A limited-size ensemble of homogeneous CNN/LSTMs for high-performance word classification

Mahya Ameryan¹ · Lambert Schomaker¹

Received: 26 December 2019 / Accepted: 11 December 2020 / Published online: 1 February 2021
© The Author(s) 2021

Abstract

The strength of long short-term memory neural networks (LSTMs) that have been applied is more located in handling sequences of variable length than in handling geometric variability of the image patterns. In this paper, an end-to-end convolutional LSTM neural network is used to handle both geometric variation and sequence variability. The best results for LSTMs are often based on large-scale training of an ensemble of network instances. We show that high performances can be reached on a common benchmark set by using proper data augmentation for just five such networks using a proper coding scheme and a proper voting scheme. The networks have similar architectures (convolutional neural network (CNN): five layers, bidirectional LSTM (BiLSTM): three layers followed by a connectionist temporal classification (CTC) processing step). The approach assumes differently scaled input images and different feature map sizes. Three datasets are used: the standard benchmark RIMES dataset (French); a historical handwritten dataset KdK (Dutch); the standard benchmark George Washington (GW) dataset (English). Final performance obtained for the word-recognition test of RIMES was 96.6%, a clear improvement over other state-of-the-art approaches which did not use a pre-trained network. On the KdK and GW datasets, our approach also shows good results. The proposed approach is deployed in the Monk search engine for historical-handwriting collections.

Keywords Coding scheme · Ensemble system · End-to-end convolutional long short-term memory · Connectionist temporal classification

1 Introduction

Convolutional neural networks (CNNs) [1] and long short-term memory networks (LSTM) [2] and its variants [3, 4] have recently achieved impressive results [5–7]. This exceptional performance comes, however, at the cost of having an ensemble of, e.g., 100–2000 recognizers [8]. The high cost of training and operation brings to mind the question of whether less costly methods can be applied to boost the performance of handwriting recognizers.

A possible direction would consist of the use of linguistic statistics [9]. A recent method for using language

information is a dual-state word-beam search [10] for decoding the connectionist temporal classification (CTC [11]) layer of neural networks, which has been shown to be effective [10].

Although the presence of dictionaries and corpora is beneficial, historical documents present a challenge. For instance, the historic spelling of a word differs from the contemporary spelling, there is often an absence of strict orthography, and there may be frequent misspellings [12]. Figure 1 shows a word image from one of the datasets used in this paper. This historical word has an extra character compared to the current spelling. Moreover, for rare languages, e.g., Aymara [13], the complete lexicon does not exist yet, and corpora are of very limited size. Handwritten-text recognition (HTR) is exactly required to *obtain* such digital linguistic resources for that language.

Another possible direction to improve performance would concern a heavy optimization of network architecture and training (hyper)parameters. The state-of-the-art

✉ Mahya Ameryan
m.ameryan@rug.nl; mahya.ameryan@gmail.com
Lambert Schomaker
l.r.b.schomaker@rug.nl

¹ Artificial Intelligence and Cognitive Engineering, Faculty of Science and Engineering, University of Groningen, Groningen, The Netherlands

Fig. 1 A historical spelling of a word, *Afdeeling*, in the historical KdK dataset. The contemporary spelling of this word would be *Afdeling*

approaches can be sensitive to the choice of hyper-parameter values. As an example, it is reported that increasing the depth of a neural network that consists of convolutional and LSTM layers, from 8 hidden layers to 10, is advantageous. Further enlarging to 12 hidden layers yielded unsatisfactory results [14]. From the perspective of e-Science services for handwriting recognition dealing with hundreds of different books, it is not feasible to tailor the recognizer models for each book based on prior knowledge, using human handcrafting of neural networks. An e-Science server is the application of computationally intensive modern scientific procedures for data gathering, preparation, experimentation, result distribution, and long-term maintenance. An e-Science server can include data modeling, digitized datasets, and analysis, e.g., the Monk system [15–18]. For an e-Science server, preferably, having an ensemble consisting of a limited number of automatically generated neural-network architectures would be practical. The Monk e-Science server is a live web-based search engine for words and character recognition, retrieval, and annotation. It contains diverse digitized historical and contemporary handwritten manuscripts in many languages: Chinese, Thai, Arabic, Dutch, English, Persian. Also, complicated machine-printed documents such as German, Fraktur, Egyptian hieroglyphs, and historical language are available in the Monk system. For such a system with almost 600 manuscripts, it is not feasible to use human effort for fine-tuning a model for each of the manuscripts to reach higher accuracy as [8]. An essential consideration is that it should be possible to add our suggested algorithm to the Monk system, with a minimum of required operational human effort.

In this paper, we explore the possibilities of exploiting the success of current CNN/LSTM approaches, using several methods at the level of linguistics and labeling systematics, as well as an ensemble method. Ideally, the approach should be robust, require a minimum of human intervention with a limited set of hyper-parameter settings (architectures), and minimum linguistic resources. For evaluation, we use a standard benchmark public dataset, RIMES [19], a historical handwritten dataset, KdK [15, 20], and the standard public benchmark George Washington dataset (GW [21]). The three datasets differ in historical period and language. The purpose of this paper is

not to handcraft a model to achieve maximum accuracy on a particular dataset but is to design a robust, high-performing recognizer that can be deployed in an e-Science server such that training occurs largely autonomously and no hyper-parameter setting should be necessary. This is important because the number of collections and the variation in styles precludes individual attention by human operators *in the back office*. In other words, the goal of this paper is to use neural networks for real-world applications involving large collections in a massive high-performance computing context.

The rest of this paper is structured as follows. In Sect. 2, we briefly survey the related works in terms of recent state-of-the-art methods and word search approaches in character-hypothesis grids. In Sect. 3, the requirements of the proposed method are explained. In Sect. 4, we present our system. The experimental evaluation and discussion are given in Sects. 5 and 6. Finally, conclusions are drawn in Sect. 7.

2 Related work

In this section, we first briefly survey the state of the art on handwriting recognition task. Afterward, we survey part of the long history of word search in character-hypothesis grids and linguistic post-processing.

2.1 The state of the art on handwriting recognition task

Offline handwriting recognition classifiers typically use the direct image values or the extracted features from an input image to predict posterior probabilities [22, 23]. These classifiers, e.g., neural networks (NNs), and hidden Markov models (HMMs) have their own merits and demerits. HMMs are relatively simple, and rely on strong assumptions. But one of the main drawbacks of HMMs is the weakness in modeling long-term input data dependency. Two well-known NNs are recurrent neural networks (RNNs) and convolutional neural networks (CNN [1]). CNNs are able to learn important features without human interference. Additionally, the convolutional approach makes CNNs translation invariant and the pooling layers make them relatively insensitive to scale variation. A distinct disadvantage is the reliance on a fixed-size input image. This is undesirable for text processing, with its variable-length sequential patterns. On the contrary, RNNs and specifically LSTMs have shown remarkable success in various sequence learning tasks [5–7, 24, 25].

Combinations of classifiers as pipeline methods and heterogeneous/homogeneous ensembles are used to reach better performance. One common pipeline concerns

ensembles with different RNN-based models using different feature extraction [7, 26] and different decoding methods [8, 14, 27–29].

CNNs are sometimes used as feature extraction method for classifiers, in particular LSTMs [10, 26, 28, 30]. In [30], a framework consisting of a deep CNN, LSTM layers as encoder/decoder, and a attention mechanism for isolated handwritten-word recognition is given. The result is reported with/without dictionary. For pre-processing, methods for baseline correction, normalization, and deslanting are applied. After pre-processing, an input image is converted to a sequence of image patches by using a horizontal sliding window. A deep CNN is used for feature extraction. Afterward, an LSTM is applied to extract the horizontal relationships existing among a sequence of overlapped horizontal patches of input images. A decoder component is used, a combination of an LSTM and an attention mechanism. To find the best performance, experiments are done to determine the optimal LSTM cell size and patch size. Although the overall architecture is interesting, this method [41] does not have a very high performance. In [28], a spatial transformer network, residual convolutional blocks (ResNet-18), stacked BiLSTMs and a CTC layer are used. Deslanting and slope normalization is performed on images, using the approach presented in [31]. A CNN-RNN is pre-trained on the IIIT-HWS dataset [32]. During training and testing on benchmark datasets, three types of augmentations are used: affine transformation; elastic distortion; multi-scale transformation. Each test image is augmented 25 times. This type of augmentation at in the operational stage has been reported earlier in other applications [33, 34] involving animal recognition.

In [35], a 12-layer convolutional neural network (CNN) is used to processes fixed-sized word images and recognize a Pyramidal Histogram of Character (PHOC) representation [36], using multiple parallel fully connected layers. Afterward, canonical correlation analysis (CCA) [37] is applied as a final stage of the word recognition task, using a predefined lexicon.

In [38], a whole-word CNN can be apply to recognize known words, defined as the 500 most frequent words in the training set of the RIMES dataset, which have a minimum confidence level of 70%. Otherwise, a Block Length CNN predicts the number of symbols in the given image block. Then, a fully convolutional neural network predicts the characters. Finally, the result is enhanced by a vocabulary-matching method. This varied-CNN method has a problem with separating common and non-common words. The separation of lexicon into a set of common and a set of uncommon words may be artificial, in view of the usual continuously decaying Zipf distribution [39].

There are two key solutions for a handwriting prediction output to be converted into a character sequence. The first

approach is using an HMM [7, 40]. HMM is the most traditional way to detect a word. The second is using CTC [11]. The approach of using an RNN followed by a CTC layer has been widely used [8, 14, 27–29].

The successful methods are unfortunately quite complicated. Most of them use a combination of CNNs and LSTMs. Therefore, it is important to consider more integrated, simplified approaches, such as a convolutional LSTM [26]. It will be treated in Sect. 3.1.

2.1.1 Ensemble system

A simple but effective method for improving an individual classifier performance is the ensemble method [27, 41–49]. In [42], it is shown that having diverse classifiers is a key point for classifier fusion. Using ensembles for handwriting recognition with hidden-Markov models as basic word classifiers, Günter and Bunke [43] compares different ensemble creation methods: Bagging, AdaBoost, Half & half bagging, random subspace, architecture as well as different voting combination methods for handwriting-recognition task. It is shown that each of these four methods increases performance.

