A Novel Approach to On-Line Handwriting Recognition Based on Bidirectional Long Short-Term Memory Networks

Marcus Liwicki[1] Alex Graves[2] Horst Bunke[1]
Jürgen Schmidhuber[2,3]

[1] Inst. of Computer Science and Applied Mathematics,
University of Bern, Neubrückstr. 10, 3012 Bern, Switzerland
[2] IDSIA, Galleria 2, 6928 Manno-Lugano, Switzerland
[3] TU Munich, Boltzmannstr. 3, 85748 Garching, Munich, Germany

Abstract

In this paper we introduce a new connectionist approach to on-line handwriting recognition and address in particular the problem of recognizing handwritten whiteboard notes. The approach uses a bidirectional recurrent neural network with long short-term memory blocks. We use a recently introduced objective function, known as Connectionist Temporal Classification (CTC), that directly trains the network to label unsegmented sequence data. Our new system achieves a word recognition rate of $74.0\,\%$*, compared with* $65.4\,\%$ *using a previously developed HMM-based recognition system.*

1. Introduction

Although the problem of handwriting recognition has been considered for more than 30 years [1, 12, 16], there are still many open issues, especially in the task of unconstrained handwritten sentence recognition. Handwriting recognition is traditionally divided into on-line and off-line recognition. In on-line recognition a time ordered sequence of coordinates, representing the movement of the tip of pen, is captured, while in the off-line mode only the image of the text is available.

In this paper we consider an on-line recognition problem, namely the recognition of notes written on a whiteboard. This is a relatively new task. As people stand, rather than sit, during writing and the arm does not rest on a table, handwriting rendered on a whiteboard is different from handwriting produced with a pen on a writing tablet. Despite some additional difficulty, the whiteboard modality is important in several applications, such as the documentation of lectures or meetings. In the particular application underlying this paper we aim at developing a handwriting recognition system to be used in a smart meeting room scenario [17], in our case the smart meeting room developed in the IM2 project [11]. Smart meeting rooms usually have multiple acquisition devices, such as microphones, cameras, electronic tablets, and a whiteboard. In order to allow for indexing and browsing [18], automatic transcription of the recorded data is needed.

In this paper, we introduce a novel approach to on-line handwriting recognition, using a single recurrent neural network (RNN) to transcribe the data. The key innovation is a recently introduced RNN objective function known as Connectionist Temporal Classification (CTC) [5]. Whereas previous objective functions only use RNNs to label individual data points within a sequence, CTC uses the network to label the entire input sequence at once. This means the network can be trained with unsegmented input data (an important requirement for on-line handwriting, where correct segmentation of individual letters is often difficult to achieve), and the final label sequence (in this case, the character level transcription) is given directly by the network output.

In our writer independent experiments on the IAM-OnDB [9][1], a word recognition rate of up to $74.0\,\%$ has been achieved. These results are significantly higher then those from previous experiments with an HMM-based system [10].

The rest of the paper is organized as follows. Section 2 gives an overview of the proposed system. In Section 3 the main steps for preprocessing the data and extracting the features are presented. Section 4 introduces the new classification approach for handwriting recognition. Experiments and results are presented in Section 5, and finally Section 6 draws some conclusions and gives an outlook to future work.

[1] http://www.iam.unibe.ch/~fki/iamondb/

Figure 1. Illustration of the recording

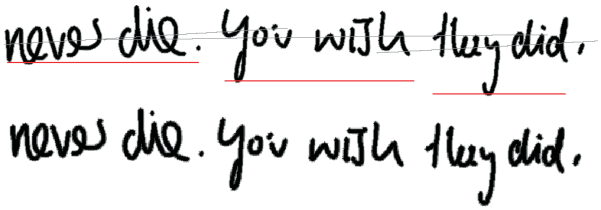

Figure 2. Splitting a text line into subparts and skew correction

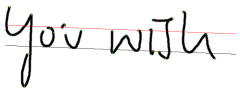

Figure 3. Baseline and corpus line of an example part of a text line

2. System Overview

The eBeam interface[2] is used for recording the handwriting. It allows the user to write on a whiteboard with a normal pen in a special casing, which sends infrared signals to a triangular receiver mounted in one of the corners of the whiteboard. The acquisition interface outputs a sequence of (x,y)-coordinates representing the location of the tip of the pen together with a time stamp for each location. The frame rate of the recordings varies from 30 to 70 frames per second. An illustration is shown in Fig. 1.

