Offline Arabic Handwriting Recognition with Multidimensional Recurrent Neural Networks

Alex Graves

Offline handwriting recognition is usually performed by first extracting a sequence of features from the image, then using either a hidden Markov model (HMM) [9] or an HMM / neural network hybrid [10] to transcribe the features.

However a system trained directly on pixel data has several potential advantages. One is that defining input features suitable for an HMM requires considerable time and expertise. Furthermore, the features must be redesigned for every different alphabet. In contrast, a system trained on raw images can be applied with equal ease to, for example, Arabic and English. Another potential benefit is that using raw data allows the visual and sequential aspects of handwriting recognition to be learned together, rather than treated as two separate problems. This kind of 'end-to-end' training is often beneficial for machine learning algorithms, since it allows them more freedom to adapt to the task [13].

Furthermore, recent results suggest that recurrent neural networks (RNNs) may be preferable to HMMs for sequence labelling tasks such as speech [5] and online handwriting recognition [6]. One possible reason for this is that RNNs are trained discriminatively, whereas HMMs are generative. Although generative approaches offer more insight into the data, discriminative methods tend to perform better at tasks such as classification and labelling, at least when large amounts of data are available [15]. Indeed much work has been in recent years to introduce discriminative training to HMMs [11]. Another important difference is that RNNs, unlike HMMs, do not assume successive data points to be conditionally independent given some discrete internal state, which is often unrealistic for cursive handwriting.

This chapter will describe an offline handwriting recognition system based on recurrent neural networks. The system is trained directly on raw images, with no manual feature extraction. It won several prizes at the 2009 International Conference on Document Analysis and Recognition, including first place in the offline Arabic handwriting recognition competition [14].

Alex Graves
Technical University of Munich, Germany, e-mail: graves@in.tum.de

The system was an extended version of a method used for online handwriting recognition from raw pen trajectories [6]. The *Long Short-Term Memory* (LSTM) network architecture [8, 3] was chosen for its ability to access long-range context, and the *Connectionist Temporal Classification* [5] output layer allowed the network to transcribe the data with no prior segmentation.

Applying RNNs to offline handwriting is more challenging, since the input is no longer one-dimensional. A naive approach would be to present the images to the network one vertical line at a time, thereby transforming them into 1D sequences. However such a system would be unable to handle distortions along the vertical axis; for example the same image shifted up by one pixel would appear completely different. A more robust method is offered by *multidimensional recurrent neural networks* (MDRNNs) [7]. MDRNNs, which are a special case of directed acyclic graph networks [1], generalise standard RNNs by providing recurrent connections along all spatio-temporal dimensions present in the data. These connections make MDRNNs robust to local distortions along any combination of input dimensions (e.g. image rotations and shears, which mix vertical and horizontal displacements) and allow them to model multidimensional context in a flexible way. We use multidimensional LSTM [7] because it is able to access long-range context along both input directions.

The problem remains, however, of how to transform two-dimensional images into one-dimensional label sequences. The solution presented here is to pass the data through a hierarchy of MDRNN layers, with subsampling windows applied after each level. The heights of the windows are chosen to incrementally collapse the 2D images onto 1D sequences, which can then be labelled by the output layer. Hierarchical structures are common in computer vision [17], because they allow complex features to be built up in stages. In particular our multilayered structure is similar to that used by convolutional networks [12], although it should be noted that because convolutional networks are not recurrent, they are difficult to apply to unsegmented cursive handwriting recognition.

The system is described in Section 1, experimental results are given in Section 2, and conclusions and directions for future work are given in Section 3.

1 Method

The three components of the recognition system are: (1) multidimensional recurrent neural networks, and multidimensional LSTM in particular; (2) the Connectionist Temporal Classification output layer; and (3) the hierarchical structure. In what follows we describe each component in turn, then show how they fit together to form a complete system. For a more detailed description of (1) and (2) we refer the reader to [4]

1.1 Multidimensional Recurrent Neural Networks

The basic idea of multidimensional recurrent neural networks (MDRNNs) [7] is to replace the single recurrent connection found in standard recurrent networks with as many connections as there are spatio-temporal dimensions in the data. These connections allow the network to create a flexible internal representation of surrounding context, which is robust to localised distortions.

An MDRNN hidden layer scans through the input in 1D strips, storing its activations in a buffer. The strips are ordered in such a way that at every point the layer has already visited the points one step back along every dimension. The hidden activations at these previous points are fed to the current point through recurrent connections, along with the input. The 2D case is illustrated in Fig. 1.

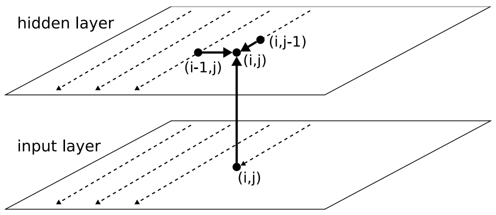

Fig. 1 Two dimensional RNN. The thick lines show connections to the current point (i, j). The connections within the hidden layer plane are recurrent. The dashed lines show the scanning strips along which previous points were visited, starting at the top left corner.

