Stochastic Gradient Descent Training of Ensembles of DT-CNN Classifiers for Digit Recognition

Christian Merkwirth * Maciej Ogorzałek † Jörg Wichard †

Abstract — **We show how to train Discrete Time Cellular Neural Networks (DT-CNN) successfully by backpropagation to perform pattern recognition on a data set of handwritten digits. By using concepts and techniques from Machine Learning, we can outperform Support Vector Machines (SVM) on this problem.**

1 INTRODUCTION

The local processing paradigm of the Cellular Neural Network mimics the processing of visual perception in biological systems and should therefore be well suited for pattern recognition tasks ([1]).

In the Neural Network community, a closely related processing structure is known as a *Convolutional Neural Network with shared weights* (see [2]). There this structure was used to construct the best performing classifier (LeNet 5, see [2]) for digit recognition that even outperforms the best performing SVM on the MNIST data set[1]. Readers should not get confused by the similar abbreviation (CNN) for both network types.

Another approach that has close parallels to the CNN paradigm is the Neocognitron ([3]). It has been used successfully to perform letter classification tasks with backpropagation style learning.

In this report, we work on the smaller ZIP code data set derived from the MNIST data base. By training several DT-CNNs with a stochastic gradient descent algorithm, we can reach better classification performance on the ZIP code test set than by using SVMs with polynomial kernel (see [4]).

The CNN *template weights* of the individual classifiers are determined by using a straightforward backpropagation scheme. We employ an online learning setting, which means that after one observation is processed, a tiny gradient step is performed. In order to reach a high generalization performance, we had to use several recent developments and insights from the field of Machine Learning:

- ϵ-insensitive absolute loss instead of mean quadratic loss. This brings the backpropagation training of the CNN template weights closer to the optimization criterion used for constructing SVM classifiers (see [5]).
- Ensembling (classifier averaging). By averaging the output of several individually trained classifiers, one can improve the generalization performance of neural network classifiers significantly (see [6], [7]).

2 THE DT-CNN CLASSIFIER

2.1 Topology

The states of the DT-CNN cells y_{ij}^t evolve for iterations $t = 0, \ldots, T-1$ according to:

$$\begin{aligned} x_{ij}^{t+1} &= \sum_{l,m=-\frac{K-1}{2}}^{\frac{K-1}{2}} (A_{l,m}^t y_{i+l,j+m}^t \\ &\quad + B_{l,m}^t u_{i+l,j+m}) + b^t \\ y_{ij}^{t+1} &= \sigma(x_{ij}^{t+1}) \end{aligned} \tag{1}$$

At each iteration t, we use different template weights (A^t, B^t and offsets b^t). The total number n of adjustable template weights used for one DT-CNN can be calculated by $N = T(2K^2+1) - K^2$, where T is the number of *template layers* and K is the number of rows of the quadratic template matrices. For $K = 5$, this results in 51 weights per template layer. A^0 is not used since $y_{ij}^0 = 0$ by initialization. The number of template weights does not depend on the spatial size of the CNN receptive field, which is 16 by 16 corresponding to the size of the input patterns.

Elements outside the boundary are treated as zero which is identical to the value of background pixels (see also section 3.1). Roughly speaking, one iteration of the DT-CNN corresponds to one layer of a Convolutional Neural Network.

The main difference to a Convolutional Neural Network lies in the presentation of input patterns. While a input patterns is presented to a Convolutional Neural Network only once at the input layer, for DT-CNNs the input pattern u_{ij} is presented to the network at every iteration t by means of the B template in equation (1).

*Computational Biology and Applied Algorithmics Group, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, e-mail: cmerk@mpi-sb.mpg.de, tel.: +49 681 9325 318

†Department of Electrical Engineering, University of Mining and Metallurgy, al. Mickiewicza 30, 30-059 Kraków, Poland, e-mail: maciej@agh.edu.pl, JoergWichard@web.de tel.: +48 12 6172890, fax: +48 12 6344825.

[1]The MNIST data base can be obtained from http://yann.lecun.com/exdb/mnist/

2.2 Deriving the Decision Variable

Since a (DT-)CNN is a spatial processing engine, we have to convert the spatial output y_{ij}^T into one output value z suitable as decision variable. This is accomplished by simply averaging these 256 states y_{ij}^T.

3 DESCRIPTION OF THE DATA SET

3.1 ZIP Code Data Set

The ZIP Code Data Set[2] consists of normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations; the images here have been deslanted and size normalized, resulting in 16 by 16 grayscale images (see [2]). There are 7291 training observations and 2007 test observations, distributed as follows:

Digit	0	1	2	3	4
Train	1194	1005	731	658	652
Test	359	264	198	166	200
Digit	5	6	7	8	9
Train	556	664	645	542	644
Test	160	170	147	166	177

Table 1: Distribution of classes in the ZIP Code data sets

The 2009 samples of the ZIP Code test set are neither used for training, nor to derive stopping criteria nor to do model selection for the ensembles of classifiers.