The system described in this paper consists of three main modules: the on-line preprocessing, where noise in the raw data is reduced and the text line is normalized with respect to skew, slant, width and height; the feature extraction, where the sequence of points is transformed into a sequence of feature vectors; and the recognition, where an ASCII transcription of the handwriting is generated.

3. Preprocessing

Before feature extraction can be applied, the recorded data has to be normalized. This is a very important step in handwriting recognition systems, because the styles of the writers differ with respect to skew, slant, height and width of the characters. If we do not apply any preprocessing and use the raw features, the recognition rate is significantly lower. The preprocessing steps applied in the current system have been introduced in [10], but for the purpose of completeness, we give a short overview below.

The recorded on-line data usually contain noisy points and gaps within strokes, which are caused by loss of data. Hence, we apply some noise filtering operations first. The cleaned text data is then automatically divided into lines using some simple heuristics. As the skew often significantly varies within the same line, we split lines into subparts. An example of splitting is shown in Fig. 2 (upper line).

Next the subparts are corrected with respect to their skew using a linear regression. This process is illustrated in Fig. 2 with the resulting text line shown in the lower part. For slant normalization, we compute the histogram over all angles between the lines connecting two successive points of the trajectory and the horizontal line [8]. Subsequently, the histogram is processed to recover the skew angle. After these operations, we remove delayed stokes, e.g. the crossing of a "t" or the dot of an "i", using simple heuristics. The next important step is the computation of the baseline and the corpus line by computing two linear regression lines through the minima and maxima of the y-coordinates of the strokes. Figure 3 illustrates the estimated baseline and the corpus line of part of the example shown in Fig. 2. The baseline is subtracted from all y-coordinates to make it equal to the x-axis. As the last preprocessing step, the width of the characters is normalized. First, the number of characters is estimated as a fraction of the number of strokes crossing the horizontal line between the baseline and the corpus line. The text is then horizontally scaled according to this value.

The set of extracted features can be divided into two classes. The first class consists of features extracted for each point p_i considering the neighbors of p_i with respect to time. The second class takes the off-line matrix representation of the handwriting into account.

The features of the first class are the following:

- *pen-up/pen-down:* a boolean variable indicating whether or not the pen-tip touches the board.
- *hat-feature:* this feature indicates if a delayed stroke

[2] eBeam System by Luidia, Inc. - www.e-Beam.com

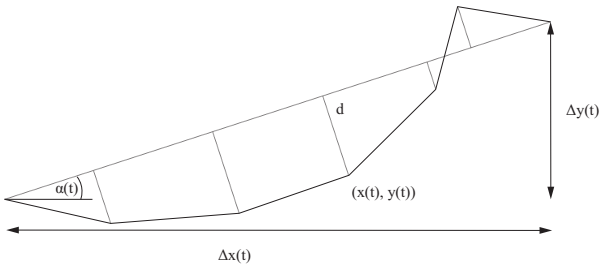

Figure 4. Features of the vicinity

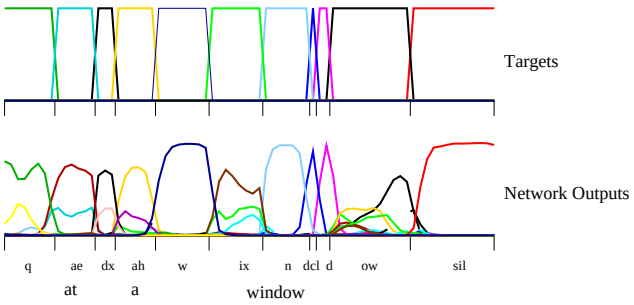

Figure 5. Individual timestep classification with a standard RNN

was removed at the considered horizontal position.