One such layer is sufficient to give the network access to all context against the direction of scanning from the current point (e.g. to the top and left of (i, j) in Fig. 1). However we usually want surrounding context in all directions. The same problem exists in 1D networks, where it is often useful to have information about the future as well as the past. The canonical 1D solution is to use *bidirectional recurrent neural networks* [18], where two separate hidden layers scan through the input forwards and backwards. The generalisation of bidirectional networks to n dimensions (*multidirectional networks*) requires 2^n hidden layers, starting in every corner of the n dimensional hypercube and scanning in opposite directions. The two dimensional case is shown in Fig. 2. All the hidden layers are connected to a single output layer, which therefore receives context information from all direction.

The error gradient of an MDRNN can be calculated with an n-dimensional extension of backpropagation through time. As in the 1D case, the data is processed in the reverse order of the forward pass, with each hidden layer receiving both the output derivatives and its own n 'future' derivatives at every timestep.

Let $a_j^{\mathbf{p}}$ and $b_j^{\mathbf{p}}$ be respectively the input and activation of unit j at point $\mathbf{p} = (p_1, \dots, p_n)$ in an n-dimensional input sequence $\mathbf{x}$ with dimensions $(D_1, \dots, D_n)$. Let $\mathbf{p}_d^- = (p_1, \dots, p_d - 1, \dots, p_n)$ and $\mathbf{p}_d^+ = (p_1, \dots, p_d + 1, \dots, p_n)$. Let w_{ij} and w_{ij}^d be respectively the weight of the feedforward connection from unit i to unit j and the

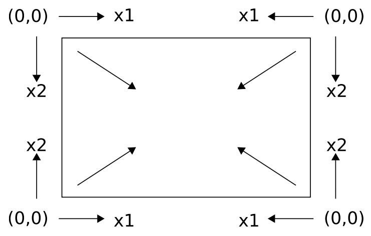

Fig. 2 Axes used by the 4 hidden layers in a multidirectional 2D RNN. The arrows inside the rectangle indicate the direction of propagation during the forward pass.

recurrent connection from i to j along dimension d. Let θ_h be the activation function of hidden unit h, and for some unit j and some differentiable objective function O let $\delta_j^{\mathbf{p}} = \frac{\partial O}{\partial a_j^{\mathbf{p}}}$. Then the forward and backward equations for an n-dimensional MDRNN with I input units, K output units, and H hidden summation units are as follows:

Forward Pass

$$a_h^{\mathbf{p}} = \sum_{i=1}^{I} x_i^{\mathbf{p}} w_{ih} + \sum_{\substack{d=1:\\ p_d>0}}^{n} \sum_{\hat{h}=1}^{H} b_{\hat{h}}^{\mathbf{p}_d^-} w_{\hat{h}h}^{d}$$

$$b_h^{\mathbf{p}} = \theta_h(a_h^{\mathbf{p}})$$

Backward Pass

$$\delta_h^{\mathbf{p}} = \theta_h'(a_h^{\mathbf{p}}) \left(\sum_{k=1}^{K} \delta_k^{\mathbf{p}} w_{hk} + \sum_{\substack{d=1:\\ p_d<D_d-1}}^{n} \sum_{\hat{h}=1}^{H} \delta_{\hat{h}}^{\mathbf{p}_d^+} w_{h\hat{h}}^{d} \right)$$

1.1.1 Multidimensional LSTM

Long Short-Term Memory (LSTM) [8, 3] is an RNN architecture designed for data with long-range interdependencies. An LSTM layer consists of recurrently connected ‘memory cells’, whose activations are controlled by three multiplicative gate units: the input gate, forget gate and output gate. The gates allows the cells to store and retrieve information over time, giving them access to long-range context. An illustration of an LSTM memory cell is shown in Figure 3.

The standard formulation of LSTM is explicitly one-dimensional, since each cell contains a single recurrent connection, whose activation is controlled by a single forget gate. However we can extend this to n dimensions by using instead n recurrent connections (one for each of the cell’s previous states along every dimension) with n forget gates.

Consider an multidimensional LSTM (MDLTSM) memory cell in a hidden layer of H cells, connected to I input units and K output units. The subscripts c, ι, ϕ and ω refer to the cell, input gate, forget gate and output gate respectively. $b_h^{\mathbf{p}}$ is the output of cell h in the hidden layer at point $\mathbf{p}$ in the input sequence, and $s_c^{\mathbf{p}}$ is the *state* of cell c at $\mathbf{p}$. f_1 is the activation function of the gates, and f_2 and f_3 are respectively the cell input and output activation functions. The suffix ϕ,d denotes the forget gate corresponding to recurrent connection d. The input gate ι is connected to previous

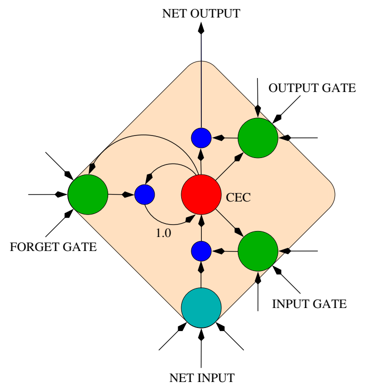

Fig. 3 LSTM memory cell. The internal state of the cell is maintained with a recurrent connection of fixed weight 1.0. The three gates collect activations from inside and outside the block, and control the cell via multiplicative units (small circles). The input and output gates scale the input and output of the cell while the forget gate scales the internal state.

