PROJECT 1: CLASSIFICATION, WEIGHT SHARING, AUXILIARY LOSSES

DENG Zhantao
zhantao.deng@epfl.ch

LI Junze
junze.li@epfl.ch

ABSTRACT

Weight sharing and auxiliary loss are widely used strategies to improve the performance of neural networks. To explore how the two strategies contribute to network performance, in this project, 4 neural networks with different structures and losses are designed to solve the two-digits-comparison problem. Among the 4 networks, a fully connected network and a basic CNN are used as the baseline models. Then, to leverage auxiliary loss and weight sharing, the other 2 networks are constructed. The final evaluation shows that weight sharing and auxiliary loss can bring more than 10% improvement to the network.

1 Introduction

In the two-digits-comparison problem, the inputs are $2 \times 14 \times 14$ tensors, consisting of 2 gray-scale images of hand-written digits from the MNIST dataset [1]. Both the train set and test set contain 1000 image pairs. There are two labels of each image pair:

- **Comparison label**: If the first digit is lesser or equal to the second one, the label of this image pair is 1. Otherwise, the label is 0.
- **Class label**: Indicate the class of each image within the range from 0 to 9.

Base on the input and label structures, we build deep neural networks to compare the two digits presenting in each input tensor. All neural networks are trained on the training set with 25 epochs and then they are tested on the test set to assess the improvement achieved with weigh sharing and/or auxiliary loss.

2 Network structure

We implement 4 different network architectures. The basic fully connected network and basic CNN are used to compare the effect of convolution kernel which shares weight in each kernel. The parallel convolutional network and Siamese network are used to compare the effect of auxiliary loss and weight sharing between sub-networks.

2.1 Basic fully connected network (FconnNet)

The fully connected network flattens the input tensor to a vector and feeds the vector to the 6 linear layers to get the binary prediction, as shown in Figure 1. The cross-entropy loss is used to compute the difference between the predictions and true labels. Batch normalization is implemented after each linear layer in order to accelerate training and decrease overfitting.

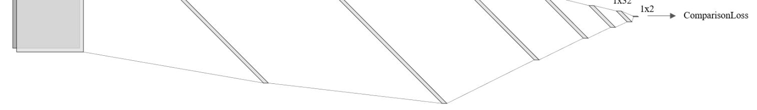

Figure 1: The structure of basic fully connected network.

2.2 Basic CNN (ConvNet)

The convolutional layer is widely used in computer vision tasks as it keeps 2D contextual information and can extract better features than the fully connected layer. The basic CNN consists of 4 convolutional layers to extract features and is then followed by 3 linear layers to do classification. Its loss is cross-entropy loss as well. The detailed network structure is shown in Figure 2.

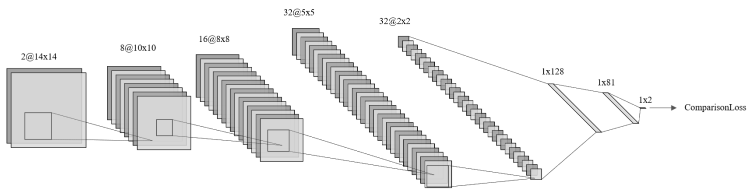

Figure 2: The structure of basic CNN.

2.3 Parallel convolutional network (parallelConv)

Since the class label of each hand-written digit image in the input tensors is provided, it can be used as an auxiliary loss term to boost the basic CNN. To this end, as shown in Figure 3, two sub convolutional networks are implemented to recognize the two digits of input tensors separately and the final linear layer compares the two digits. To make it comparable with the basic CNN, the two sub convolutional networks have the same structure as the basic CNN except for the input size (2× to 1×) and output size (32× to 10×). In the parallel convolutional network, two sub networks are trained separately without weight sharing. The total loss of this architecture consists of the original loss of digit comparison plus cross-entropy loss for the classification of each digit which is also denoted as the auxiliary loss. Comparing to the basic CNN, the parallel convolutional network involves auxiliary loss which could boost the performance. However, as it contains two independent CNN, the computation cost is higher than the basic CNN.

