1 Introduction

The aim of this project is to show the impact of the architecture of a Neural Network (NN) on its performance. To achieve this we trained several different NNs to perform the same prediction task and compare their performances. The predictions are based on the Mnist dataset of handwritten digits, introduced by LeCun et al.[1] in 1998. This dataset has been widely used to assess performance of simple networks. It contains 60k train samples and 10k test samples of 28x28 grayscale images. The goal of our neural network is to compare two handwritten digit images from Mnist and predict which of the two digits is larger. In this report we will often refer to this task as binary classification.

We start with a simple Multi-layer Perceptron to establish a baseline. We then try to improve performance by first constructing a convolutional neural network. Subsequently, we add weight sharing and an auxiliary loss. We also try a transfer learning architecture. We finally try to predict the digits directly and compare the classes predicted.

2 Architectures

2.1 Baseline

The first architecture we try is a simple Multi-Layer Perceptron. We use 5 fully connected layers with rectified linear unit (ReLu) as an activation function.

2.2 CNN

A first improvement is to use convolutional layers and create a convolutional Neural Network(CNN). The architecture is based on the LeNet5 architecture from LeCun et al.[1]. We modify it to work on a 2x14x14 input instead of the 1x28x28 format the original architecture uses. The convolutional layers start from two channels, the image pair, and go to successively 32 and 64 features maps. We use ReLu as an activation function and add a MaxPooling layer after each convolution to reduce sensitivity to local differences between digits.

We follow this by 3 fully connected layers still using ReLu as activation function. This allows to go from a tensor of size 1x256 to 1x2. Where the last two nodes indicate whether the first or the second image is larger.

To avoid over-fitting of the network, Dropout layers with a probability of dropout of 0.3 are added after each activation function.

2.3 Weight Sharing

The naive implementation of the CNN, taking the image pair as two channels in the input can be improved. We create an architecture that starts with each image of the pair independently. This means that we start from a 1x14x14 tensor which is convoluted twice leading to 64 activation maps. Those activation maps are then reduced to a 1x84 tensor by two fully connected layers. At this point we have two 1x84 tensors - one per image - which are then merged to a single 1x168 tensor. This tensor is reduced to the 1x2 classification output by two fully connected layers. This model has one linear layer more than the previous, needed for dimensionality reduction of the merged tensor. This results in an increase in the number of parameters used.

2.4 Auxiliary Loss

The next improvement we try is to make use of the digit classes. We can train the two channels to explicitly learn digit recognition before drawing comparisons. To do this, we introduce as an auxiliary loss the loss of digit recognition. The total loss is the weighted sum of this auxiliary loss and the loss of the comparison, computed as:

$$L = \frac{E-e}{E} L_{\text{auxiliary}} + (1 - \frac{E-e}{E}) L_{\text{target}}$$

where e is the training epoch and E is total number of training epochs. The auxiliary loss will dominate the total loss in the beginning but the comparison loss takes over with time. The network will therefore start by learning to recognize digit and then shift to learning the target of binary prediction.

This architecture is similar to the one of the previous section expect that the tensors resulting from the independent training of both images are used for digit classification as well as merged to produced the binary classification. This implies one extra fully connected layer allowing to reduce the tensors from 84 to 10, the one hot encoded classes prediction.

2.5 Transfer learning

Another way of using the classes information is using transfer learning. Instead of learning both the digit recognition task and the prediction task together, we fully separate both training phases. We use a structure that first learns the digit classification task alone and then reuses the layers for the binary classification. We therefore split the model from the previous section into two parts: we train the digit classification for the first half of the epochs and train the binary classification for the second half.

The architecture for both tasks is the same that was used in the previous section. The second network reuses the parameters of the first up to the last layers.

2.6 Digit Prediction

We finally use the first network of the transfer learning to predict the digit classes and the comparisons are performed outside of the CNN. In this architecture, the network does not have to learn how to compare the digits.

3 Results

All models are trained on the same training set of 1000 image pairs, 500 pairs validation set and 500 pairs test set. They are trained on 100 epochs each. Transfer learning is training both its networks on 50 epochs each. We use Cross-entropy as criterion and Adam with a learning rate of 0.001 as optimizer.

The accuracy for each of the different architectures is shown in Figure 1:

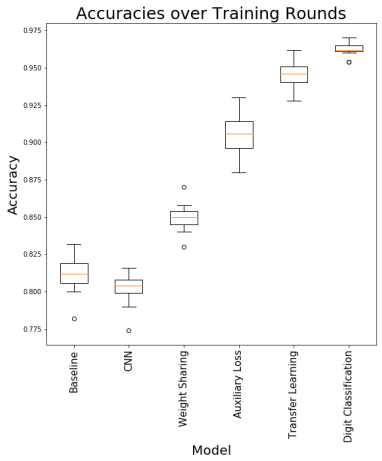

Figure 1: Accuracies of models on 15 training rounds with different weights initialization on same 500 pairs test set

Models	Mean Accuracy	Parameters
Baseline	0.81	146'588
CNN	0.8	60'278
Weight Sharing	0.85	63'242
Auxiliary Loss	0.9	64'092
Transfer Learning	0.94	64'092
Digit Classification	0.96	60'670

3.1 MLP

A basic model such as a MLP is already able to classify the image pairs with a mean accuracy above 80%. This is expected as the Mnist dataset is quite simple. The numbers of parameters used is however quite high.

3.2 CNN

The CNN performs slightly worse than the MLP but is more stable across the training rounds. It also requires only 2/5 of the parameters needed for the MLP.

