University of Groningen

Improving the robustness of LSTMs for word classification using stressed word endings in dual-state word-beam search

Ameryan, Mahya; Schomaker, Lambert

Published in:

Proceedings of the 2020, 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)

DOI:

[10.1109/icfhr2020.2020.00014](https://doi.org/10.1109/icfhr2020.2020.00014)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version

Publisher's PDF, also known as Version of record

Publication date:

2020

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):

Ameryan, M., & Schomaker, L. (2020). Improving the robustness of LSTMs for word classification using stressed word endings in dual-state word-beam search. In *Proceedings of the 2020, 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)* (pp. 13-18). [9257737] (Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR; Vol. 2020-September). Institute of Electrical and Electronics Engineers (IEEE). <https://doi.org/10.1109/icfhr2020.2020.00014>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Improving the robustness of LSTMs for word classification using stressed word endings in dual-state word-beam search

Mahya Ameryan*, Lambert Schomaker †

Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, Faculty of Science and Engineering

University of Groningen

Groningen, The Netherlands

Email: *m.ameryan@rug.nl, †l.r.b.schomaker@rug.nl

Abstract—In recent years, long short-term memory neural networks (LSTMs) followed by a connectionist temporal classification (CTC) have shown strength in solving handwritten text recognition problems. Such networks can handle not only sequence variability but also geometric variation by using a convolutional front end, at the input side. Although different approaches have been introduced for decoding activations in the CTC output layer, only limited consideration is given to the use of proper label-coding schemes. In this paper, we use a limited-size ensemble of end-to-end convolutional LSTM Neural Networks to evaluate four label-coding schemes. Additionally, we evaluate two CTC search techniques: Best-path search vs dual-state word-beam search (DSWBS). The classifiers in the ensemble have comparable architectures but variable numbers of hidden units. We tested the coding and search approaches on three datasets: A standard benchmark IAM dataset (English) and two more difficult historical handwritten datasets (diaries and field notes, highly multilingual). Results show that stressing the word endings in the label-coding scheme yields a higher performance, especially for DSWBS. However, stressing the start-of-word shapes with a token appears to be disadvantageous.

Keywords—end-to-end convolutional LSTM neural network, historical handwriting recognition, label-coding scheme, ensemble

I. INTRODUCTION

In end-to-end training of handwriting recognizers based on long short-term memory neural networks (LSTMs) [1], it is mostly assumed that the method is powerful enough to find the proper mapping from shape elements along the x-axis (a *pseudo-time* axis) to the code elements in the target label (e.g., the letters). For example, in an Arabic isolated-word dataset IFN/ENIT [2], this mapping is facilitated by adding the bar sign, ‘|’ as an extra separator between characters and ligatures. For individual letters in a Latin-based script, it is usually assumed that there is no problem. However, the presence of blanks between words introduces shape-context effects at both the beginning and the end of words. These run-in and run-out writing-context effects exist both in full line-strip images as well as in isolated, segmented words. The context-dependency of allographs is very apparent when one wants to simulate handwriting [3, 4]. For instance, in the connected-cursive Latin-based writing, the letter ‘n’ in the

Figure 1: Different allographs, depending on the serial position in a word. These examples are taken from the MkS dataset [5]: (a) The French word, *convexe* contains two different shapes of ‘e’; (b) The German word, *Geruchsnerven*, has two allographs for the letter ‘n’.

beginning or middle of a word will often be well-formed, whereas a final letter ‘n’ in a word may consist of an oscillatory slur, Figure 1.