cell c along all dimensions with the same weight ($w_{c\iota}$) whereas the forget gates are connected to cell c with a separate weight $w_{c(\phi,d)}$ for each dimension d. Then the forward and backward pass are as follows:

Forward Pass

<u>*Input Gate:*</u>
$$b_\iota^{\mathbf{p}} = f_1\left(\sum_{i=1}^{I} x_i^{\mathbf{p}} w_{i\iota} + \sum_{\substack{d=1:\\ p_d>0}}^{n} \left(w_{c\iota} s_c^{\mathbf{p}_d^-} + \sum_{h=1}^{H} b_h^{\mathbf{p}_d^-} w_{h\iota}^d\right)\right)$$

<u>*Forget Gate:*</u>
$$b_{\phi,d}^{\mathbf{p}} = f_1\left(\sum_{i=1}^{I} x_i^{\mathbf{p}} w_{i(\phi,d)} + \sum_{\substack{d'=1:\\ p_{d'}>0}}^{n} \sum_{h=1}^{H} b_h^{\mathbf{p}_{d'}^-} w_{h(\phi,d)}^{d'} + \begin{cases} w_{c(\phi,d)} s_c^{\mathbf{p}_d^-} \text{ if } p_d > 0 \\ 0 \text{ otherwise} \end{cases}\right)$$

<u>*Cell:*</u>
$$a_c^{\mathbf{p}} = \sum_{i=1}^{I} x_i^{\mathbf{p}} w_{ic} + \sum_{\substack{d=1:\\ p_d>0}}^{n} \sum_{h=1}^{H} b_h^{\mathbf{p}_d^-} w_{hc}^d$$

<u>*State:*</u>
$$s_c^{\mathbf{p}} = b_\iota^{\mathbf{p}} f_2(a_c^{\mathbf{p}}) + \sum_{\substack{d=1:\\ p_d>0}}^{n} s_c^{\mathbf{p}_d^-} b_{\phi,d}^{\mathbf{p}}$$

<u>*Output Gate:*</u>
$$b_\omega^{\mathbf{p}} = f_1\left(\sum_{i=1}^{I} x_i^{\mathbf{p}} w_{i\omega} + \sum_{\substack{d=1:\\ p_d>0}}^{n} \sum_{h=1}^{H} b_h^{\mathbf{p}_d^-} w_{h\omega}^d + w_{c\omega} s_c^{\mathbf{p}}\right)$$

<u>*Cell Output:*</u>
$$b_c^{\mathbf{p}} = b_\omega^{\mathbf{p}} f_3(s_c^{\mathbf{p}})$$

Backward Pass

$$\underline{\textit{Cell Output:}}\quad \epsilon_c^{\mathbf{p}} \overset{\text{def}}{=} \frac{\partial O}{\partial b_c^{\mathbf{p}}} = \sum_{k=1}^{K} \delta_k^{\mathbf{p}} w_{ck} + \sum_{\substack{d=1:\\ p_d<D_d-1}}^{n} \sum_{h=1}^{H} \delta_h^{\mathbf{p}_d^+} w_{ch}^d$$

$$\underline{\textit{Output Gate:}}\quad \delta_\omega^{\mathbf{p}} = f_1'(a_\omega^{\mathbf{p}})\epsilon_c^{\mathbf{p}} f_3(s_c^{\mathbf{p}})$$

$$\underline{\textit{State:}}\quad \epsilon_s^{\mathbf{p}} \overset{\text{def}}{=} \frac{\partial O}{\partial s_c^{\mathbf{p}}} = b_\omega^{\mathbf{p}} f_3'(s_c^{\mathbf{p}})\epsilon_c^{\mathbf{p}} + \delta_\omega^{\mathbf{p}} w_{c\omega} + \sum_{\substack{d=1:\\ p_d<D_d-1}}^{n} \left(\epsilon_s^{\mathbf{p}_d^+} b_{\phi,d}^{\mathbf{p}_d^+} + \delta_\iota^{\mathbf{p}_d^+} w_{c\iota} + \delta_{\phi,d}^{\mathbf{p}_d^+} w_{c(\phi,d)} \right)$$

$$\underline{\textit{Cell:}}\quad \delta_c^{\mathbf{p}} = b_\iota^{\mathbf{p}} f_2'(a_c^{\mathbf{p}})\epsilon_s^{\mathbf{p}}$$

$$\underline{\textit{Forget Gate:}}\quad \delta_{\phi,d}^{\mathbf{p}} = \begin{cases} f_1'(a_{\phi,d}^{\mathbf{p}}) s_c^{\mathbf{p}_d^-} \epsilon_s^{\mathbf{p}} \text{ if } p_d > 0 \\ 0 \text{ otherwise} \end{cases}$$

$$\underline{\textit{Input Gate:}}\quad \delta_\iota^{\mathbf{p}} = f_1'(a_\iota^{\mathbf{p}}) f_2(a_c^{\mathbf{p}})\epsilon_s^{\mathbf{p}}$$

1.2 Connectionist Temporal Classification

Connectionist temporal classification (CTC) [5] is an output layer designed for sequence labelling with RNNs. It does not require pre-segmented training data, or postprocessing to transform its outputs into transcriptions. It trains the network to predict a conditional probability distribution over all possible output label sequences, or *labellings*, given the complete input sequence.

