International Journal on Document Analysis and Recognition (IJDAR)
https://doi.org/10.1007/s10032-018-0313-2

ORIGINAL PAPER

A comparative study of delayed stroke handling approaches in online handwriting

Esma F. Bilgin Tasdemir[1] · Berrin Yanikoglu[1]

Received: 17 August 2017 / Revised: 22 October 2018 / Accepted: 27 October 2018

Abstract

Delayed strokes, such as i-dots and t-crosses, cause a challenge in online handwriting recognition by introducing an extra source of variation in the sequence order of the handwritten input. The problem is especially relevant for languages where delayed strokes are abundant and training data are limited. Studies for handling delayed strokes have mainly focused on Arabic and Farsi scripts where the problem is most severe, with less attention devoted for scripts based on the Latin alphabet. This study aims to investigate the effectiveness of the delayed stroke handling methods proposed in the literature. Evaluated methods include the removal of delayed strokes and embedding delayed strokes in the correct writing order, together with their variations. Starting with new definitions of a delayed stroke, we tested each method using both hidden Markov model classifiers separately for English and Turkish and bidirectional long short-term memory networks for English. For both the UNIPEN and Turkish datasets, the best results are obtained with hidden Markov model recognizers by removing all delayed strokes, with up to 2.13% and 2.03% points accuracy increases over the respective baselines. In case of the bidirectional long short-term memory networks, stroke order correction of the delayed strokes by embedding performs the best, with 1.81% (raw) and 1.72% (post-processed) points improvements above the baseline.

Keywords Online handwriting · Delayed strokes · Accented characters

1 Introduction

Online handwriting recognition is the task of interpreting handwritten input, at character, word, or line level. The handwriting is represented in the form of a time series of coordinates that represent the movement of the pen-tip which is captured by a digitizer equipment.

One of the well-known problems in online handwriting recognition domain is the so-called *delayed strokes* that increase timing variations in online handwriting. A delayed stroke is 'a stroke, such as the crossing of a "t" or the dot of an "i," written in delayed fashion (not immediately after the corresponding character's body).' Writers have different writing practices as to when they write such strokes (right after the character body or after the word is written), which cause variations in the resulting sequence, which in turn degrades recognition performance.

✉ Esma F. Bilgin Tasdemir
efbilgin@sabanciuniv.edu

[1] Faculty of Engineering and Natural Sciences, Sabancı University, 34956 Istanbul, Turkey

As with other sources of variations, one option is to try to remove the variation by putting the data in a canonical form (e.g., reordering the strokes) or using large amounts of data to represent all possible variations in the training data. As large amounts of data are not always available, different approaches to the problem have concentrated on reducing the source of the variations. One suggested alternative is to remove delayed strokes altogether, which may be suitable for languages where delayed strokes are either not very common or where words are not differentiated by such strokes. For instance, accents are common in French, but words can still be recognized to the large extent even if the accents were removed. A recent variation of this approach uses the *hat feature* to mark sampling points deemed to be associated with the removed delayed strokes. Yet another alternative is to try to embed the delayed strokes in the writing sequence in a canonical order (e.g., always right after the corresponding letter body is drawn). Finally, there are also systems that try to overcome the problem by using only *offline* features in order to gain invariance toward writing order variations, while losing some or all of the timing information.

Hidden Markov models (HMMs) have been the most popular technique for online handwriting recognition until recent years [15,16,21], to be surpassed by deep learning techniques, especially in problems where large amount of training data are available [10,22]. In particular, recurrent neural networks (RNNs) and a special kind of RNNs—long short-term memory neural networks (LSTMs)—have been very successful in both online and offline handwritten and machine-print recognition problems in recent years [11]. LSTMs are capable of learning long-range temporal dependencies from unsegmented input streams, which makes them suitable for sequence recognition tasks such as handwriting recognition.

Despite the success of deep learning systems, HMMs remain a viable alternative, especially when the computational resources are limited or in domains where training data are not abundant or in hybrid systems together with various kinds of artificial neural networks (ANNs) [17,23,28,29]. A comprehensive survey of handwriting recognition approaches is out of scope of this paper, but can be found in [18,24,25].

While delayed stroke handling is used as a preprocessing in some studies [5,11,17,22], very few studies report how delayed stroke handling affects performance. Jaeger et al. report 0.5% points improvements for English by identifying and removing delayed strokes [17] using the hat feature. Delayed strokes pose a big problem, especially in languages written with many diacritical marks and accents (e.g., Arabic, Farsi, Turkish). Ghods et al. report 6.8% points improvement in Farsi, using reordering of delayed strokes with sub-word models [7]. The most extreme improvement are reported by Abdelaziz et al., where an increase from 2 to 92% is reported with reordering of delayed strokes in Arabic. Authors report that more than 60% of characters have delayed strokes or diacritical marks [2]. Note that if there is no special processing for handling of delayed strokes, they can affect recognition performance since the variability in the writing order translates into variability in the alignment of the input to the states in the models.

This study proposes a new method for automatically detecting delayed strokes and evaluates the effects of different delayed stroke handling approaches proposed in the literature. The evaluation is done separately for English and Turkish using hidden Markov models (HMMs) which have been the main approach in recognizing handwritten text, and Bidirectional LSTM (BLSTM) networks, which have outperformed other methods on the problem of recognizing unsegmented cursive handwriting recently.

We review existing definitions for defining delayed strokes and propose a new definition in Sect. 2. Then, suggested delayed stroke handling alternatives from the literature are given in Sect. 3. Section 4 describes the HMM and BLSTM recognizers, and Sect. 5 presents experimental results, for both the UNIPEN dataset for English and ElementaryTurkish dataset for Turkish.

2 Delayed strokes

A stroke is a pen trajectory starting with a pen-down point and ending with a pen-up point. It can thus be a full character, a part of a character or several characters written consecutively. When a stroke is separated from the character body it belongs to by one or more strokes, it is said to be ‘delayed.’ For instance, the dot of an ‘i’ or the cross of a ‘t’ can be delayed, when the dot or cross is not written immediately after the corresponding letter body.

Delayed strokes occur in multi-stroke characters, but not every multi-stroke character is written in delayed fashion. For instance, uppercase characters are typically written one character at a time; hence, even multi-stroke letters (e.g., ‘E’) are not written with delay. In fact, each script has different strokes that are typically written in delayed fashion. These strokes can be either diacritical marks or integral parts of characters. Hence, the delayed stroke problem should ideally be examined for each language/script.

An exact delayed stroke detection can only be done *after* recognition, or more specifically after letter boundaries are known, by considering those letter parts that are written separately from the corresponding character bodies. For instance, the dot of an ‘i’ is not considered delayed if it is written right after the letter body, even though it involves a pen-up movement with a backward move of the pen. Nonetheless, there have been various definitions, such as calling all backward moves after pen-up as delayed strokes, so as to detect and handle delayed strokes automatically during preprocessing.

Once such a working definition is at hand, the delayed strokes can be detected and then handled according to a chosen method, of which there are a few. In the remainder of the paper, we use the terms ‘definition’ (to be consistent with previous work) and ‘algorithm’ interchangeably, to refer to the algorithm used to describe/detect delayed strokes automatically.