The patterns are given to the training algorithm as input/output pairs with inputs being 16 by 16 matrices containing gray levels coded from 0.0 (background pixels) to 1.0 (foreground pixels). Outputs are class labels out of 0,1,...,9.

3.2 Handling Multi-Class Problems

This multiclass classification problem (10 classes corresponding to digits $0, 1, \ldots, 9$) is attacked by converting it into 10 binary classification problems and solving these independently (one against all approach, cmp [8]). All ten classifiers are trained on the same 7291 input patterns. The desired outputs presented to the first classifier are $+1$ for observations that have class label 0, otherwise -1. For the second classifier the desired outputs are $+1$ for observations with class label 1, and so on.

To classify a new input pattern, it is presented to all ten trained classifiers and the class label belonging to the classifier with highest output value z is chosen as final output label.

4 TRAINING

4.1 Stochastic Gradient Descent

This variant of training a Neural Network with the well known backpropagation algorithm is also known as online learning. In contrast to batch learning, where gradients are accumulated over all samples of the data set for the same weight setting before a gradient step is performed, a tiny gradient step[3] is done here after each observations has been processed. For every epoch[4] the order in which the samples of the training data set are processed is randomized. In all simulations below, the number of training epochs is fixed to 50.

A detailed discussion of the advantages of stochastic gradient descent over batch learning and how to compute the gradient for networks with shared weights can be found in [2].

4.2 Loss Function

Using an ϵ-insensitive absolute loss (see figure 1) with $\epsilon = 0.9$ has two important aspects:

- It mitigates the function estimation problem since positive (negative) training observations for that the DT-CNN output is between 0.1 and 1.9 (-0.1 and -1.9) have zero loss and gradient. This behavior is desired since patterns are classified correctly as soon as the CNN output value is positive (negative). This mimics the SVM concept where observations lying on the correct side of the separating hyperplane do not influence the positioning of that hyperplane. There the hyperplane is determined only by observations that lie on the margin or violate it. These observations are called support vectors (cmp [5]).

- Training patterns that can't be correctly classified have only a linear contribution to the loss. This allows easier tolerating of outliers in the data set than with quadratic error.

Using a smaller ϵ degrades the classification performance!

[2]The ZIP Code data set can be obtained from `http://www-stat-class.stanford.edu/~tibs/ElemStatLearn`

[3]The global stepsize $\mu = 0.0007$ is exponentially decreased every 5 epochs by a factor of 0.8.

[4]One complete pass of the algorithm through all training samples is called *epoch*.

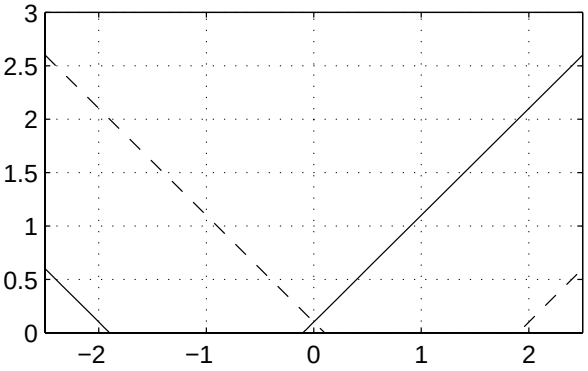

Figure 1: ϵ-insensitive loss function used for the training of the DT-CNNs with $\epsilon = 0.9$. Training observations with desired label $+1$ (-1) have nonzero error when the CNN output is outside the ϵ-insensitive margin given by the dashed (solid) lines. Though the derivative of this function is not continuous, it can be seamlessly used with the backpropagation method.

4.3 Activation Function

The activation function $\sigma(x)$ is a piecewise linear activation function that does no fully saturate for input values x far from 0 (see figure 2). This feature enhances the ability of the backpropagation algorithm to escape from solutions where a neuron has such high activation that the derivative of a sigmoid activation function nearly vanishes (cmp [9]).

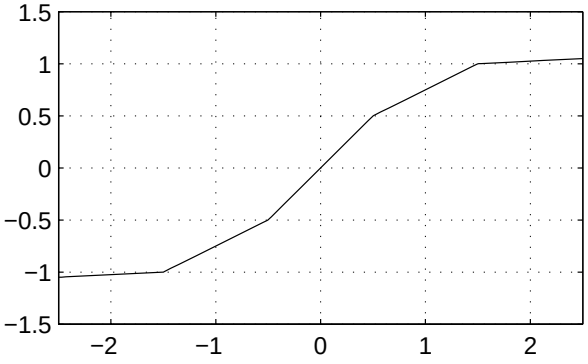

Figure 2: Piecewise linear activation function used as nonlinearity. Note that the function does not saturate at ± 1, instead slope decreases to 0.01. Though the derivative of this activation function is not well defined in a mathematical sense, this does not adversely affect the backpropagation algorithm.

