Offline Handwritten Recognition : Influence of the Baseline information

Killian Barrere [*] and Florent Bartoccioni[*] *supervised by* Bertrand Coüasnon[†]
[*]ENS Rennes, Univ Rennes, *France*
[†]Univ Rennes, CNRS, IRISA, *France*

Abstract—Known as Handwritten Text Recognition (HTR), the task of transcribing handwritten inputs into a sequence of numerical character is still an open problem. The offline variant of HTR, that is to say, making transcription from an image of a text, has been extensively researched. Therefore, we provide an exhaustive survey of the evolution and progress in this field, introducing Hidden Markov Models (HMM) as well as Neural Network (NN) approaches. It shows that NN architectures are more promising as they need less human supervision than HMM only methods. From this survey we provide a NN architecture for HTR that should compete with the state of the art performances. Finally, we discuss of the potential improvement the text baseline's position could bring to the recognition performance.

Index Terms—Deep Learning, Handwritten Text Recognition, Offline Text Recognition, Pre-Segmented lines.

1 Introduction

Even in our digital age there is still a non-negligible number of documents that are handwritten. Storing handwritten documents and searching through them efficiently are both difficult tasks. We define Handwritten Text Recognition (HTR) as the ability of transcribing into a sequence of numerical character handwritten inputs from sources such as paper documents, pictures, touch-screens and other devices. Two variants of HTR exist, *Online* and *Offline*. For the former the input is captured while the person is writing. For example the movements of the pen tip is registered by a graphics tablet. Meanwhile, for the later the input is a picture of the entire text (the raw values of pixels). Online HTR is know as being easier as more information is available such as the velocity of the tip, the pressure applied and the points drawn as a sequence through time. We narrow down the scope of the project by only working on the offline variant of HTR. Therefore, by simplicity and from this point, we will refer *Offline HTR* as only *HTR*.

Like other subjects, HTR was struck by the Deep Learning wave. Before HTR used to work with Hidden Markov Model (HMM) as an entire system, but with the new Deep Learning methods, the gain in performance was important [1]. Now, almost every HTR system uses Neural Networks. Many teams now work on many parts of HTR Neural Networks to slightly increase the performance.

But one thing that seems to be not so researched is the influence of the baseline's position. We think that adding the knowledge of a line could help the Neural Network to recognize letters (and texts at the end), especially those who have to cross the standard character size such as a p, l, etc.

In this project, we will first try to build an architecture that stick to state of the art results. Then we will try to evaluate the influence of the text baseline position. Especially, we will have to check different ways to add the baseline's information to the architecture.

The first part of the paper consists of an historical background of Text Recognition. HMM systems for text recognition is discussed in section 2, while the section 3 explains why HMM are now replaced with Neural Networks (NN). Then in section 4, some architectures used for HTR, and their main components are described. Section 5 explains some optimization that could be done to achieve better results. Finally, section 6 will explain our choices regarding the architecture that we will implement.

2 Hidden Markov Model for Handwritten Text Recognition

Hidden Markov Model (HMM) used to be the most common solution for HTR. Now almost each HMM pipeline have been replaced by Neural Networks. But HMM are still used for some part of the HTR pipeline (presented in section 4.2.1).

HMM are composed of multiple states. Each state can be linked to another state thanks to transitions. The HMM learns the probability of each transition according to some examples. Once learned, the HMM is able to recognize texts. It first computes the features for the inputs (the inputs are the pixels the HMM is currently working), and then uses the Viterbi's algorithm [2] to select the most likely sequence. It then returns the decided text.

But HMMs have some problems. First, each descriptor are provided by hand. People choose what descriptor to use based on their knowledge (Edge detection, number of black pixels, etc). Another problem is the lack of relations with the other characters (also called context). HMM have only access to its last result. It is a problem mainly because a HMM does not work on entire character, but on some pixels of the character.

3 Deep learning

Recently, Deep learning methods has been introduced in HTR pipeline to compensate for HMM's drawbacks. In this section we will introduce the deep learning tools used for feature extraction in HTR. We introduce Convolutional Neural Networks (CNN) (in section 3.2), which are able to learn relevant features extractor instead fixed, as well as Recurrent Neural Networks (RNN)(presented in section 3.3) that can access a larger context.

