Classification, weight sharing, auxiliary losses

Deep Learning – Project 1

Marie Biolková **Minh Tri Pham**
Department of Computer Science, EPFL, Switzerland
{marie.biolkova, minh.pham}@epfl.ch

***Abstract*—In this report, we assess the effect of weight sharing and/or use of auxiliary losses on model performance on a MNIST-based image comparison task. In addition, batch normalization and dropout were considered to improve learning and generalization. We found that in particular auxiliary losses significantly improve the performance. The highest accuracy was achieved by a modified LeNet with weight sharing, auxiliary loss and batch normalization, which achieved a mean accuracy 92.5 ± 0.8 %.**

I. Introduction

For image classification, the best-performing models on benchmark datasets such as ImageNet [1], but also in practice, are built using deep neural architectures leveraging convolutions to perform feature extraction which is subsequently used as input into a feed-forward network for classification. Such models can have tens or hundreds of billions parameters, making their training challenging. Besides the need for sufficient computational resources, one also needs to ensure the gradients do not vanish as they back-propagate through the numerous layers, and to mitigate the tendency to overfit.

In this report, we investigate different approaches to addressing the above challenges – in particular the use of weight sharing and/or auxiliary losses – on a binary classification task of grayscale images.

II. Methods

The classification task consisted in comparing two digits in a two-channel image. If the digit in the first image is less than (or equal to) the digit in the second image, the model should output 1, and 0 otherwise. To generate our inputs, we used the popular MNIST dataset [2] of handwritten digits from 0 to 9. The original images of size 28×28 pixels were compressed to 14×14 and then random pairs were generated to create a training set and a test set of $1,000$ samples each.

III. Models

In this section we introduce the four different models we explored.

A. Baseline (`BaseNet`)

As a starting point, we chose to construct a modified LeNet [4] which has proven to achieve high accuracies on the MNIST digit classification task. We keep the architecture unchanged, but decrease the kernel sizes of the convolutional and max-pooling layers to suit the input size. The input maintains the original shape $2 \times 24 \times 14$, i.e. the baseline processes both digits at once. The network consists of two convolutional layers with 32 and 64 filers of size 3×3, respectively, followed by 2×2 max-pooling and the ReLU non-linearity. Finally, the pre-processed input goes through two fully connected layers of size 256 and 200 in order to obtain a prediction.

B. Auxiliary Loss (`BaseNetAux`)

Adding auxiliary classifiers to the network's intermediate layers was first suggested by Szegedy et al. (2014) at Google. It aims at encouraging the classifier to develop discriminatory capabilities even in the early layers, improving back-propagation of the gradient signal and increasing the generalization power. More precisely, this method consists in using the output of an intermediate layer as input to a classifier which performs some partial classification task, and then summing up these auxiliary losses with the loss of the network.

The proposed model implements this by splitting the input such that each digit is processed independently and introducing an extra fully connected layer into the baseline model. The output of this layer is a 10-dimensional vector, which is then compared to the true digit class in order to obtain the auxiliary loss. The outputs (one for each digit) are then combined and fed into a final classifier to produce a decision for the main task.

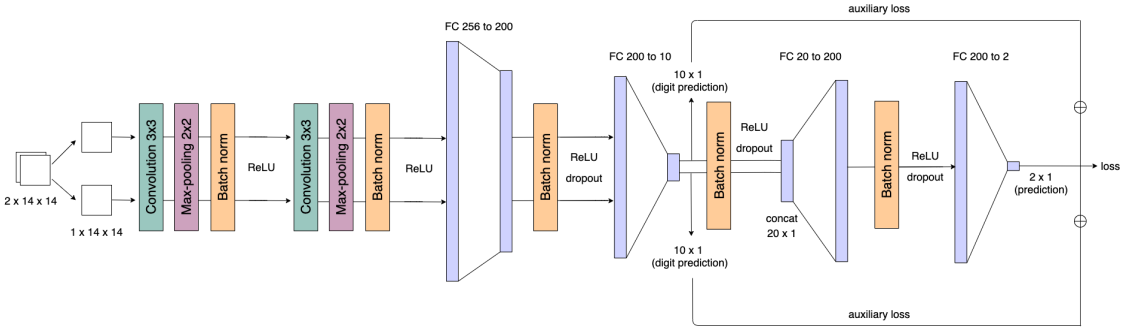

Fig. 1. A diagram of the model with auxiliary loss and weight sharing. The input is split into two single-channel images which go through the preprocessing independently, producing an auxiliary loss which is later added to the main loss. The preprocessed data are concatenated to perform the final digit comparison task.