The IAM dataset is an English standard benchmark for words, lines and sentences, [6]. Here, the ground truth of an isolated word contains the corresponding character string. The space character between words in the line or sentence transcription is presented by the bar sign. However, each recognizer method has its specific demands. This importance was realized and addressed by introducing new shape-based alphabets or feature extraction methods [7–9]. In [7], a new alphabet is defined for the core shape of Arabic characters by separating them from diacritics. In [8], a Hidden-Markov Model (HMM [10]) is trained on a core-shape alphabet and another HMM is trained on diacritic element labels. In [9], graph-similarity features are used as the input of a HMM for handwriting recognition on a Middle High-German dataset. In [11], a multidimensional LSTM network [12] is used for paragraph recognition using a special token at the end of an input sequence. From several internal pilot studies that we conducted earlier, it was apparent that the performance of a LSTM [1] on Arabic handwriting recognition heavily depended on the used label-coding scheme. This raises the question whether more attention should be focused on the labeling of Latin-based scripts.

Table I: Four label-coding schemes.

Label-coding scheme	Notation	Example
Plain	W	<i>home</i>
Stressed word-starting	$\hat{W}$	$\hat{h}ome$
Stressed word-start and end	$\hat{W} $	$\hat{h}ome $
Extra-separator	$W $	<i>home </i>

II. DUAL-STATE WORD-BEAM SEARCH

In this paper, we use a framework consists of convolutional neural network (CNN [13]) and bidirectional long short-term memory (BiLSTM [14]) followed by a connectionist temporal classification (CTC [15]) layer. For decoding this layer, a relatively new approach is used, named the dual-state word-beam search (DSWBS), [16]. This method is based on Vanilla Beam-Search Decoding [17]. The output activation strip of a BiLSTM is one of the inputs of DSWBS. The other input is a prefix tree that is constructed from the given lexicon. The method has two states: a word-state and a non-word-state. The temporal evolution of a beam depends on the state transitions. A character is considered either as *word-character* or as *non-word-character*. In the non-word-state, the beam can be extended by a *non-word-character*. The entry of a *word-character* carries the system from non-word-state into the word-state. This *word-character* therefore identifies the start of a word. The further extension of a word in the word-state is possible by using the prefix tree, checking the plausibility of the path at that point in the sequence.

III. METHOD

In this section, first, we explain four label-coding schemes. Then, we use the ensemble of five CNN-LSTMs suggested in [18], for acknowledging their effect. Such networks can handle sequence variability and geometric variation.

A. Label-coding scheme methods

In RNNs, the precise correlation between the word image and its ground-truth label is not apparent. Therefore, for prediction at every time step, a probability distribution is computed over possible tokens. This makes the importance of a proper label-coding scheme very clear. The question can be asked is: "How to model and label transitional patterns that exist between the major patterns, e.g., words?". Hence, in this paper, we explore whether a stressed word start and ending in the ground truth helps the neural network. Using stressed word-start and ending signs results in four possible label-coding schemes (Table I):

- 1) *W* The Plain label-coding scheme: The corresponding ground-truth label exclusively shows the characters which are present in the word image, [18].
- 2) $\hat{W}$ The stressed word-start label-coding scheme: An extra character (e.g., $\hat{''}$) is added to the ground-truth label of a word image to provide the recognizer an extra hint relating the start-of-word shape condition. This character must not appear in the original character set of the given dataset and is required to be the same for all ground truths.
- 3) $\hat{W}|$ The stressed word-start and end label-coding scheme: Two extra characters are defined as word-starting and ending signs. This gives extra information to

Table II: Configuration of each of the five recognizers used in the ensemble from the input image (bottom) to output (top). 'K', 'W', 'S' and 'P' denote kernel size, window size, stride and padding, [18].