The impact of dictionary size, the training set size, and the number of recognizers in ensemble systems are studied for off-line cursive handwritten-word recognition in [44]. The ensemble methods are Bagging, AdaBoost and the random subspace, while the recognizers used are HMMs with different configurations. It is verified that increasing the size of the training set and the number of recognizers elevates the performance of the system, while the larger dictionary pulls down the performance.

Recently, in [45], ensemble classifiers for Persian handwriting recognition were used. This study used AdaBoost and Bagging to combine weak classifiers created from hand-crafted families of simple features.

In the deep learning domain, Yang et al. [46] obtained very high accuracy for Chinese handwritten character recognition using deep convolutional neural networks and a hybrid serial-parallel ensemble strategy which tries to find an ‘expert’ network for each example that can classify the example with high accuracy, or if such a network cannot be found, falls back to the majority vote overall networks.

An ensemble of NNs and HMM methods is used in [27]. This ensemble uses eight recognizers for handwriting recognition which includes four variants of a multidimensional long short-term memory neural network (MDLSTM [4]), a grapheme-based MLP-HMM, and two variants of a context-dependent sliding window based on Gaussian Mixture Model (GMM-HMM). The ensemble system is a simple sum rule. This example illustrates that some studies involve highly complicated and heterogeneous algorithm

architectures requiring a lot of traditional engineering efforts.

In an ensemble system, majority or, alternatively, the plurality can be used if the output of each individual recognizer is only the best hypothesis label. If recognizers of the ensemble system produce a ranked hypotheses list, Borda count is possible [47, 48] to determine the result. In this case, it is required that the ranked list shows a sufficient diversity of intuitive candidates, i.e., with a low edit distance [50] from the target. Two ensemble system of handwritten recognition methods are presented in [49]: using word-list merging; and linear combination.

In [8], two architectures are used to generate more than a thousand networks to construct an ensemble. Each network is either a two-layer BiLSTM or a three-layer MDLSTMs. BiLSTMs are fed by HOG [51], and the input of the MDLSTM is raw images. The *best path* algorithm [52] is applied for CTC decoding. This approach uses a lexicon verification method. After training 2100 networks and evaluating the validation set of RIMES dataset, the lowest performance networks are removed, which results in 118 networks. It is reported that the pruned ensemble of 118 networks has a 0.16 pp drop in performance compared to the ensemble of 2100 networks on the RIMES dataset. On another dataset, IAM [53], the size of ensemble is different ($n_{\text{rec}} = 1039$). Because of the simplicity of the system and the high number of recognizers, the complexity is medium to high.

The good results represented in literature are often based on a fairly complex system with many hyperparameters. In an e-Science service such as Monk which currently has about almost 600 different manuscripts, it is clear that human attendance and detailed selection of hyperparameters for each of those documents by humans and crafting are impossible.

2.2 Word search and linguistic post-processing

Character-oriented approaches create a data structure representing the character hypotheses, their position in the text, and the confidence value. For example, an LSTM produces a final map with character hypothesis activations, ordered from left-to-right or right-to-left with some stride (step size). Other approaches generate a grid or graph of character hypotheses. The final processing step involves finding the most likely character path, given a dictionary and potential other linguistic resources (statistics). For the LSTM, a well-known first step toward this is connectionist temporal classification (CTC) [11].

Given a dictionary containing possible input words, an easy method can be used for error detection and correction of a word recognizer. In the case of the existence of the word hypothesis in the dictionary, the result is accepted as

the label of the input image. Otherwise, if a similar word exists in the dictionary, it can be accepted as a final label candidate by using the Levenshtein distance and its variants [50, 54–56], or n-gram distances [57], as common measures for comparing (dis)similarity. If required, it is possible to use suitable linguistic statistics to further refine the ranking [58–60].

A data structure for contextual word recognition is presented in [61] for a quick dictionary look-up using limited memory.

An approach of providing contextual information by giving a dictionary to predict the most probable label in a graph search is presented in [62], which is robust to dictionary errors. In this approach, for every lexical word, the most probable path and related confidence is calculated to predict dictionary ranking.

Shannon [63, 64] was one of the first researchers working on the letter prediction task. Based on this idea, using a trainable variable memory length Markov model (VLMM), a linguistic post-processing model for character recognizers is introduced in [65]. The next character is predicted by a variable-length window of previous characters.

In [66], on the linguistic corpora, a statistical n-gram language model of syllables is trained. In [67], for Japanese mail addresses, a character recognition method uses a dictionary in a trie tree. The dictionary matching is controlled by a beam search approach. The dictionary includes all the address names and principal postal offices in Japan. After pre-processing and segmentation character hypotheses are produced by a combination of successive segments. Then, a version of a nearest-neighbor classifier that exploits the trie structure is made for a fast prediction of the final label. In [68], an on-line handwritten recognition system for cursive words uses simple character features to reduce a given large dictionary. The outputs of a Time-Delay Neural Network (TDNN) are converted into a character sequence. The result of the system is a matched word in the reduced dictionary using a variant of Damerau–Levenshtein distance. For on-line handwriting recognition, a search technique is proposed in [69], which is a post-processing phase of a recognition system that calculates posterior probabilities of characters based on Viterbi decoding.

In [70] a version of beam and Viterbi search-recognizer is presented. This search method provides the use of discrete probabilities generated by many character recognition systems based on stroke. Powalka et al. [71] introduces a technique combining word segmentation and character recognition with a lexical search to deal with segmentation ambiguities. A depth-first trace of dictionary tree for text recognition using recursive procedure presented in [72]. For online handwriting recognition, in [73], by applying simple feature extraction a given dictionary is reduced.

Afterward, the reduced dictionary is refined by AI techniques. In [74], for isolated cursive handwritten-word recognition, contextual knowledge is used. A dictionary tree representation with an efficient pruning method, as a fast search method for a large dictionary for an on-line handwriting recognition system, is proposed in [75].

Of all these approaches, a dual-state word-beam search for CTC decoding currently enjoys increased interest, Scheidl et al. [10], and will be described in Sect. 3.2.

3 Background

In this section, we detail two essential approaches for our proposed method. Firstly, a convolutional recurrent neural network is briefly detailed [26]. Afterward, a dual-state word-beam search (DSWBS [10]) for CTC decoding is explained.

3.1 Convolutional recurrent neural network

The convolutional recurrent neural network is an end-to-end trainable system presented in [26]. It outperforms the plain CNN in four aspects: (1) it does not need precise annotation for each character and it can handle a string of characters for the word image; (2) it works without a strict preprocessing phase, hand-crafted features, or component localization/segmentation; (3) it benefits from the state preservation capability of a recurrent neural network (RNN) in order to deal with character sequence; (4) it does not depend on the width of the word image. Only, height normalization is needed.

The model is composed of seven layers of convolutional layers followed by two layers of BiLSTM units containing 256 hidden cells and a transcription layer. Although the model is made up of two distinct neural network varieties, it can be trained integrally using one loss function.

Figure 2 shows the pipeline of the convolutional recurrent neural network [26]. The input of the model is a height-normalized and grayscale word image. The feature extraction is performed by convolutional layers directly from the input image. The output of CNN is a frame of features sequence and acts the input of the recurrent neural network, which provides raw character hypotheses. Finally, the transcription layer translates the resulting prediction into a label sequence.

3.2 A dual-state word-beam search for CTC decoding

The dual-state word-beam search for CTC decoding, [10], is based on Vanilla Beam Search decoding (VBS) [76] for decoding of the CTC layer. The output of RNN is a matrix,

Fig. 2 The architecture of a convolutional recurrent neural network is composed of three components: convolutional and recurrent layers, and a transcription layer. The phases are as follows: first, feature extraction is carried out by convolutional layers directly from a height-normalized and grayscale word image. Secondly, for each frame, the prediction of label distribution is performed by RNN layers. Thirdly, the transcription layer transcribes the regarding prediction into a label sequence [26]. In this paper, handwritten character sequences are the input

and it is the input of the dual-state word-beam search method. In the dual-state word-beam search, a prefix tree is made of ground truth labels of the training set. It consists of two states: the word-state and the non-word-state (Fig. 3). The next character of the current beam is either a *word-character* or a *non-word-character*, and it determines the

Fig. 3 The dual-state word-beam search for CTC decoding [10] used for our proposed system

subsequent state of the beam. The sets of *word-characters* and *non-word-characters* are predefined.

The temporal evolution of a beam depends on its state. For a beam in the non-word state, it is possible to be extended by a *non-word-character*, and it will stay in the non-word state. A word-character entering brings the system to the word state. Such a *word-character* is the beginning of a word. For a beam in the word-state, the feasible following characters are presented by a prefix tree. This procedure iteratively repeats until a complete word is reached. Scoring can be done in four ways:

1. Words: a dictionary is used without employing a language model (LM).
2. N-grams as LM: as a beam goes to the non-word state from the word state, the LM scores beam-labeling.
3. Ngram+forecast: as a word-character appends a beam, the prefix tree presents all possible words. LM scores all of the relevant beam-extensions.
4. Ngram+forecast+sample: to restrain the following potential words, first some samples are randomly selected. Then, LM scores them. The total score value has to be refined to account for the random-sampling step.

The pseudo-code of the dual-state word-beam search is illustrated in Algorithm 1. The list of symbols is as follows.

- RNN_o : the sequence of RNN output activations over time.
- B : the set of beams at the present time step.
- $Width$: beam width.
- P_b : the probability of finishing the paths of a beam with blank.
- P_{nb} : the probability of not finishing the paths of a beam with blank.
- P_{tot} : $P_b + P_{nb}$.
- P_{txt} : the probability allocated by the LM.
- T : the final iteration of the algorithm, $t = T$.
- $\emptyset$: Empty beam.
- -1 : the last character of the beam.
- x : a beam.
- c : a character.
- $x(t)$: a beam character at t .
- $numWords(x)$: the number of words in the beam x .
- $GetbestBeams(B, Width)$: best $Width$ beams based on the highest value of $P_{txt} * P_{tot}$.
- $NumWord's(x)$: the number of words exists in the beam x .
- $scoreBeam(LM, x, c)$: the probability of seeing character c for extension of the beam x .