3.1 Artificial Neural Network

Neural Networks (NN) try to mimic the way the human brain works. A neuron is a simple unit combining input from the data with a set of weights. These weights assign significance to inputs for the task the algorithm is trying to learn. (For example, which input is most helpful in classifying data without error?). A neuron performs a weighted sum of input on which is applied a non-linear activation function. This activation function determines, according to the weights, to what extent the input signal will progress to the output, say, an act of classification. So, in essence, a neuron is a mathematical operation of its input.

Multiple neurons can be stacked together into what we call a layer. We will see in the following sections that different types of layer useful to HTR, such as convolutional and recurrent (section 3.2 and 3.3 exist. Deep Neural Networks (DNN) is the name we use for networks composed of several layers where each layer's output is simultaneously the subsequent layer's input.

In order for a NN to fulfill its task, we give the NN some examples. The NN then processes the input through its various neurons, this phase is called Forward Propagation. Then we get the output of the NN and compare the output with the desired output. Based on the error, we change the weights of the connections between neurons, this is called Backpropagation. This whole process is called training. DNN are difficult to train as the gradient of the Backpropagation tends to vanish when passing troughs a lot of layers, effectively preventing the weight from changing its value. This issue is called the vanishing gradient. We will explain this phenomenon and present different existing solutions to overcome it in section 3.4 .

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are mostly used in image features extraction as they capture local information from the input.

An intuitive explanation of CNN is that it learns local filters. Trough its convolutional layers, it learn to extract features from every location in the input. Formally, we make a convolution (exactly the mathematical operation) across the input data, but instead of using a predefined matrix, we learn this one as being the weights applied to the input data. Convolution has the nice property of being translational invariant. The output signal strength is not dependent on where the features are located, but simply whether the features are present.

Moreover, Various operations can be applied between convolutional layers. For example, inputs from the convolution layer can be "smoothened" to reduce the sensitivity of the filters to noise and variations (subsampling). Subsampling methods (for image signals) include reducing the size of the image, or reducing the color contrast across red, green, blue (RGB) channels.

They are several architectures in the field of CNNs that have a name. The most common are LeNet [3], AlexNet [4], VGG [5] and Inception [6] and are well known for images classification task.

With all these characteristics, CNNs are of great help regarding HTR [7] [8] and convolutional layers is often, if not always, found in architecture used for HTR.

3.3 Recurrent Neural Networks

Feed-forwards Neural Networks (FNN), like CNNs, falls short when working on sequential information. In FNN, we assume that inputs are independents from outputs, hence they can't retain what they have already seen (compute). However, if you want to predict the next element of a sequence, you have to know the preceding elements. Recurrent Neural Networks (RNN) address this issue.

RNNs are called recurrent because the output is depended of the previous computations, allowing information to persist. You can think of it as a memory that stores informations about previous computations.

In the figure 1, C is the repeating module (composition of different layers, more to say in section 3.4.1), it is the "memory" of this network. It captures information about what happened in all the previous steps. At step t, it takes an input x_t and outputs a value h_t based on the previous memorizations.

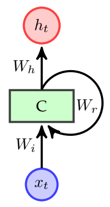

Fig. 1: A single layer Recurrent Neural Network

In other words, a RNN is simply a sequence of the same network, each of them passing informations to the next one. It is more explanatory when we unfold the network as in figure 2:

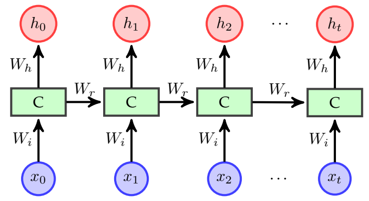

Fig. 2: A single layer Recurrent Neural Network unfolded

Once unfolded an RNN reveals its natural architecture for sequences. As an RNN is recursively connected to its previous state, it has informations of its entire history. This architecture embed properties that make RNNs suitable for sequence learning, such as robustness to input warping and the ability to learn which context to use. Theoretically it is able to memorize every past state. But in experiment, it struggles a lot to memorize informations from a far previous state. Long Short-Term Memory (introduced in section 3.4.1) have been proposed as a solution to better memorize context from a long time.