Layer		Configuration
Output string		Label hypotheses
Transcription		A dual-state word-beam search CTC decoding
Bidirectional-LSTM	L1	512 hidden units
	L2	512 hidden units
	L3	512 hidden units
Max Pooling		W and S:1×2
non-linear ReLU		-
Normalization		-
Convolution		K:3 × 3, S:1, p:1
Max Pooling		W and S:1×2
non-linear ReLU		-
Normalization		-
Convolution		K:3 × 3, S:1, p:1
Max Pooling		W and S:1×2
non-linear ReLU		-
Normalization		-
Convolution		K:3 × 3, S:1, p:1
Max Pooling		W and S:2×2
non-linear ReLU		-
Normalization		-
Convolution		K:3 × 3, S:1, p:1
Max Pooling		W and S:2×2
non-linear ReLU		-
Normalization		-
Convolution		K:3 × 3, S:1, p:1
input image		128×32 gray-scale image

↑

the recognizer about the conditions of start-of-word and end-of-word shape contexts. These two characters should not be present in the original character set of the given dataset.

- 4) $W|$ The extra-separator label-coding scheme: In order to provide more information about the condition of the end-of-word shape, an extra, unique character (e.g., '|') is appended to the ground-truth label.

The stressed word starting and ending signs should not exist in the given lexicon. Please note that the tests are not on line strips but are on words.

B. Recognizer

We use an ensemble framework consisting of end-to-end trained networks [18]. This approach consists of five individual CNN-LSTM networks and one voter module. The network is summarized in Table II. This setup showed a good performance in the earlier work [18]. Moreover, the ensemble does not need handcrafted features or extensive network-architecture engineering efforts.

1) *Pre-processing and data augmentation*: The preprocessing stage consists of three steps: a) randomly stretching/squeezing the input image in the direction of width. b) mapping the image into an image block-sized 128 × 32 pixels and c) a normalization process. Please note that the preprocessing step does not contain baseline alignment or deslanting. Augmentation is applied to the training set in such a way that the natural samples themselves are not present in the training set.

Table III: The number of hidden units in front-end CNN. Each row shows the number of hidden units of a CNN used in five Architectures ($A_1 \dots A_5$). Each CNN has five layers ($l_1 \dots l_5$), as in [18].

Arc.	Hidden unit size				
	l_1	l_2	l_3	l_4	l_5
A_1	128	256	256	256	512
A_2	128	256	512	512	512
A_3	128	128	256	256	512
A_4	128	128	512	512	512
A_5	128	128	128	256	512

2) *CNN*: Afterward, for feature extraction, the pixel-intensity values are fed into the first layer of a CNN. The CNN contains of five layers, which each layer consists a convolution operation, intensity normalization, the ReLU activation function [19], and a max-pooling layer. Each layer has kernel filters sized 3×3 . Apart from the number of units in the hidden layers, important hyperparameter values are the same in the classifiers of the ensemble. Table III shows the number of hidden layers, as in [18]. The batch size is 50. The individual recognizers used in [18] were inspired by [20]. However, for gradient descent, RMSProp [21] was used instead of ADADELTA [22] used in [20]. Furthermore, the number of convolutional layers in [18] was five, which is two layers less than the method in [20]. There is no dropout.

3) *BiLSTM*: The extracted features by the front-end CNN is the input of a three-layer BiLSTM. Since the last layer of the CNN has 512 hidden units, the number of hidden units in each BiLSTM layer is 512.

4) *Connectionist temporal classification (CTC)*: The three-layer BiLSTM is followed by CTC. The number of units in the output layer of CTC differs for each label-coding scheme.

- 1) *Plain*: When the Plain label-coding scheme is used, the number of units of the CTC output layer is unit more than the size of the character set of the goal dataset. This extra unit belongs to a common blank, which is different from the space character between the words [15]. The activation of this unit represents the probability of observing a common CTC blank, or no label at a given time step.

$$A' = A \cup \text{blank} \quad (1)$$

Here, A and A' denote the size of the original character set and the size of the CTC output layer.