Algorithm 1: The dual-state word-beam search for CTC decoding [10]

Data: RNN output matrix RNN_o , $Width$ and LM
Result: most probable labeling

```

 $B = \{\emptyset\};$ 
 $P_b(\emptyset, 0) = 1;$ 
for  $t = 1 \dots T$  do
     $bestBeams = GetbestBeams(B, Width)$ 
     $B = \{\};$ 
    for  $x \in bestBeams$  do
        if  $x \neq \emptyset$  then
             $P_{nb}(x, t) += P_{nb}(x, t-1) * RNN_o(x(-1), t);$ 
        end
         $P_b(x, t) += P_{tot}(x, t-1) * RNN_o(blank, t);$ 
         $B = B \cup x;$ 
         $nextChars = GetNextChars(x);$ 
        for  $c \in nextChars$  do
             $x' = x + c;$ 
             $P_{txt}(x') = scoreBeam(LM, x, c);$ 
            if  $x(t) == c$  then
                 $P_{nb}(x', t) += RNN_o(c, t) * P_b(x, t-1);$ 
            else
                 $P_{nb}(x', t) += RNN_o(c, t) * P_{tot}(x, t-1);$ 
            end
             $B = B \cup x';$ 
        end
    end
 $B = completeBeams(B);$ 
return  $bestBeams(B, 1);$ 

```

In RNNs, such as LSTM, the exact alignment of the observed word image with the ground truth label is not clear. Hence, a probability distribution at each time step is used for prediction which makes it more important to use an adequate coding scheme.

However, even after the CTC stage, additional processing steps from the above-mentioned repertoire are needed to boost classifications.

Unfortunately, although using linguistic resources is clearly advantageous, there are cases where this is not, or only partly possible:

- Not all problems enjoy the presence of an abundance of digitally encoded text, comparable to, e.g., the massive contemporary-English text corpora;
- in historical collections, there may be virtually no resources, not even a lexicon;
- many collections, e.g., administrative ones, have a dedicated jargon, abbreviations and non-standard phrasing. Diaries often contain family-specific or other idiosyncratic neologisms;

- many collections have outdated geographical and scientific terminology, such as the historical document collection that belonged to the Natuurkundige Commissie's scientific exploration of the Indonesian Archipelago between year 1820 and 1850 [77]. This heterogeneous handwritten manuscript contains 17,000 pages of the field notes based of the scientists' natural observation in German, Latin, Dutch, Malay, Greek, and French. Biological terms vary greatly over periods in history [78].

There is, however, an additional way to improve the classification performance. Impressive results using an ensemble method were presented in [8]; however, the number of networks was so large (118) that the need for a less drastic approach is becoming urgent. We will therefore focus on the probabilities of a small-scale ensemble.

4 Method

In this section, we present a limited-size ensemble system for word recognition with a minimum of human intervention. The suggested system uses an adequate label-coding scheme and a dictionary as the only resource for the language model. This system is adequate for being applied on e-Science servers. The system is described as follows.

4.1 The Extra-separator label-coding scheme

In the common label-coding scheme, we call it '*Plain*,' only the characters which are present in the word image appear in the corresponding label. In the '*Extra-separator*' label-coding scheme, one more character is appended at the end of each label. The appended character, named the extra separator (e.g., 'l'), must not exist in the alphabet of the dataset. The aim of adding the Extra-separator character is to give the recognizer an extra hint concerning the end-of-word shape condition.

4.2 Neural network

The neural network is a convolutional BiLSTM neural network, and it is an end-to-end trainable framework inspired by [26]. The main configuration of the networks is detailed in Table 1. In this section, we explain the essential components of our approach.

4.2.1 Pre-processing

The preprocessing is performed in each epoch of training. It consists of: (a) data augmentation through randomly stretching/squeezing the grayscale images in the width

direction, (b) re-sizing the images into 128×32 and (c) normalization. Data augmentation is performed to increase the size of the training set, and it is achieved by changing the width of an image randomly by a factor between 0.5 and 1.5. Next, both the original grayscale images and those added through data augmentation are resized so that either the width is 128 pixels or the height is 32 pixels. After that, we pad the image with white pixels until the size is 128×32 . Then we normalize the intensity of the grayscale image. Note that our method does not need baseline alignment or precise deslanting. Please note that one of our datasets was already deslanted to 90° .

4.2.2 A 5-layer CNN

The pixel-intensity values after preprocessing are fed to the first of 5-layers of a CNN to extract feature sequences. Each layer of the CNN contains a convolution operation, normalization, the ReLU activation function [79], and a max pooling operation. The size of the kernel filters in each layer is 3×3 . Given the fixed important hyperparameter setting, such as the number of layers, the only variable control parameters concern the number of units in the hidden layers. The simple table of three possible sizes {128, 256, 512} is used with the random probability of 0.33 for selecting the sizes of hidden units. The network has no dropout. The sizes of the numbers of hidden units used in our experiments are shown in Table 2. The number of layers, the size of the kernel and the optimizer is our configuration, and differ from Shi et al. [26].

Furthermore, instead of using ADADELTA [80] used in [26], we used RMSProp [81]. Moreover, we used five convolutional layers instead of seven suggested in [26].

4.2.3 BiLSTM

The five convolutional layers are followed by three layers of BiLSTM. Because the last convolutional layer contains 512 hidden units, each BiLSTM has 512 hidden units.

4.2.4 Connectionist temporal classification (CTC)

The CTC output layer contains two units more than characters in the alphabet (A) of the given dataset: the suggested extra separator (e.g., 'l'), and a common blank for CTC, which differs from the space character. Therefore, the alphabet of CTC output is:

$$A' = A \cup \text{extra separator} \cup \text{blank}$$

The $|A + 2|$ output units determine the probability of detecting the relevant label at the time. Further, the blank unit determines the probability of observing blank, or 'no label.' For CTC decoding, we use the dual-state beam

Table 1 Configuration of our a convolutional recurrent neural network from input image (bottom) to last output (top)

Layer#	Description	Configuration	Nh_{units}
7	Transcription	A dual-state word-beam Search CTC decoding	$ A' $
6	Bidirectional-LSTM	L3 512 hidden units	512
		L2 512 hidden units	512
		L1 512 hidden units	512
5	Max pooling	W and S:1×2	512
	Nonlinear ReLU	–	
	Normalization	–	
	Convolution	K:3 × 3, S:1, p:1	
4	Max pooling	W and S:1×2	Nh_4
	Nonlinear ReLU	–	
	Normalization	–	
	Convolution	K:3 × 3, S:1, p:1	
3	Max pooling	W and S:1×2	Nh_3
	Nonlinear ReLU	–	
	Normalization	–	
	Convolution	K:3 × 3, S:1, p:1	
2	Max pooling	W and S:2×2	Nh_2
	Nonlinear ReLU	–	
	Normalization	–	
	Convolution	K:3 × 3, S:1, p:1	
1	Max pooling	W and S:2×2	128
	Nonlinear ReLU	–	
	Normalization	–	
	Convolution	K:3 × 3, S:1, p:1	
0	Input image	128×32 grayscale image	n_{pixel}
↑			

'K','W','S' and 'P' denote kernel size, window size, stride and padding. Nh_{units} shows the number of hidden units (Table 2)

search presented in [10]. This method is explained in Sect. 3.2.

4.3 The ensemble system

In order to construct an ensemble automatically, the number of hidden units in layers 2,3 and 4 is selected at random from a list of possible sizes [128, 256, 512] (Table 2). In each of the five CNN-BiLSTMs, for an input image, the outcome of the CTC decoder is a string as a word hypotheses with its relative likelihood. The word hypothesis obtained from five networks are sent to the voter component. Plurality voting [82] with a solution for ties is then applied, where the alternatives are divided to subsets with identical strings. The subset with the largest number of voters is selected. In case of a tie, the subset with the highest averaged likelihood is the winner. If the

number of subsets is equal to the number of alternatives, the alternative with the highest likelihood is the winner. The winning string is considered as the final, best label of the input image. This approach was chosen after a pilot experiment, using Borda-count voting, without good results. This may be due to the lack of diversity in the ranked candidate lists. Therefore, the more simple approach using plurality voting with exception handling was performed. Please note that analyzing the different, randomly drawn CNN-BiLSTM architectures in the ensemble is not the research goal of this paper. We just need a number of networks that sufficiently support each other in the ensemble by sufficiently independent votes.

Table 2 Number of hidden units in the CNNs front ends, in the five architectures (A_i , $i = 1 \dots 5$)

Layer	Hidden unit size				
	Nh_1	Nh_2	Nh_3	Nh_4	Nh_5
Arch.					
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

Nh_j shows the number of hidden units of the j th layer of a CNN, where $j = 1 \dots 5$

5 Results

In this section, we will first describe the datasets used in the experiments. Then, we explain how our experiments were carried out. Finally, we report the quantitative results.

5.1 Datasets

In this paper, we used three datasets which differ in time period and language, summarized in Table 3. The first dataset is named RIMES, which was used to be comparable with the state-of-the-art methods. This database has different versions. We used isolated words of the version of ICDAR 2011 for evaluation of the methods and making the comparison with the published results possible [19]. The RIMES database is drawn from different types of handwritten manuscripts: postal mails and faxes. It contains 12,723 pages written by 1300 volunteers using black ink on white paper. The RIMES dataset consists of 51,738 images of French handwriting for training, 7464 images for validation and 7776 images for testing. The dictionary size of the training set is 4943 words, the validation set is 1612 and the test set is 1692, and the dictionary size of the whole dataset is 5744 words. The comparison is accomplished case insensitive as it is common for the RIMES dataset, and the accent were contemplated. In the evaluation

(a) garnizoensplaats|

(b) advocaat|

(c) Staatsblad|

(d) wetenschappelijk|

Fig. 4 Samples of the KdK dataset (the year 1903). **a–d** Show the images labeled, using the Extra-separator label-coding scheme

process of our model on RIMES, two dictionaries were used: *Concise* and *Large*. The *Concise* dictionary contains the whole words within the RIMES dataset, $n_{words} = 5744$ (6K). A French dictionary called *Large* (50K) is used to study the effect of a larger dictionary.