Delayed strokes of Latin-based scripts can be investigated in three groups: (1) those that are written spatially above other strokes of the character, mostly without touching them, such as i-dots, umlauts (pair of dots) or other similar accents (e.g., accents grave and breve); (2) those that are written spatially below other strokes of the character, with or without touching them (e.g., cedilla and hook); and (3) those that are spatially overlapping with other strokes of the character, such as crosses of ‘f,’ ‘t,’ ‘z’ and ‘x.’ Figure 1 shows some examples of characters with diacritical marks as delayed strokes from the UNIPEN dataset.

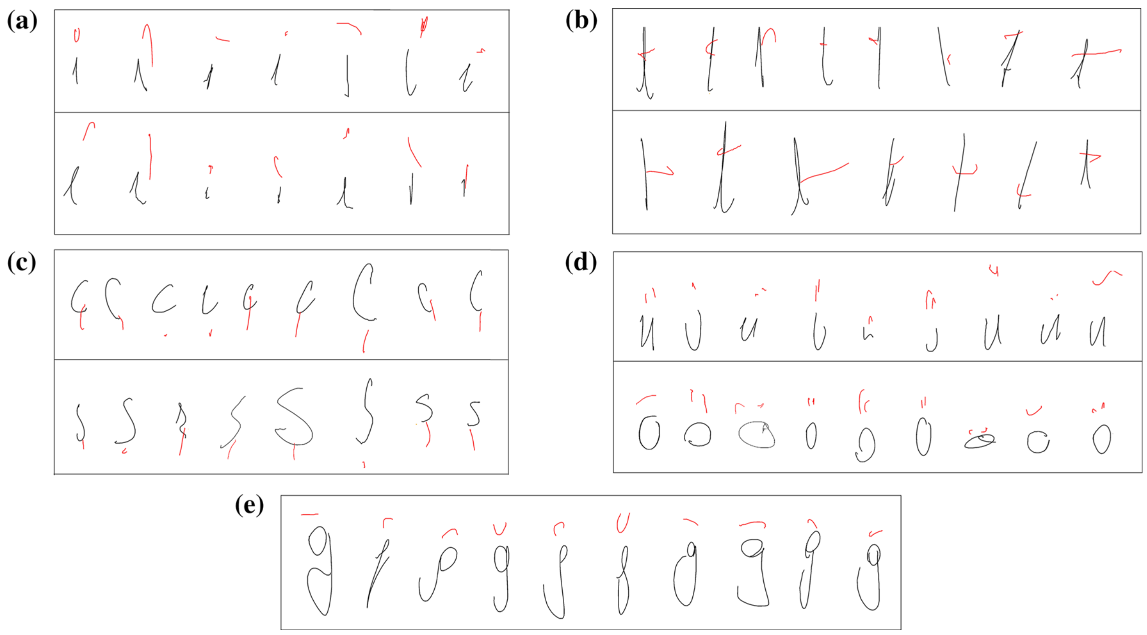

Fig. 1 Samples of characters with potential delayed strokes: **a** 'i' with dot, **b** 't' with cross, **c** 'ç' and 'ş' with cedilla, **d** 'ü' and 'ö' with umlaut and **e** 'ğ' with breve

2.1 Existing definitions

The definition given in the beginning of Sect. 2 ['strokes separated from the corresponding character body by other stroke(s)'] is not very useful for *automatically* detecting delayed strokes. There are other definitions in the literature for delayed strokes, proposed in the context of automatically detecting and handling them. For instance, [16] defines delayed strokes as:

> *…strokes such as the cross in 't' or 'x' and the dot in 'i' or 'j,' which are sometimes drawn last in a handwritten word, separated in time sequence from the main body of the character.*

Another definition is given by [17] as:

> *…usually a short sequence written in the upper region of the writing pad, above already written parts of a word, and accompanied by a pen movement to the left.*

Finally, [11] identify delayed strokes as:

> *…those strokes that are written above already written parts, followed by a pen movement to the left.*

In this work, we make a new working definition which can be used for detection of delayed strokes. We start with the minimal definition based on a backwards movement, which expectedly marks too many strokes as delayed due to its very general/simple description:

> *…a new stroke starting with a backwards pen movement from the last pen-up point.*

Improving the minimal definition is possible through incorporation of script-specific features such as absolute and relative size and x- and y-position of the stroke with threshold values learned from samples from the target script. Adding more constraints increases detection precision for the cost of increasing complexity of the definition.

In the next section, the minimal definition is expanded for English to obtain the proposed definition. The new definition is learned *automatically* from the handwriting statistics learned from the UNIPEN dataset. Specifically, a subset of 1000 random words are marked manually for the presence and type of delayed strokes: Each sample is visually inspected at stroke level and the strokes that correspond to a dot or a cross of a character are marked, along with whether they are 'delayed' or 'regular.'

This 1000-word training set contains a total of 5124 strokes and a total of 816 dots and crosses that can be written in delayed fashion. Of these 816 strokes, 332 are delayed (225 i-dots and 107 t-crosses), while the rest (484) are not. Overall, the number of non-delayed strokes is 4792. Details of the UNIPEN dataset itself can be found in Sect. 5.1.

After generating the ground truth dataset, the decision tree learning algorithm is used to minimize the delayed stroke classification error, subject to some constraints regarding the tree size.

2.2 Proposed definition for delayed strokes in English

The English script uses 26 letters from the Latin alphabet. Parts of letters and diacritical marks can be written in delayed fashion: dots for the letters 'i' and 'j,' bar-like strokes (crosses) in 'f,' 't,' 'z,' and 'x,' and diacritical marks in borrowed words. Delaying dot-type strokes is very common, followed by crosses, while diacritical marks like accent, umlaut and cedilla are used mostly in loan words like naïve, café and façade.

We formulate a delayed stroke definition for English by concentrating on dots and crosses, as they cover the overwhelming majority of delayed strokes in English. Indeed, all of the strokes that are delayed in the randomly selected 1000-word training subset of UNIPEN are either i/j-dots or crosses.

We start with describing each stroke of a word in terms of the following set of measurements which conveys information about the shape of the stroke itself and its position within the global context of the word it belongs to. In this study, the baseline and corpus line refer to the baseline of the text and the top of the lowercase letter bodies as in [17], while midline and corpus height are derived from them as the midpoint and height of the region between the two. The new features are:

- positions w.r.t baseline, corpus line and midline: as percentage of sampling points lying above these lines
- height of bounding box/width of bounding box
- normalized height of bounding box : height/corpus _height
- normalized width of bounding box : width/corpus_height
- depth of the stroke: distance to the middle point from line connecting two ends
- normalized stroke length: stroke_length/corpus_height
- stroke curvature : angle between lines connecting ends to the middle point

After feature extraction, we train a decision tree classifier using the CART decision tree learning algorithm and evaluate its performance using tenfold cross-validation on the 1000-word dataset.

As the data are highly unbalanced (332 delayed strokes vs. 4792 regular strokes), random subsampling is applied to regular strokes, so that the ratio of positive and negative examples is 1/4. Also, a higher cost (x2) is set for the misclassification of the delayed strokes (false negatives). Class prior probabilities are empirically determined from class frequencies in the dataset. When the training is complete, the full tree is pruned to keep the number of rules small, to make the definition simple and for better generalization.