- 2) *The stressed word-starting label-coding scheme*: In the case of using this label-coding scheme, one unit exists in the CTC output layer presenting the stressed word-starting sign. Therefore, the size of this layer is $|A|+2$.

$$A' = A \cup \text{start sign} \cup \text{blank} \quad (2)$$

- 3) *The stressed word-starting and ending label-coding scheme*: When this label-coding scheme is applied the CTC output layer has three units more than the size

of character set of the original dataset (A): one for the stressed word-starting sign (e.g., ``^''), one for stressed word-starting sign (e.g., ``|'') as an extra separator, and a common blank for CTC. Therefore, the final character set of CTC output is:

$$A' = A \cup \text{starting sign} \cup \text{extra separator} \cup \text{blank} \quad (3)$$

- 4) *The extra-separator label-coding scheme*: by using this label-coding scheme, the size of CTC output layer is $|A|+2$. The extra two is for the common blank and the extra separator (eg, ``|'').

$$A' = A \cup \text{extra separator} \cup \text{blank} \quad (4)$$

Regardless of the label-coding approach, we use a dual-state word beam search [16] explained in II.

C. The ensemble system

The output of a CTC for a given image is a string of characters representing a word hypothesis with its corresponding likelihood. The voter module receives one hypothesis from each of the five CNN-LSTM recognizers and subsequently applies *Plurality voting* on them [23]. The hypothesis with the most votes wins the plurality-voting method. If a tie occurs, the winner is the hypothesis with the highest averaged likelihood. If all recognizers provide a different hypothesis, we accept the one with the highest likelihood. The winner will provide the final label of the input image. Plurality voting with two exception rules was selected over Borda-count voting [24] after a pilot experiment because it showed better results. This plurality-voting method is summarized as follows:

- 1) There is plurality $\rightarrow$ take it.
- 2) There is a tie $\rightarrow$ take subset with the best average likelihood.
- 3) Only a single vote per class $\rightarrow$ take hypothesis with the best likelihood.

IV. RESULTS

In this section, first, the datasets used in our experiments are described. Afterward, the condition of our experiments is explained. Finally, the numerical results are reported.

A. Datasets

In this paper, three handwritten datasets were used which differ in the levels of difficulty, summarized in Table IV. The first is a commonly used standard benchmark dataset, IAM [6]. IAM contains 78 characters including alphabetic characters in lower and upper case, digits 0 to 9 and signs in the set $S = \{-, ; : ! ? / . ' " () * \& \# +\}$. In our evaluation, all characters were allowed, and it was conducted case-sensitively. As it is common, we use the correctly-segmented images for evaluation, [25].

The second dataset, *van Oort*, is derived from the field diaries of Pieter van Oort (years 1825-1834) collected in the Indonesian Archipelago. Van Oort was one of the best expert

Table IV: Characteristics of the three datasets (IAM, van Oort and MkS) used in our experiments showing: Number of word images; Number of words for case sensitive (CS) and case insensitive (CI) methods.

set	IAM			Van Oort			MkS		
	Images	Words $_{CS}$	Words $_{CI}$	Images	Words $_{CS}$	Words $_{CI}$	Images	Words $_{CS}$	Words $_{CI}$
Train	53,839	7,748	7,140	9,177	3,650	3,484	5,778	2,648	2,497
Validation	16,465	3,928	3,696	3,059	1,696	1,641	1,926	1,188	1,142
Test	13,753	3,107	2,904	3,059	1,696	1,641	1,926	1,188	1,142
Total dataset	84,057	7,748	10,112	15,295	5,066	4,813	9,630	3,730	3,494

draftsmen of natural specimens in the early nineteenth century Netherlands. The rate of out-of-vocabulary (OOV) words has 26.2% case-sensitively and 24.6% case-insensitively.

The third dataset, *MkS*, concerns fieldnotes of the Natuurkundige Commissie in the Indonesian Archipelago between 1820-1850 [5]. The manuscript is highly multilingual, such as German, Latin, Dutch, Malay, Greek, and French (see Figure 2). The rate of OOV words is 31.9% case-sensitively and 29.5% case-insensitively. The lexica of *van Oort* and *MkS* contain words with alphabetic characters. The evaluations on van Oort and MkS were conducted case-insensitively. We ran 5-fold cross-validation for assessing a more accurate estimation of the performance of our model on these two datasets. It was no attempt to balance classes and instances. Therefore, the folds are quite diverse.