3.3 Weight Sharing

The weight sharing approach increases the performance of the network to about 85% accuracy. This can be expected as the parameters of the first layers are now trained twice by each pairs.

3.4 Auxiliary Loss

The use of the auxiliary loss brings a drastic improvement to the performance. The added information brought by the classes is a likely reason for this. We can see how the losses for both the digit classification and the binary classification tasks change over time in Figure 2:

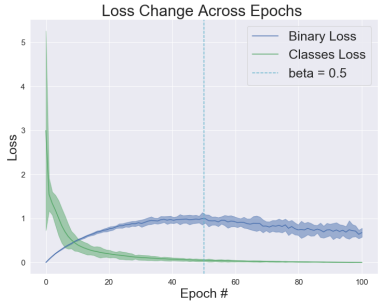

Figure 2: Change of weighted loss of the digit classification and the digit comparison during the training of the Auxiliary Loss model with $\beta = (\text{nb epochs} - e)/\text{nb epochs}$. Binary loss is rescaled for clarity.

The classification loss dominates in the beginning, resulting in an increase of the binary loss. This increase slows down as the model gets better at recognizing digits and the weight of the binary loss grows. When the binary loss starts dominating the total loss ($\beta > 0.5$) it starts decreasing again. The classes loss does not increase after $\beta > 0.5$ indicating that the model is not "forgetting" the digit classification.

3.5 Transfer learning

Separating the digit classification and the binary classification tasks allows the network to perform better and achieve an mean accuracy of 94.5%. This architecture is more closely related to a human way of solving the task: first recognize both numbers and then compare them. This shows that letting the knowledge of the problem guide the architecture improves performance.

3.6 Digit Prediction

Digit prediction outperforms all other architectures. This is expected as the task to learn is easier: this network is not required to learn how to compare the digits. The prediction accuracy is also much more stable across training rounds as seen in Figure 1.

3.7 Error analysis

In an effort to understand and compare the downfalls and advantages of the different architectures, we look

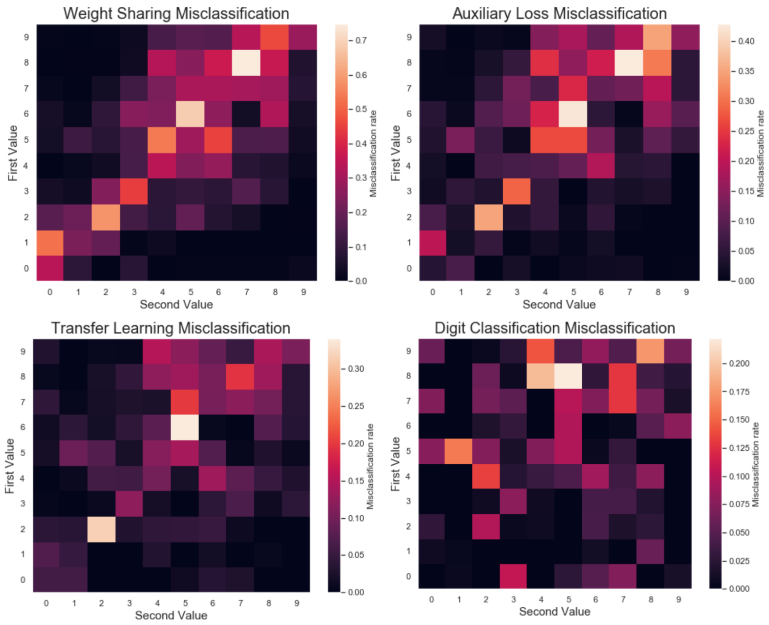

Figure 3: Miss-classification of digit pairs normalized by pair occurrences

at the pairs that are miss-classified for each model. This is shown as heat maps in Figure 3. We do not show the results for the MLP and the basic CNN as they are very similar to those of the weight sharing model and have no added informational value.

Note that the heat maps are non-symmetric across the diagonal since the problem is whether the first digit is smaller or equal than the second one. Note also that not all pairs are equally represented and high miss-classification could also be due to few pair occurrences. Note finally that all models are trained on the same data and comparison of one heat map with another is therefore coherent.

The heat maps of the models trained only on the binary labels (digit comparison) display a clear increase of miss-classification along the diagonal. Prediction on identical number pairs is clearly problematic. We also see that the comparison of close numbers is often wrong. Without further indication other than the binary labels, the networks can at best learn whether a digit is rather high or rather low. This is easier to do for the extremes and is more often right when comparing very different values.

The difficulty of classifying close values or equal pairs limits the predictive power of the models and explains the sudden increase in accuracy when class labels start being used.

The correlation of the miss-classification with the value of the digits dissolves when doing the digit classification task alone. This is expected as this classification is only based on the shape of the digit and is naive to its value.

This correlation still persists in the auxiliary loss model. While it seems better at classifying equal pairs, the classes information did not completely solve the limiting effect of close value comparison.

However, the correlation seems lower on the transfer learning architecture. This could indicate that the model is now learning sequentially: to first recognize a value and then to tell whether this value is higher than another given value.

We see that the equal pairs are still limiting the transfer learning model but since they are edge cases this is expected.

4 Conclusion

The architecture of a network has a clear impact on its performance. A well designed architecture can significantly improve the accuracy of a model. However, we see that crafty architectures such as weight sharing cannot replace the improvement brought by adding more information during training. We also see that an architecture that is driven by the knowledge of the problem is able to perform better. It is also interesting to note that we tried to find a specific solution to a problem that is easily solvable in general. Once the digit are classified, values comparison can be done in a single line. Using the general solution performs better and should be preferred.

References

[1] 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*, 1(4):541–551, 1989.