3.4 Vanishing gradient problem

A problem caused by the use of Deep Learning architectures is often faced during backpropagation. The process of backpropagation consists in updating the network's weights according to the error. More formally we have

$$\Delta W_i = \frac{\partial E}{\partial W_i}$$

Using the chain rule, it translate into

$$\Delta W_i = \frac{\partial E}{\partial output} \frac{\partial output}{\partial hidden_n} \cdots \frac{\partial hidden_1}{\partial W_i}$$

Recall that each neuron in hidden layers usually use sigmoid activation and its derivative outputs values between 0 and 1/4. Hence, we are multiplying numerous values which are between 0 and 1. The error propagated will become small very fast. Because of this, the first layers are the slowest to train. And since the latter layers, especially the output, is functionally dependent on the earlier layers, inaccurate early layers will cause the latter layers to simply build on this inaccuracy, corrupting the entire neural net. This phenomenon is called the Vanishing gradient Problem. The following sections proposes existing solutions to solve this problem.

3.4.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) [9] is a solution to RNN (introduced in section 3.3) that are facing the Vanishing Gradient Problem. It consists of several memory cells. Each cell has an input, an output and a forget gate. It can remember previous states like a RNN. The forget gate is used to tell the cell to lose some knowledge. The results show that LSTM can store informations for a longer time than RNN. LSTM have increased the RNN's performances and are now used almost in every architecture (like Bi-Directional LSTM presented in section 4.1.1). It is also more robust to the Vanishing Gradient Problem. Some new alternatives exist like Gated Recurrent Unit (GRU) [10] or Gated Memory Unit (GMU) [11]. They provide an access to even longer term dependency, giving better results. While GRUs may be used, GMUs are just recent and struggle to have better results than LSTMs.

The activation layer controls how the signal flows from one layer to the next, emulating how neurons are fired in our brain. Output signals which are strongly associated with past references would activate more neurons, enabling signals to be propagated more efficiently for identification.

3.4.2 Rectified Linear Unit

The vanishing gradient is a problem for the training of neural networks. Standard activation functions limit the backpropagation of the gradient. It is explained by the bounds of the activation functions. Sigmoid function returns a result between 0 and 1. Rectified Linear Unit (ReLU) solves this problem and shows some improvements to the Vanishing gradient problem.

$$ReLU(x) = max(0, x)$$

For negative numbers, the derivate is equal to 0 while for positive numbers, it is equal to 1. Thanks to the derivate value, NNs that use ReLU are less affected by the Vanishing Gradient Problem. Through Backpropagation, the values to change still have a derivate equal to 1. A problem with this function is that its derivate has no value at $x = 0$.

4 PIPELINE

In this section we will discuss of the full pipeline. We will talk about how the basic Deep Learning tools (section 3) have been extended to handle the feature extraction in the HTR task. An example of an architecture using Multi-Dimensional LSTM (MDLSTM), CNN, Dropout and Connectionist temporal classification (CTC) (respectively section 4.1.2, 3.2, 5.2 and 4.2.3) is showed in figure 3. However the feature extraction is only a component of a complete recognition system, which typically also include an optical model, and a language model. Both of them will be formalized and explained in section 4.2.

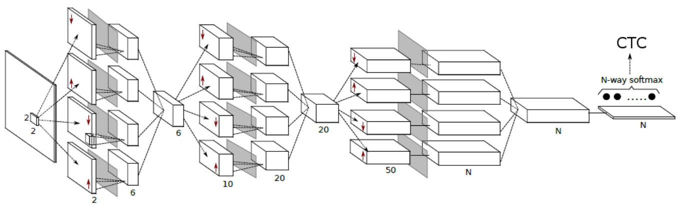

Fig. 3: Architecture designed for feature extraction in HTR (source: [12]). Full size image with additional information is available in figure 6.

4.1 Feature Extraction

Feature extraction is the process of transposing the data from its original space into a feature space used for classification. As explained in section 3.3 RNNs are a natural tool for sequence processing. To make use of all the information available in the input, it has been extended to handle two directions (in section 4.1.1). It has been further extended on multiple dimensions and direction by Alex Graves with multi-directional Multiple-Dimensional RNN [13] (MDRNN) which we are describing in section 4.1.2.

4.1.1 Bidirectional Long Short-Term Memory

When classifying a word in an image of handwritten text, it helps to look at the characters at the end and the beginning of the word. Even if a RNN can retains past information it does not take advantage of this contextual information. Bidirectional RNN [14] (BRNN) is an effective solution to this problem. In a BRNN, two separate recurrent net takes the input respectively in forward and backward direction and are connected to the same output layer (see figure 4).

