Abstract—We present an approach to leveraging both offline and online handwriting samples to build a single recognizer for recognizing both offline and online handwritings. Given a training set of offline handwriting samples and another set of online handwriting samples, a skeleton is derived first from each offline handwriting sample via vectorization. Then both the skeleton samples and online handwriting samples are normalized and rendered by using the same method to generate a combined training set of skeleton images. Finally a handwriting recognizer based on Deep Bidirectional Long Short-Term Memory (DBLSTM) and Hidden Markov Model (HMM) (e.g., [7]) by using the skeleton images derived from offline and online handwritten characters to enrich the training data for improving the recognition accuracy and robustness of an offline handwriting recognition system (e.g., [6]).

In this paper, we also want to leverage both offline and online handwriting data. Online handwriting samples are rendered to images and then merged with offline handwriting samples to build a single recognizer. However, naively rendered images do not help much because real-world offline handwriting images can have various styles. Although certain success has been demonstrated in [6], how to render realistic offline handwriting images from online samples remains largely an open problem. From a different perspective, we propose to address this problem by converting the two kinds of handwritings to “handwriting skeletons”, which can hopefully map offline and online handwritings to a common observation space. Such skeleton images can then be merged together to build a single recognizer which can recognize well the skeleton image derived from either an online or offline handwriting sample. To verify the effectiveness of the above idea, we build a handwriting recognizer based on Deep Bidirectional Long Short-Term Memory (DBLSTM) and Hidden Markov Model (HMM) (e.g., [7]) by using the skeleton images derived from two IAM benchmark databases [8], [9] of offline and online English handwritings plus an internal online handwriting corpus, and compare its performance with the recognizers built from either offline or online handwriting samples only.

The rest of this paper is organized as follows: Section II introduces our approach to deriving skeleton images from handwriting samples. Section III describes how to build a handwriting recognition system from skeleton images. Section IV presents experiments and results. Finally, Section V summarizes our work and draws some conclusions.

II. HANDWRITING SKELETON

A. Skeletonization for Handwriting

Skeletonization of character images is a process to extract a set of thin lines (usually one-pixel thick) which preserves the topological and geometric properties, and especially the literal content of original images. It is widely used as a method to reduce font variability in printed text recognition, usually along with skeleton matching algorithms (e.g., [10], [11]). In
this paper we want to use skeletonization as a way to blur the boundary between offline and online handwriting data so that we can use one set of models to recognize both of them.

The skeletonization used in this paper consists of three modules: stroke extraction, normalization and rendering. Fig. 1 shows the skeletonization pipeline and output examples of each step. The stroke extraction module is designed for extracting \((x, y)\) coordinates of skeletons from offline images (Fig. 1(a)). Since online handwriting data contains \((x, y)\) coordinates sequences naturally, they don’t need to pass the stroke extraction module. The normalization module then uses the skeleton coordinates to correct the baseline and slant (Figs. 1(b)(c)). Finally, the rendering module is used to convert coordinates to skeleton images. As a result, clear, baseline-corrected, and de-slanted handwriting images with sharp edges from both offline and online data sources are generated. They are visually similar in styles and can be used together for the following recognition system (Fig. 1(d)).

B. Stroke Extraction for Offline Handwriting Images

Many methods have been proposed to extract stroke coordinates from offline character images. There are methods based on thinning algorithms (e.g., [12]), correction of junctions points (e.g., [13]), curve fittings (e.g., [14]), use of principal curves (e.g., [15]), wavelet transform (e.g., [16]), self-organizing maps (e.g., [17]) and others. In this paper, a proprietary Microsoft tool for image vectorization is used, which can convert a pixel-based image into a vectorized one. One can experience its vectorization capability by trying out the feature of “convert images into Word and PowerPoint files” in Office Lens app [18]. We use the intermediate results (i.e., the center-line coordinates of handwritings) of this tool for stroke extraction purpose.

While skeleton coordinates are a compact descriptor for the natural shape of offline handwritings, the current imperfect stroke extraction algorithm may discard some useful details or introduce certain distortions. However, as we will demonstrate in later sections, these side-effects can be made up by the inclusion of a larger online or offline training data set.

C. Skeleton Normalization

The normalization module used in this paper consists of baseline and slant correction. An approach similar to that in [19] is used, which is based on Run Length Smoothing Algorithm (RLSA) ([20], [21]) and projection profile techniques (e.g., [22], [23]). Images are rotated by angles within a certain range and then smoothed by RLSA. Each rotation is evaluated by different objective functions to find an optimal angle for baseline and slant correction. Rotations are done on coordinates rather than images so unnecessary noises can be avoided.