B. Quantitative results

We implemented the CNN-LSTM model in Tensorflow framework. The hat character (^) as the stressed-starting sign and the bar character (|) as the stressed-ending sign were chosen. These two characters do not appear in the lexicon of IAM, van Oort, and MkS. Figure 2 shows some samples of MkS, annotated using the extra-separator label-coding scheme. The networks were trained from the scratch. The comparison of word accuracy (%) between four label-coding schemes on IAM is reported in Table V. The comparison is conducted using the best-path CTC decoder (BP) [26] and the DSWBS using the standard dictionary in terms of average $\pm$ standard deviation. The result shows $W|$ out-

Table V: The comparison of word accuracy (%) between the four label-coding schemes on IAM using the best-path CTC decoder (BP) and the dual-state word-beam search (DSWBS) in terms of average $\pm$ standard deviation.

Label-coding scheme	Plain	
	BP	DSWBS
W	75.55 $\pm$ 0.37	87.65 $\pm$ 0.18
$\hat{W}$	72.73 $\pm$ 1.92	85.39 $\pm$ 2.06
$\hat{W} $	74.24 $\pm$ 1.66	89.05 $\pm$ 1.3
$W $	75.55 $\pm$ 0.25	89.97 $\pm$ 0.23

Figure 2: The samples of the strongly multilingual MkS dataset. (a) and (b): German, (c) and (d) Dutch, (e): Latin, and (f): French. The images labeled using extra-separator label-coding scheme.

performs the other label-coding schemes. The second-best result is achieved by $\hat{W}|$ using DSWBS. While $\hat{W}$ gives the least performance. The results on IAM confirm that: using DSWBS improves the performance (Chi-squared test, $p < 0.05$, significant); using $W|$ increases the performance (Chi-squared test, $p < 0.05$, significant); using DSWBS and $W|$ results in the highest performance.

The Table VI shows the result on IAM. The comparison with the state-of-the-art methods on IAM is shown in Table VII. The number of recognizers in each method is denoted by n_{rec} . Checking the homogeneity ($Hom.$) is not applicable (N/A) on the single methods. If the used lexicon contains digits and signs (78 characters) '*Punct.*' has ($\checkmark$). '*Case*' shows whether the evaluation is conducted case-sensitively. As it is not clear in [28] whether the evaluation is conducted case-sensitively, we report as unknown (?). It is written in [31] that the evaluation in [28] is done case-insensitively. Pre_{tr} is denoted pre-trained recognizer. The $word_{acc}$ shows word accuracy. The approximate sign ($\approx$) shows that, in [30], the word images containing only punctuation marks are removed from the test set (84.3% of the test set used in [30]).

Moreover, the augmentation approaches of these state-of-the-art methods is summarized here. In [29], the rotation and warping techniques were applied for data augmentation. In [27], the RIMES [32] and NIST datasets were used to pre-train the recognizer. In [28], the intensive preprocessing method made the train set, and the test set 37 times bigger. In [31], the network is trained on the IIIT-HWS dataset [33]. In [28, 30, 31], augmentation methods are applied on the test set. Then, the best output is chosen as the final result.

We performed 5-fold cross-validation on the van Oort and the MkS dataset. For each dataset, the five architectures were trained using the Plain and the Extra separator coding schemes separately. It resulted in 50 trained networks for each dataset (5 architectures $\times$ 2 coding schemes $\times$ 5 folds). Afterward, each trained network was evaluated twice on the test set, using the dictionary-free BP and the DSWBS CTC decoders, resulting in 100 evaluations for each dataset.