- *speed:* the velocity at point p_i is computed before re-sampling and then interpolated.
- x*-coordinate:* the x-position of point p_i is taken after high-pass filtering (subtracting a moving average)
- y*-coordinate:* the vertical position after normalization.
- *writing direction:* the cosine and sine of the angle between the line segment starting at p_i and the x-axis.
- *curvature:* the cosine and sine of the angle between the lines to the previous and the next point.
- *vicinity aspect:* the aspect of the trajectory (see Fig. 4): $(\Delta y(t) - \Delta x(t)) \ / \ (\Delta y(t) + \Delta x(t))$
- *vicinity slope:* cosine and sine of the angle $\alpha(t)$ of the straight line from the first to the last vicinity point.
- *vicinity curliness:* the length of the trajectory in the vicinity divided by $\max(\Delta x(t), \Delta y(t))$.
- *vicinity linearity:* the average square distance d^2 of each point in the vicinity to the straight line from the first to the last vicinity point.

The features of the second class are all computed using a two-dimensional matrix representing the off-line version of the data. The following features are used:

- *ascenders/descenders:* the number of points above/below the corpus in the x-vicinity of p_i.
- *context map:* the two-dimensional vicinity of p_i is transformed to a 3×3 map. The resulting nine values are taken as features.

4. The New Approach

Recurrent neural networks (RNNs) are a natural choice for on-line handwriting recognition, since they are able to access a wide range of context when transcribing letters, words or word sequences. Until recently however, RNNs were limited to making separate classifications at every timestep in an input sequence. This severely limited their applicability to domains such as cursive handwriting recognition, since it required that the training data be pre-segmented, and that the network outputs be post-processed to give the final transcriptions. Figure 5 shows an example output sequence for a standard RNN as it is used in speech recognition. The network has one output neuron for each phoneme. Note that the targets, i.e. the phonemes to be recognized, are pre-segmented.

Connectionist Temporal Classification (CTC) [2, 5] is an RNN objective function designed to overcome the above problem. It uses the network to define a probability distribution over a fixed set of labels plus an additional 'blank', or 'no label' unit. It then interprets the sequence of network outputs as a probability distribution over all possible transcriptions for a given input sequence, and trains the network by maximizing the log probabilities of the correct transcriptions on the training set. Figure 6 illustrates how CTC transcribes an on-line handwriting sample. At the bottom there is an image of a handwritten input word. Above the input, three on-line feature streams representing x-position, y-position and time, together with a fourth binary feature indicating the end of a stroke, are shown. Above the feature stream, the outputs of the neurons in the hidden layer are given. Finally, he network output is shown in the uppermost layer in Figure 6.

As mentioned above, the effectiveness of RNNs lies in their ability to access contextual information. It is therefore important to choose an RNN architecture that maximizes the amount of context available.

Long Short-Term Memory (LSTM) [4, 7] is an RNN architecture specifically designed to bridge long time delays between relevant input and target events, making it suitable for problems (such as handwriting recognition) where long range context is required to disambiguate individual labels.

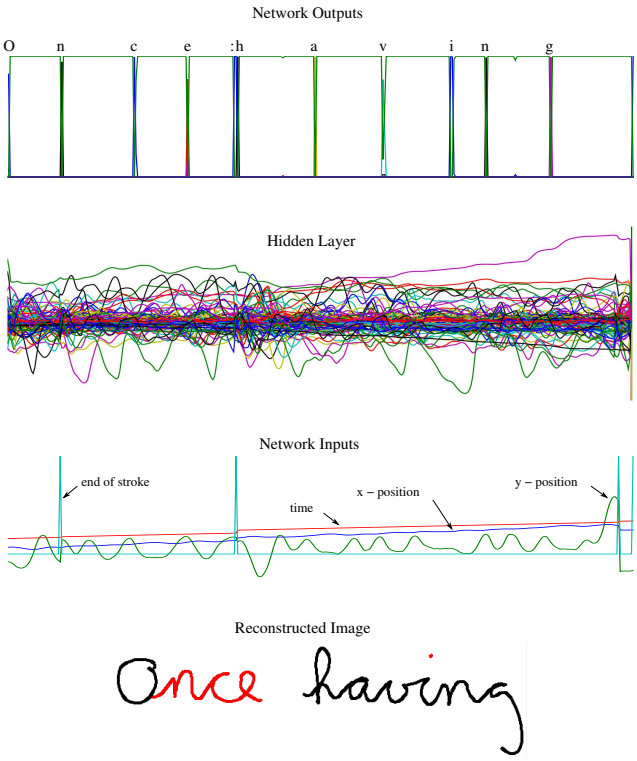

Figure 6. Online handwriting recognition with CTC (with four simple input features)