The second dataset belongs to the National Archive of the Netherlands, named KdK (Het Kabinet der Koningin or Dutch Queen's Office) [15, 20]. The manuscript was written between years 1798 and 1988, and the year 1903 was used. The KdK dataset contains 172,440 Dutch word images. The number of word classes of the total dataset is 11,749 and 10,747, case-sensitively and case-insensitively, respectively. Regardless of case-sensitivity, there are 1–5628 sample(s) in each class. The length of the word samples is 1–28 character. In the case-sensitive manner, 5% of the test words does not occur in the training words, and is 'out of vocabulary (OOV).' OOV in the case-insensitive manner is 4.5%. The remaining words are considered as is referred to as 'in vocabulary (INV).' Figure 4 shows four original samples of the KdK dataset. For

Table 3 Datasets

Language	RIMES			KdK			GW		
	French			Dutch			English		
Set	Image	Word	Word CI	Image	Word	Word CI	Image	Word	Word CI
Training (T)	51,738	4943	4639	103,464	8717.6	8006.8	2433	835	799
Validation	7464	1612	1509	34,488	4486.6	4155.4	1293	606	588
Test	7776	1692	1606	34,488	4486.6	4155.4	1168	525	508
Whole dataset	66,978	5744	5378	172,440	11,749	10,747	4894	1471	1412

evaluation, two dictionaries are used: *Concise* and *Large*. The *Concise* dictionary contains all the words in the KdK dataset (12 K); the size of the Dutch *Large* dictionary is 384 K, [83].

The third dataset is George Washington (GW [21]). The GW dataset is harvested from 20 pages written by George Washington and George Mercer in the year 1755. The GW datasets for handwriting word recognition consists of 4894 word images. The ground truth of GW contains the upper and lower case English letters, punctuation marks, digits, and the historical special characters, e.g., *long s*(f) which all were encountered in our evaluations. As it is common, we used the first partition of the dataset [28, 84] (Table 3). The *Concise* dictionary contains the whole words within the GW dataset, $n_{words} = 1471$ (1K). A *Large* dictionary (12 K) is made used to study the effect of our model. The *Large* dictionary contains all the words and signs from *Pamphlets of the American Revolution* [85] from year 1750 to 1776.

5.2 Quantitative results

In this section, we evaluate our model on the RIMES, the KdK and the GW datasets in terms of label-coding scheme (Plain vs Extra separator) and ensemble/single network in terms of word accuracy. Moreover, for the RIMES and the GW datasets, the results of our model are compared with the-state-of-the-art methods. We train the model from scratch. Although using synthetically generated images can boost the result for a particular task, this is not a general solution. There are no synthetic resources for rare languages, e.g., Aymara, or rare script types. This argument also applies to the difficult note-fields, e.g., the MkS dataset [77]. Generating artificial (synthetic) samples in the proper language and style is very interesting but much more complicated than random morphing of existing data [86]. There are several methods for data augmentation, [87, 88]. Whether one uses algorithmic synthesis or generative adversarial networks (GANs [89]), this requires human expertise and research labor, in addition to the recognizer's design and training. This dependency on human input is actually in contrast with the current data-driven philosophy in machine learning and AI.

For the Extra-separator label-coding scheme, a character that is absent in the given dataset was found automatically as the Extra-separator character, the bar sign (!); hence, the bar sign is annexed to the end of each image label (Fig. 4). As a result, the size of the output of the CTC layer increases. The RIMES dataset contains 80 unique characters. Meaning that the size of the output layer of the CTC layer is 82 (80 unique character, one extra separator, and one common blank). The KdK dataset contains 52 unique characters. Therefore, the size of the output layer of CTC

layer is 54 (52 unique characters, one extra separator, and one common blank). We compare the result of this addition to the Plain label-coding scheme. Two CTC decoder methods are used: dictionary-free (Best path) and with a dictionary (dual-state word-beam search [10]). For the dual-state word-beam search, two dictionaries are used for each dataset: Concise and Large.

Table 4 shows the effect of the two label-coding schemes, single recognizer, and ensemble voting on the RIMES dataset showing word accuracy (%). For each of the two label-coding schemes (Plain and Extra separator), the five architectures were trained, which resulted in 10 trained networks. Then the networks were evaluated using the Best-path CTC decoder and the dual-state word-beam search CTC decoder applying the Concise (6 K) and the Large (50 K) dictionaries. The result of each evaluation and the relative average $\pm$ standard deviation (avg $\pm$ sd) are reported. In the bottom row of Table 4, the voting-based result of the ensemble of the five networks is presented.

Best path vs Dual-state word-beam search: the results confirm that using a decoder with a dictionary considerably improves the performance (95–97%) as expected (t -test, $p < 0.05$, significant). The dictionary-free best-path CTC decoder is given a low performance, still at 88–89%. Moreover, when the dual-state word-beam search CTC decoder is used, adding an Extra-separator character enhances the model.

Plain vs Extra separator: for the Best-path CTC decoder, both Plain and Extra separator have an average of 84.5% (t -test, $p > 0.05$, N.S.). Therefore, the extra separator has no effect. However, for a dual-state word-beam search CTC decoder using the Concise dictionary, Plain has an average of 94.3%, and Extra separator has an average of 95.2% (t -test, $p < 0.05$, significant). Hence, the extra separator is effective; for a dual-state word-beam search CTC decoder, using the Large dictionary, Plain has an average of 92.9%, and Extra separator has an average of 94.1% (t -test, $p < 0.05$, significant). Therefore, the Extra separator is effective again, for the case of a large dictionary.

Single network vs ensemble: ensemble voting increases the performance where its effect is more on a weaker recognizer (4 pp increase in performance for the dictionary-free CTC decoder using the Plain/Extra-separator label-coding scheme, final row vs average and individual). An ensemble of five recognizers, using the CTC decoder with the Concise dictionary combined with the Extra-separator label-coding scheme results in the highest performance among the methods trained from scratch (96.6%, column 6, bottom).

Table 4 The result of the RIMES dataset

Coding scheme	Plain			Extra separator		
	Best path	The dual-state word-beam search		Best path	The dual-state word-beam search	
Dictionary	-(Dictionary-free)	Concise (6 K)	Large (50 K)	-(Dictionary-free)	Concise (6 K)	Large (50 K)
<i>Arch.</i>						
A ₁	84.6	94.1	92.9	83.8	95.2	94.1
A ₂	84.6	94.5	93.2	84.4	94.7	93.4
A ₃	84.2	94.4	93.1	84.9	95.5	94.6
A ₄	84.5	94.2	92.8	84.9	95.3	94.3
A ₅	84.7	94.2	92.6	84.3	95.1	94.0
Avg $\pm$ sd	84.5 $\pm$ 0.2	94.3 $\pm$ 0.2	92.9 $\pm$ 0.2	84.5 $\pm$ 0.4	95.2 $\pm$ 0.3	94.1 $\pm$ 0.4
Ensemble	88.4	95.7	94.8	88.9	96.6	95.8

The table shows comparison of word accuracy (%) between two label-coding schemes (Plain and Extra separator) using Best-path CTC decoder and the dual-state word-beam search with different dictionary sizes in terms of average $\pm$ standard deviation (avg $\pm$ sd) and the ensemble. Two dictionaries are used: the dictionary which contains words of the training, the validation and the test sets (Concise) and a dictionary which contains more than 50 K words (Large)

Fig. 5 The graph shows the effect of number of networks in the voting ensemble on the final accuracy (%) for the RIMES dataset, with diminishing returns as the number of voters increases

To study the effect of the number of networks in the ensemble on the final accuracy, the results of randomly selected 1, 3, 5, 10, and 15 network(s) are shown in Fig. 5 for the RIMES dataset. The label-coding scheme is Extra separator, and the CTC decoder is the dual-state word-beam search using the Concise dictionary. The networks in the ensemble only differ in the random initialization and number of the units over the layers, also randomly selected from the set $n = \{128, 256, 512\}$ in 1 through 4. The maximum accuracy is obtained by the ensemble of 15 networks, 96.72%, which is just 0.09 pp is more than using 10 networks.

Table 5 shows the comparison of our method on the RIMES dataset with [8, 27, 29, 30, 35, 38] in the terms of a

number of characteristics: number of recognizers, homogeneity of the algorithm, word accuracy (%), and the complexity of the approach, not to be confused with computational complexity, e.g., deep learning method without extra complicated modules.

Here, model complexity means complexity in general, but this includes the *computational complexity* that for LSTMs is in the order of the total number of coefficients or weights [90]. Indeed, there is another aspect of complexity, i.e., the intricacy of the models in terms of the number of layers, other hyperparameters, and the consequent human effort spent on training.

The importance of a homogeneous ensemble for an e-Science server is its practicality. In this paper, we will use the term 'homogeneous' to indicate constrained heterogeneity because absolute homogeneity would not be sensible due to the lack of stochastic independence of opinions in an ensemble. While [91] rightly suggested that heterogeneity is important in ensemble voting, our results indicate that even constrained heterogeneity, i.e., with a limited number of random hyper-parameter variations still is beneficial. In this work, we pursue an adequate method for e-Science servers. In an e-Science server, tailoring an ensemble for slightly higher performance is not feasible because an e-Science server handles massive different datasets. Therefore, for an e-Science server, it is feasible to use an ensemble composed of a limited number of automatically generated architectures.