Table VI: The result of the IAM dataset. The Table shows the comparison of word accuracy (%) between two label-coding schemes (Plain and Extra separator) using the best-path CTC decoder (BP) and the dual-state word-beam search (DSWBS) applying the standard dictionary, and the ensemble in terms of average $\pm$ standard deviation (avg $\pm$ sd).

Label-coding scheme		Plain		Extra separator	
Arch.	CTC decoder	BP	DSWBS	BP	DSWBS
A ₁		75.63	87.68	76.17	90.13
A ₂		75.83	87.86	75.58	89.65
A ₃		75.53	87.77	75.57	89.76
A ₄		75.69	87.78	75.79	90.24
A ₅		74.55	86.92	75.05	90.07
avg $\pm$ sd		75.55 $\pm$ 0.37	87.65 $\pm$ 0.18	75.55 $\pm$ 0.25	89.97 $\pm$ 0.23
Ensemble		79.90	90.24	79.95	92.21

Table VII: The comparison of our system to the state-of-the-art methods on the IAM dataset in terms of number of recognizers (n_{rec}), homogeneity of the algorithm (Hom.), recognizing punctuation and digits $Punct.$, case sensitivity ($Case$), using pre-trained recognizer Pre_{tr} , and word accuracy (%) ($word_{acc}$). Please, refer to the text for further explanation.

system	n_{rec}	Hom.	$Punct.$	$Case$	Pre_{tr}	$word_{acc}$
Ours	1	N/A	✓	✓	-	89.83 $\pm$ 0.32
Ours (Table III)	5	✓	✓	✓	-	92.21
Ptucha et al. 2019 [27]	3	-	✓	✓	✓	91.78
Poznanski and Wolf 2016 [28]	1	N/A	-	?	-	93.55
Stuner et al. 2016 [29]	1,039	-	✓	✓	-	94.07
Wigington et al. 2017 [30]	1	N/A	≈	✓	-	94.17
Dutta et al. 2018 [31]	1	N/A	-	-	✓	95.20

Table VIII: The result of the van Oort and the MkS datasets. Please, refer to the text for further explanation.

(a) The van Oort dataset

Label-coding scheme		Plain		Extra separator	
Arch.	CTC decoder	BP	DSWBS	BP	DSWBS
A ₁		64.91	72.64	65.86	89.87
A ₂		65.49	70.94	65.68	90.14
A ₃		65.52	69.92	64.90	89.88
A ₄		65.20	68.76	65.80	89.91
A ₅		63.48	67.44	64.44	89.11
avg $\pm$ sd		64.92 $\pm$ 0.83	86.85 $\pm$ 0.73	65.34 $\pm$ 1.35	89.78 $\pm$ 0.68
Ensemble		71.61 $\pm$ 0.57	89.30 $\pm$ 0.61	72.58 $\pm$ 0.67	92.02 $\pm$ 0.41

(b) The MkS dataset

Label-coding scheme		Plain		Extra separator	
Arch.	CTC decoder	BP	DSWBS	BP	DSWBS
A ₁		55.02	80.19	53.57	82.65
A ₂		55.39	80.91	54.30	83.29
A ₃		52.97	78.99	54.27	81.86
A ₄		53.56	79.56	53.26	82.22
A ₅		52.62	78.55	54.80	82.58
avg $\pm$ sd		53.91 $\pm$ 1.33	79.64 $\pm$ 1.06	54.04 $\pm$ 1.14	82.52 $\pm$ 1.05
Ensemble		61.11 $\pm$ 1.37	83.30 $\pm$ 0.59	61.57 $\pm$ 1.52	85.72 $\pm$ 0.70

The Table VIII shows the comparison of word accuracy (%) between two label-coding schemes (Plain and Extra separator) using BP and DSWBS applying the standard dictionary, and the ensemble on van Oort and MkS in terms of average $\pm$ standard deviation. The results in Tables VI and VIII reveal that: *Best path vs Dual-state word-beam search*: The DSWBS search outperforms BP, as expected. *Single network vs Ensemble*: The ensemble increases the performance, from 2 percentage points (pp) to 7pp. The positive effect of the ensemble on the weaker recognizers is more. *Plain vs Extra separator*: Extra separator outperforms Plain (Chi-squared test, $p < 0.05$). Moreover, using the extra-separator label-coding scheme boosts the performance

more, when the DSWBS is used as a CTC decoder.

