4	3	0.0083 ± 0.0077	0.6818 ± 0.2138	82.80 ± 4.66
	5	0.0056 ± 0.0051	0.6253 ± 0.1305	82.60 ± 5.04
A5	3	0.0050 ± 0.0074	0.7623 ± 0.1263	83.30 ± 2.21
	5	0.0061 ± 0.0059	0.6079 ± 0.0868	$\mathbf{84.60 \pm 2.22}$
A6	3	0.0012 ± 0.0011	0.9685 ± 0.2474	81.30 ± 3.83
	5	0.0010 ± 0.0009	0.8269 ± 0.1872	83.10 ± 2.02

In Fig. 5, we see the evolution of the validation and training losses and the validation accuracy over training epochs for the setup with best final validation accuracy.

In both Figs. 4 and 5, we see a substantial increase in the validation loss after epoch 5 while the training loss reduces,

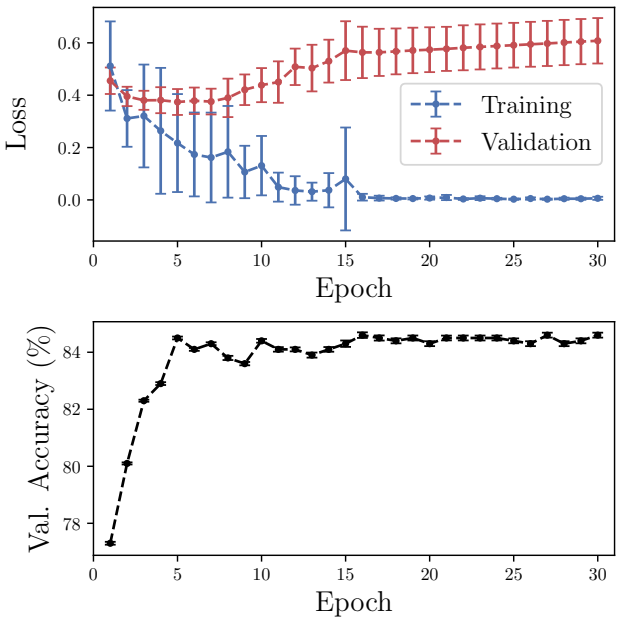

Figure 5. Best training performance with weight sharing: evolution of training and validation losses and validation accuracy over training epochs.

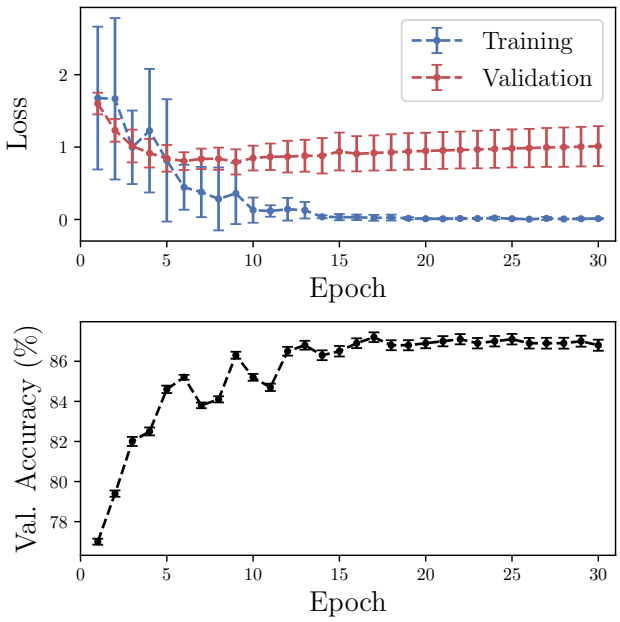

Figure 6. Best training performance with weight sharing and auxiliary losses: evolution of training and validation losses and validation accuracy over training epochs.

which should indicate overfitting of the model. The best performance achieved using the second structure seems to generalize slightly better than the first structure.

The training and validation performance for the third structure, with weight sharing and intermediate outputs used in the multitask loss can be seen in Table III.

Table IV
TRAINING PERFORMANCE USING WEIGHT SHARING AND AUXILIARY LOSSES. A: CNN ARCHITECTURE, K: CONVOLUTION KERNEL

A	K	Train Loss	Val. Loss	Val. Acc. (%)
A1	3	0.1483 ± 0.1041	0.9782 ± 0.1539	79.60 ± 4.65
	5	0.1118 ± 0.0499	0.9414 ± 0.1089	82.70 ± 2.26
A2	3	0.0445 ± 0.0425	1.0992 ± 0.2603	78.40 ± 4.40
	5	0.0852 ± 0.0480	1.0723 ± 0.2348	80.20 ± 4.32
A3	3	0.0645 ± 0.0525	1.0879 ± 0.3943	83.20 ± 2.53
	5	0.0650 ± 0.0577	$\mathbf{0.8759 \pm 0.1659}$	82.10 ± 3.51
A4	3	0.0126 ± 0.0099	1.2369 ± 0.4372	79.50 ± 5.50
	5	0.0230 ± 0.0154	0.9925 ± 0.2163	82.80 ± 2.53
A5	3	0.0118 ± 0.0199	1.1330 ± 0.3197	84.10 ± 1.91
	5	0.0647 ± 0.0445	0.8795 ± 0.2153	84.20 ± 4.96
A6	3	0.0046 ± 0.0036	1.4947 ± 0.2293	82.50 ± 2.80
	5	0.0119 ± 0.0093	1.0130 ± 0.2766	$\mathbf{86.80 \pm 3.52}$

In Fig. 6, we see the evolution of the validation and training losses and the validation accuracy over training epochs for the setup with best final validation accuracy.

The overfitting of the model in Fig. 6 seems to be less pronounced than in the previous models. The accuracy achieved by the best model is significantly improved with respect to the previous strategies.

Finally we compare the test accuracy and losses for the three best trained models depicted in Figs. 4, 5,6. For the first model, we obtain a test loss of 0.3592 ± 0.2742 and test accuracy of $83.49 \pm 0.87\%$; test loss 0.3418 ± 0.2566 and accuracy of $85.46 \pm 0.71\%$ for the second model, and 0.9449 ± 0.6908 and $87.05 \pm 0.70\%$ for the third model.

IV. SUMMARY

In this work, we have showed experimentally the added value of weight sharing in association with using auxiliary loss functions. In the experiments using images of hand-written digits, where the main task was to compare both digits, the best generalization performance was achieved by minimizing a multitask loss with auxiliary cross-entropy terms using a three-convolutional-layered structure. In this strategy, exploring commonality across tasks, weight sharing is used to regularize the optimization problem and improve generalization performance.

REFERENCES

[1] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[2] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[3] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.