The resulting tree classifies a stroke in a given word as 'delayed' or 'regular' based on the features of that stroke. The rules of the tree can be extracted, yielding a working definition for automatic detection of delayed strokes. In Algorithm 1, we present the procedure for detecting the delayed strokes according to the new definition derived from the tree rules.

The threshold for backward movement, which is the distance skipped backwards over the last written letter, is set to average character width. The number of characters is estimated using a heuristic method given in [22], while the baseline and corpus line are calculated by regression through minima and maxima method as described in [11].

```
Input:  W: A "word" (a set of strokes)
        S: A stroke in W
Output: Return True if S is a delayed stroke and False otherwise
$W_{end}$ = x-coordinate of the last pen-up before S
$S_{beg}$ = minimum of the x-coordinates in S
height = normalized height of bounding box of S
$W_{ch\_width}$ = average character width in W
$W_{c\_line}$ = y-coordinate of the corpus line of W
$W_{c\_height}$ = difference between y-coordinates of the corpus line
and the base line of W
if $W_{end}$-$S_{beg} \geq W_{ch\_width}$
 AND 0.86% or more of points in S are above $W_{c\_line}$
 AND height $< 1.45*W_{c\_height}$ then
    Return True;
else
    Return False;
end
```

Algorithm 1: Proposed definition for detecting delayed strokes (see above for definitions).

Based on the upper and lower regional characteristics of strokes, a discrimination for the type is also made, by simply considering whether there are points in the upper region of the detected delayed stroke. Those with points in the upper region are labeled as crosses, while others are considered dots.

2.3 Detecting all dots and crosses

The new definition finds dots and crosses that are delayed, but any subsequent handling of delayed strokes can potentially increase variation in writing if *all* (delayed or not) dots and crosses are not handled in the same way. For instance, with the approach of removing delayed strokes, some of the characters will be stripped off the delayed parts while their counterparts with *non-delayed* strokes are left intact.

In order to study this issue, we developed a new definition for detecting *all* dots and crosses—whether they are delayed or not—using the same decision tree learning approach (without enforcing a backward movement constraint), and using the appropriate data (the 816 strokes corresponding to the *all* dots and crosses in the training dataset and randomly selected 3000 strokes from the rest). The procedure for detecting the delayed strokes according to the new definition is shown in Algorithm 2.

```
Input: W: A "word" (a set of strokes)
       S: A stroke in W
Output: Return True if S is a delayed stroke and False otherwise
height = height of bounding box of S
depth = depth of S
above_C = percentage of points above corpus line in S
above_M = percentage of points above
          (baseline + corpus line)/2 in S

if above_M ≥ 87%
AND height < 1.9*corpus height
AND ((above_C ≥ 42% AND depth < 381)
OR (above_C < 42% AND height < 0.56*corpus height)) then
    Return True;
else
    Return False;
end
```

Algorithm 2: DetectAll: Definition for detecting all (delayed or not) dots and crosses.

The performance of the three definitions (minimal, proposed and DetectAll) is given in Table 1. As can be seen here, the minimal definition based on the often cited backward movement has a very high false positive rate (25.1%), while the proposed definition has about 10.3% error rate (equal false positive and negative). Detecting all dot and cross-like strokes has a lower error rate; however, as it is computed over all strokes (delayed or not), the absolute number of errors is almost twice as much as the proposed method (297 versus 158).

Table 1 Accuracies of different delayed stroke definitions

Definition	FN	FP
1 Minimal	7.5% (25/332)	25.1% (301/1200)
2 Proposed	10.2% (34/332)	10.3% (124/1200)
3 DetectAll	2.6% (21/816)	9.2% (276/3000)

The type labeling is evaluated separately, following the proposed definition. For the 298 ($= 332 - 34$) strokes that are correctly detected as delayed stroke, type identification is correct for 95.30%. On the other hand, out of the 34 missed delayed strokes (not recognized as delayed), about 40% are dot strokes and 60% are crosses.

3 Delayed stroke handling alternatives

There have been a variety of alternatives to address the variations caused with delayed strokes, namely discarding (removing) delayed strokes altogether; embedding delayed strokes as if they are written after each corresponding character (also called reordering); and their variations.

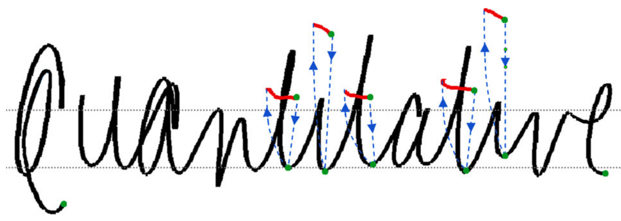

Fig. 2 Detected delayed strokes (shown in red) are embedded as the arrows indicate. Best viewed in color (color figure online)

3.1 Removal

In this approach, delayed strokes are identified and removed before the recognition phase [1,3]. Here, the assumption is that they are somewhat superfluous and may not be necessary for recognition. While this assumption may largely hold for English, many more words may become indistinguishable in the absence of dots and accents in certain languages, such as Turkish and Arabic.

Information about removed delayed strokes can be used in post-processing for lexicon reduction [1] and disambiguation of similar word bodies [13]. While useful, the shortcoming of these post-processing attempts is that the recognition process itself cannot make use of the information about delayed strokes. Also, there is a risk of removing correct words during lexicon reduction due to faulty delayed stroke detections.

3.2 Embedding

Another alternative to deal with delayed strokes is to embed them in the writing sequence such that they are reordered to appear after the corresponding letter. This reordering normalizes the sequence to a canonical writing order and thus reduces or removes sequence variations and extensions. Embedding is illustrated in Fig. 2.

Some handcrafted rules are used for deciding attachment points of delayed strokes in [5]. For Arabic, a vertical projection of detected delayed strokes is used in [4], while a more complex approach of segmenting the word body into strokes and sub-strokes to find correct projection locations of delayed strokes is used in [2].

The main difficulty with embedding is to find the correct attachment points of delayed strokes, which can in turn adversely affect recognition. For English, the rules are designed for each type of delayed strokes separately. While there is some difficulty in deciding attachment points for i-dots, this is less of a problem for t-crosses.

In this study, for any stroke that is classified as an i-dot by some detection mechanism, we first find the nearest local

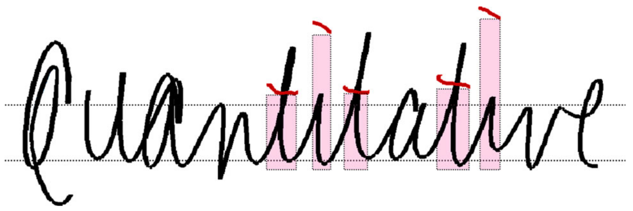

Fig. 3 Hat feature value is 1 for points lying below the delayed strokes (in the pink rectangular area) and 0 for the rest (color figure online)

maximum of the y-coordinate among the subsequent points (called *nearest_max*). Next, we find the first local minimum of the y-coordinates of the points after *nearest_max* along the x-axis (called *nearest_min*). The i-dot is attached right after *nearest_min* by modification of the *nearest_min* as a pen-up point.

As for the t-crosses, we utilize the observation that crosses are in relation with letter bodies most of the time. If there are any pen-up points lying under a t-cross-stroke and within its x-coordinate range, the cross-stroke is attached right after the one with the greatest x-coordinate. Otherwise, we make a sequential search for a convenient attachment position starting from the point with greatest y-coordinate lying under the t-cross. We stop the search when the next point in the sequence has greater y-coordinate than the previous one. The cross-stroke is inserted to its new position afterward.