As BRNN is based on the standard RNN it also suffers from the vanishing gradient. To address this issue BRNN has been extended to Bidirectional LSTM (BLSTM). BLSTM has been successfully applied to speech [15] [16] and text [17] [18] [19] recognition

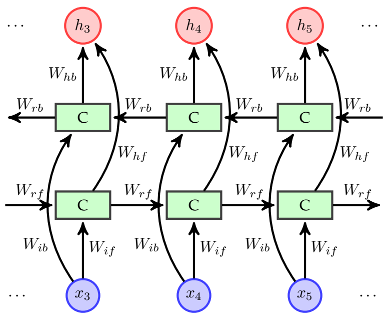

Fig. 4: A BRNN unfolded

4.1.2 *Multi-Dimensional Recurrent Neural Networks*

Because of the one unidimensional nature of standards RNNs (section 3.3), they are poorly suited to multidimensional data such as images. Therefore, the data must be preprocessed to one dimension, for example by presenting one vertical line of an image at a time to the network. Alex Graves proposed the Multi-Dimensional Recurrent Neural Networks (MDRNN) [13] as an efficient way of building multidimensional context into RNNs.

The idea behind MDRNNs is to replace the single recurrent connection found in standard RNNs (C in figure 1) with as many recurrent connections as there are dimensions in the data. Hence, at each point in the data sequence, the hidden layer of the network receives information from one step back along all dimensions.

For the same reasons as discussed in section 4.1.1, we might want to have access to the surrounding context in all directions. BRNNs can be extended to n-dimensional data by using 2^n separate hidden layers. The 2 dimensional case is illustrated in figure 5a .

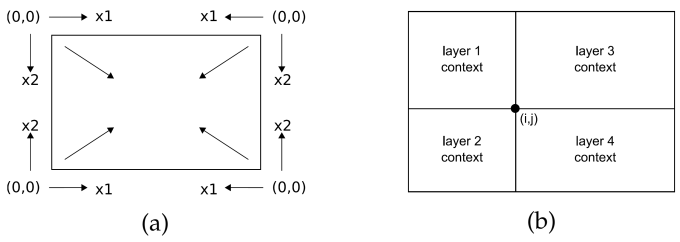

Fig. 5: (a) Axes used by the 4 hidden layers in a multi-directional MDRNN. The arrows inside the rectangle indicate the direction of propagation during the forward pass. (b) Context available at (i,j) to a multi-directional MDRNN (source: [13])

The MDRNN have several advantages that others NN architectures don't have. For example, CNN must have a hand specified kernel size which means they do not scale well to large images. Moreover, with CNNs, tasks such as handwritten digits recognition require a presegmentation into individual characters [20]. MDRNN, on the other hand, scale naturally in input size and dimensionality.

Multi-dimensional HMMs have also been proposed for multi-dimensional tasks. However, the time required to run the Viterbi's algorithm [2], and thereby calculate the optimal state sequences, grows exponentially with the number of data points. Likewise, the number of transition probabilities, and hence the required memory, grows exponentially with the data dimensionality. Hence, this method is not able to exploit the full multi-dimensional structure of the data.

MDRNN address these issues [13], and its variant using LSTMs, the MDLSTM [13], showed great results on HTR [1] [12] [21]

4.2 Decoding

In this section, we focus on the transformation of feature vectors (the output of the NN) into text. We first need to formalize the task of recognition. Given an input x (a feature vector), we want to find the sentence s^* that maximize the conditional probability:

$$s^* = \arg\max_{s \in \mathcal{S}} P(s|x)$$

As $P(x)$ is fixed, by Bayes rule the maximization can be reduced to:

$$s^* = \arg\max_{s \in \mathcal{S}} P(x|s)P(s)$$

In this section we present state of the art methods to estimate these probabilities.

4.2.1 *Optical Model*

The term $P(x|s)$ describes the probability of feature vector x that can be produced from sentences: this is known as the optical model (similarly as the acoustic model in speech recognition), and is often a HMM (which gives the so called hybrid-HMM).

Hybrid Hidden Markov Model (Hybrid HMM), use both a Neural Network and a HMM. The Neural Network can be a basic NN or a more complex network like MDLSTMs (see section 4.1.2). The NN is used to estimate the probabilities of the HMM's transitions. Hybrid HMM are faster than HMM as a full pipeline, and the number of parameters is lower [22].

4.2.2 *Language Model*

A complete system for large vocabulary handwriting text recognition includes a lexicon and a Language model (LM), which greatly decrease the error rate by inducing lexical constraints and rescoring the hypotheses produced by the optical model.