A CTC output layer contains one more unit than there are elements in the alphabet L of labels for the task. The output activations are normalised at each timestep with the softmax activation function [2]. The first $|L|$ outputs estimate the probabilities of observing the corresponding labels at that time, and the extra output estimates the probability of observing a 'blank', or no label. For a length T input sequence $\mathbf{x}$, the complete sequence of CTC outputs therefore defines a probability distribution over the set L'^T of length T sequences over the alphabet $L' = L \cup \{blank\}$. We refer to the elements of L'^T as *paths*. Since the probabilities of the labels at each timestep are conditionally independent given $\mathbf{x}$, the conditional probability of a path $\pi \in L'^T$ is given by

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi(t)}^t. \tag{1}$$

where y_k^t is the activation of output unit k at time t.

Paths are mapped onto labellings $\mathbf{l} \in \mathbf{L}^{\leq T}$ by an operator $\mathscr{B}$ that removes first the repeated labels, then the blanks. So for example, both $\mathscr{B}(a,-,a,b,-)$ and $\mathscr{B}(-,a,a,-,-,a,b,b)$ yield the labelling (a,a,b). Since the paths are mutually exclusive, the conditional probability of some labelling $\mathbf{l} \in \mathbf{L}^{\leq T}$ is the sum of the probabilities of all paths corresponding to it:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathscr{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}). \tag{2}$$

This 'collapsing together' of different paths onto the same labelling is what allows CTC to use unsegmented data, because it means that the network only has to learn the order of the labels, and not their alignment with the input sequence.

Although a naive calculation of Eq. 2 is unfeasible, it can be efficiently evaluated with a dynamic programming algorithm, similar to the forward-backward algorithm for HMMs.

To allow for blanks in the output paths, for each labelling $\mathbf{l} \in \mathbf{L}^{\leq T}$ consider a modified labelling $\mathbf{l}' \in \mathbf{L}'^{\leq T}$, with blanks added to the beginning and the end and inserted between every pair of labels. The length $|\mathbf{l}'|$ of $\mathbf{l}'$ is therefore $2|\mathbf{l}|+1$.

For a labelling $\mathbf{l}$, define the *forward variable* $\alpha(s,t)$ as the summed probability of all path beginnings reaching index s of $\mathbf{l}'$ at time t, and the *backward variables* $\beta(s,t)$ as the summed probability of all path endings that would complete the labelling $\mathbf{l}$ if the path beginning had reached s at time t. Both the forward and backward variables are calculated recursively [5]. The label sequence probability is given by the sum of the products of the forward and backward variables at any timestep, i.e.

$$p(\mathbf{l}|\mathbf{x}) = \sum_{s=1}^{|\mathbf{l}'|} \alpha_(s,t)\beta(s,t). \tag{3}$$

1.2.1 Objective Function

Let S be a training set, consisting of pairs of input and target sequences $(\mathbf{x},\mathbf{z})$, where $|\mathbf{z}| \leq |\mathbf{x}|$. Then the objective function $\mathscr{O}$ for CTC is the negative log probability of the network correctly labelling all of S:

$$\mathscr{O} = -\sum_{(\mathbf{x},\mathbf{z}) \in S} \ln p(\mathbf{z}|\mathbf{x}) \tag{4}$$

The network can be trained with gradient descent by first differentiating $\mathscr{O}$ with respect to the outputs, then using backpropagation through time to find the derivatives with respect to the weights.

Note that the same label (or blank) may be repeated several times for a single labelling $\mathbf{l}$. We define the set of positions where label k occurs as

$$lab(\mathbf{l},k) = \{s : \mathbf{l}'_s = k\}, \tag{5}$$

which may be empty. Setting $\mathbf{l} = \mathbf{z}$ and differentiating $\mathscr{O}$ with respect to the network outputs for a particular element $(\mathbf{x},\mathbf{z})$ in the training set, we obtain:

$$\frac{\partial \mathscr{O}}{\partial a_k^t} = -\frac{\partial \ln p(\mathbf{z}|\mathbf{x})}{\partial a_k^t} = y_k^t - \frac{1}{p(\mathbf{z}|\mathbf{x})} \sum_{s \in lab(\mathbf{z},k)} \alpha(s,t)\beta(s,t), \tag{6}$$

where a^t_k and y^t_k are respectively the input and output of CTC unit k at time t for some $(\mathbf{x}, \mathbf{z}) \in S$.

1.2.2 Decoding

Once the network is trained, we can label some unknown input sequence $\mathbf{x}$ by choosing the labelling $\mathbf{l}^*$ with the highest conditional probability, i.e.

$$\mathbf{l}^* = \arg\max_{\mathbf{l}} p(\mathbf{l}|\mathbf{x}). \tag{7}$$

In cases where a dictionary is used, the labelling can be constrained to yield only sequences of complete words using the CTC token passing algorithm [6]. For the experiments in this paper, the labellings were further constrained to give single word sequences only, and the n most probable words were recorded. For words with variant spellings, the summed probability of all variants was used as the probability.

