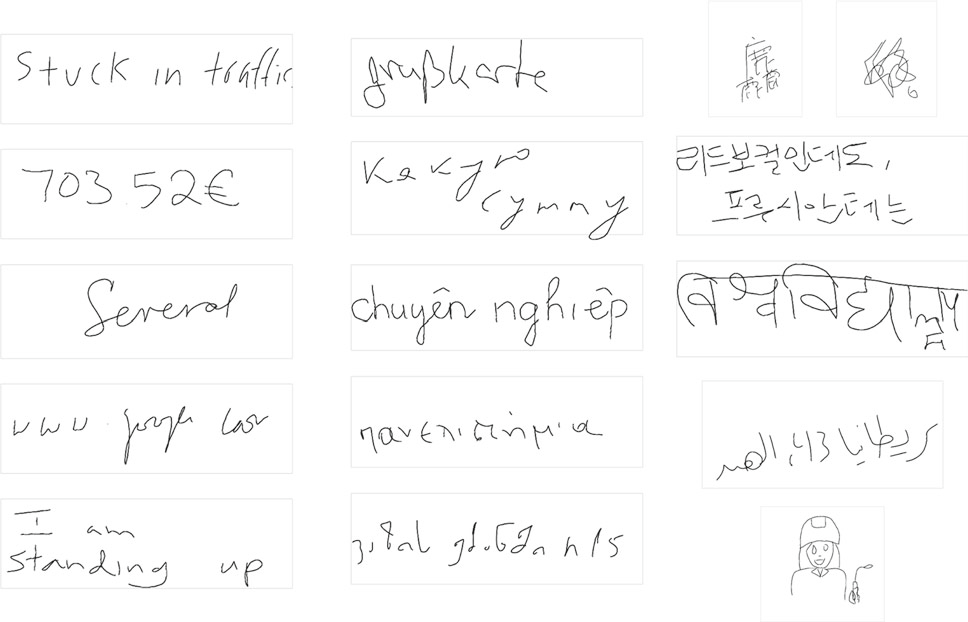
### Abstract

The paper introduces a deep neural network-based online handwriting system that supports 102 languages. It reduces error rates by 20-40%, achieves top results on IAM-OnDB, and uses Bézier curves for faster recognition. The system’s effectiveness was tested on various public datasets.

1 Introduction

This paper focuses on online handwriting recognition, which interprets user input in the form of an ink, a list of touch or pen strokes, into text. A stroke is defined as a sequence of points with position (x, y) and timestamp t.

Figure 1 demonstrates the versatility of an online handwriting recognition system that can interpret various languages and scripts. The system can handle different writing styles, content types, and multi-line inputs in English, as shown in the left column. The center column presents examples from five alphabetic languages similar to English, namely German, Russian, Vietnamese, Greek, and Georgian. The right column displays scripts significantly different from English. Chinese, with its complex and overlapping characters; Korean, with its syllable-based alphabet; Hindi, with its connecting “Shirorekha” line and grapheme clusters; and Arabic, which is written right-to-left and has position-dependent character shapes. The system can also recognize non-text Unicode symbols like emojis.



Online handwriting recognition is gaining importance due to the rise of mobile devices with touchscreens and styluses, and the challenges of typing certain languages on these devices. Early research in this field used segment-and-decode classifiers and hidden Markov models (HMMs), with some hybrid approaches combining HMMs and Feed-forward Neural Networks. The first HMM-free models were based on time delay neural networks (TDNNs), and recent work focuses on recurrent neural network (RNN) variants such as long short-term memory networks (LSTMs). The representation of online handwriting data has evolved from feature-based approaches to learned representations through deep learning, which is now prevalent in many domains like speech, computer vision, and natural language processing.