The result of Dutta et al. [28] in Table 5 appears to be the highest performance; however, in [28], the special characters are not counted in the evaluation. On the other hand, in our approach, the punctuations and digits are

Table 5 The comparison of our system to the state-of-the-art systems on the RIMES dataset in terms of number of recognizers (n_{rec}), homogeneity of the algorithm (Hom.), complexity of the approach (Compl.), and word accuracy (%) ($word_{acc}$)

System	n_{rec}	Hom.	Compl.	$word_{acc}$
Ours	1	N/A	Low	95.1 ± 0.3
Ours (Table 2)	5	✓	Low	96.6
Dutta et al. [28]	1	N/A	Medium	98.2
Stuner et al. [8]	2100	✗	Medium	96.5
Stuner et al. [8]	118	✗	Medium	96.4
Poznanski and Wolf [35]	1	N/A	Medium	96.1
Menasri et al. [27]	8	✗	High	95.3
Ptucha et al. [38]	3	✗	High	94.3
Menasri et al. [27]	1	N/A	Low	91.1
Stuner et al. [29]	1	N/A	Low	89.9
Sueiras et al. [30]	1	N/A	Low	86.9

Please refer to the text for further explanation

Fig. 6 The samples of the pre-processed KdK dataset. **a–f** show the images labeled using the Extra-separator label-coding scheme. After the binarization process, all the word images were sheared 45° in the anticlockwise direction**Table 6** The results of the KdK dataset

Arch.	Avg	Sd
A ₁	94.4	2.7
A ₂	94.4	2.6
A ₃	94.3	2.8
A ₄	94.4	2.6
A ₅	94.2	2.8

The table shows the average (avg) and standard deviation (sd) of word accuracy (%) of five architectures using fivefold cross-validation and varying per architecture, over the following parameters: dictionary (none, Concise, Large) and label-coding scheme (Plain, Extra separator) ($5 \times 3 \times 2$). Each row is derived from 30 network evaluations

counted for RIMES. The model in [28] is pre-trained on the IIIT-HWS dataset, and then training and evaluation were done on the RIMES dataset.

For the KdK dataset, the results are as follows. The samples of the KdK dataset for our experiment were binarized, then sheared 45° in the anticlockwise direction to the slant angle in this style approximately 45° . Afterward, the white borders of images were removed horizontally and vertically, until the place where the first black pixel is observed. In Fig. 6, the deslanted, white-removed images are shown. To derive a more accurate estimation of the performance of our model, we ran fivefold cross-validation. Each architecture, A_i , where $i = 1$ to 5, is trained, either using the Plain label-coding scheme or using the Extra separator, resulting in 50 trained networks ($5 \times 5 \times 2$). Then, each network is tested three times: using the dictionary-free Best-path CTC decoder; using the dual-state word-beam search CTC decoder applying Concise (12 K), and Large (384 K).

Table 6 shows the average (avg) and standard deviation (sd) of word accuracy (%) of five architectures using

Table 7 The results of the KdK dataset

CTC decoder	Dictionary	Avg	Sd
Dual-state word-beam search	Concise (12 K)	96.5	0.3
	Large (384 K)	95.8	0.4
Best-path method	Dictionary-free	90.6	0.3

The table shows the average (avg) and standard deviation (sd) of word accuracy (%) of using a dictionary on fivefold cross-validation and varying per dictionary, over the following parameters: architecture (A_i , $i = 1$ to 5), label-coding scheme (Plain, Extra separator) ($5 \times 5 \times 2$). Each row is derived from 50 network evaluations

Fig. 7 The behavior of a single network A_2 , using the Extra-separator label-coding scheme and the dual-state word-beam search CTC decoder for different word lengths for the OOV and INV conditions in the KdK dataset. The continuous black line indicates the word-length proportion of the training set of one round of fivefold cross-validation for the KdK dataset. The dots represent the accuracy of the network A_2 on OOV and INV using the dual-state beam search and an extra separator for labeling. Please note that OOV words can be recognized with high accuracy in a range in which there are few numbers of samples in the training set (i.e., even in case of infrequently used words) (color figure online)

fivefold cross-validation and varying per architecture, over the following parameters: dictionary (none, Concise, Large), and label-coding scheme (Plain, Extra separator) ($5 \times 3 \times 2$). Each row is derived from 30 network evaluations. In other words, each row is the result of one architecture, regardless of the used CTC decoding method, dictionary, and label-coding scheme. Slightly lower performance is expected as the Best-path CTC decoder pulls it down. A similar result is achieved for each label-coding scheme, regardless of the used CTC decoding method, dictionary, and architecture. The Extra-separator has a higher performance, 94.5%, which is 0.4 pp higher than the Plain decoding scheme.

Table 7 shows the average (avg) and standard deviation (sd) of word accuracy (%) of using a dictionary on fivefold cross-validation and varying per dictionary, over the following parameters: architecture (A_i , $i = 1$ to 5), label-coding scheme (Plain, Extra separator) ($5 \times 5 \times 2$). Each row is derived from 50 network evaluations.

Figure 7 shows the behavior of a single network A_2 , using the Extra-separator coding scheme and the dual-state word-beam search CTC decoder for different word lengths for the OOV and INV condition in the KdK dataset. The blue and red dots represent the accuracy of OOV and INV, respectively.

The continuous green and black lines in Fig. 7 indicate the word-length occurrence of the training and the test sets

Fig. 8 Accuracy of test words obtained by network A_2 on one round of fivefold cross-validation on the KdK dataset. The horizontal axis represents the number of instances per word class, sorted in order of $\log(\text{frequency})$ in the training set. The parentheses show the number of samples per class. The blue circles show the test words which are present in the training set, in vocabulary, where the darker the blue circle, the more word classes. The dark red circle indicates the average accuracy (89.9%) of out-of-vocabulary samples at $\text{frequency} = 0$ (color figure online)

in the KdK dataset in one round of the fivefold cross-validation. It is demonstrated that the single network A_2 on INV words with a length of up to 17 characters has high accuracy and is promising. For longer words the performance becomes erratic. The single network A_2 does not perform satisfactorily on short OOV words with 1–4 characters. The performance on OOV words which have 5–15 characters is highly adequate. For OOV words whose length is between 16 and 20, the performance is variable. Surprisingly, for OOV samples longer than 21 characters, the model has a high performance.

Figure 8 shows the accuracy of words achieved by network A_2 on one round of fivefold cross-validation on the KdK dataset. The X axis shows the number of instances per word class, sorted in order of log frequency in training set. The blue circles indicate INV words. The dark red circle reveals the average accuracy and the log occurrence of OOV (89.9%). Note the different 'threads' in the curve, revealing groups of easy and difficult (slow-starting) classes. In a lifelong machine-learning, the horizontal axis corresponds to time, starting with just a few examples on the left. The average of the performance on OOV samples is high, at $\log(f) = 0$, where f is the frequency in the training set. From the curves, it can be seen that more examples imply a higher accuracy, but even words that are not in the (training) lexicon can obtain decent performance.

Table 8 shows the comparison of the effect of the two label-coding schemes (Plain and Extra separator) and the CTC decoder application on the ensemble for the five rounds of the cross-validation of the KdK dataset.

Table 8 The result of the KdK dataset

Coding scheme	Plain			Extra separator		
	Best path	The dual-state word-beam search		Best path	The dual-state word-beam search	
Dictionary	-(Dictionary-free)	Concise (12 K)	Large (384 K)	-(Dictionary-free)	Concise (12 K)	Large (384 K)
<i>Arch.</i>						
A ₁	90.62	96.27	95.52	90.84	96.79	96.16
A ₂	90.77	96.34	95.55	90.87	96.76	96.15
A ₃	90.45	96.23	95.43	90.50	96.77	96.13
A ₄	90.72	96.32	95.56	90.88	96.81	96.20
A ₅	90.23	96.13	95.34	90.50	96.72	96.09
Avg $\pm$ sd	90.56 $\pm$ 0.22	96.26 $\pm$ 0.12	95.48 $\pm$ 0.15	90.72 $\pm$ 0.26	96.77 $\pm$ 0.12	96.14 $\pm$ 0.13
Ensemble	93.38 $\pm$ 0.12	97.00 $\pm$ 0.09	96.51 $\pm$ 0.10	93.52 $\pm$ 0.13	97.37 $\pm$ 0.09	96.99 $\pm$ 0.11

The table shows a comparison of word accuracy (%) between two label-coding schemes (Plain and Extra separator) using best-path CTC decoder and the dual-state word-beam search with different dictionary sizes in terms of average $\pm$ standard deviation (avg $\pm$ sd) and The ensemble. Two dictionaries are used: the dictionary which contains words of the training, the validation, and the test sets (Concise) and a dictionary which contains 384 K words (Large)

Best path vs dual-state word-beam search: using no dictionary conditions in more than 93% accuracy. Using a decoder with dictionary boosts the performance (t -test, $p < 0.05$, significant). Adding an extra separator enhances the model when a CTC decoder with a dictionary is used.

Plain vs extra separator: for the Best-path CTC decoder, Plain has an average of 90.6%, and Extra separator has an average of 90.7% (t -test, $p > 0.05$, N.S.). Therefore, the extra separator has no effect; for a dual-state word-beam search CTC decoder using the Concise dictionary (12 K), Plain has an average of 96.3%, and Extra separator has an average of 96.8% (t -test, $p < 0.05$, significant). Therefore,

the extra separator is effective; for a dual-state word-beam search CTC decoder using the large dictionary (384 K), Plain has an average of 95.5%, and extra separator has an average of 96.1% (t -test, $p < 0.05$, significant). Therefore, the extra separator is effective.

Single network vs Ensemble: ensemble voting increases the performance where its effect is more on a weaker recognizer (3 pp increase in performance for the dictionary-free CTC decoder for Plain/Extra separator). The ensemble of five recognizers that used the CTC decoder with the Concise dictionary combined with the Extra

Table 9 The result of the GW dataset

Coding scheme	Plain			Extra separator		
	Best path	The dual-state word-beam search		Best path	The dual-state word-beam search	
Dictionary	-(Dictionary-free)	Concise (1 K)	Large (12 K)	-(Dictionary-free)	Concise (1 K)	Large (12 K)
<i>Arch.</i>						
A ₁	68.92	82.36	78.68	70.63	87.67	85.44
A ₂	71.15	85.19	83.05	71.92	86.56	84.33
A ₃	69.86	84.93	81.85	70.38	86.13	83.81
A ₄	68.15	83.56	81.08	68.15	85.45	81.76
A ₅	69.69	83.90	81.42	69.78	85.79	83.64
Avg $\pm$ sd	69.55 $\pm$ 1.0	83.99 $\pm$ 1.0	81.21 $\pm$ 1.4	70.2 $\pm$ 1.2	86.13 $\pm$ 0.4	83.56 $\pm$ 0.9
Ensemble	74.46	86.89	84.49	74.66	89.55	86.73

The table shows the comparison of word accuracy (%) between two label-coding schemes (Plain and Extra separator) using best-path CTC decoder and the dual-state word-beam search with different dictionary sizes in terms of average $\pm$ standard deviation (avg $\pm$ sd) and the ensemble. Two dictionaries are used: the dictionary which contains words of the training, the validation, and the test sets, Concise (1 K), and a dictionary that contains 12 K words (Large)

separator label-coding scheme results in the highest performance (97.4%).