3.3 Hat feature

In this approach, the delayed strokes are *removed*, but the existing feature vector is expanded with a binary feature, to indicate the location of the removed strokes [11,17,22]. The binary feature takes on the value of 1 at locations that were under a removed delayed stroke, and zero otherwise (see Fig. 3).

An alternative is to keep the original input and *add* the hat feature to obtain an extended feature set. In this case, the hat feature serves to highlight the presence of a delayed stroke event.

3.4 Delayed strokes as characters

In this approach, delayed strokes are considered as special characters in the alphabet [15,16,33]. Words which contain characters that can be written in delayed fashion are represented with all the alternative forms (corresponding to the possible writing orders) in the lexicon. For instance, if the i-dot is represented with a '.' and a cross is represented by a '-,' then, the word '*it*' has three possible spellings in the dictionary: {i.t-, it.-, it-.}. While this seems like the best approach in terms of not losing information or interfering with the writing order, it is not suitable for mid-to-large scale vocabulary tasks, as the hypotheses space can grow dramatically.

3.4.1 Adding at the end

Another method for handling delayed strokes is moving them to the end of the word such that they are appended to the last sampling point of the word, in the order of appearance in the writing order [7]. The main issue with this method is that it is not suitable for sub-word-based recognition systems, such as character HMMs, as it separates delayed strokes from corresponding letter bodies with respect to time.

3.4.2 Removal and embedding hybrid

We have also tried a hybrid method of removing the i-dots and embedding the t-crosses. This hybrid approach was due to the observation that deciding on the correct position of a dot and embedding it is difficult, as dots are not constrained to be in a precise position with respect to the corresponding letter body, while crosses are. Furthermore, while rare, some writers omit writing dots altogether in certain languages, so removing all i-dots may be beneficial, as it would make the dotted and not dotted versions of the words match.

3.4.3 Definitions for different handling methods

We have made two extensions to the minimal definition and presented the related detection procedures in Algorithm 1 (proposed) and Algorithm 2 (detection of all dots and crosses). While each of them can be used with each handling method, some are more suitable for particular methods than the others. For a fair evaluation, it is important to detect delayed strokes using the appropriate definition for the chosen handling method. With this in mind, embedding and hat feature approaches are evaluated by detecting delayed strokes according to the proposed definition, while removing approach is evaluated by detecting delayed strokes using both of the definitions. Stroke-type labeling can be of use if character-specific heuristics will be used for finding the attachment points.

4 Recognition systems

We used hidden Markov model and bidirectional long short-term memory (BLSTM) classifiers that are trained from scratch for each considered definition-handling method pair.

HMMs can be used for modeling strokes, characters or words; in this work, we train HMM character models since the model set increases with the size of the vocabulary in case of word models and it is more convenient to use character models when there are not enough samples for each word model. For training and recognition, we use the standard forward and backward passes and Viterbi decoding, as explained in [26].

The BLSTM systems can use unsegmented sequences of varying lengths to indirectly model constituent characters. We train the networks by truncated backpropagation algorithm [30]. The letter probabilities that are output by the networks are used by the Connectionist Temporal Classification (CTC) beam search decoding algorithm [8] with a beam width of 100 and without a dictionary, for finding the most probable output transcription.

4.1 Preprocessing

Slant and height normalization are applied for preprocessing, to remove size and slant variations. Number of characters are estimated as described in [22] for each word. Then, equidistant resampling is performed at a 50 sampling points per character rate.

Delayed stroke handling is done at this stage according to the chosen definition and processing method.

4.2 Feature extraction

We use a set of 10 features that are summarized below. The first 9 features are defined in [20], while the last one is newly designed to distinguish among similar strokes by incorporating more neighborhood information. Together, the features are capable of capturing important characteristics of writing and many of them are used in previous studies.

- (1–2) delta: differences between the x- and y-coordinates of the current and previous points;
- (3–4) sine and cosine of the angle between x-axis and the line joining consecutive points;
- (5) curvature angle: the angle between the lines to the previous and the next point;
- (6) vicinity linearity: average squared distance of each point in the vicinity to the straight line from the first to the last vicinity point;
- (7) vicinity slope: a pair of features such that cosine and sine of the angle of the straight line from the first to the last vicinity point;
- (8) pen-up/down: a binary feature showing whether a sampling point is an up point (pen is lifted up here) or a down point (pen is touching the writing pad);
- (9) normalized x: the x-position taken after high-pass filtering (subtracting a moving average of 5 previous points' x values from the current x).
- (10) distance to median y-value: for each point, distance to the median y-coordinate value of the given sample point sequence.

When the hat feature method is employed for delayed stroke handling, the binary hat feature is used along with this feature set.

4.3 Classifiers

We evaluate the delayed stroke detection and handling methods with HMM and BLSTM recognizers.

An HMM is built for each of the 52 characters in the English alphabet with all upper and lowercase letters, all with linear topology (self and next transitions) and 20 states per character which is decided empirically. In the emitting states, the observation probability distributions are estimated by 35 mixtures of Gaussian components.

The number of Gaussians is decided using an iterative approach as suggested in [12].

Starting from an initial model with 3 mixtures, the number of mixture components is increased by one via splitting the Gaussian distribution with the highest weight until no further improvement was obtained on a validation set. The mean vectors of resulting Gaussian distributions are defined as the mean of the original one perturbed by plus or minus 0.2 standard deviations. After each split, the whole system of models is trained for a particular number of iterations until the next split. In our case, four iterations of training is applied before the next splitting. The average total training time was 40 h.

Once trained, the character models are concatenated to represent words. For system implementation, the well-known HTK software [14] is used.

The LSTM recognizer has two bidirectional recurrent layers (i.e., two forward and two backward) each with 100 LSTM cells without peepholes. The input layer size is 10 in accordance with the features listed in Sect. 4.2 (except for the hat feature method which has $10+1 = 11$ features). There are 52 output nodes for the symbols in the English alphabet and an extra node for the special blank symbol which is required by the CTC loss function [9] that we used in our recognition system. We utilize the open source, parallel implementation of CTC by Baidu.[1]

An adaptive learning rate optimization algorithm, root mean square prop (RMSProp) is used for minimizing the objective function, with an initial learning rate of 0.001. The networks are trained with mini-batches of 64 variable-length sequences that are padded to the same length before they are fed to the network. The training continues until no improvement is observed in recognition accuracy of the validation set for 10 epochs, and the best performing network is used on the test set.

The whole system is implemented with the open-source TensorFlow framework and the experiments are run on a NVIDIA Quadro K4000 graphical processing unit (GPU) card. Utilization of the GPU brings 7x improvement in training time (1.5 h per epoch on average) over training the

[1] https://github.com/baidu-research/warp-ctc.

network with a Intel© Xeon © CPU E5-2630 v2 2.60 GHz CPU.

5 Experiments

We evaluate the effectiveness of different delayed stroke handling approaches with both HMM and BLSTM recognizers. The BLSTM networks are evaluated on 7300 samples from a 3845-word lexicon.