Let D be a dictionary of words. All words in a subset U of D are unique and all other words in D are variants of some word in U. For each word $u \in U$, define $v(u)$ as the set of variants of u, which includes u itself. For each word w, define the modified word w' as w with blanks added at the beginning and end and between each pair of labels. Therefore $|w'| = 2|w| + 1$. For segment s of word w' at timestep t in the output sequence, the value of $tok(w,s,t)$ is defined as the probability of the most probable partial output path $\pi(1:t)$ such that $\pi(t) = w'(s)$ and $\mathscr{B}(\pi(1:t)) = w(1:s/2)$, where $A(b:c)$ denotes the subsequence of sequence A from index b to index c.

At every timestep t of the length T output sequence, each segment s of each modified word w' holds a single token $tok(w,s,t)$. This is the highest token reaching that segment at that time. The *output token* $tok(w,-1,t)$ is the highest token leaving word w at time t.

Pseudocode is provided in Algorithm 1. Note that in cases where decoding speed is important, the algorithm could be optimised by storing the words in a trie structure.

1.3 Network Hierarchy

Many computer vision systems use a hierarchical approach to feature extraction, with the features at each level used as input to the next level [17]. This allows complex visual properties to be built up in stages. Typically, such systems use subsampling, with the feature resolution decreased at each stage. They also generally have more features at the higher levels. The basic idea is to progress from a small number of simple local features to a large number of complex global features.

We created a hierarchical structure by repeatedly composing MDLSTM layers with feedforward layers. The basic procedure is as follows: (1) the image is divided into pixel windows, each of which is presented as a single input to the first set of

```
1: **Initialisation:**
2: for all words $w \in D$ do
3:     $tok(w,1,1) = \ln y_b^1$
4:     $tok(w,2,1) = \ln y_{w_1}^1$
5:     if $|w| = 1$ then
6:         $tok(w,-1,1) = tok(w,2,1)$
7:     else
8:         $tok(w,-1,1) = -\infty$
9:     $tok(w,s,1) = -\infty$ for all other $s$
10:
11: **Algorithm:**
12: for $t = 2$ to $T$ do
13:     sort output tokens $tok(w,-1,t-1)$ by ascending value
14:     for all words $w \in D$ do
15:         for segment $s = 1$ to $|w'|$ do
16:             $P = \{tok(w,s,t-1), tok(w,s-1,t-1)\}$
17:             if $w'(s) \neq blank$ and $s > 2$ and $w'(s-2) \neq w'(s)$ then
18:                 add $tok(w,s-2,t-1)$ to $P$
19:             $tok(w,s,t) = \max(P) + \ln y_{w'(s)}^t$
20:         $tok(w,-1,t) = \max(tok(w,|w'|,t), tok(w,|w'|-1,t))$
21:
22: **Termination:**
23: for all unique words $u \in U$ do
24:     $tok(u,-1,T) = \sum_{w \in v(u)} tok(w,-1,T)$
25: output $n$ best $tok(u,-1,T)$
```

Algorithm 1: CTC Token Passing Algorithm for single words.

MDLSTM layers (e.g. a 4x3 window is collapsed to a length 12 vector). If the image does not divide exactly into windows, it is padded with zeros. (2) the four MDLSTM layers scan through the window vectors in all directions. (3) the activations of the MDLSTM layers are collected into windows. (4) these windows are given as input to a feedforward layer. Note that all the layers have a 2D array of activations: e.g. a 10 unit feedforward layer with input from a 5x5 array of MDLSTM windows has a total of 250 activations.

The above process is repeated as many times as required, with the activations of the feedforward layer taking the place of the original image. The purpose of the windows is twofold: to collect local contextual information, and to reduce the area of the activation arrays. In particular, we want to reduce the vertical dimension, since the CTC output layer requires a 1D sequence as input. Note that the windows themselves do not reduce the overall amount of data; that is done by the layers that process them, which are therefore analogous to the subsampling steps in other approaches (although with trainable weights rather than a fixed subsampling function).

For most tasks we find that a hierarchy of three MDLSTM/feedforward stages gives the best results. We use the standard ‘inverted pyramid’ structure, with small layers at the bottom and large layers at the top. As well as allowing for more features at higher levels, this leads to efficient networks, since most of the weights are concentrated in the upper layers, which have a smaller input area.

Unless we know that the input images are of fixed height, it is difficult to choose window heights that ensure that the final feature map will always be one-dimensional, as required by CTC. A simple solution is to collapse the final array by summing over all the inputs in each vertical line, i.e. the input at time t to CTC unit k is given by

$$a_k^t = \sum_x a_k^{(x,t)} \tag{8}$$

where $a_k^{(x,y)}$ is the uncollapsed input to unit k at point (x,y) in the final array.

Furthermore, the widths of the windows must be chosen to prevent the final feature map from being shorter (horizontally) than the number of labels for a particular sequence, since CTC assumes that the input sequence is at least as long as the label sequence. If the trained system is to be applied to images of unknown dimensions, it is therefore a good idea to ensure that the final feature map is considerably longer than the target label sequence for every element of the training set.

1.4 Combined System

Fig. 4 shows how MDLSTM, CTC, and the layer hierarchy combine to form a complete recogniser.