The aim of the LM is to provide a density estimation $P(s)$ over all possible sentences. Hence, we could distinguish two sentences that are similar visually. In particular, we want the model to satisfy $P(s_1) > P(s_2)$ where s_1 is the correct sentence and s_2 the erroneous one.

To estimate this probability, *n-grams* are often used. *n-grams* are defined as a succession of n characters. LM can use NN to improve some parts of the LM. These LMs are specifically called Neural Network - Language Model (NNLM). They learn a vocabulary given some example words. They are focusing on the probabilistic transition between characters, syllabus or *n-grams*. They are useful because of the difficulty of the task [23] [24].

4.2.3 *Connectionist Temporal Classification*

RNNs require pre-segmented training data and post-processing to transform their outputs into label sequences. Therefore their applicability has so far been limited. To address this issue, Connectionist Temporal Classification (CTC) [25] has been introduced by Alex Graves and al, and directly estimate $P(s|x)$. CTC is a cost function for NN which enables it to learn a mapping between sequences of unequal lengths. Specifically, we assume that the output sequence is shorter than the input. To compensate for difference in length, a blank label is added to the output labels. Moreover it allows classification without segmentation as it aligns each label prediction with the corresponding part of the sequence. CTC showed better results [25] [1] than HMM as it provide probabilistic estimation of the output sequence (characters and blanks) regarding the sequence of inputs.

In [26] and [27], the authors have also extended their CTC with RNN transducer method, where a LM is added in conjunction with the CTC leading to a jointly trained optical and LM.

5 Training & Optimization process

It is known that NN are difficult to train [28]. In this section we discuss about techniques and optimizations to train NN.

5.1 training RNN

The feedforward Backpropagation algorithm cannot be directly transferred to RNNs because the standard backpropagation pass assumes the connections between units are cycle-free. The solution of the Backpropagation Trough Time approach is to unfold the recurrent network in time (as shown in figure 2), by stacking identical copies of the RNN, and redirecting connections within the network to obtain connections between subsequent copies. This gives a feedforward network, on which we can execute the backpropagation algorithm.

An efficient training framework for recurrent neural networks is critical to reach competitive results [29]. One simple optimization for the training of RNN is to initialize the weight of the network to random values [30]. When initialized this way, the RNN converges faster.

5.2 Dropout

The Dropout [12] selects some neurons in the network randomly during each epoch, and disable them. The outputs of these neurons are null, and they have no impact on forward propagation and Backpropagation. The resulting trained NN is less impacted by over-fitting. Recently, some researchers were working on a possible alternative called DropFilter [31]. DropFilter shows goods results. First the error rate is lowest thanks to their use, and the number of parameters is lower. But it is only true for some specific models. For others NN the training seems to be a problem.

5.3 Curriculum Learning

Generally, NN are trained with random examples at a given epoch. Curriculum Learning [32] mimics school teaching. Rather than training with random examples, the neural network is trained with simple examples firstly. The difficulty of the given examples increases during the draining.

5.4 Transfer Learning

In addition to Curriculum Learning, training a model already trained on other data or languages shows good results. The process is called Transfer Learning. It starts with a NN and its trained weight, and the NN is slightly modified. Only some weights can be targeted for instance. It usually converges faster than a full training.

5.5 Database augmentation

The number of examples in a dataset is a key to have good performances. Because it can be hard to get an important number of examples in a dataset, the dataset is increased with some operations. The examples in the dataset are cropped, mirrored, rotated, etc. Then the dataset is bigger, and it got better results, more generality power.

6 Conclusion

Recently, the performance of offline HTR has been improved greatly [33] by leveraging deep learning technologies such as RNNs (section 3.3) and CNNs (section 3.2). Deep BLSTM (e.g., [34], [19], [35]) and MDLSTM (e.g., [36], [37], [1], [12] [21]) seem to be the state of the art. Results such as word error rate and character error rate respectively below 10% and 5% are now reachable on evaluation sets such as RIMES [38].

But despite these astonishing results HTR is still an open problem. Recently, the necessity of MDLSTM for HTR was debated [39]. On the other hand, advances in the deep learning field such as Deep Residual Network [40] might open new ways for improving the state of the art.

Deep BLSTM an MDLSTM trained by using a CTC objective function will learn both local character image dependency for character modeling and long-range contextual dependency for implicit language modeling. Hence, we will firstly work on these type of architectures as they are well suited for the HTR task.