D. Skeleton Rendering

In the rendering module Bézier curve is used to generate smooth strokes. Every 4 points inside a stroke, namely \(P_1, P_2, P_3, P_4\), are connected by a cubic Bézier curve. The curve starts at \(P_1\), goes toward \(P_2\) and arrives at \(P_3\) by coming from the direction of \(P_3\). In order to preserve the connectivity inside a stroke, the end point of one curve \(P_3\) is used as the start point of next. The remaining points (if less than 4) are simply connected. Stroke thickness is controlled by the thickness of Bézier curves. Resizing of output image is done
by the transformation of coordinates. In this paper, skeletons with 1 pixel thick drawn on text line images with a height of 60 pixels are used for both training and recognition.

III. HANDWRITING SKELETON RECOGNITION WITH DBLSTM-HMM RECOGNIZER

As mentioned above, we have built a DBLSTM-HMM based handwriting recognition system from skeleton images (e.g., [7]). Fig. 2 illustrates how to build and deploy such a system with the details explained in the following subsections.

A. Feature Extraction

For each handwriting skeleton image, a sliding window with a width of 30 pixels and a frame shift of 3 pixels is used to extract a sequence of frames. A horizontal cosine window is applied for smoothing. Because the sliding window for feature extraction has a size of $30 \times 60$ and we use raw pixel values as original features, the dimension of original feature vector is 1800. Then, Principal Component Analysis (PCA) is used to reduce the dimension of each feature vector to 50. Finally, 50-dimensional feature vectors are normalized such that each dimension of feature has a zero sample mean and unit sample variance on training set. The above control parameters are determined empirically by evaluating the performance on a validation set.

B. GMM-HMM based Alignment

A context-independent Gaussian mixture model HMM (GMM-HMM) based system is built first. For simplicity, every character is modeled by a 3-state left-to-right GMM-HMM with self-loop transitions, and each state has an observation probability density function (PDF) of a mixture of 64 Gaussian PDFs. There are 78 character classes in our character set, including 52 case-sensitive English letters, 10 digits, 15 punctuation marks (’! # $ % ’ ( ) * + , - . ; : ? ) and a “space” symbol. Consequently, we have 234 HMM states in total, which serve as the output classes of the DBLSTM. GMM-HMMs are trained by maximum likelihood training, and used to generate frame-level state targets for the training set by forced alignment.

C. DBLSTM Training

Long Short-Term Memory (LSTM) (e.g., [24], [25], [26]) is a special type of Recurrent Neural Networks (RNN) (e.g., [27]), which has been used to build handwriting recognition systems for a long time by using a Bidirectional LSTM (BLSTM) [28] and a Multi-Dimensional LSTM (MDLSTM) [29]. Recently, researchers of [30] used a bidirectional LSTM (BLSTM) as a feature extractor to build a GMM-HMM based offline handwriting recognition system. Inspired by the success of using DBLSTM-HMM for speech recognition [31], later, they built DBLSTM-HMM based offline handwriting recognition systems and achieved state-of-the-art performance on several benchmark tasks [7], [32]. In this paper, we also build a DBLSTM-HMM based system using BLSTM memory blocks with the same topology as that in [31]. Other state-of-the-art offline handwriting recognition systems (e.g., [33], [34]) were built by using MDLSTM with a Connectionist Temporal Classification (CTC) output layer [35].

In our experiments, we train a DBLSTM with five hidden layers, each containing 256 memory blocks (128 for forward and 128 for backward states). Each block has an input gate, an output gate, a forget gate and peephole connections [28], [31]. This results in roughly 1.8 million weights. The network parameters are trained by using an epochwise Back Propagation Through Time (BPTT) algorithm (e.g., [27]). Convergence is detected by using a validation set independent from training set. We stop training if we do not observe a validation error reduction for 10 epochs. The “Current toolkit” [36] is used for DBLSTM training.

D. Language Model and Decoding

The language model (LM) we used is a standard word trigram model with Good-Turing discounting. It is built by using a text corpus from Linguistic Data Consortium (LDC) (catalog number LDC2008T15) [37]. 500 (out of 4495) documents of the corpus are used. The lexicon has a vocabulary of 200k words with top occurring frequencies in the training corpus, which leads to 8% and 10% out-of-vocabulary (OOV) rates on IAM offline validation and test sets, and 6% and 8% OOV rates on IAM online validation and test sets respectively. We use SRILM toolkit [38] to train the language model.

In the decoding phase, we apply the DBLSTM in a feed-forward mode, converting every input frame to a posterior probability vector, which defines a distribution over the set of HMM states $\{s\}$. Then, the posterior estimates are divided by the prior $p(s)^{\alpha}$ of each HMM state $s$ to serve as a scaled state-dependent likelihood score, where a prior scaling factor $\alpha$ is empirically selected (e.g., [7]). The prior $p(s)$ is calculated from the forced-alignment results of the training set. Then, the HMM state likelihood scores are combined with state transition probabilities and the language model scores to determine the best recognition result. This is done efficiently.
A. Experimental Setup

DBLSTM-training. Furthermore, an internal Ink (i.e., online handwriting) database can outperform the baseline on test set. The model trained by combined data set including 30k lines of IAM-online data and 60k lines of IAM-offline data can outperform the baseline on test set. The results show that by using original IAM-offline handwritings. This may be due to the side-effects caused by the current imperfect vectorization algorithm, the above results are very encouraging. We are keeping improving as we add more skeletons from Ink database.