V. DISCUSSION AND CONCLUSION

Surprisingly, it is helpful to stress the word ending with an extra token that is to be associated with the end-of-word shape in the input image by the CNN-LSTM network. Apparently, the LSTM is helped by explicit end-of-word stress. It should be mentioned the beneficial effect of the vertical bar in labeling is not caused by image white space or variable white space in the word endings, because the van Oort and MkS datasets were tightly cropped. The improved performance is most apparent when the dual-state word-beam search (DSWBS) is applied as the CTC

decoder. Finally, it should be noted that the use of the ensemble increases the single-recognizer performance 2-3pp relative to the DSWBS performance while yielding a 4-7pp improvement in the case of raw best-path search.

In this paper, we proposed an effective label-coding scheme in DSWBS for CTC. We used a limited-size ensemble of five end-to-end convolutional BiLSTM neural networks to evaluate several label-coding schemes. We found out that stressing the end-of-word shape clearly increases the performance when DSWBS is used. Other label-coding schemes had a less clear effect, and stressing the begin-of-word condition with a token decreases the accuracy. The ensemble of five networks improves the results, as in [18]. For future work, we want to extend our approach to address the line recognition task and Zero-shot learning [34].

VI. ACKNOWLEDGMENT

This work is part of the research program Making Sense of Illustrated Handwritten Archives with project number 652-001-001, which is financed by the Netherlands Organization for Scientific Research (NWO). We would like to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing cluster.