The HMM systems are tested on the UNIPEN dataset for changing lexicon sizes (3500-word and 1000-word lexicons), in order to be comparable to previous work in the literature. Later on, we also carry out the same experiments on a small (9860-sample) dataset of Turkish, with a $\sim$ 2000-word lexicon, so as to see the results in a language where delayed strokes are more prominent.

5.1 Dataset

There are a few standard online handwriting datasets (IRONOFF [32], UNIPEN [31], IAMonDB [19]) that have been widely used in handwriting recognition research. We use isolated word collection of the UNIPEN dataset as it is the largest publicly available collection with a large vocabulary. The UNIPEN online handwriting database is a collection of handwritten samples (containing sets of characters, digits, words and texts), collected by a consortium of 40 companies and institutes over time. The whole collection consists of a total of 5 million characters by writers from all around the world, representing a wide variety of writing styles. The number of words contributed by different writers is highly unbalanced, ranging from less than 5 to more than 500. In fact, more than 2000 writers submitted very small number of words, specifically between 1 and 5, whereas there are 59 writers who contributed more than 400 words.

In this work, we use the isolated word collection (category 6) of the current publicly available version called $train\_r01\_v07$ training dataset. This collection contains 75,529 cursive or mixed-style words in total. The lexicon size is 13,913 words, with separate upper and lowercase versions for some of the words.

There is no common standard for how to split the UNIPEN data as training, test and development/validation sets. There is a development release called devtest_r01_v02 used in some works [16], but it is not publicly available. Hence, different works use different splits and thus report results on different portions of the UNIPEN dataset, rendering them not directly comparable.

In order to have reproducible and comparable results, we had previously split the data randomly into 10 random but similar subsets for tenfold cross-validation experiments and made the splits public.[2] Specifically, the subsets are roughly equal in number of writers ($\sim$ 320), number of samples ($\sim$ 7000) and size of lexicon (3500). The split also takes the skewed contribution per writer distribution of the UNIPEN (i.e., some with 1 sample and some with more than 400) into account and maintains similar distributions in each of the subsets. No writer appears in more than one subset, so the evaluation is writer independent. Lexicons of the subsets are not the same, but 1000-words were found to be shared between 5 of the subsets.

5.2 Methodology

We use different experimental protocols for HMM and BLSTM recognizers depending on the amount of data and the required training times.

For the HMM systems on UNIPEN, two series of HMM experiments are conducted with 3500-word and 1000-word lexicons in order to be comparable to results with same lexicon sizes reported in the literature (Tables 2 and 3). We do not use the tenfold cross-validation due to time complexity. Instead, five subsets (*set1*, *set2*, *set4*, *set5*, *set8*) described in 5.1 and that contain a common set of 1000-words in their lexicons, are each used as the hold-out test set while the rest of the 9 subsets are used for training. In the second set of experiments with the 1000-word lexicon, the same test sets are used, but they are reduced so that they contain only samples of these 1000 common words.

For the HMM experiments using the Turkish data (Table 4), all of the UNIPEN dataset and the training set of the ElementaryTurkish dataset are used together for training. The tests are conducted with the 2500 samples from a 2089-word Turkish lexicon which constitutes the test set of the ElementaryTurkish dataset.

For the BLSTM systems (Table 5), we randomly chose 2 of the 10 UNIPEN subsets for validation and testing, while the others are used in training the networks. Specifically, *set9* (319 writers, 3548 unique words, 7040 samples) is chosen as the validation set and *set7* (321 writers, 3845 unique words, 7300 samples) is used for testing the system performances.

Both HMM and BLSTM approaches are used together with the following delayed stroke methods:

- (1) No handling: The recognition performance without any delayed stroke handling is given as the baseline.
- (2) Removal of all delayed strokes: Delayed strokes are detected and removed from the sample word.
- (3) Embed all: All of the delayed strokes, i.e., i-dot and t-cross type delayed strokes, are reordered in time so that they are repositioned after the corresponding letter body.

[2] https://tinyurl.com/y7m3rv5w.

Table 2 HMM results for the 3500-word task on UNIPEN, using five-fold cross-validation

Handling method	Definition	Accuracy (%)
Baseline		81.01 ± 2.66
Remove all	Proposed	83.02 ± 2.32
Remove dots—embed crosses	Proposed	82.73 ± 2.35
Embed all	Proposed	82.20 ± 2.82
Remove all	DetectAll	80.79 ± 2.29
Hat feature, without removal	Proposed	79.12 ± 2.66
Hat feature, with removal	Proposed	79.07 ± 2.69

Table 3 HMM results for the 1000-word task on UNIPEN, using five-fold cross-validation

Handling method	Definition	Accuracy (%)
Baseline		83.99 ± 3.75
Remove all	Proposed	86.12 ± 3.03
Remove dots—embed crosses	Proposed	85.44 ± 3.29
Embed all	Proposed	85.08 ± 3.73
Remove all	DetectAll	84.54 ± 3.15
Hat feature, with removal	Proposed	82.76 ± 3.28
Hat feature, without removal	Proposed	81.72 ± 4.14

Table 4 HMM results for the ElementaryTurkish dataset with a 2089-word lexicon

Handling method	Definition	Accuracy (%)
Baseline		89.14
Remove all	Proposed	91.17
Embed all	Proposed	90.58
Remove dots—embed crosses	Proposed	89.94

– (4) Remove i-dots, embed t-crosses as proposed in this paper.
– (5) Binary hat feature representation of delayed strokes with removal.
– (6) Binary hat feature representation of delayed strokes without removal, but using extended feature representation (tested with the HMM systems only).

The handling methods are tested with appropriate definitions to achieve their intended effect. Thus, embedding and hat feature approaches are tested with detection according to the proposed definition while removing method is tested with detection using both the Proposed (Algorithm 1) and DetectAll (Algorithm 2) definitions.

5.3 HMM results on UNIPEN

We report results obtained over 5 test subsets with the 3500-word lexicon in Table 2 and those obtained with the 1000-word lexicon in Table 3. According to these results, removal of the delayed strokes using the detection method in Algorithm 1 (proposed definition) performs best, with 2.01 and 2.13% points above the baseline of no special handling. The second best approach is the hybrid method, followed by embedding. All three methods are found to show statistically significant improvements over the baseline in the five experiments, while the hat feature-based approaches resulted in performance degradation. The statistical significance was calculated using paired t-tests with results of the five sets of experiments.

5.4 HMM results on ElementaryTurkish dataset

We evaluate different handling approaches in a small dataset of Turkish text as well, in order to see their effectiveness on a script with more diacritical marks.

The modern Turkish alphabet is based on the Latin alphabet and differs from the English alphabet by the addition of six more characters (‘ç,’ ‘ğ,’ ‘ı,’ ‘ö,’ ‘ş’ and ‘ü’) and omission of three others (‘q,’ ‘w’ and ‘x’). Delayed strokes pose a serious problem in Turkish script due to increased number of characters with diacritics. In this work, we treat the cedilla in ‘ç’ and ‘ş’ as a t-cross as they are mostly attached to the body and umlaut and accent breve in ‘ğ,’ ‘ ö’ and ‘ü’ as an i-dot, as they are written disconnectedly and similar to a dot.