An LSTM layer consists of a set of recurrently connected blocks, known as memory blocks. Each one contains one or more recurrently connected memory cells and three multiplicative units, namely the input, output and forget gates that provide write, read and reset operations on the cells. More precisely, the input to the cells is multiplied by the activation of the input gate, the output to the net is multiplied by the activation of the output gate, and the previous cell values are multiplied by the forget gate. The net can only interact with the cells via the gates. The cells are extended by peephole connections [3] that allow them to inspect their current internal states.

Whereas standard RNNs make use of previous context only, bidirectional RNNs (BRNNs) [14] are able to incorporate context on both sides of every position in the input sequence. This is useful in handwriting recognition, since it is often necessary to look to both the right and left of a given letter in order to identify it. Figure 7 shows a comparison of a standard RNN and a BRNN in two time states. In the BRNN there exist two hidden layers, i.e. one layer for each direction, and the input/output neurons (marked with I/O) are connected to both.

Combining the above two concepts gives bidirectional

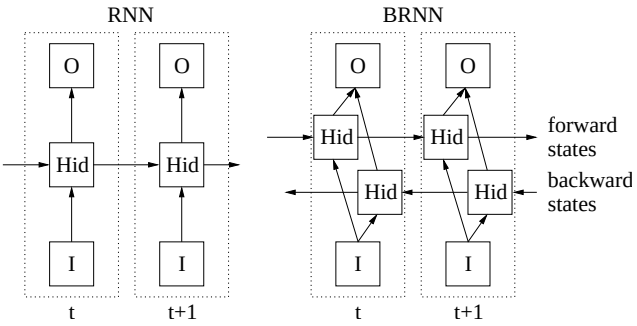

Figure 7. RNN and BRNN (unfolded in time)

LSTM (BLSTM), which has been shown to outperform other neural network architectures in framewise phoneme recognition [6]. In the experiments described below, we use a BLSTM network with a CTC output layer to transcribe samples of handwritten text.

5. Experiments and Results

For our experiments we used the IAM-OnDB, a large online handwriting database acquired from a whiteboard [9]. This database contains 86,272 word instances from a 11,050 word dictionary written down in 13,040 text lines.

We used the sets of the benchmark task with the open vocabulary IAM-OnDB-t2[3]. There the data is divided into four sets: one set for training; one set for validating the meta parameters of the training; a second validation set which can be used, for example, for optimizing a language model; and an independent test set. No writer appears in more than one set. Thus, a writer independent recognition task is considered. The size of the general vocabulary is $20,000$. Please note that the perfect accuracy would be $94.4\,\%$ because the remaining words of the test set are not present in the vocabulary. In our experiments, we did not include a language model. Thus the second validation set has not been used.

The CTC system used a BLSTM [6] network with 100 extended LSTM memory blocks in each of the forward and backward layers. Each memory block contained one memory cell, an input gate, an output gate, a forget gate and three peephole connections. The tanh activation function was used for the block input and output squashing functions, while the gate activation function was the logistic sigmoid.

The input layer was size 25 (one input for each feature), and the CTC output layer had one output for each character, plus one for 'blank'. The input layer was fully connected to both hidden layers, and these were fully connected to themselves and to the output layer.

[3] see benchmark task on http://www.iam.unibe.ch/~fki/iamondb/

The network was trained with online gradient descent with momentum, using a learning rate of $1e^{-4}$ and a momentum of 0.9. Training was stopped when performance ceased to improve on the validation set. The results are quoted as an average over four runs, $\pm$ the standard error. At the start of each run, the network was initialized with a Gaussian distribution of mean 0 and std. deviation 0.1. Note that because of this random initialization we run the experiments four times in contrast to the reference system, where no random initialization takes place (see below).