Table 9 shows the effect of the two label-coding schemes, Plain and Extra separator, single recognizer, and ensemble voting on the GW dataset showing word accuracy (%). We follow the use of this measure to compare our results to other studies. Sometimes character classification rates or edit distances are reported. There are caveats here. For instance, asymmetric distance measurement has been reported to be more relevant in the case of historical spelling words compared to contemporary spelling [92]. Since the word accuracy is a strict, conservative measure, we will use it here. For each of the two coding schemes, five CNN-BiLSTMs described in Sect. 4.2 with a different number of hidden units (Table 2) were trained. Then the evaluation conducted using the Best-path (BP) and the dual-state word-beam search (DSWBS) CTC decoders applying the Concise (1 K) and the Large (12 K) dictionaries. The word recognition accuracy (%) of each evaluation and the relative average $\pm$ standard deviation (avg $\pm$ sd) are reported in Table 9. The bottom row of Table 9 shows the result of the ensemble.

Best path vs dual-state word-beam search: the results confirm that using a CTC decoder with a dictionary significantly improves the performances as expected (t -test, $p < 0.05$, significant). Additionally, the dual-state word-beam search CTC decoder coupled with an Extra-separator character enhances the model more (86%). The dictionary-free best-path CTC decoder results in low performance, 70%.

Plain vs Extra separator: when the best-path CTC decoder is used, the performance for both Plain and Extra separator is low (t -test, $p > 0.05$, N.S.). It is shown that using the Extra-separator label-coding scheme is not beneficial when the Best-Path CTC decoder is used. However, for a dual-state word-beam search CTC decoder using the Concise dictionary, Plain has an average of 84%, and Extra separator has an average of 86% (t -test, $p < 0.05$, significant). For the dual-state word-beam search CTC decoder

Table 10 The results of the GW dataset

System	Word recognition accuracy (%)
Ours (single, Table 2)	86.13 $\pm$ 0.48
Ours (ensemble, Table 2)	89.55
Dutta et al. [28]	87.41

The table shows the average (avg) and standard deviation (sd) of word accuracy (%) of five architectures using fivefold cross-validation and varying per architecture, over the following parameters: dictionary (none, Concise, Large) and label-coding scheme (Plain, Extra separator) ($5 \times 3 \times 2$)

Fig. 9 Comparison of the effect of the two label-coding schemes (Plain vs Extra-separator) and dictionary application on the single architecture and ensemble voting on the RIMES, the KdK, and GW datasets showing the weighted average based on test set sizes. The two datasets are rather different. The spread of a distribution is not very informative

Table 11 Weighted average of word accuracy (%) on the RIMES, KdK and GW datasets, using the dual-state word-beam search applying the Concise dictionary and the Extra-separator label-coding scheme, for the two CTC methods and single vs ensemble voting

CTC decoder	Framework	
	Single	Ensemble
Best path	85.3	89.2
Dual-state word-beam search	94.9	96.2

using the Large dictionary, Plain has an average of 81% and Extra separator has an average of 84% (t -test, $p < 0.05$, significant).

Single network vs ensemble: ensemble voting boosts the performance. The ensemble further affects a weak classifier. An ensemble of five recognizers increases the performance 4 pp using the Plain/Extra separator label-coding scheme when the dictionary-free CTC decoder is used. The ensemble using the Plain/Extra separator increases the performance 3 pp when a dictionary is used.

Table 10 shows the comparison of the performance of our approach on the GW dataset with a recent paper [28]. In this paper, we focused on *word recognition* rather than *character recognition*. Unfortunately, the other studies on this dataset have reported *character recognition accuracy* [36, 84].

Figure 9 shows a comparison of the effect of two label-coding schemes and dictionary application on single architecture and ensemble voting on the RIMES, the KdK, and GW datasets showing the weighted average. Table 11 shows the average of word accuracy (%) on the RIMES,

KdK, the GW datasets, using the Concise dictionary and the Extra-separator label-coding scheme.

The analysis of the ensemble shows that the suggested solution for ties is more beneficial in case of weaker classifiers. The increase in accuracy is 1.3 pp for GW, 0.3 pp for RIMES, and 0.2 pp for the KdK dataset.

6 Discussion

The results indicate that it is possible to achieve a high word accuracy (%) in comparison with the state of the art with a limited-size ensemble, a homogeneous algorithmic approach, and a low complexity [8, 27–30, 35, 38] (cf. Tables 5, 10). In those studies, numerous networks (up to 118 or 2100 network instances) are required in the ensemble. Our method only uses five networks, yielding comparable or better results, also considering that no extraneous training sets were used. The other approach to improve recognition rates is using pre-trained classifiers on the bases of synthetic data [28]. However, such an approach will only work if a language model is available, the allographs of the script style are known including details of punctuation and diacritics. In general, this is not the case and human effort is necessary to implement the training setup. This is not acceptable as a general solution in a large and diverse e-Science server for historical document processing. In the proposed method, handcrafted feature descriptors such as histogram of oriented gradients (HOG [51]) are not used, the process starts with a pixel image and is trained end to end.

The results confirm that the DSWBS CTC decoder using a prefix tree made of a given dictionary significantly increases performance as anticipated (5–16 pp). The results indicate that adding an end-of-word separator is beneficial specifically when dual-state word beam search is used for CTC decoding, not in case of the basic best-path/dictionary-free CTC decoding. In other words, the Extra-separator character, 'l,' tagging the end of the word, boosts the result of the dual-state word-beam search CTC decoding. This increase in performance occurs despite the slight increase in the model size by adding the Extra-separator character. However, the effect on the result of CTC best-path decoding, i.e., a non-dictionary method, is limited. Finally, ensemble voting clearly improves the word accuracy (1–4 pp); its effect is stronger on weaker recognizers.

It should be noted that the reported result is based on realistic images with many word-segmentation problems and therefore can be considered as a conservative estimate (cf. Fig. 6).

We have shown that medium length OOV words (5–11 characters), i.e., words that are in the test set, but not in the training set, profit from training information that is present within short words in the training set (cf. Fig. 7). Longer OOV words (11–23 characters) profit from the training on words whose length is 1–11 characters. Interestingly, OOV words can have a high performance in a word-length range for which there are not many examples (cf. Fig. 7). In addition, for INV words shorter than 18 characters, the accuracy is higher than 95%. Therefore, it can be inferred that both OOV words and INV words are recognized with high accuracy, if they are of a commonly occurring word length. Furthermore, it is apparent that some INV words can be considered 'easy' in the sense that they only need a limited number of examples in the training set, whereas other words are 'difficult,' i.e., needing more than 100 examples to obtain a high accuracy (Fig. 8).

The goal of this research is not a record attempt toward maximized accuracy on the RIMES, the KdK, and GW datasets. Higher performance can undoubtedly be achieved using a larger ensemble (e.g., from Fig. 5 it can be derived that 15 or more NNs in the ensemble would yield 97% accuracy: this is not the point). However, our choice for an ensemble of 5 voting elements results in a compromise with a very good and stable performance. The more than 1 pp jump in performance from one individual classifier to five classifiers is larger than the less than 0.3 pp increase in performance from 5 to 10 classifiers, and the increase in the performance is even smaller for higher numbers of classifiers in the ensemble, showing diminishing returns.

Furthermore, we have shown that by providing a more than 30 times larger dictionary (i.e., the case of the KdK dataset), only a slight drop in performance occurred. In addition, for the dictionary-free approach, using an ensemble system results in much higher performance with more stability than a single network. However, in the higher-performing approach, the ensemble-based improvement is present but less prominent when a dictionary is used. Moreover, as expected from previous research, using the CTC decoder with a dictionary increases the performance compared to a dictionary-free CTC decoder.

From the literature, it is known that using synthetic data for pre-training can be beneficial for contemporary and common languages, e.g., the contemporary French RIMES dataset [28]. Also for the English GW dataset with special characters and styles, it was noted that some characters were absent in the handcrafted synthetic augmentation data. However, the positive effect of synthetic data is expected to be less even when the problem is highly multilingual, e.g., as in the case of the MkS dataset, or a barely-used language, e.g., Aymara. The use of handcrafted synthetic data makes the recognizer highly dependent on

the human labor that is needed for its implementation. An alternative to augmenting the data from a training set would be to exploit pre-existing text-shape knowledge from a pre-trained network. However, this was not the purpose of this paper, because we want to provide training on the data itself, not using extraneous background knowledge.