TABLE I

<table>
<thead>
<tr>
<th>Data Sets</th>
<th># of text lines</th>
<th># of words</th>
<th># of characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAM-Offline Training</td>
<td>6,159</td>
<td>53,823</td>
<td>227,731</td>
</tr>
<tr>
<td>IAM-Offline Validation</td>
<td>900</td>
<td>7,897</td>
<td>33,769</td>
</tr>
<tr>
<td>IAM-Offline Test</td>
<td>1,861</td>
<td>17,615</td>
<td>65,920</td>
</tr>
<tr>
<td>IAM-Online Training</td>
<td>5,364</td>
<td>29,093</td>
<td>127,398</td>
</tr>
<tr>
<td>IAM-Online Validation</td>
<td>1,438</td>
<td>7,152</td>
<td>31,654</td>
</tr>
<tr>
<td>IAM-Online Test</td>
<td>3,859</td>
<td>20,272</td>
<td>89,153</td>
</tr>
<tr>
<td>Ink 30k Training</td>
<td>29,981</td>
<td>192,032</td>
<td>1,046,079</td>
</tr>
<tr>
<td>Ink 60k Training</td>
<td>59,963</td>
<td>377,742</td>
<td>2,095,885</td>
</tr>
<tr>
<td>Ink 90k Training</td>
<td>89,963</td>
<td>512,008</td>
<td>3,191,770</td>
</tr>
</tbody>
</table>

by using a WFST-based decoder modified from the open source Kaldi toolkit [39].

IV. EXPERIMENTS

A. Experimental Setup

We use both IAM offline [8] and online [9] databases, which consist of handwritten English sentences captured by different methods. IAM-online data is acquired on a Whiteboard with E-Beam system, while IAM-offline data is scanned from papers with handwritten text. By following the standard handwriting recognition tasks provided by IAM [40], [41], we split the databases into 4 sets: a training set, two validation sets, and a test set. The text lines of all data sets are mutually exclusive. In this study, only the first validation set is used for monitoring DBLSTM-training. Furthermore, an internal Ink (i.e., online handwriting) database is also used for enriching our training and testing data. 90k lines of text images are rendered. We split them into 3 partitions to investigate the effect of training set size. Table I shows the sizes of different data sets in detail.

We firstly use separate training sets (IAM-offline, IAM-online) to build recognition systems and test on their corresponding test sets. Then we combine different training sets to build a general system and test on separate test sets. We use word error rate (WER) and character error rate (CER) as evaluation metrics for our experiments.

B. Experimental Results

Table II summarizes how skeletonization and combination of skeletons from offline and online data sets can affect the recognition performance on IAM-offline handwriting recognition task. By using only about 6k lines of IAM-offline skeleton images for training, the result is worse than the one by using original IAM-offline handwritings. This may be due to the side-effects of skeletonization. However, by combining about 5k lines of IAM online and 6k lines of IAM offline skeleton images together, we can get better results. The results keep improving as we add more skeletons from Ink database. The model trained by combined data set including 30k lines from Ink database can outperform the baseline on test set. The combined data set including 11k lines of IAM online-offline skeletons and 90k lines of Ink skeletons can help the recognizer to reduce WER on IAM offline test set from 28.04% to 23.84%. Table III shows how data combination can help the recognition of online handwritings. The model trained by using rendered online handwritings achieves a reasonable performance, but adding offline handwriting data helps the system to achieve a better performance. The model trained by using IAM on-off data can reduce the WER of test set from 28.67% to 23.42%. By adding 90k lines of Ink data, the test WER can be further reduced to 19.71%.

V. CONCLUSION

In this paper, we propose a method of using one system for both offline and online handwriting recognition. By converting offline and online handwritings into skeleton images, different kinds of handwritings are mapped to a common observation space so that a single robust recognizer can be built from a merged training set of offline and online handwritings, which is much larger than the individual ones and hopefully has a better coverage of different writing styles. By evaluating on offline and online data sets, it is observed that the combination of offline and online handwriting data can indeed help each other. Compared with the baseline systems trained on each type of data only, the system trained by using the proposed approach achieves a relative word error rate reduction of 8% and 31% on IAM offline and online test sets, respectively. Given the side-effects caused by the current imperfect vectorization algorithm, the above results are very encouraging. We are investigating and developing better vectorization techniques so that even more promising results can be achieved in future.

ACKNOWLEDGMENT

The authors would like to thank Filip Panjevic and Ivan Razumenic of Microsoft for providing an image vectorization tool to enable us to conduct this study.
REFERENCES