2 Experiments

Variants of the above system won several competitions at the 2009 International Conference on Document Analysis and Recognition (ICDAR 2009). In this section we describe the winning entry to the offline Arabic handwriting recognition competition.

2.1 Data

The competition was based on the publicly available IFN/ENIT database of handwritten Arabic words [16]. The data consists of 32,492 images of individual handwritten Tunisian town and village names, of which we used 30,000 for training, and 2,492 for validation. The images were extracted from artificial forms filled in by over 400 Tunisian people. The forms were designed to simulate writing on a letter, and contained no lines or boxes to constrain the writing style.

Each image was supplied with a ground truth transcription for the individual characters, and the postcode of the corresponding town. There were 120 distinct characters in total, including variant forms for initial, medial, final and isolated char-

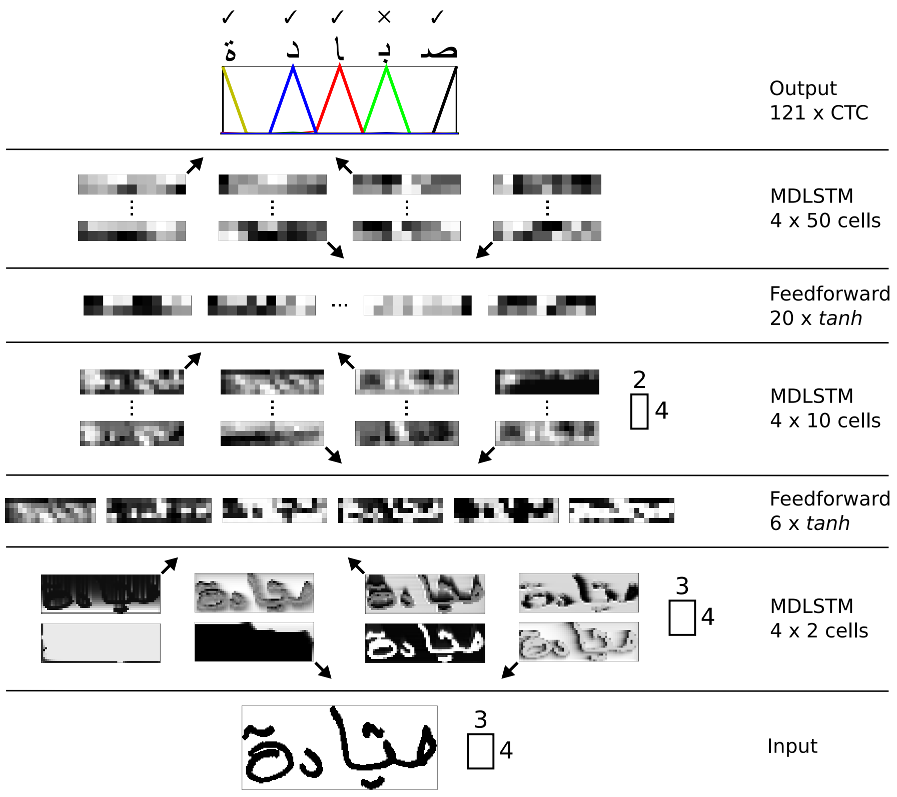

Fig. 4 A complete handwriting recognition system. First the input image is collected into windows 3 pixels wide and 4 pixels high which are then scanned by four MDLSTM layers. The activations of the cells in each layer are displayed separately, and the arrows in the corners indicates the scanning direction. Next the MDLSTM activations are gathered into 4 x 3 windows and fed to a feedforward layer of tanh summation units. Again the activations are. This process is repeated two more times, until the final MDLSTM activations are collapsed to a 1D sequence and transcribed by the CTC layer. In this case all characters are correctly labelled except the second last one.

acters. The goal of the competition was to identify the postcode, from a list of 937 town names and corresponding postcodes. Many of the town names had transcription variants, giving a total of 1,518 entries in the complete dictionary.

The test data (which is not published) was divided into sets ‘f’ and ‘s’. The main competition results were based on set ‘f’. Set ‘s’ contains data collected in the United Arab Emirates using the same forms; its purpose was to test the robustness of the recognisers to regional writing variations. The systems were allowed to choose up to 10 postcodes for each image, in order of preference. The test set performance using the top 1, top 5, and top 10 answers was recorded by the organisers.

2.2 Network Parameters

Three versions of the MDLSTM handwriting recognition system were entered for the competition, with slightly different parameters. Within the competition, they were given the collective group ID ‘MDLSTM’. For the first two networks (assigned system IDs 9 and 10 in the competition) the topology shown in Figure 4 was used, with each layer fully connected to the next layer in the hierarchy, all MDLSTM layers connected to themselves, and all units connected to a bias weight. These networks had 159,369 weights in total. The third network (system ID 11) had twice as many units in each of the hidden layers. That is, the four MDLSTM layers in the first level had four cells each, the first feedforward layer had twelve units, the MDLSTM layers in the second level had 20 cells each, the second feedforward layer had 40 units and MDLSTM layers in the third level had 100 cells each. This gave a total of 583,289 weights.

For all networks the activation function used for the LSTM gates was the logistic sigmoid $f_1(x) = 1/(1+e^{-x})$, while tanh was used for f_2 and f_3 (c.f. Section 1.1.1).

