LOCALLY RECURRENT NEURAL NETWORKS FOR EFFICIENT REALIZATION OF A SPEECH RECOGNIZER

Klaus Kasper, Herbert Reininger, Dietrich Wolf, and Harald Wüst
Institut für Angewandte Physik
Johann Wolfgang Goethe-Universität
60054 Frankfurt am Main, FRG
wuest@apx00.physik.uni-frankfurt.de

ABSTRACT

The computational complexity of speech recognizers based on fully connected recurrent neural networks, i.e. the large number of connections, prevents a hardware realization. We introduced locally connected recurrent neural networks in order to keep the properties of recurrent neural networks and to reduce the connectivity density of the network. A special form of feature presentation and output coding is developed which reduces the computational complexity and allows learning of long-term dependencies.

By applying all these methods a locally recurrent neural network results, which has only one third of the weights as a fully connected recurrent network. Thus, with this concept a speech recognition system can be realized on a single VLSI-Chip.

1 INTRODUCTION

In former work [1, 2] it has been shown that recurrent neural networks (RNN) are powerful tools for speech recognition. A speech recognizer system (SRS) which is based on a single RNN achieves a recognition performance for isolated words which outperforms SRS based on sophisticated Hidden Markov Models. Actually, the computational complexity of the SRS, e.g. the large number of connections, prevents a VLSI realization.

In order to take into account a realizable hardware complexity we introduced locally connected recurrent neural networks (LRNN). A LRNN consists of an input layer, a recurrent hidden layer and an output layer. The inter-layer connectivity is unidirectional and sparse. The connectivity structure of the hidden layer is regular and local, and therefore well suited for hardware realization.

In the following a detailed definition of LRNN is given and simulation experiments concerning the optimization of a LRNN for speaker independent recognition of a 23 word vocabulary are reported.

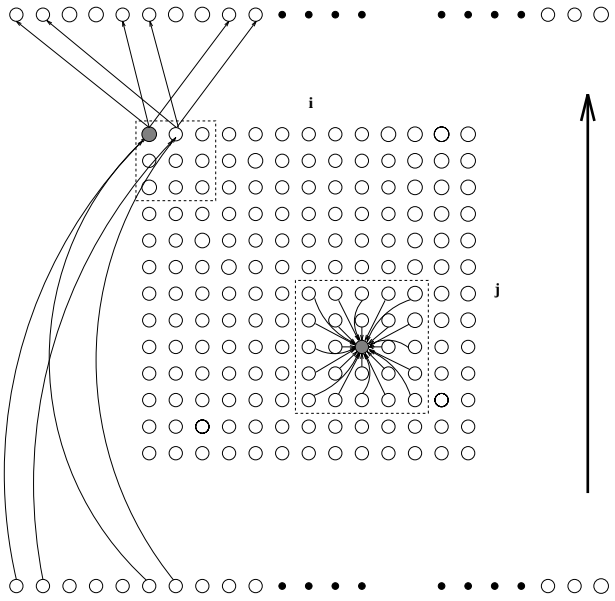

Figure 1: LRNN where neurons of the hidden layer are connected to their $n = 2$ nearest neighbours.

2 DEFFINITION OF LOCALLY RECURRENT NEURAL NETWORKS

LRNN are layered non-fixpoint networks mapping time functions at the input on to time functions at the output. A LRNN consists of an input layer, a hidden layer, and an output layer. The interactions between the input and the hidden layer as well as between the hidden and the output layer are unidirectional and sparse. The hidden layer consists of neurons arranged on a 2-dimensional grid. The neurons of the hidden layer are locally connected to their n nearest neighbours and to a subset of the neurons belonging to the input or output layer. The recurrent connections of the hidden neurons are ending at the edges of the grid and selfloops do not occur. Figure 1 shows an exemplary LRNN with 169 neurons in the hidden layer placed on a 13×13 grid with connections to their $n = 2$ nearest neighbours. The activity x of the hidden neuron ij at time $t + 1$ can be calculated

as

$$x_{ij}(t+1) = \frac{1}{1+\exp(-h_{ij}(t+1))}$$

$$h_{ij}(t+1) = \sum_{k=i-n}^{i+n} \sum_{l=j-n}^{j+n} w_{ij,kl} x_{kl}(t) + \sum_{m \in \mathcal{I}} w_{ij,m} x_m(t) + T_{ij}$$

with $h_{ij}(t+1)$ denoting the activation of neuron ij at time t, $\mathbf{W} = \{\mathrm{w_{ij,kl}}\}$ the weight-matrix, T_{ij} the threshold of neuron ij and $\mathcal{I}$ the indices of all input neurons which have connections to neuron ij. LRNN are trained by truncated back-propagation through time (BPTT) [3].

The definition of the LRNN topology results in a regular structure and leads to an only linear growth of the number of weights with increasing number of neurons instead of the quadratic growth for RNN. Especially, the regular structure is suitable for hardware realizations and could not be achieved by unsupervised pruning methods.