7 Conclusions

Implementing algorithms that perform very well on standard benchmark datasets may yield sub-optimal results on other, large historical manuscript collections in a real application. In this study we wanted to find an LSTM architecture and CTC decoding approach that shows a high performance, and is easy to implement without human supervision in training and operation. Our model consists of an ensemble of just five homogeneous end-to-end trainable recognizers, using plurality voting with a solution for ties. Each recognizer is composed of five convolutional layers and three BiLSTM layers, followed by a CTC layer. Diversity is fostered by a various number of units in the hidden layers of the CNNs. For CTC decoding, a dual-state word-beam search is applied, using only the given dictionary as the language model. For the labeling of words, we show that adding a token to stress the end-of-word state is significantly beneficial. Training the system is done from scratch, exclusively on the given dataset, and data augmentation is not used during testing. The word accuracy (%) of our model is 96.6% on RIMES, 89.55% for the George Washington dataset, and 97.4% on the KdK dataset, a locally collected historical handwritten dataset. The contributions of this paper are:

- (a) To illustrate that—even in a deep learning paradigm—careful, one could say, handcrafted design of the labeling systematics in LSTMs plays an important role. Adding a separator that marks the word ending has a beneficial effect;
- (b) to make the point that LSTM architectures are difficult to design and train. They do not lend themselves easily to large-scale operations where training and deployment take place as autonomous as possible. The goal is to design an architecture that can be generated randomly on the basis of a limited number of hyper-parameter values, with good performance;
- (c) to introduce an ensemble-based approach, that unlike recent examples in literature, does not require hundreds or thousands of individual networks, but just a handful;

- (d) a specific solution for ties in plurality voting, yielding 0.2–1.3 pp improvement on plain plurality voting;
- (e) to provide empirical results on very large datasets to give an insight into: (1) the effect of the word length on the accuracy of in-vocabulary and out-of-vocabulary test samples; (2) the in-vocabulary vs out-of-vocabulary results; (3) the effect of the number of examples per word class; (4) the presence of easy and difficult classes in training; (5) the effect of the ensemble size on the accuracy; and (6) the effect of the size of a word class on the recognition rate for easy and difficult test samples.

Word-based LSTMs cannot make use of the larger textual content. Therefore, as future work, we plan to extend our approach to handle line-strip images. Moreover, we will explore the applicability of our model on other datasets with different languages, and increase the performance on out-of-vocabulary words. Furthermore, the challenge of high-performance recognition of long words will be addressed.

Acknowledgements 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 Organisation for Scientific Research (NWO). We would like to thank Gideon Maillette de Buy Wenniger for thoughtful advice and the Center for Information Technology of the University of Groningen for providing access to the Peregrine high-performance computing cluster.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. LeCun Y, Bengio Y (1998) Convolutional networks for images, speech, and time series. In: Arbib MA (ed) *The handbook of brain theory and neural networks*. MIT Press, Cambridge, pp 255–258
2. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
3. Graves A, Fernández S, Schmidhuber J (2005) Bidirectional LSTM networks for improved phoneme classification and recognition. In: Duch W, Kacprzyk J, Oja E, Zadrozny S (eds) *Artificial neural networks: formal models and their applications—ICANN 2005*. Springer, Berlin, Heidelberg, pp 799–804

4. Graves A, Schmidhuber J (2009) Offline handwriting recognition with multidimensional recurrent neural networks. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) *Advances in neural information processing systems*, vol 21. Curran Associates Inc, New York, pp 545–552
5. Li X, Wu X (2014) Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In: 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 4520–4524
6. Maillette de Buy Wenniger G, Schomaker L, Way A (2019) No padding please: efficient neural handwriting recognition. In: 15th International conference on document analysis and recognition (ICDAR 2019), Sydney, Australia
7. Doetsch P, Kozielski M, Ney H (2014) Fast and robust training of recurrent neural networks for offline handwriting recognition. In: 14th International conference on frontiers in handwriting recognition, pp 279–284. <https://doi.org/10.1109/ICFHR.2014.54>
8. Stuner B, Chatelain C, Paquet T (2016) Handwriting recognition using Cohort of LSTM and lexicon verification with extremely large lexicon. [arXiv:1612.07528](https://arxiv.org/abs/1612.07528)
9. Puigcerver J (2018) A probabilistic formulation of keyword spotting. PhD thesis, University of Valencia
10. Scheidl H, Fiel S, Sablatnig R (2018) Word beam search: a connectionist temporal classification decoding algorithm. In: The international conference on frontiers of handwriting recognition (ICFHR), IEEE Computer Society, pp 253–258
11. Graves A, Fernández S, Gomez F, Schmidhuber J (2006) Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on machine learning, ACM, New York, ICML'06, pp 369–376. <https://doi.org/10.1145/1143844.1143891>
12. Hauser AW, Schulz KU (2007) Unsupervised learning of edit distance weights for retrieving historical spelling variations. In: Proceedings of the first workshop on finite-state techniques and approximate search, pp 1–6
13. Emlen NQ (2017) Perspectives on the Quechua–Aymara contact relationship and the lexicon and phonology of Pre-Proto-Aymara. *Int J Am Linguist* 83(2):307–340
14. Voigtlaender P, Doetsch P, Ney H (2016) Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In: 15th international conference on frontiers in handwriting recognition (ICFHR), pp 228–233. <https://doi.org/10.1109/ICFHR.2016.0052>
15. Van der Zant T, Schomaker L, Haak K (2008) Handwritten-word spotting using biologically inspired features. *IEEE Trans Pattern Anal Mach Intell* 30(11):1945–1957. <https://doi.org/10.1109/TPAMI.2008.144>
16. Van der Zant T, Schomaker L, Zinger S, Van Schie H (2009) Where are the search engines for handwritten documents? *Interdiscip Sci Rev* 34(2–3):224–235. <https://doi.org/10.1179/174327909X441126>
17. He S, Samara P, Burgers J, Schomaker L (2016) Image-based historical manuscript dating using contour and stroke fragments. *Pattern Recogn* 58:159–171
18. Schomaker L (2019) A large-scale field test on word-image classification in large historical document collections using a traditional and two deep-learning methods. [arXiv:1904.08421](https://arxiv.org/abs/1904.08421)
19. Grosicki E, Carré M, Geoffrois E, Augustin E, Preteux F (2006) La campagne d'évaluation RIMES pour la reconnaissance de courriers manuscrits. In: *Actes colloque international franco-phonie sur l'écrit et le document (CIFED'06)*, Fribourg, Switzerland, pp 61–66
20. Van Oosten JP, Schomaker L (2014) Separability versus prototypicality in handwritten word-image retrieval. *Pattern Recogn* 47(3):1031–1038
21. Fischer A, Keller A, Frinken V, Bunke H (2012) Lexicon-free handwritten word spotting using character HMMs. *Pattern Recogn Letters* 33(7):934–942 (special Issue on Awards from ICPR 2010). <https://doi.org/10.1016/j.patrec.2011.09.009>. <http://www.sciencedirect.com/science/article/pii/S0167865511002820>
22. Impedovo S (2014) More than twenty years of advancements on frontiers in handwriting recognition. *Pattern Recogn* 47(3):916–928 (handwriting Recognition and other PR Applications). <https://doi.org/10.1016/j.patcog.2013.05.027>. <http://www.sciencedirect.com/science/article/pii/S0031320313002513>
23. Plamondon RJ, Srihari SN (2000) On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Trans Pattern Anal Mach Intell* 22:63–84
24. Graves A, Jaitly N (2014) Towards end-to-end speech recognition with recurrent neural networks. In: 31st international conference on machine learning, vol 32
25. Mahasseni B, Lam M, Todorovic S (2017) Unsupervised video summarization with adversarial lstm networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 202–211
26. Shi B, Bai X, Yao C (2017) 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* 39(11):2298–2304. <https://doi.org/10.1109/TPAMI.2016.2646371>
27. Menasri F, Louradour J, Bianne-Bernard AL, Kermorvant C (2012) The A2iA French handwriting recognition system at the Rimes-ICDAR2011 competition. In: Viard-Gaudin C, Zanibbi R (eds) *Document recognition and retrieval XIX*, International Society for Optics and Photonics, SPIE, vol 8297, pp 263–270. <https://doi.org/10.1117/12.911981>
28. Dutta K, Krishnan P, Mathew M, Jawahar CV (2018) Improving CNN-RNN hybrid networks for handwriting recognition. In: 2018 16th international conference on frontiers in handwriting recognition (ICFHR), pp 80–85. <https://doi.org/10.1109/ICFHR-2018.2018.00023>
29. Stuner B, Chatelain C, Paquet T (2017) Self-training of BLSTM with lexicon verification for handwriting recognition. 14th international conference on document analysis and recognition (ICDAR 2017). Kyoto, Japan, pp 633–638
30. Sueiras J, Ruiz V, Sanchez A, Velez JF (2018) Offline continuous handwriting recognition using sequence to sequence neural networks. *Neurocomputing* 289:119–128
31. Vinciarelli A, Luetin J (2001) A new normalization technique for cursive handwritten words. *Pattern Recogn Lett* 22(9):1043–1050
32. Krishnan P, Jawahar CV (2016) Generating synthetic data for text recognition. [ArXiv: abs/1608.04224](https://arxiv.org/abs/1608.04224)
33. Okafor E, Smit R, Schomaker L, Wiering M (2017) Operational data augmentation in classifying single aerial images of animals. In: 2017 IEEE international conference on innovations in intelligent systems and applications (INISTA), Gdynia, Poland, pp 354–360
34. Okafor E, Schomaker L, Wiering MA (2018) An analysis of rotation matrix and colour constancy data augmentation in classifying images of animals. *J Inf Telecommun* 2(4):465–491
35. Poznanski A, Wolf L (2016) CNN-N-Gram for handwriting word recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 2305–2314. <https://doi.org/10.1109/CVPR.2016.253>
36. Almazán J, Gordo A, Fornés A, Valveny E (2014) Word spotting and recognition with embedded attributes. *IEEE Trans Pattern Anal Mach Intell* 36(12):2552–2566. <https://doi.org/10.1109/TPAMI.2014.2339814>
37. Hardoon DR, Szedmak S, Shawe-Taylor J (2004) Canonical correlation analysis, an overview with application to learning methods. *Neural Comput* 16(12):2639–2664