The ElementaryTurkish dataset that contains around 10K samples of isolated words from a 2089-word lexicon of 1st and 2nd Grade Turkish books, written by 113 different writers including children. We have made the dataset publicly available.[3]

The train set includes 7360 samples from a 1950-word lexicon by 79 writers, and the test set contains 2500 samples from a 2089-word lexicon written by 34 writers in the test set. The dataset split is designed such that the writers are not overlapping. The training set lexicon is covered by the test set lexicon.

Since the number of training samples of ElementaryTurkish dataset is very small, we use UNIPEN training data and Turkish data together for training. Specifically, models of characters which are common for English and Turkish are trained with both UNIPEN and ElementaryTurkish samples, while the others are trained with relevant samples in the corresponding dataset.

For the sake of brevity, we evaluate only embedding, removing and hybrid methods, as they performed the best on UNIPEN data.

[3] https://tinyurl.com/yc93rcf5.

Table 5 BLSTM word recognition accuracies obtained by detecting delayed strokes according to the proposed definition, on 7300 UNIPEN samples from a 3845-word lexicon (see Sect. 5.2)

Handling method	Validation raw (%)	Validation post-processed (%)	Test raw (%)	Test post-processed (%)	Best epoch
Baseline	39.76	69.40	42.79	74.40	125
Hat feature, with removal	41.69	71.76	44.46	75.84	125
Embed all	41.05	70.85	44.60	76.12	155
Remove all	38.92	69.17	40.38	73.74	120
Remove dots—embed crosses	39.20	69.21	42.04	75.08	135

Table 4 shows performances of the three methods on the ElementaryTurkish dataset. Removal of delayed strokes (detected according to the proposed definition) is again the best method, with a 2.03% points improvement over the baseline. Performances of embedding method and the hybrid method are above the baseline by 1.44 and 0.8 points, respectively.

5.5 BLSTM results on UNIPEN

In this section, we present the recognition results obtained by the BLSTM networks, each of which uses a different delayed stroke handling method, with delayed strokes detected according to our proposed definition.

We report two types of results for each recognizer in terms of word recognition accuracy in Table 5 for the validation and test sets. The first one is the accuracy of raw recognition result that is obtained by the CTC beam search decoder without a dictionary constraint. We use the raw recognition results in a post-processing step to find the most probable word matches in the test set lexicon. The second accuracy corresponds to the recognition performance after the dictionary matching.

In the post-processing stage, we use *PyEnchant*, a Python library for spell checking which use many popular spell checking algorithms including Ispell, Aspell and MySpell. For each string output by the recognizer, we search the possible matches in the lexicon of the relevant test set. The nearest word is found using a slightly modified version of the Ratcliff–Obershelp string matching algorithm [27] which calculates the similarity of two strings as as twice the number of number of matching characters divided by the total number of characters in the two strings.

According to the results in Table 5, stroke order correction of the delayed strokes by embedding performs the best, with 1.81 (raw) and 1.72 (post-processed) percent points improvements over the baseline (i.e., no special handling). The second best approach is the hat feature method. The other two approaches, removal and the hybrid method fail to excel over the baseline.

5.6 Error analysis

We observe that different handling methods have different shortcomings. Embedding delayed strokes for order correction is heavily afflicted with unaligned delayed strokes which are observed to be of i-dots most of the time (Fig. 4a, b). Unusual writing styles which do not comply with the heuristics for deciding attachment points are another source of error (Fig. 4c). Also, incorrect type detection of delayed strokes sometimes leads to incorrect repositioning when type-specific embedding strategies are applied (Fig. 4d). Lastly, since this method always integrates dots and crosses to character bodies, recognizers model such characters with those strokes; consequently, they suffer from omitted dots and crosses more than some other methods. The main problem with removing delayed strokes method is losing distinguishing information, which in turn leads to confusion with other characters. Case errors are common for all methods. If we do not count the case errors, scores increase to 85.92% for the 3500-word task and 87.73% for the 1000-word task.

5.7 Comparison with state-of-art

In this section, we give a summary of the previous results on the UNIPEN data to give context and also to show that our results are state-of-art level.

There are a limited number of studies on isolated word recognition using the UNIPEN dataset. In one of the earlier studies, Hu et al. [16] use a two-stage delayed stroke modeling combined with N-best decoding for large vocabulary tasks. For testing, they use a subset of devtest_r01_v02 that include more than 1500 samples from 100 writers which are not in the training set and report recognition rates of 90.5% and 87.2% for 1000- and 2000-word lexicons, respectively. Unfortunately, these test sets are not publicly available.

Marukatat et al. use neural networks (NNs) to predict the emission probabilities in a hybrid system combining HMMs with NNs [23]. They train the system with 30K words by 256 writers from UNIPEN dataset. They do not give particulars of their test set, but report 80.1% and 77.9% word recognition

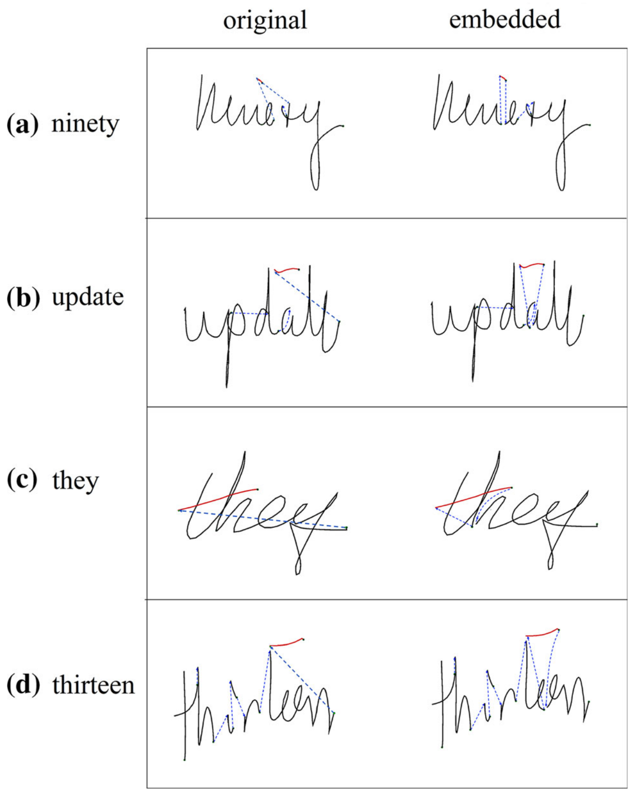

Fig. 4 Examples for sources of error with resulting incorrect embeddings: **a**, **b** unaligned delayed strokes; **c** unusual writing style; **d** incorrect type detection. The original writings are given in the first column

rates for multi-writer and writer-independent (omni-writer) recognition tasks, respectively, with a 2000-word lexicon.

Gauthier et al. also use a hybrid approach, with the combination of an online HMM-NN and an offline HMM system [6]. Using 40K words written by 256 writers from UNIPEN dataset for training, the combined classifier is tested on *parts of* a 1K sample set written by the *same* training set writers, all chosen from lowercase words. They report a 87% recognition rate with a 1500-word lexicon, which decreases to 79% when the lexicon size is increased to 10,000 words.

Our average best result with the 1000-word lexicon (86.12%) by the HMM approach is in accordance with what has been reported on the UNIPEN database before, using similar size lexicons.