3 SPEECH DATA

The vocabulary of the data base used in the evaluation of the recognition capability of LRNN consists of the 10 German digits, the word *zwo*, and 12 telephone command words. The speech signals were limited to telephone bandwidth and sampled with 8kHz. From these signals feature vectors were extracted every 12 ms, each consisting of 12 cepstral coefficients derived from LPC parameters.

For the computing of the SRS parameters, feature vectors from 100 utterances of each word spoken from different speakers, were used. Speaker independent recognition rates were measured on a set containing 100 utterances of each word from speakers not included in the training set. The networks considered in the experiments were trained with BPTT. The error back-propagation was initiated every 30 time steps and considers the 40 preceding time steps. 400 training epoches were sufficient for all LRNN. In recognition mode the activities of the output neurons were accumulated in order to generate word hypothesises.

4 LEARNING OF LONG-TERM DEPENDENCIES

RNN have in principle the capability to learn long-term dependancies but as experiments have shown, without special modifications of a RNN, only dependencies in the range of about 15 time steps could be learned by gradient-descent learning [4]. This is not sufficient for word recognition becouse with feature vectors derived from the speech signal every 12 ms one word consists of about 50 feature vectors. Information from all of these feature vectors must be taken into account in order to reach a high recognition rate.

A recurrent network structure optimized for learning long-term dependencies is proposed in [5]. In this network the delayed values of the output neurons within the interesting time span are used as additional input patterns to overcome the problem of learning long-term dependencies. In that way, the recognition task may be managed but the number of necessary input neurons is increased dramatically.

The increase in complexity could be avoided by taking into account that dependencies in a short range could be learned. Only information concerning time distances which can't be captured have to be provided as additional input. This could be achieved by only feeding back the activities from time instances $(T-\Delta\tau, T-2\cdot\Delta\tau, ..., T-N\cdot\Delta\tau)$. The delay $\Delta\tau$ can be at maximum 15 in order to stay in the learnable time span. Dependencies with time distances longer than 15 requires feed back of additional activity values. An appropriat value of N can be determined by the time span of the dependencies and $\Delta\tau$. By this means, the number of additional input neurons is kept small and all the needed information for learning long-term dependencies is available.

Instead of feeding back output activities it is also possible to use the activities of some arbitrarily selected hidden neurons due to the recurrent connections in the hidden layer. This allows the optimization of the number of feedback neurons independently from the number of output neurons and the target function.

Another method to overcome the problem of learning long-term dependencies is to combine N_b consecutive feature vectors to a super-vector

$$\underline{X}(t') = (\underline{x}(N_b \cdot t'), \underline{x}(N_b \cdot t' + 1)), ..., \underline{x}(N_b \cdot t' + N_b - 1)).$$

The number of necessary network calculations per feature sequence is reduced by the factor of N_b. By appropriate choice of N_b it is therefore possible to bring the length of a sequence of super-vectors to a value below 15. Thus the necessary information could be exploited without special modifications of the network.

The different approaches to solve the problem of learning long-term dependencieswere evaluated in simulation experiments. Figure 2 shows the resulting recognition rates (R) as function of N_b for LRNN with feedback of hidden activity values (HF-LRNN). As can be seen, for $N_b = 1$, which means a single feature vector as input, LRNN achieves only a poor recognition rate. Feeding back $N = 1$ vector of 40 hidden activity values with a delay of $\Delta\tau = 12$ the recognition rate R increases to 80%. With $N = 2$ the HF-LRNN achieves with $R = 97\%$ a high recognition rate for this task and uses 80 additional input neurons. By raising the super-vector size to $N_b = 2$, HF-LRNN with $N = 1$ reaches also the rate of $R = 97\%$. The performance of LRNN with super-vectors at the input increasing continously and with $N_b = 5$ the recognition rate of $R = 97\%$ is

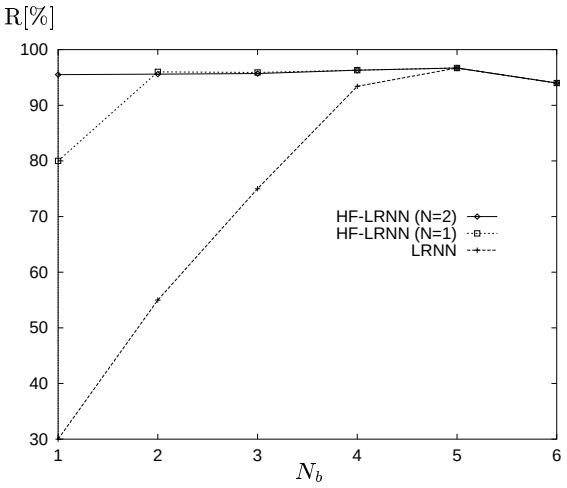

Figure 2: Recognition rates (R) of LRNN networks with different super-vector sizes (N_b).

obtained. Therefore the input layer consists of 60 input neurons and the length of the super-vector sequence is reduced to about 12. Although, the number of input neurons for processing super-vectors is nearly the same as for the HF-LRNN this approach has still the advantage that the computational load is reduced by a factor of 5.