For the decoding part, Hybrid HMM (section 4.2.1) and NNLM (section 4.2.2) show a significant improvement. They lower the error rate and seem to be better for the training process. We will use one of the two models, and eventually test the influence of changing to the other.

In addition to just recognize handwritten text, we also want to study if the addition of the text baseline's position have a positive effect on the NN. We could easily have this information thanks to systems of the competition on Baseline Detection [41]. Adding text baseline's position to the network's available informations could eventually lead to a faster convergence, and a better accuracy to recognize the text. Some articles ([36] [32] [42]) already discuss about the impact of pre-segmented lines, and it seems to have good results. But the subject is not so studied. That is why we want to test the impact of pre-segmented lines. We think that it will be important to test how to add this information to the network. It can be given through network inputs as some numbers, by modifying the text images, or by adding it in deeper layers of the network.

References

[1] Alex Graves and Juergen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 545–552. Curran Associates, Inc., 2009.

[2] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] D. Suryani, P. Doetsch, and H. Ney. On the benefits of convolutional neural network combinations in offline handwriting recognition. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 193–198, Oct 2016.

[8] Yi-Chao Wu, Fei Yin, and Cheng-Lin Liu. Improving handwritten chinese text recognition using neural network language models and convolutional neural network shape models. *Pattern Recognition*, 65:251–264, 2017.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[11] Sun Li, Su Tonghua, and Yu Shengjie, Zhou Lijun. Gmu: A novel rnn neuron and its application to handwriting recognition. In *2017 14th International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[12] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. Dropout improves recurrent neural networks for handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 285–290, Sept 2014.

[13] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Multi-dimensional recurrent neural networks. *CoRR*, abs/0705.2011, 2007.

[14] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, Nov 1997.

[15] A. Graves, N. Jaitly, and A. r. Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, Dec 2013.

[16] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke. The Microsoft 2017 Conversational Speech Recognition System. *ArXiv e-prints*, August 2017.

[17] A. Ray, S. Rajeswar, and S. Chaudhury. Text recognition using deep blstm networks. In *2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR)*, pages 1–6, Jan 2015.

[18] Marcus Liwicki, Alex Graves, Horst Bunke, and Jürgen Schmidhuber. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In *In Proceedings of the 9th International Conference on Document Analysis and Recognition, ICDAR 2007*, 2007.

[19] V. Frinken and S. Uchida. Deep blstm neural networks for unconstrained continuous handwritten text recognition. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 911–915, Aug 2015.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[21] T. Bluche, J. Louradour, M. Knibbe, B. Moysset, M. F. Benzeghiba, and C. Kermorvant. The a2ia arabic handwritten text recognition system at the open hart2013 evaluation. In *2014 11th IAPR International Workshop on Document Analysis Systems*, pages 161–165, April 2014.

[22] Morgan Nelson and Boulard Hervé. An introduction to hybrid hmm/connectionist continuous speech recognition.

[23] F. Zamora-Martínez, V. Frinken, S. España-Boquera, M.J. Castro-Bleda, A. Fischer, and H. Bunke. Neural network language models for off-line handwriting recognition. *Pattern Recognition*, 47(4):1642 – 1652, 2014.

[24] V. Frinken, F. Zamora-Martínez, S. España-Boquera, M. J. Castro-Bleda, A. Fischer, and H. Bunke. Long-short term memory neural networks language modeling for handwriting recognition. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 701–704, Nov 2012.

[25] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[26] A. Graves, A. r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013.

[27] Alex Graves. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711, 2012.

[28] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[29] P. Doetsch, M. Kozielski, and H. Ney. Fast and robust training of recurrent neural networks for offline handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 279–284, Sept 2014.

[30] Hans-Georg Zimmermann, Christoph Tietz, and Ralph Grothmann. Forecasting with recurrent neural networks: 12 tricks. In *Neural Networks: Tricks of the Trade*, pages 687–707. Springer, 2012.

[31] Kang Woo-Young, Park Kyung-Wha, and Zhang Byoung-Tak. Extremely sparse deep learning using inception modules with dropfilters. In *2017 14th International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[32] J. Louradour and C. Kermorvant. Curriculum learning for handwritten text line recognition. In *2014 11th IAPR International Workshop on Document Analysis Systems*, pages 56–60, April 2014.

