Deep Learning - Project 1

Maxime Gardoni, Hila Vardi, Niccolò Stefanini

May 22, 2020

Abstract

The aim of this project is to compare the effects of different architectures on the training and accuracy of a neural network. In particular, the effects of the use of weight sharing and auxiliary losses on the performance are assessed.

I. Introduction

i. Data

The input is comprised of a set of pairs of grey-scale images of hand-written digits (2x14x14 tensors). The networks should predict if the digit in the first image of the pair is lesser or equal to the digit in the second image.

The input dataset is symmetric, meaning that the training set can be doubled by swapping between the pairs of images and flipping the targets. This data augmentation trick is used throughout this project in order to increase the size of the training set. Data were normalized as a preprocessing step for faster learning.

II. Architectures

The two main architectures presented in the report include a network using transfer learning and a Siamese network. In both of the architectures, the digit recognition is done with two convolutional layers and two fully connected layers. The loss criterion for the digit recognition is also kept the same, as cross entropy loss, in order to render the comparison more complete.

A comparison between the two architectures is presented in this report. The analysis of the effects of weight sharing and auxiliary loss is done with the Siamese neural network as it is a more powerful neural network.

The training was done with adam optimizer and a minibatch size of 100 samples.

i. Transfer Learning

Transfer learning is the use of knowledge gained from solving one problem in order to solve a related problem. In the context of this project, a neural network is used to recognise the digits of all the images and the comparison between the pairs of images is done using a logic function.

ii. Siamese Net

A Siamese neural network receives two different input tensors in parallel and produces comparable outputs. As the input in this project is a pair of images, the two parallel input tensors are easily separated into the first and second images in the pair and then compared after the digit recognition is done, as shown in Figure 1.

The comparison between two digits is a linearly separable problem. The choice was made to give all the 10 digits probabilities to the comparison part, which is done using three linear layers.

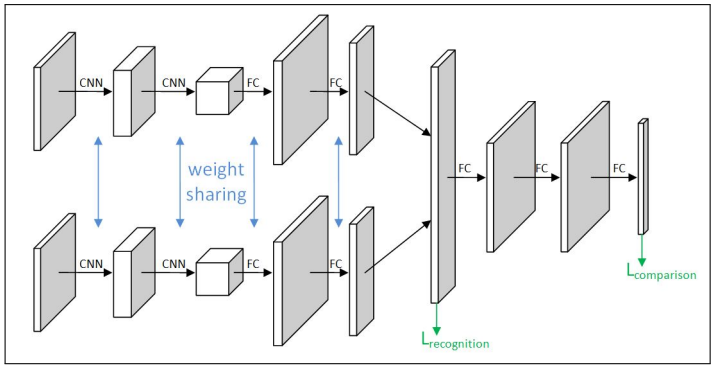

Figure 1: *Siamese Neural Network*

ii.1 Loss

The main loss, $L_{comparison}$, is computed using a binary cross entropy.

An auxiliary loss $L_{recognition}$ was added to the Siamese neural network, taking into account the intermediate error of the digits recognition:

$$L = L_{comparison} + \alpha L_{recognition}$$

The value of α was tuned to 0.5 by grid testing. The criterion for $L_{recognition}$ is Cross Entropy, which performed way better than L2 norm.

iii. Weight Sharing

Weight sharing is implemented in the Siamese neural network by using the same weights for both digit recognition branches, as seen in Figure 1. The main advantage of weight sharing is the reduction of parameters to optimize.

iii.1 Batch Normalisation and Dropout

Batch normalisation and dropout are usually used in order to improve the speed and performance of the neural network. Batch normalisation is done by normalising data between layers. In Dropout, some elements of the input tensor are 'dropped', preventing co-adaptation of feature during training, hence reducing overfitting.

III. Results

Two types of analyses can be done using the different architectures implemented; First is evaluating the performance of the different architectures, given in Table 1. Second is analysing the effects of adding auxiliary losses, weight sharing, batch normalisation and dropout on the training loss and test accuracy, see Figure 2.

The performances of the different architectures are presented in Table 1. These were estimated through 10 rounds, in which the data and the weight initialisation were randomised. The standard deviations over the 10 rounds are also shown.

The accuracy of the transfer learning network is better than that of the basic Siamese Network (with weight sharing), 0.9686 and 0.7790 respectively. This is due to the fact that a logic comparison function that is easily written as a mathematical expression is hard for a a neural network to learn, especially with the small dataset used here. In this project, the transfer learning architecture is shown to be more accurate than the Siamese network. However, in a more complex case where a simple logic function can't be found, the siamese network can prove itself very useful.

Figure 2 presents the effects of auxiliary losses, weight sharing, batch normalisation and dropout on the training loss and the test accuracy. In order to have a common basis for comparison of the loss, we plot only $L_{comparison}$, without the auxiliary component, $\alpha L_{recognition}$, if used. This explains why the yellow loss curve converges quicker; "comparison" is the only objective being optimized. It's however a greedy method, because adding the auxiliary loss helps to have a better final accuracy.