REFERENCES

- [1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, pp. 1735–1780, 1997.
- [2] M. Pechwitz, S. S. Maddouri, V. Mrgner, N. Ellouze, and H. Amiri, "IFN/ENIT database of handwritten Arabic word," in *7th Colloque Int. Francophone sur l'Ecrit et le Document*, 2002.
- [3] H. CHOI and J. hyung, "Probabilistic synthesis of personal-style handwriting," *IEICE Trans. Inf. Syst.*, pp. 653–661, 2009.
- [4] J.-P. Orliaguet, S. Kandel, and L.-J. Bo, "Visual perception of cursive handwriting: Influence of spatial and kinematic information on the anticipation of forthcoming letters," *Perception*, vol. 26, pp. 905–912, 1997.
- [5] A. Weber, M. Ameryan, K. Wolstencroft, L. Stork, M. Heerlien, and L. Schomaker, "Towards a digital infrastructure for illustrated handwritten archives," in *Final Conf. of the Marie Skłodowska-Curie Initial Training Network for Digital Cultural Heritage*, M. Ioannides, Ed. Springer, 2017.
- [6] H. B. U.-V. Marti, "The IAM-database: an English sentence database for offline handwriting recognition," *Int. J. on Document Anal. and Recognit.*, vol. 5, 2002.
- [7] F. Menasri, N. Vincent, M. Cheriet, and E. Augustin, "Shape-based alphabet for off-line Arabic handwriting recognition," in *Ninth Int. Conf. on Document Anal. and Recognit.*, 2007, pp. 969–973.
- [8] I. Ahmad and G. A. Fink, "Multi-stage HMM based arabic text recognition with rescoring," in *13th Int. Conf. on Document Anal. and Recognit.*, 2015, pp. 751–755.
- [9] A. Fischer, K. Riesen, and H. Bunke, "Graph similarity features for HMM-based handwriting recognition in historical documents," in *12th Int. Conf. on Frontiers in Handwriting Recognit.*, 2010, pp. 253–258.
- [10] L. E. B. T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *Ann. Math. Statist.*, pp. 1554–1563, 1966.
- [11] T. Bluche, J. Louradour, and R. Messina, "Scan, attend and read: End-to-end handwritten paragraph recognition with MDLSTM attention," *Arxiv:1604.0328*, 2016.
- [12] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Inf. Process. Syst.*, 2009, pp. 545–552.
- [13] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. MIT Press, 1998.
- [14] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, May 2009.
- [15] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *23rd Int. Conf. on Machine Learning*, 2006, pp. 369–376.
- [16] H. Scheidl, S. Fiel, and R. Sablatnig, "Word beam search: A connectionist temporal classification decoding algorithm," in *The Int. Conf. on Frontiers of Handwriting Recognit.* IEEE Comput. Soc., 2018, pp. 253–258.
- [17] K. Hwang and W. Sung, "Character-level incremental speech recognition with recurrent neural networks," in *IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, 2016, pp. 5335–5339.
- [18] M. Ameryan and L. Schomaker, (2019) A limited-size ensemble of homogeneous CNN/LSTMs for high-performance word classification. [Online]. Available: <https://arxiv.org/abs/1912.03223>
- [19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *27th Int. Conf. on Machine Learning*, 2010, pp. 807–814.
- [20] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 2298–2304, 2017.
- [21] T. Tieleman and G. Hinton, "Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for Mach. Learning*, vol. 4, no. 2, pp. 26–31, 2012.
- [22] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.
- [23] B. Peleg, "Consistent voting systems," *Econometrica: J. of the Econometric Society*, pp. 153–161, 1978.
- [24] Tin Kam Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 66–75, Jan 1994.
- [25] J. Sueiras, V. Ruiz, A. Sanchez, and J. F. Velez, "Offline continuous handwriting recognition using sequence to sequence neural networks," *Neurocomputing*, pp. 119–128, 2018.
- [26] A. Graves, *Supervised sequence labelling with recurrent neural networks*. Springer-Verlag Berlin Heidelberg, 2012.
- [27] R. Ptucha, F. P. Such, S. Pillai, F. Brockler, V. Singh, and P. Hutkowski, "Intelligent character recognition using fully convolutional neural networks," *Pattern Recognit.*, pp. 604–613, 2019.
- [28] A. Poznanski and L. Wolf, "CNN-N-Gram for handwriting word recognition," in *IEEE Conf. on Comput. Vision and Pattern Recognit.*, 2016, pp. 2305–2314.
- [29] B. Stuner, C. Chatelain, and T. Paquet, (2016) Handwriting recognition using Cohort of LSTM and lexicon verification with extremely large lexicon. [Online]. Available: <https://arxiv.org/abs/1612.07528>
- [30] C. Wigginton, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen, "Data augmentation for recognition of handwritten words and lines using a CNN-LSTM Network," in *14th IAPR Int. Conf. on Document Anal. and Recognit.*, 2017, pp. 639–645.
- [31] K. Dutta, P. Krishnan, M. Mathew, and C. V. Jawahar, "Improving CNN-RNN hybrid networks for handwriting recognition," in *16th Int. Conf. on Frontiers in Handwriting Recognit.*, Aug 2018, pp. 80–85.
- [32] E. Grosicki, M. Carre, J.-M. Brodin, and E. Geoffrois, "RIMES evaluation campaign for handwritten mail processing," in *11th Int. Conf. on Frontiers in Handwriting Recognit.*, 2008, pp. 1–6.
- [33] A. Vinciarelli and J. Luetttin, "A new normalization technique for cursive handwritten words," *Pattern Recognit. Lett.*, pp. 1043–1050, 2001.
- [34] S. Chanda, J. Baas, D. Haitink, S. Hamel, D. Stutzmann, and L. Schomaker, "Zero-shot learning based approach for medieval word recognition using deep-learned features," in *16th Int. Conf. on Frontiers in Handwriting Recognit.*, 2018, pp. 345–350.