[33] J. A. Sánchez, V. Romero, A. H. Toselli, and E. Vidal. Icfhr2016 competition on handwritten text recognition on the read dataset. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 630–635, Oct 2016.

[34] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, May 2009.

[35] Q. Liu, L. Wang, and Q. Huo. A study on effects of implicit and explicit language model information for dblstm-ctc based handwriting recognition. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 461–465, Aug 2015.

[36] B. Moysset, T. Bluche, M. Knibbe, M. F. Benzeghiba, R. Messina, J. Louradour, and C. Kermorvant. The a2ia multi-lingual text recognition system at the second maurdor evaluation. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 297–302, Sept 2014.

[37] P. Voigtlaender, P. Doetsch, and H. Ney. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 228–233, Oct 2016.

[38] Emmanuèle Grosicki, Matthieu Carre, Jean-Marie Brodin, and Edouard Geoffrois. RIMES evaluation campaign for handwritten mail processing. In *ICFHR 2008 : 11th International Conference on Frontiers in Handwriting Recognition*, pages 1 – 6, Montreal, Canada, August 2008. Concordia University.

[39] Joan Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? 2017.

[40] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[41] Diem Markus, Kleber Florian, Fiel Stefan, Gatos Basilis, and Tobias Grüning. Icdar 2017 competition on baseline detection in archival documents. 2017.

[42] T. Bluche, B. Moysset, and C. Kermorvant. Automatic line segmentation and ground-truth alignment of handwritten documents. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 667–672, Sept 2014.

APPENDIX

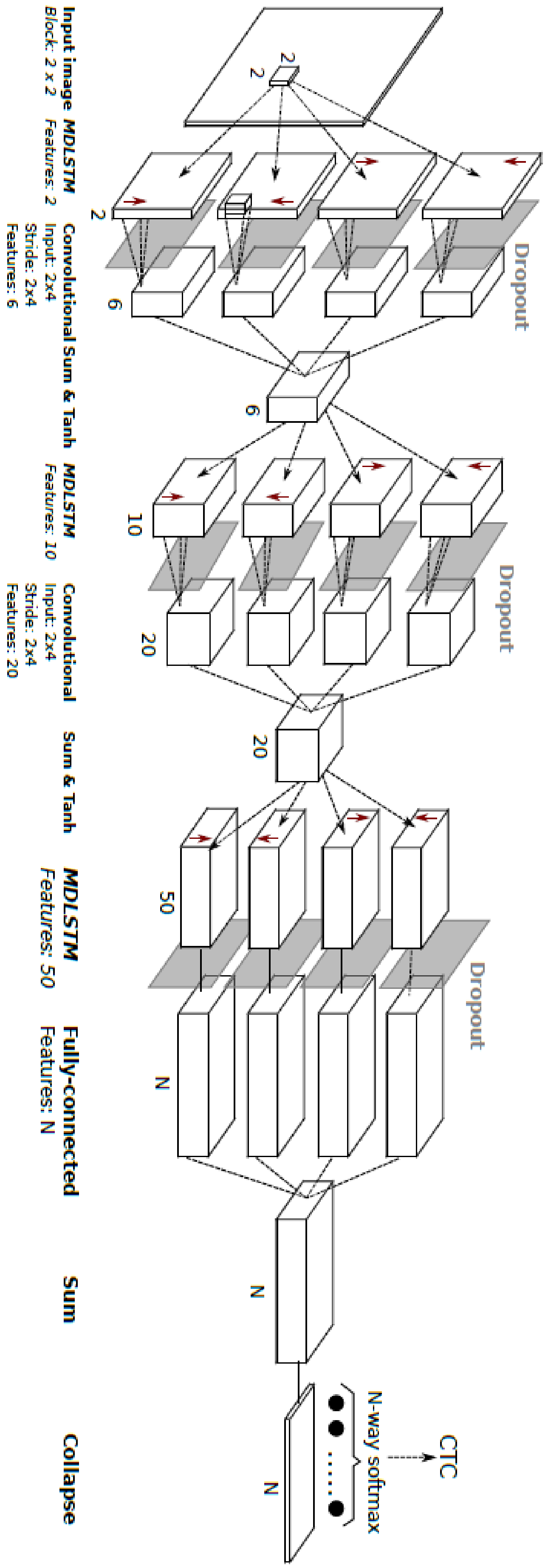

Fig. 6: architecture designed for feature extraction in HTR (source: [12])