Table 1: *Performance Comparison*

Architecture	Test Accuracy	SD	#Param
TransferLearn	0.9686	0.0067	72′418
Siamese_{NoWS}	0.8074	0.0089	145′737
Siamese_{WS}	0.7790	0.0152	73′299
$\text{Siamese}_{WS,AUX}$	0.8373	0.0220	73′299
$\text{Siamese}_{WS,AUX,BN}$	0.8476	0.0142	73′299
$\text{Siamese}_{WS,AUX,BN,DROP}$	0.8474	0.0115	73′299

With weight sharing, the recognition layers are trained on twice as many training data as without weight sharing. This explains the higher stability with weight sharing. With weight sharing, the loss converges to the minima faster, but by looking at the test accuracy the model seems to overfit.

Selectively adding batch normalisation and dropout were also tested in order to prevent the gradient vanishing and overfitting of the model, resulting in a higher accuracy. It seems to also help the model converge more quickly and to be more stable. The addition of dropout on top of the batch normalisation does not result in a significant improvement, probably because the network used is too shallow to really benefit from it.

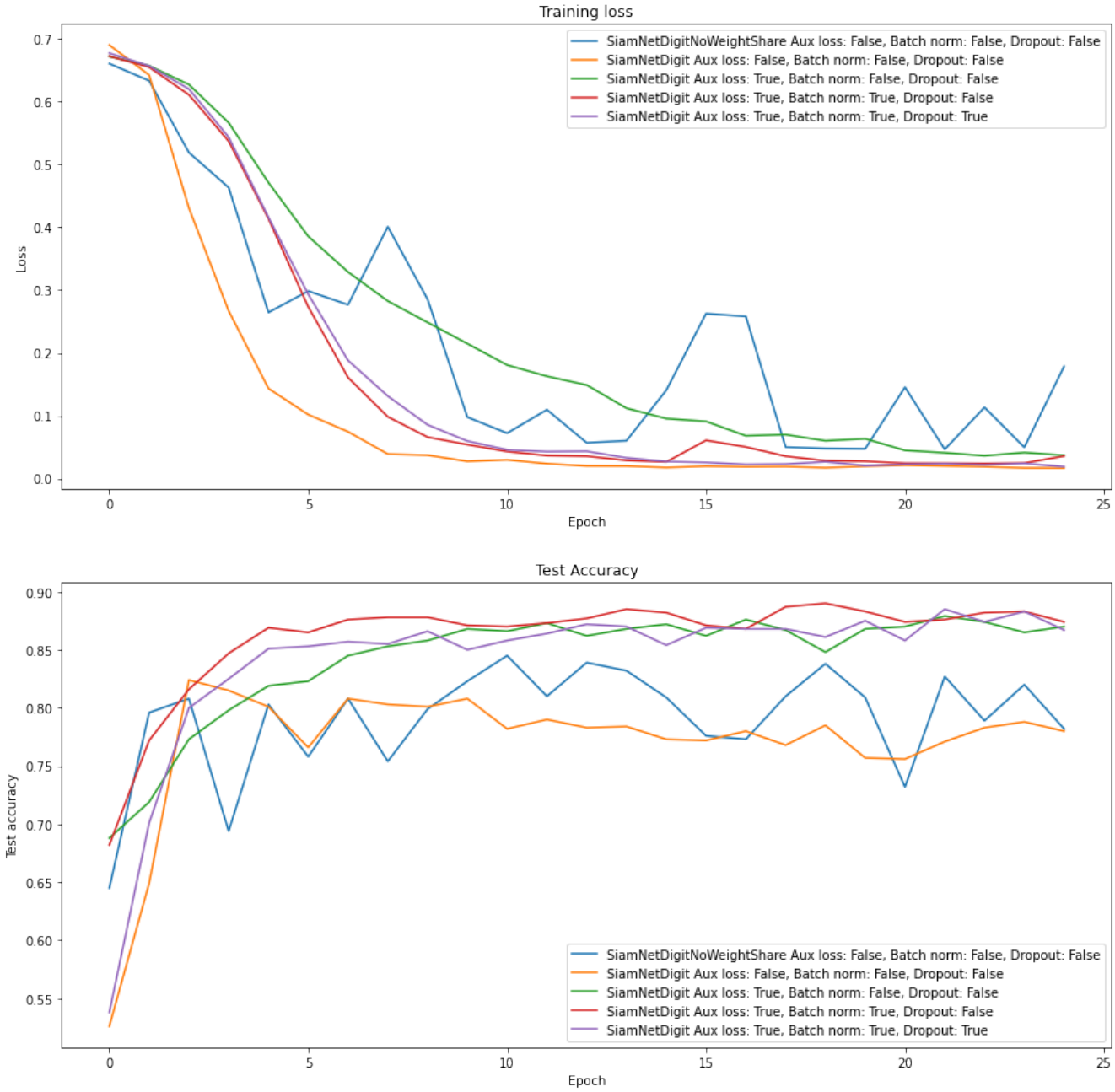

Figure 2: *Train Loss and Test Accuracy*