C. Weight Sharing (`BaseNetWeightShare`)

Weight sharing consists of using the same parameters while processing a pair of signals of the same type in tandem. The idea is that if a processing is good for one element of the pair, it is also good for the other element. Using weight sharing is a way of reducing the model size, so the network can be trained faster. This comes at the price of reduced flexibility, however this is often beneficial as it provides a sort of regularization.

We implement weight sharing by splitting the input into two single-channel just like in the previous model with auxiliary loss, but use the same weights for both sub-inputs.

D. Both auxiliary loss and weight sharing (`BaseNetWeightShareAux`)

Our final model utilises uses both auxiliary loss and weight sharing, as discussed above. Figure 1 shows a diagram of the network.

IV. Batch normalisation and dropout

In addition, we chose to explore the potential benefit of using batch normalisation and dropout in the aforementioned models.

A. Batch normalisation

Batch normalisation was proposed by Ioffe and Szegedy (2015) and consists of re-normalising the activation statistics during the forward pass. It forces the activations' first and second order moments, so that the following layers do not need to adapt to their drift.

In our implementation, each model has a Boolean parameter `batch_normalization` which controls whether the model uses batch normalisation. Batch normalisation layers are placed after each convolutional or linear layer (except the final classification layer).

B. Dropout

Dropout was proposed by Srivastava et al. (2014) and consists of removing units at random during the forward pass on each sample in training. This method increases independence between units, and distributes the representation. It generally improves performance.

In our implementation, each model has a double parameter `dropout` which controls the drop probability for the model. Dropout layers are placed after linear layer, except the final classification layer.

V. Results

All models were trained with stochastic gradient descent for 30 epochs with a batch size of 25 samples, with learning rate $\eta = 0.1$. Datasets were standardised prior to training.

For each model, we tuned the hyperparameters: whether to use batch normalisation and the dropout probability. This was done via a grid search on an initial dataset of $1,000$ samples and choosing the parameter combination that maximises the test accuracy for the $1,000$ test samples. The candidate drop probabilities were 0, 0.1, 0.2, 0.5 and 0.8.

The performance of each model (with its best parameter setting) was evaluated across 10 randomized runs. The results are shown in Table I.

In all four models, using batch normalisation was beneficial, however the best dropout probability was zero for the model with auxiliary loss and the model with weight sharing only.

TABLE I
MEAN TEST ACCURACY AND STANDARD DEVIATION OF EACH MODEL, COMPUTED ACROSS 10 RUNS.

Model	Batch norm.	Dropout p	Accuracy (%)
`BaseNet`	Yes	0.5	83.0 ± 1.2
`BaseNetAux`	Yes	0.0	92.3 ± 0.8
`BaseNetWeightShare`	Yes	0.0	85.9 ± 1.5
`BaseNetWeightShareAux`	Yes	0.2	**92.5 ± 0.8**

Although all the models with auxiliary loss and/or weight sharing outperformed the baseline, it appears that introducing an auxiliary loss helped more than weight sharing (BaseNetAux gave a 9% increase in accuracy compared to BaseNet, whereas BaseNetWeightShare yielded a 3% increase in accuracy compared to BaseNet).

The best result was achieved with combining both an auxiliary loss and weight sharing. This yielded average test accuracy of 92.5 ± 0.8 %.

VI. CONCLUSION

In this project, we examined whether weight sharing and/or the use of an auxiliary loss, as well as the use of batch normalisation and/or dropout would help improve the perfomance of a LeNet-like model on a MNIST-based image comparison task. Weight sharing, batch normalisation and especially introducing an auxiliary loss were shown to be improve the performance while dropout did not seem to have much of an effect.

REFERENCES

[1] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *In 2009 IEEE conference on computer vision and pattern recognition*, (pp. 248–255).

[2] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6), 141–142.

[3] Ioffe, S., Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *In Proceedings of ICML*, (pp.448-456).

[4] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, & L. D. Jackel (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541-551.

[5] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929-1958.

[6] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 1-9).