The CTC network was trained to identify individual characters. In the recognition phase, an adapted version of the token passing algorithm [15] was used to find the most probable sequence of complete words, given the dictionary.

The recognizer of [10] was used as a reference system. It applies the same preprocessing and feature extraction steps as described in Section 3. It is based on Hidden Markov Models (HMMs). For more details we refer to [10].

System	Accuracy
HMM	65.4 %
CTC	74.0 % ($\pm$ 0.3 %)

Table 1. Results on IAM-OnDB-t2 benchmark

Table 1 shows the results of the CTC approach compared to the HMM-based system. The recognition accuracy of $74.0\,\%$ is a significant improvement. The accuracy is calculated using the following formula:

$$acc = 1 - \frac{\#insertions + \#substitutions + \#deletions}{\#words\_in\_transcription}$$

6. Conclusions and Future Work

In this paper we described a novel approach for recognizing on-line handwritten text on a whiteboard, using a single recurrent neural network (RNN). The key innovation is a recently introduced RNN objective function known as Connectionist Temporal Classification (CTC). CTC uses the network to label the entire input sequence at once. This means the network can be trained with unsegmented input data, and the final label sequence can be read directly from the network output. In our experiments we have achieved a word recognition rate of $74.0\,\%$ which is significantly higher than the recognition rate of the HMM-based system.

In future we plan to develop a strategy for including a statistical n-gram language model in the RNN [13]. It is reasonable to integrate a statistical language model, since we are performing handwritten text recognition on text lines and not only on single words. We also plan to overcome the problem of Out-Of-Vocabulary words (OOV). This could be done by using the network likelihood and the edit distance to the nearest vocabulary word.

References

[1] H. Bunke. Recognition of cursive roman handwriting - past present and future. In *Proc. 7th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 448–459, 2003.

[2] S. Fernández, A. Graves, and J. Schmidhuber. Sequence labelling in structured domains with hierarchical recurrent neural networks. In *Proc. 20th Int. Joint Conf. on Artificial Intelligence, IJCAI 2007*, pages 774–779, 2007.

[3] F. Gers and J. Schmidhuber. Recurrent nets that time and count. In *Proc. IEEE-INNS-ENNS International Joint Conf. on Neural Networks*, volume 3, pages 189–194, 2000.

[4] F. Gers, N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143, 2002.

[5] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proc. Int. Conf. on Machine Learning*, pages 369–376, 2006.

[6] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[7] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *NC*, 9(8):1735–1780, 1997.

[8] S. Jaeger, S. Manke, J. Reichert, and A. Waibel. Online handwriting recognition: the NPen++ recognizer. *Int. Journal on Document Analysis and Recognition*, 3(3):169–180, 2001.

[9] M. Liwicki and H. Bunke. IAM-OnDB - an on-line English sentence database acquired from handwritten text on a whiteboard. In *Proc. 8th Int. Conf. on Document Analysis and Recognition*, volume 2, pages 956–961, 2005.

[10] M. Liwicki and H. Bunke. HMM-based on-line recognition of handwritten whiteboard notes. In *Proc. 10th Int. Workshop on Frontiers in Handwriting Recognition*, pages 595–599, 2006.

[11] D. Moore. The IDIAP smart meeting room. Technical report, IDIAP-Com, 2002.

[12] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):63–84, 2000.

[13] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? In *Proc. IEEE 88*, volume 88, pages 1270–1278, 2000.

[14] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45:2673–2681, November 1997.

[15] Y. SJ, R. NH, and T. JHS. Token passing: A simple conceptual model for connected speech recognition systems. Technical report, Cambridge University Engineering Dept, 1989.

[16] A. Vinciarelli. A survey on off-line cursive script recognition. *Pattern Recognition*, 35(7):1433–1446, 2002.

[17] A. Waibel, T. Schultz, M. Bett, R. Malkin, I. Rogina, R. Stiefelhagen, and J. Yang. Smart: The smart meeting room task at ISL. In *Proc. IEEE ICASSP*, volume 4, pages 752–755, 2003.

[18] P. Wellner, M. Flynn, and M. Guillemot. Browsing recorded meetings with Ferret. In *Machine Learning for Multimodal Interaction*, pages 12–21, 2004.