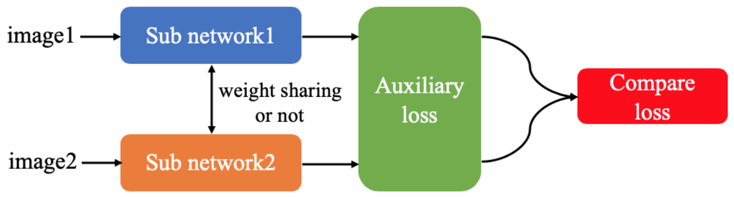

Figure 3: The structure of parallel convolutional network/Siamese network.

2.4 Siamese network

The first Siamese network was used for signature verification on US checks. It verifies whether the signature on the check is consistent with the reserved signature in bank [2]. As it compares two inputs, which is similar to our task, we implement the Siamese network. The structure is same as the parallelConv except for that the sub network 1 and sub network 2 share weights with each other. Therefore, the Siamese network has less parameters and thus has less computation cost than the parallelConv network.

3 Hyper-parameter tuning

Before comparing the 4 networks, grid search is conducted to find the optimal learning rate and batch size for each network. In this way, we could compare the best performance that these networks could achieve. Considering the basic observation of the experiment results and the time complexity, we define the following grid search space.

- **learning rate**: [0.001, 0.005, 0.01, 0.05, 0.1]
- **batch size**: [25, 50, 100, 200]

For each combination of hyper-parameters, we run 10 rounds containing 25 epochs and compare the average accuracy. The optimal hyper-parameters and the corresponding average accuracy are listed in Table 1. The basic CNN is only a bit better than the basic fully connected network which is anti-intuition. This could be because the fully connected network has much more trainable parameters than the basic CNN which means it has more expressive power. In this way, with the same training strategy on this simple task, the fully connected would not perform worse than the basic CNN. Differently, with the help of auxiliary loss and weight sharing, the parallel convolutional network and Siamese network perform much better than the other 2 networks.

Table 1: The hyper-parameter tuning results

	FconnNet	ConvNet	parallelConv	Siamese
number of paras	377,826	37,005	69,886	34,964
learning rate	0.1	0.05	0.01	0.005
batch size	100	25	100	50
accuracy	80.65%	80.87%	88.36%	**91.14%**

4 Auxiliary loss and weight sharing

To understand how the auxiliary loss and weight sharing help the training, we run 20 rounds evaluations for parallelConv and Siamese both with/without auxiliary loss. Figure 4 shows the accuracy curves of the 4 cases on test set. In Table 2, the accuracy and standard deviation are listed.

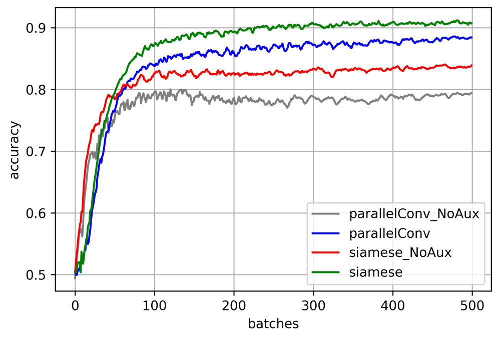

Figure 4: The accuracy curves for different architectures

Table 2: The accuracy and std for different combinations

Technique	Accuracy	Std
weight sharing, auxiliary loss	90.70%	1.62%
weight sharing	83.90%	1.09%
auxiliary loss	88.42%	1.24%
none	79.41%	1.22%

It can be seen that when there is no weight sharing nor auxiliary loss, the accuracy is only $79.41\% \pm 1.22\%$, which is even sightly lower than the accuracy of the basic CNN models. On the contrary, when both weight sharing and auxiliary loss are implemented, the accuracy is $90.70\% \pm 1.62\%$, which is a decent performance. When we consider these two techniques separately, we notice that the auxiliary loss leads to a higher improvement on accuracy than that of the weight sharing. This could be explained as the auxiliary loss guides the learning procedure of the sub networks, making the two sub networks learn similar parameters, which could be achieved by weight sharing.

5 Conclusion

When we compare each image pair directly without using class labels, the FconnNet and ConvNet have similar performances and ConvNet has a slightly higher accuracy. Considering the parallel convolutional network and Siamese network, it shows that both weight sharing and auxiliary loss can improve the performance. The possible reason is that the weight sharing forces the convolutional layers to learn more effective features and the auxiliary loss guides different parts of the network to learn better features. However, there is a trade off between accuracy and standard deviation considering weight sharing and auxiliary loss individually. Overall, the Siamese network could achieve the best performance for our task.

References

[1] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[2] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.