38. Ptucha R, Such FP, Pillai S, Brockler F, Singh V, Hutkowski P (2019) Intelligent character recognition using fully convolutional neural networks. *Pattern Recogn* 88:604–613
39. Zipf GK (1935) *The psycho-biology of language*. Houghton, Mifflin, Oxford
40. Bluche T, Ney H, Kermorvant C (2013) Feature extraction with convolutional neural networks for handwritten word recognition. In: 2013 12th international conference on document analysis and recognition, pp 285–289
41. Ho T (1992) A theory of multiple classifier systems and its application to visual word recognition. PhD thesis, State University of New York at Buffalo, Buffalo, UMI Order No GAX92-22062
42. Romesh Ranawana VP (2006) Multi-classifier systems: review and a roadmap for developers. *Int J Hybrid Intell Syst* 3(1):35–61
43. Günter S, Bunke H (2003) Ensembles of classifiers for handwritten word recognition. *Int J Doc Anal Recogn* 5(4):224–232. <https://doi.org/10.1007/s10032-002-0088-2>
44. Günter S, Bunke H (2005) Off-line cursive handwriting recognition using multiple classifier systems on the influence of vocabulary, ensemble, and training set size. *Opt Lasers Eng* 43(3):437–454
45. Karimi H, Esfahanimehr A, Mosleh M, Jadval Ghadam FM, Salehpour S, Medhati O (2015) Persian handwritten digit recognition using ensemble classifiers. *Proced Comput Sci* 73:416–425
46. Yang W, Jin L, Xie Z, Feng Z (2015) Improved deep convolutional neural network for online handwritten Chinese character recognition using domain-specific knowledge. In: Proceedings of the 2015 13th International conference on document analysis and recognition (ICDAR), ICDAR '15, pp 551–555
47. Ho TK, Hull JJ, Srihari SN (1994) Decision combination in multiple classifier systems. *IEEE Trans Pattern Anal Mach Intell* 16(1):66–75. <https://doi.org/10.1109/34.273716>
48. Van Erp M, Schomaker L (2000) Variants of the Borda count method for combining ranked classifier hypotheses. In: Proceedings 7th international workshop on frontiers in handwriting recognition (7th IWFHR), pp 443–452
49. Powalka RK, Sherkat N, Whitrow RJ (1995) Recognizer characterisation for combining handwriting recognition results at word level. In: Proceedings of 3rd international conference on document analysis and recognition, vol 1, pp 68–73. <https://doi.org/10.1109/ICDAR.1995.598946>
50. Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions and reversals. *Sov Phys Dokl* 10:707
51. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), vol 1, pp 886–893. <https://doi.org/10.1109/CVPR.2005.177>
52. Graves A (2012) Supervised sequence labelling with recurrent neural networks, vol 385. Springer, Berlin, Heidelberg
53. Marti U-V, Bunke H (2002) The IAM-database: an English sentence database for offline handwriting recognition. *Int J Doc Anal Recogn* 5:39–46
54. Wagner RA, Fischer MJ (1974) The string-to-string correction problem. *J ACM* 21(1):168–173
55. Seni G, Kripasundar V, Srihari RK (1996) Generalizing edit distance to incorporate domain information: handwritten text recognition as a case study. *Pattern Recogn* 29(3):405–414
56. Oommen B, Loke R (1997) Pattern recognition of strings with substitutions, insertions, deletions and generalized transpositions. *Pattern Recogn* 30(5):789–800
57. Angell RC, Freund GE, Willett P (1983) Automatic spelling correction using a trigram similarity measure. *Inf Process Manag* 19(4):255–261
58. Youssef Bassil MA (2012) OCR post-processing error correction algorithm using Google's online spelling suggestion. *J Emerg Trends Comput Inf Sci* 3(1):90–99
59. Asonov D (2010) Real-word typo detection. In: Horacek H, Métais E, Muñoz R, Wolska M (eds) *Natural language processing and information systems*. Springer, Berlin, Heidelberg, pp 115–129
60. Chantal Amrhein SC (2018) Supervised OCR error detection and correction using statistical and neural machine translation methods. *J Lang Technol Comput Linguist* 3(1):49–76
61. Wells C, Evett L, Whitby P, Whitrow R (1990) Fast dictionary look-up for contextual word recognition. *Pattern Recogn* 23(5):501–508
62. Favata JT (2001) Off-line general handwritten word recognition using an approximate beam matching algorithm. *IEEE Trans Pattern Anal Mach Intell* 23(9):1009–1021
63. Shannon CE (1951) Prediction and entropy of printed English. *Bell Syst Techn J* 30(1):50–64
64. Shannon CE (1948) A mathematical theory of communication. *Bell Syst Techn J* 27:379–423 (Part I) 623–656 (Part II)
65. Guyon I, Pereira F (1995) Design of a linguistic postprocessor using variable memory length markov models. In: Proceedings of 3rd international conference on document analysis and recognition, Montreal, QC, Canada, vol 1, pp 454–457. <https://doi.org/10.1109/ICDAR.1995.599034>
66. Swaileh W, Paquet T, Mohand K (2016) A syllabic model for handwriting recognition [Un modèle syllabique pour la reconnaissance de l'écriture]. In: CORIA 2016—conference en recherche d'informations et applications—13th French information retrieval conference. CIFED 2016—colloque international francophone sur l'écrit et le document, association francophone de recherche d'information et applications (ARIA), pp 23–37. <https://doi.org/10.1109/ICDAR.2015.7333846>
67. Liu C-L, Koga M, Fujisawa H (2002) Lexicon-driven segmentation and recognition of handwritten character strings for Japanese address reading. *IEEE Trans Pattern Anal Mach Intell* 24(11):1425–1437. <https://doi.org/10.1109/TPAMI.2002.1046151>
68. Seni G, Srihari RK, Nasrabadi N (1996) Large vocabulary recognition of on-line handwritten cursive words. *IEEE Trans Pattern Anal Mach Intell* 18(7):757–762. <https://doi.org/10.1109/34.506798>
69. Seni G, Anastasakos T (2000) Non-cumulative character scoring in a forward search for online handwriting recognition. In: 2000 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No.00CH37100), vol 6, pp 3450–3453. <https://doi.org/10.1109/ICASSP.2000.860143>
70. Seni G, Seybold J (1999) Forward search with discontinuous probabilities for online handwriting recognition. In: Proceedings of the 5th international conference on document analysis and recognition, ICDAR '99 (Cat. No. PR00318), IEEE Computer Society, Washington, pp 741–744. <https://doi.org/10.1109/ICDAR.1999.791894>
71. Powalka RK, Sherkat N, Evett LJ, Whitrow RJ (1993) Multiple word segmentation with interactive look-up for cursive script recognition. In: Proceedings of 2nd international conference on document analysis and recognition (ICDAR '93), Tsukuba City, Japan, pp 196–199. <https://doi.org/10.1109/ICDAR.1993.395750>
72. Ford DM, Higgins CA (1990) A tree-based dictionary search technique and comparison with n-gram letter graph reduction. In: Plamondon R, Leedham G (eds) *Computer processing of handwriting*. World Science Publishing Co., Singapore, pp 291–312
73. Bramall PE, Higgins CA (1995) A cursive script-recognition system based on human reading models. *Mach Vis Appl* 8(4):224–231
74. Côté M, Lecolinet E, Cheriet M, Suen C (1998) Automatic reading of cursive scripts using a reading model and perceptual

- concepts. *Int J Doc Anal Recogn* 1(1):3–17. <https://doi.org/10.1007/s100320050002>
75. Manke S, Finke M, Waibel A (1996) A fast search technique for large vocabulary on-line handwriting recognition. In: *Proceedings of the 5th international workshop on frontiers in handwriting recognition*, UK, pp 183–188
 76. Hwang K, Sung W (2016) Character-level incremental speech recognition with recurrent neural networks. In: *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp 5335–5339. <https://doi.org/10.1109/ICASSP.2016.7472696>
 77. Weber A, Ameryan M, Wolstencroft K, Stork L, Heerlen M, Schomaker L (2017) Towards a digital infrastructure for illustrated handwritten archives. In: Ioannides M (ed) *Final conference of the Marie Skłodowska-Curie initial training network for digital cultural heritage, ITN-DCH 2017*, Springer, Olmje, Slovenia, vol 10605
 78. Schuh RT (2003) The Linnaean system and its 250-year persistence. *Botan Rev* 69(1):59–78
 79. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: *Proceedings of the 27th international conference on international conference on machine learning*, Omnipress, USA, ICML'10, pp 807–814
 80. Zeiler MD (2012) ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701
 81. Tieleman T, Hinton G (2012) Lecture 6.5-RMSProp: divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Netw Mach Learn* 4(2):26–31
 82. Peleg B (1978) Consistent voting systems. *Econometrica* 46(1):153–161. <https://doi.org/10.2307/1913652>
 83. BV SW (2018) *Woorden.org, Nederlandse woordenboek*. <https://www.woorden.org>. Accessed 17 October 2017
 84. Toledo JI, Dey S, Fornés A, Lladós J (2017) Handwriting recognition by attribute embedding and recurrent neural networks. In: *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, vol 01, pp 1038–1043
 85. (1994) *Pamphlets of the American Revolution: [selections]/edited by Bernard Bailyn*, Oxford Text Archive. <http://hdl.handle.net/20.500.12024/2021>
 86. Bulacu M, Brink A, Van der Zant T, Schomaker L (2009) Recognition of handwritten numerical fields in a large single-writer historical collection. In: *2009 10th international conference on document analysis and recognition*, pp 808–812. <https://doi.org/10.1109/ICDAR.2009.8>
 87. Frid-Adar M, Klang E, Amitai M, Goldberger J, Greenspan H (2018) Synthetic data augmentation using gan for improved liver lesion classification. In: *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pp 289–293
 88. Mikołajczyk A, Grochowski M (2018) Data augmentation for improving deep learning in image classification problem. In: *2018 International interdisciplinary PhD workshop (IIPhDW)*, pp 117–122
 89. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative Adversarial Nets. In: *Proceedings of the 27th international conference on neural information processing systems*, vol 2, MIT Press, Cambridge, NIPS'14, pp 2672–2680
 90. Sak H, Senior AW, Beaufays F (2014) Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR* abs/1402.1128. <http://arxiv.org/abs/1402.1128>
 91. Seijo-Pardo B, Porto-Díaz I, Bolón-Canedo V, Alonso-Betanzos A (2017) Ensemble feature selection: homogeneous and heterogeneous approaches. *Knowl-Based Syst* 118:124–139. <https://doi.org/10.1016/j.knsys.2016.11.017>. <http://www.sciencedirect.com/science/article/pii/S0950705116304749>
 92. Stehlik M, Ruiz MP, Stehliková S, Lu Y (2020) *Contemporary experimental design, multivariate analysis and data mining, Chap on equidistant designs, symmetries and their violations in multivariate models*. Springer, New York

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.