The networks were trained with online gradient descent, using a learning rate of 10^{-4} and a momentum of 0.9. Both the CTC objective function $\mathscr{O}$ (Section 1.2.1) and the character error rate (total number of insertions, deletions and substitutions needed to transform the network outputs into the target sequences, divided by the total length of the target sequences) were evaluated on the validation set after every pass through the training set. For networks 9 and 11 the error measure was the character error rate, while for network 10 the error measure was the CTC objective function. Networks 9 and 10 were created during the same training run, with the two different error measures used as a stopping criterion. For all networks training was stopped after 30 evaluations with no reduction in the error measure on the validation set. The weights giving the lowest error on the validation set were passed to the competition organisers for assessment on the test sets.

Figure 5 shows the error curves for networks 9 and 10 during training. Note that, by the time the character error is minimised, the CTC error is already well past its minimum and has risen substantially. This is typical for networks trained with CTC output layers.

Network 9 took 86 passes through the training set to complete training, network 10 took 49 passes and network 11 took 153 passes. The time per pass, which grows with the number of network weights, was around 62 minutes for networks one and two, and around 180 minutes for network three. The fact the network three required more training passes than network two is untypical, since usually the more weights a network has the fewer passes it takes to minimise a particular error measure. However the same network minimised the CTC error on the validation set after only 22 passes, and that the decrease in validation character error rate between 22 and 152 passes was only 0.2.

Table 1 summarises the differences between the three networks.

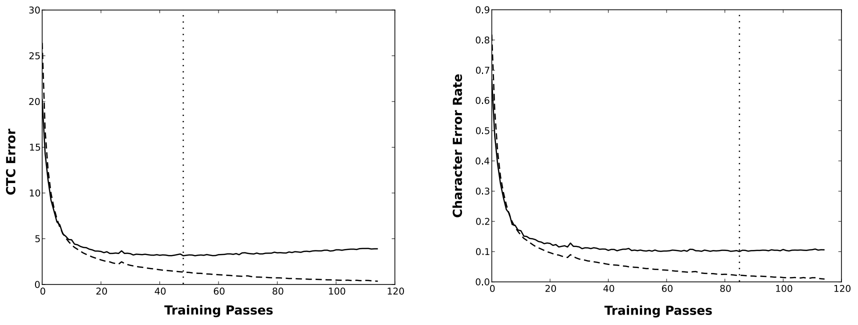

Fig. 5 Error curves during training of networks 9 and 10. The CTC error is shown on the left, and the character error is shown on the left. In both plots the solid line shows the error on the validation set, the dashed line shows the error on the training set, and the vertical dotted line indicates the point of lowest error on the validation set.

Table 1 The three MDLTSM networks entered for the Arabic handwriting competition.

ID	Weights	Error Measure	Passes	Approx. Pass time (mins)
9	159,369	Character	86	62
10	159,369	CTC	49	62
11	583,289	Character	153	180

2.3 Results

Table 2 [14] shows that all three MDLSTM networks (group ID MDLSTM, system ID 9–11) outperformed all other entries in the 2009 International Conference on Document Analysis and Recognition (ICDAR 2009), in terms of both recognition rate and speed. The recognition rates were also better than any of the entries in the ICDAR 2007 competition, which used the same training and test data, although the Siemens and MIE systems were faster.

The overall difference in performance between networks 9 and 10 is negligible, suggesting that it isn't that important which error measure is used for early stopping. This is significant, since, as discussed above, using CTC error for early stopping can leading to much shorter training times. Of particular interest is that the performance on set *s* (with handwriting from the United Arab Emirates) is about the same for both error measures. One hypothesis was that, because using CTC error leads to fewer training passes, network 10 would overfit less on the training data and therefore generalise better to test data drawn from a different distribution.

Network 11 gave about a 2% improvement over networks 9 And 10 in word recognition for both test sets, if only the best word was used. Although significant, this improvement comes at a cost of a more than threefold increase in word recognition time. For applications where time must be traded against accuracy, the number of units in the network layers (and hence the number of network weights) should be tuned accordingly.

Table 2 ICDAR 2009 Arabic offline handwriting recognition competition results . Results are % of correctly recognised images on reference datasets d and e, new datasets f and s, subsets f_a, f_f, and f_g. The average recognition time in ms per image on subsets t and t_1 is shown in the last two columns. (G-ID: Group ID, S-ID: System ID).