To the best of our knowledge, there have been no previous studies that apply a deep learning method on the UNIPEN dataset. In this work, we present the first results obtained by using BLSTM networks with UNIPEN word collection. Our best system achieves 76.12% word accuracy with the 3845-word lexicon test set with no dictionary constraint during decoding.

However, none of the results reported so far are directly comparable, as they have been obtained using different subsets of the UNIPEN databases or different conditions (e.g., writer-independent vs. multi-writer) or different test sample selection procedures. In this work, we have split the UNIPEN dataset into writer-independent subsets randomly to avoid any biases. Also, we have used larger test sets in terms of both number of samples (3.5–4K for the 1000-word test) and number of writers for more reliable evaluation.

5.8 Evaluating the proposed definition

We conduct another set of experiments in order to measure to which extent errors in the proposed detection algorithm affect the recognition performance with different handling methods. For this, we compare the accuracies obtained when detecting delayed strokes with the proposed delayed stroke definition versus ground truth (manual) labels, using the 1000-sample dataset where delayed strokes are manually labeled (see Sect. 2.1). Specifically, we measure the performance of the five previously trained classifiers from Sect. 5.3 on the 1000-sample dataset, excluding the samples overlapping with the test and training subsets for an unbiased testing. As a result, 503 test samples from a 428-word lexicon are tested in total out of the 1000 samples.

The results are given in Table 6 along with the recognition accuracies when the proposed definition is used for detection of delayed strokes. As seen in this table, different handling approaches achieve very similar scores with both the ground-truth labels and the proposed definition. Furthermore, the removal of delayed strokes performs the best on the chosen test set, with both the ground truth and the proposed definition as well. The results are higher than those found in Table 3, due to the smaller test set and the variations in UNIPEN.

A perfect detection is of course not possible without first recognizing the input. However, our proposed working definition appears to be a viable solution to automatically detect delayed strokes, considering the very small differences from those obtained with the ground truth.

It should be noted that for this experiment, each HMM system is trained with samples where the delayed strokes are detected according to the *proposed* definition and handled according to the considered handling method. For a fully unbiased evaluation, the recognition systems should have been trained with the ground truth detection as well. Unfortunately, its too labor-intensive to label all of the samples in the UNIPEN dataset manually as we did for the selected 1000 words. So, the additional results we provide in this section may be taken with some caution as they may contain some bias toward the proposed definition.

5.9 Discussion

We make a set of observations based on the results of experiments and analysis of error cases.

Table 6 Comparison of the perfect (manual) detection and detection according to the proposed definition, for different handling approaches

Handling method	Perfect detection	Detection according to the proposed definition
Baseline (no handling)	89.53 ± 2.95	89.53 ± 2.95
Remove all	90.58 ± 4.23	90.24 ± 4.90
Remove dots—embed crosses	90.32 ± 4.03	88.19 ± 3.23
Embed all	88.57 ± 3.52	87.09 ± 4.77
Hat feature, with removal	87.40 ± 3.18	88.15 ± 3.94

Performance shows the average accuracy with the 5 previously trained HMMs on 503 UNIPEN samples from the 1000-sample UNIPEN subset (see Sect. 5.8)

– Removal, embedding and their hybrid, all perform better than baseline, for both English and Turkish with the HMM approach, while removal and hybrid removal approaches lead to performance degradation with BLSTMs. We can conjecture that BLSTMs can deal with delayed strokes sufficiently well so that a drastic measure such as the removal of delayed strokes harms the performance.
– With the HMM systems, removing all delayed strokes performs the best for both English and Turkish. This is a surprising finding for Turkish, since diacritical marks help differentiate between many similar words in Turkish (e.g., 'ol' vs. 'öl'; 'oldu' vs. 'öldü'). On closer inspection, we found that recognition performance is not affected by collisions between similar words since such collisions did not occur in the small lexicon; however, 303 of the 1442 test words would collide with a larger lexicon of 50K words.
  Furthermore, extra Turkish characters become the same as their diacritic-free counterparts in English (e.g., 'ö' becomes 'o' and 'ç' becomes 'c') with the removal methods. Considering this, we have used the character models trained with both UNIPEN and ElementaryTurkish datasets with the removal methods, rather than the ones trained with the latter one only. The use of the larger training data can also be a factor in the higher performance of the removal method for Turkish.
– The relatively lower success of the embedding approach with HMM systems can be attributed to faulty decisions at choosing the attachment point when relocating a delayed stroke. However, the BLSTMs can capitalize on the order-corrected strokes, even if the attachment points are not precisely found. While RNNs are able to learn long-range dependencies, reordering the delayed strokes such that they are closer to their corresponding letter bodies may have made it easier to associate these strokes with their corresponding letter bodies.
– The hat feature approaches which were used in a number of studies [11,17,22] and were reported to bring a small (0.5%) improvement [17], are observed to underperform the baseline for English and subsequently not tried for Turkish in HMM systems. However, the BLSTM systems benefit from the hat feature considerably.
– The deep learning techniques are shown to be superior to HMM systems in online handwriting recognition task [10,11,22]. In this work, the best HMM system achieves better results (83.02%, Table 2) than the best BLSTM system (76.12%, Table 5). However, it should be noted that these systems are not directly comparable since the experimental settings including the training and test set sizes and the lexicons are not the same for them. More importantly, the BLSTM systems do not benefit from any dictionary constraint during the decoding phase.
– According to the results in Table 5, the number of training epochs of a BLSTM network increases as the handwriting signal contains more information, which is as expected. Unlike HMMs, network performances decrease as the signal is overly simplified by the removal of some or all of the delayed strokes. This can be explained by the capability of deep networks in capturing long-term dependencies.

6 Conclusions

While delayed strokes are known to cause difficulties in the recognition of online handwriting, there has been no work comparing different alternatives that are suggested in the literature, to the best of our knowledge. Also, very few studies report how delayed stroke handling affects performance individually.

In this work, we provide a comprehensive evaluation of the techniques mentioned in the literature, for two of the most common handwriting recognition approaches (HMMs and RNNs) and for two languages (English and Turkish). Our contributions are as follows: (1) We develop and evaluate a working definition for detecting delayed strokes. (2) We evaluate delayed stroke handling methods suggested in the literature for their effectiveness in English and Turkish text with HMM and BLSTM systems. (3) We present reproducible results that are comparable to the state-of-the-art results on the well-known UNIPEN dataset. (4) We make our online handwritten word dataset for Turkish publicly available.

The best delayed stroke handling method achieves around 2% points improvement over the baseline in various tasks. However, the difference between the best and the worst method can exceed 4% points. We believe that this large difference among viable alternatives shows the importance of choosing the appropriate method for the task at hand.

We observe that delayed strokes have an impact in recognition performance with both the conventional HMM systems and the BLSTM networks of the deep learning approach, but the most effective handling method differs for each technique. While all preprocessing alternatives improve over the baseline for HMMs (for both languages), removal and the hybrid removal result in performance degradation in BLSTMs, for which we believe removal is too harsh a strategy. Also considering that removal would perform worse with large lexicons, we suggest to use the embedding method with HMMs and the hat feature with LSTM approaches, for large vocabulary tasks in English or Turkish.

Acknowledgements This work was supported by The Scientific and Technological Research Council of Turkey (TÜBITAK), under Grant Number 113E062.