5 REDUCING THE NUMBER OF OUTPUT NEURONS

In most neural classification tasks one output neuron represents one class in a (1 out of N) coding. The classification result could be found in this case by a simple maximum search throughout the activities of the output neurons. The disadvantage of this is that the number N_0 of output neurons, and in particular, the number of connection weights to these neurons, increase linearly with the number of classes. With a binary coding of the class numbers the linear increase can be avoided but the calculation of the classification result becomes difficult due to the need of a threshold value.

As a solution to this problem we introduce a (m out of N) coding which allows to represent

$$N_c = \frac{N_0!}{m!(N_0 - m)!} \tag{1}$$

classes with N_0 output neurons. If m is choosen as $N_0/2$ the maximum number of possible classes with the smallest number of necessary output neurons could be achieved. So, with 7 output neurons 35 classes could be recognized which leads to a reduction of the number of output neurons by a factor of 5 in comparison to the (1 out of N) coding. In order to get the classification result the maximum search must be slightly modified to find not only the best but the m-best output neurons. Simulation experiments have shown that the recognition performance is not influenced by the use of the (m out of N) coding. In the case of recognizing the 23 word vocabulary this reduces the number of connections from the hidden to the output layer by more than 70%.

Table 1: Amount of weights (W_Σ) and word recognition rates (R) for different neighbourhood width (n).

n	W_Σ	$R(\%)$
0	11499	70.6
1	12699	88.3
2	14811	93.9
3	17571	95.1
4	20739	96.4
5	24099	97.1

6 MINIMIZING THE AMOUNT OF WEIGHTS IN LRNN

In first experiments the neighbourhood width n of the neurons in the hidden layer of a LRNN was optimized to achieve the same recognition rate as a RNN but with minimum number of weights. The RNN with 169 hidden neurons reaches a recognition rate of 97.1% and has 40480 connections.

Table 1 shows the resulting recognition rates and the amount of weights for LRNN with 13x13 neurons in the hidden layer, 60 input neurons for storing a super-vector with $N_b = 5$ and 7 output neurons. A LRNN with n=0, which has no recurrent connections and is therefore equal to a multilayer perceptron, achieves a rather low recognition rate of 70%. Introducing recurrent connections to the nearest neighbours, LRNN with n=1, leads to a significant increase of 18% in recognition rate. Obviously, recurrent connections in the hidden layer are substantial for exploiting dynamical dependencies of the feature vectors. With increasing neighbourhood width the recognition performance increase until n=5 where the rate of the fully connected RNN is reached. Thereby, the LRNN consist of only one half of the number of connections than the fully connected RNN.

Additionally, the maximum reduction of the connection density between the input and the hidden layer as well as between the hidden and the output layer was determined. Table 2 shows the recognition rates for the LRNN with n=5 at different reduction factors r which denotes the reduction of the inter-layer connections in respect to the number of connections of fully connected layers. It could be seen that reductions up to $r = 1/4$ are possible without a loss in recognition performance. In total, the number of weights could be reduced to 15431 which is nearly one third of the weights of the baseline

Table 2: Amount of weights (W_Σ) and word recognition rates (R) for different reduction factors (r).

r	W_Σ	$R(\%)$
1/1	23036	97.1
1/2	17966	97.0
1/4	15431	97.0
1/5	14586	96.2

RNN.

7 CONCLUSIONS

With the concept of LRNN it is possible to reduce the complexity of recurrent neural networks without affecting the recognition performance in a speaker independent recognition task of 23 words. By merging several feature vectors together to build a super-vector the learning of long-term dependencies becomes possible and the computational complexity is reduced by a factor of up to five. With a (m out of N) coding the number of output neurons could be reduced. The inter-layer connectivity density could be reduced to one fourth and the connectivity in the hidden layer could be restricted to the $n = 5$ nearest neighbours.

By applying all these methods a LRNN results, which has only one third of the weigths as a RNN but the same recognition capability. LRNN is therefore an efficient concept for realizing SRS based on recurrent neural networks. Currently, a cooperation with hardware experts started in order to realize a SRS on the basis of LRNN on a single VLSI-chip.

This work is supported by the Deutsche Forschungsgemeinschaft in the research program "System- und Schaltungstechnik für hochgradige Parallelverarbeitung".

References

[1] K. Kasper, H. Reininger, D. Wolf, and H. Wüst. A monolithic speech recognizer based on fully recurrent neural networks. In *Neural Networks for Signal Processing IV*, pages 335–344, Ermioni, 1994.

[2] K. Kasper, H. Reininger, and H. Wüst. Strategies for reducing the complexity of a rnn based speech recognizer. In *Proc. IEEE Conf. on Acoust., Speech, and Signal Processing*, Atlanta, 1996. in print.

[3] R.J. Williams and J. Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2:490–501, 1990.

[4] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. on Neural Networks*, 5:157–166, 1994.

[5] T. Lin, B. Horne, P. Tino, and L. Giles. Learning long-term dependencies is not as difficult with narx recurrent neural networks. Technical Report UMIACS-TR-95-78, University of Maryland, 1995.