G-ID	S-ID	set d	set e	set f$_a$	set f_f	set f_g	set f			set s			time (**ms**)	
		top 1	top 1	top 1	top 1	top 1	top 1	top 5	top 10	top 1	top 5	top 10	set t	set t_1
UOB-ENST	1	92.52	85.38	83.57	84.77	85.09	82.07	89.74	91.22	69.99	81.44	84.68	812.69	841.25
	2	89.06	81.85	79.49	80.90	81.11	78.16	89.06	91.88	65.61	81.44	85.95	2365.48	2755.01
	3	89.84	83.52	80.89	82.15	82.17	79.55	90.60	92.16	67.83	83.47	86.65	2236.58	2754.08
	4	92.59	86.28	85.42	86.96	87.21	83.98	91.85	93.00	72.28	85.19	87.92	2154.48	2651.57
REGIM	5	79.52	63.53	58.81	59.27	60.42	57.93	73.43	78.10	49.33	65.10	71.14	1564.75	1712.15
Ai2A	6	93.90	87.25	86.73	88.54	89.36	85.58	92.57	94.12	70.44	82.01	84.87	1056,98	956,82
	7	94.92	82.21	83.53	84.86	84.67	82.21	91.24	92.47	66.45	80.52	83.13	**519,61**	1616,82
	8	97.02	91.68	90.66	91.92	92.31	**89.42**	95.33	95.94	**76.66**	88.01	90.28	2583,64	1585,49
MDLSTM	9	99.72	98.64	92.59	93.79	94.22	91.43	96.11	96.61	78.83	87.98	90.40	115.24	122.97
	10	99.60	97.60	92.58	94.03	94.40	91.37	96.24	96.61	78.89	88.49	90.27	**114.61**	122.05
	11	99.94	99.44	94.68	95.65	96.02	**93.37**	96.46	96.77	**81.06**	88.94	90.72	371.85	467.07
RWTH-OCR	12	99.91	98.71	86.97	88.08	87.98	85.51	93.32	94.61	71.33	83.66	86.52	17845.12	18641.93
	13	99.79	98.29	87.17	88.63	88.68	85.69	93.36	94.72	72.54	83.47	86.78		
	14	99.79	98.29	87.17	88.63	88.68	**85.69**	93.36	94.72	72.54	83.47	86.78		
	15	96.72	91.25	86.97	88.08	87.98	83.90	-	-	65.99	-	-	**542.12**	560.44
LITIS-MIRACL	16	93.04	85.46	83.29	84.51	84.35	82.09	90.27	92.37	**74.51**	86.14	88.87	143269.81	145157.23
LSTS	17	18.58	14.75	15.34	16.00	15.65	15.05	29.58	35.76	11.76	23.33	29.62	612.56	685.42
Results of the 3 best systems at ICDAR 2007														
Siemens	08	94.58	87.77	88.41	89.26	89.72	87.22	94.05	95.42	73.94	85.44	88.18	109.406	125.31
MIE	06	93.63	86.67	84.38	85.21	85.56	83.34	91.67	93.48	68.40	80.93	83.73	188.439	210.55
UOB-ENST	11	92.38	83.92	83.39	84.93	85.18	81.93	91.20	92.76	69.93	84.11	87.03	2172.55	2425.47

3 Conclusion

This chapter introduced a general offline handwriting recognition system based on MDLSTM recurrent neural networks. The system works directly on raw pixel data, and therefore requires minimal changes to be used for languages with different alphabets. It won several competitions at the ICDAR 2009 conference, including the Arabic offline handwriting recognition competition.

Various extensions to the system are currently being explored, including more efficient decoding, complete page transcription and weight pruning for increased speed.

References

1. P. Baldi and G. Pollastri. The principled design of large-scale recursive neural network architectures–dag-rnns and the protein structure prediction problem. *J. Mach. Learn. Res.*, 4:575–602, 2003.
2. J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fogleman-Soulie and J.Herault, editors, *Neurocomputing: Algorithms, Architectures and Applications*, pages 227–236. Springer-Verlag, 1990.
3. F. Gers, N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143, 2002.
4. A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Ph.D. in Informatics, Fakultat für Informatik – Technische Universität München, Boltzmannstraše 3, D - 85748, Garching bei München, Germany, 2008.
5. A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning, ICML 2006*, Pittsburgh, USA, 2006.
6. A. Graves, S. Fernández, M. Liwicki, H. Bunke, and J. Schmidhuber. Unconstrained online handwriting recognition with recurrent neural networks. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.
7. A. Graves, S. Fernández, and J. Schmidhuber. Multidimensional recurrent neural networks. In *Proceedings of the 2007 International Conference on Artificial Neural Networks*, Porto, Portugal, September 2007.
8. S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.
9. J. Hu, S. G. Lim, and M. K. Brown. Writer independent on-line handwriting recognition using an HMM approach. *Pattern Recognition*, 33:133–147, 2000.
10. S. Jaeger, S. Manke, J. Reichert, and A. Waibel. On-line handwriting recognition: the NPen++ recognizer. *International Journal on Document Analysis and Recognition*, 3:169–180, 2001.
11. H. Jiang. Discriminative training of hmms for automatic speech recognition: A survey. *Computer Speech Language*, 24(4):589 – 608, 2010.
12. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.
13. Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp. Off-road obstacle avoidance through end-to-end learning. In *Advances in Neural Information Processing Systems (NIPS 2005)*. MIT Press, 2005.
14. V. Margner and H. El Abed. Icdar 2009 arabic handwriting recognition competition. pages 1383 –1387, jul. 2009.

15. A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, pages 841–848, 2001.
16. M. Pechwitz, S. S. Maddouri, V. Mrgner, N. Ellouze, and H. Amiri. IFN/ENIT-database of handwritten arabic words. In *7th Colloque International Francophone sur l'Ecrit et le Document (CIFED 2002)*, Hammamet, Tunis, 2002.
17. M. Reisenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.
18. M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45:2673–2681, November 1997.