References

1. Abdelazeem, S., Eraqi, H.M.: On-line Arabic handwritten personal names recognition system based on HMM. In: 2011 International Conference on Document Analysis and Recognition, ICDAR 2011, Beijing, China, September 18–21, 2011, pp. 1304–1308 (2011)
2. Abdelaziz, I., Abdou, S., Al-Barhamtoshy, H.: Large vocabulary Arabic online handwriting recognition system. CoRR. arxiv:abs/1410.4688 (2014)
3. Alimi, A.M.: An evolutionary neuro-fuzzy approach to recognize on-line Arabic handwriting. In: 4th International Conference Document Analysis and Recognition, ICDAR 1997, 2-Volume Set, August 18–20, 1997, Ulm, Germany, Proceedings, pp. 382–386 (1997)
4. Biadsy, F., El-Sana, J., Habash, J.: Online Arabic handwriting recognition using Hidden Markov Models. In: Tenth International Workshop on Frontiers in Handwriting Recognition, IWFHR 2007, IAPR. Newyork,USA, 2006 (2006)
5. Flann, N.S.: Recognition-based segmentation of on-line cursive handwriting. In: Advances in Neural Information Processing Systems 6, 7th NIPS Conference, Denver, Colorado, USA, 1993, pp. 777–784 (1993)
6. Gauthier, N., Artières, T., Gallinari, P., Dorizzi, B.: Strategies for combining on-line and off-line information in an on-line handwriting recognition system. In: 6th International Conference on Document Analysis and Recognition, ICDAR 2001, 10–13 September 2001, pp. 412–416. Seattle, WA, USA (2001)
7. Ghods, V., Kabir, E., Razzazi, F.: Effect of delayed strokes on the recognition of online Farsi handwriting. Pattern Recognit. Lett. **34**(5), 486–491 (2013)
8. Graves, A., Jaitly, N.: Towards end-to-end speech recognition with recurrent neural networks. In: Proceedings of the 31st International Conference on Machine Learning, ICML 2014, Beijing, China, 21–26 June 2014, pp. 1764–1772 (2014)
9. Graves, A., Fernández, S., Gomez, F.J., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25–29, 2006, pp. 369–376 (2006)
10. Graves, A., Fernández, S., Liwicki, M., Bunke, H., Schmidhuber, J.: Unconstrained on-line handwriting recognition with recurrent neural networks. In: Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, NIPS, Vancouver, British Columbia, Canada, December 3–6, 2007, pp. 577–584 (2007)
11. Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for unconstrained handwriting recognition. IEEE Trans. Pattern Anal. Mach. Intell. **31**(5), 855–868 (2009)
12. Günter, S., Bunke, H.: Optimizing the number of states, training iterations and Gaussians in an HMM-based handwritten word recognizer. In: 7th International Conference on Document Analysis and Recognition, (ICDAR 2003), 2-Volume Set, 3–6 August 2003, pp. 472–476. Edinburgh, Scotland, UK (2003)
13. Ha, J., Oh, S., J.K., Kwon, Y.: Unconstrained handwritten word recognition with interconnected Hidden Markov Models. In: Third International Workshop on Frontiers in Handwriting Recognition, IWFHR 1993, IAPR. Buffalo, USA, May 1993, pp 455–560 (1993)
14. HTK The Hidden Markov Model toolkit htk. http://htk.eng.cam.ac.uk/. Accessed 29 Oct 2016
15. Hu, J., Brown, M.K., Turin, W: Handwriting recognition with Hidden Markov Models and grammatical constraints. In: Fourth International Workshop on Frontiers in Handwriting Recognition, IWFHR 1994, IAPR.Taipei, Dec 1994 (1994)
16. Hu, J., Lim, S.G., Brown, M.K.: Writer independent on-line handwriting recognition using an HMM approach. Pattern Recognit. **33**(1), 133–147 (2000)
17. Jäger, S., Manke, S., Reichert, J., Waibel, A.: Online handwriting recognition: the NPen++ recognizer. IJDAR **3**(3), 169–180 (2001)
18. Koerich, A.L., Sabourin, R., Suen, C.Y.: Large vocabulary off-line handwriting recognition: a survey. Pattern Anal. Appl. **6**(2), 97–121 (2003)
19. Liwicki, M., Bunke, H.: IAM-OnDB—an on-line English sentence database acquired from handwritten text on a whiteboard. In: Eighth International Conference on Document Analysis and Recognition, ICDAR 2005, 29 August–1 September 2005, pp. 956–961. Seoul, Korea (2005)
20. Liwicki, M., Bunke, H.: Hmm-based on-line recognition of handwritten whiteboard notes. In: Tenth International Workshop on Frontiers in Handwriting Recognition, IWFHR 2006 (2006)
21. Liwicki, M., Bunke, H.: Handwriting recognition of whiteboard notes—studying the influence of training set size and type. IJPRAI **21**(1), 83–98 (2007)
22. Liwicki, M., Graves, A., Bunke, H., Schmidhuber, J.: A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In: Proceedings of the 9th International Conference on Document Analysis and Recognition, ICDAR 2007 (2007)
23. Marukatat, S., Artières, T., Gallinari, P., Dorizzi, B.: Sentence recognition through hybrid neuro-Markovian modeling. In: 6th International Conference on Document Analysis and Recognition, ICDAR 2001, 10–13 September 2001, pp. 731–737. Seattle, WA, USA (2001)
24. Plamondon, R., Srihari, S.N.: On-line and off-line handwriting recognition: a comprehensive survey. IEEE Trans. Pattern Anal. Mach. Intell. **22**(1), 63–84 (2000)
25. Plötz, T., Fink, G.A.: Markov models for offline handwriting recognition: a survey. IJDAR **12**(4), 269–298 (2009)

26. Rabiner, L.R.: A tutorial on Hidden Markov Models and selected applications in speech recognition. In: Proceedings of the IEEE, pp. 257–286 (1989)
27. Ratcliff, J.W., Metzener, D.E.: Pattern matching: the gestalt approach. Dr Dobbs J. **13**(7), 46–72 (1988)
28. Schenk, J., Rigoll, G.: Novel hybrid nn/hmm modelling techniques for on-line handwriting recognition. In: 10th International Workshop on Frontiers in Handwriting Recognition, IWFHR 2006, IAPR. , La Baule, France, Oct 2006, pp. 619–6230 (2006)
29. Schenkel, M., Guyon, I., Henderson, D.: On-line cursive script recognition using time-delay neural networks and Hidden Markov Models. Mach. Vis. Appl. **8**(4), 215–223 (1995)
30. Sutskever, I.: (2013) Training Recurrent Neural Networks. Ph.D. thesis, Toronto, Ont., Canada
31. The UNIPEN Consortium The UNIPEN project. http://www.unipen.org/products.html. Accessed 20 Oct 2016
32. Viard-Gaudin, C., Lallican, P.M., Binter, P., Knerr, S.: The IRESTE on/off (IRONOFF) dual handwriting database. In: Fifth International Conference on Document Analysis and Recognition, ICDAR 1999, 20–22 September, 1999, pp. 455–458. Bangalore, India (1999)
33. Vural, E., Erdogan, H., Oflazer, K., Yanikoglu, B.A.: An online handwriting recognition system for Turkish. In: Document Recognition and Retrieval XII,DRR 2005,San Jose, California, USA, January 16–20, 2005, Proceedings, pp. 56–65 (2005)