4.4 Shrinking

A reduction in training time can be achieved by using a shrinking heuristics. Here we postpone the evaluation of training observations that are easy to classify and try to concentrate on harder to classify samples. In detail this means that if a training observation is repeatedly inside the eps-insensitive interval of the loss function and therefore it's error and gradient vanishes, it's frequency of calculation is decreased according to the following rules:

- For each observation store a counter that tells how many epochs to wait until this observation is evaluated again. In every epoch, decrease this counter by 1. Negative counters are set to zero. All counters are initialized to zero.
- Evaluate an observation if it's counter is zero.
- If an observation is inside the ϵ-insensitive margin, increase it's counter. If this observation is first time inside the ϵ-insensitive margin, increase it's counter by 1, second time by 2 ...
- If an observation is outside the ϵ-insensitive margin, reset it's counter to 0 and the next increase to 1.

4.5 Classifier Ensembling

We employed a simple *classifier averaging strategy*:

1. Several (6-16) DT-CNNs are trained independently on randomly chosen subsets of the training set. Each subset consists of 80% of all training observations, without any duplicates.
2. Training of the DT-CNNs starts from different random template weight initializations.
3. After training, the DT-CNNs are evaluated on the full training set. DT-CNNs with lowest error on the full training set are selected to join the final ensemble.
4. To compute the output of the ensemble for one input pattern, the output decision variables of all DT-CNNs belonging to that ensemble are averaged. The averaging is not done on the template weights!

Please note that we have to construct one ensemble for each of the ten binary classification tasks.

5 RESULTS

5.1 Classification Performance

Computational demand did not allow to perform a systematic check of all possible combinations of

topological parameters K and T. So the template size K employed during the simulations (see table 2) was fixed to 5. With standard nearest-neighbor coupling $K = 3$ we couldn't achieve an adequate classification performance, for template size $K = 7$ we could observe a beginning gap between training and test errors (overfitting).

From table 2 one can observe that with T from 1 to 10 layers, classification rate increases significantly. At higher number of template layers the rate saturates around 96.5%. For comparison, we constructed a polynomial SVM classifier of degree 4 with $C = 100$ (see [4]) on the ZIP Code training set and applied it to the test set.

T	1	2	3	4
Class.rate	79.7%	91.6%	94.9%	95.9%
T	5	6	8	10
Class.rate	96.0%	96.1 %	95.6%	96.5%
T	14	18	24	SVM
Test	96.6%	96.2%	96.6%	95.4%

Table 2: Classification rates of final classifiers on the ZIP Code test set versus number of template layers T. Note that uniform random guessing would yield a trivial classification rate of 10%.

5.2 Interpretation of Templates

Unfortunately, the interpretation of the processing steps performed by the so determined templates is difficult and not straightforward. The solutions obtained from training several classifiers on the same binary classification task tend to be diverse, which might be caused by the existence of multiple local minima in the error landscape. This is common for neural networks. However, the existence of suboptimal and nonunique solutions is not a major obstacle for achieving good classification performance, instead it can even be utilized for constructing ensembles of decorrelated classifiers with increased generalization ability compared to single classifiers (see [6], [7]).

6 CONCLUSIONS

This article discusses a backpropagation type algorithm for training CNN classifiers for digit recognition. The results on a database of handwritten digits outperform a polynomial Support Vector classifier. An important aspect here is the usage of an alternative loss function during the training process instead of quadratic error.

A main drawback of the method is the higher computational effort for training ensembles of DT-CNN classifiers than constructing SVM classifiers on the same task. Despite of that, the authors hope that a high speed hardware implementation can be realized based on the templates determined by the proposed method.

Acknowledgments

The authors would like to thank the inventor of the libSVM, Chih-Jen Lin, and Jochen Bröcker for stimulating comments and discussions. Parts of this work are supported by the Research Training Network *COSYC of SENS* No. HPRN-CT-2000-00158 within the 5th Framework Program of the EU. We would like to thank the people of the Akademia Górniczo–Hutnicza in Kraków for their support.

References

[1] L.O. Chua, T. Roska, "The CNN Paradigm", IEEE Trans. Circ. Syst., part I, 40, 147–156, 1993

[2] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition",Proceedings of the IEEE 86, 11, 2278–2324, 1998

[3] K. Fukushima, N. Wake, "Handwritten alphanumeric character recognition by the neocognitron", IEEE Transactions on Neural Networks, 2[3], pp. 355–365, 1991

[4] Chih-Chung Chang, Chih-Jen Lin, "LIBSVM : a library for support vector machines", Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.

[5] V. Vapnik, The Nature of Statistical Learning Theory, Springer, 1999

[6] M.P. Peronne, L.N. Cooper, "When networks disagree: Ensemble methods for neural networks", Neural Networks for Speech and Image Processing, Chapman Hall, 1993

[7] J. Krogh, A. Vedelsby, "Neural Network Ensembles, Cross Validation and Active Learning", Advances in Neural Information Processing Systems 7, MIT Press, 1995

[8] C. Hsu, C. Lin, "A comparison of methods for multi-class support vector machines", Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001

[9] Y. LeCun, L. Bottou, G. Orr, K. Müller, "Efficient BackProp" in Neural Networks: Tricks of the trade, Springer, 1998