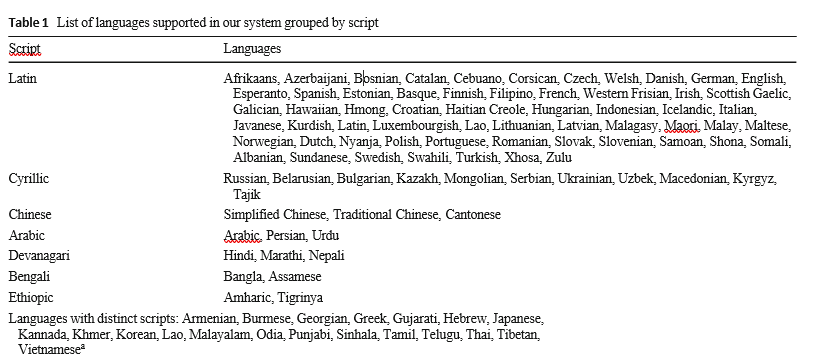
This paper presents a new online handwriting recognition system that leverages deep learning methodologies, marking a significant shift from the previous segment-and-decode system. The older system relied heavily on preprocessing, segmentation, and feature extraction heuristics, which involved over-segmenting the ink, grouping the segments into character hypotheses, and then classifying each character hypothesis using a relatively shallow neural network. The recognition result was obtained using a best path search decoding algorithm on the lattice of hypotheses, incorporating additional knowledge sources such as language models.

The new system, on the other hand, simplifies the process by eliminating the need for these heuristics. It consists of a simple stack of bidirectional Long Short-Term Memory networks (BLSTMs), a single Logits layer, and the Connectionist Temporal Classification (CTC) loss. A separate model is trained for each script, and to support potentially many languages per script, language-specific language models and feature functions are used during decoding. For instance, a single recognition model for the Arabic script is combined with specific language models and feature functions for Arabic, Persian, and Urdu language recognizers.

The new models have proven to be more accurate, smaller, and faster than the previous segment-and-decode models, eliminating the need for a large number of engineered features and heuristics. The paper presents an extensive comparison of the differences in recognition accuracy for eight languages and compares the accuracy of models trained on publicly available datasets where available.

In addition, the paper introduces a novel input representation based on Bézier curve interpolation, which produces shorter input sequences, resulting in faster recognitions. This system achieves a new state of the art on the IAM-OnDB dataset, both for open and closed training sets.

Furthermore, the paper proposes a new standard experimental protocol for the IBM-UB-1 dataset to enable easier comparison between approaches in the future. The main contributions of the paper include a detailed description of the recurrent neural network-based recognition stack, a novel input representation, and an evaluation protocol for the IBM-UB-1 dataset. These advancements mark a significant step forward in the field of online handwriting recognition



An overview our recognition models. In our architecture, the input representation is passed through one or more bidirectional LSTM layers, and a final softmax layer makes a classification decision for the output at each time step



The document discusses an end-to-end handwriting recognition model inspired by research in handwriting recognition, optical character recognition, and acoustic modeling in speech recognition. The model architecture is constructed from common neural network blocks, namely bidirectional Long Short-Term Memory (LSTM) and fully connected layers, and is trained using the Connectionist Temporal Classification (CTC) loss.

The architecture is similar to what is often used in the context of acoustic modeling for speech recognition, referred to as a CLDNN (Convolutions, LSTMs, and DNNs). However, it differs in four points:

1. It does not use convolution layers, which are found to not add value for large networks trained on large datasets of relatively short sequences typically seen in handwriting recognition.
2. It uses bidirectional LSTMs, which due to latency constraints is not feasible in speech recognition systems.
3. The architecture does not make use of additional fully connected layers before and after the bidirectional LSTM layers.
4. The system is trained using the CTC loss, as opposed to the Hidden Markov Models (HMMs) used in speech recognition.

This structure eliminates many components of the previous system, such as feature extraction and segmentation. The output of the final LSTM layer is passed through a softmax layer, leading to a sequence of probability distributions over characters for each time step. For CTC decoding, beam search is used to combine the softmax outputs with character-based language models, word-based language models, and information about language-specific characters.

The document also discusses the input representation. In earlier models, 23 per-point features were used to represent the input. However, in deeper and wider models, engineered features are found to be unnecessary and their removal leads to better results. This confirms the observation that learned representations often outperform handcrafted features when sufficient training data are available.

In the experiments presented, two representations are used:

1. Raw touch points: The simplest representation of stroke data is as a sequence of touch points. In the current system, a sequence of 5-dimensional points is used where the dimensions represent the coordinates of the touchpoint, the timestamp of the touchpoint since the first touch point in the current observation in seconds, whether the point corresponds to a pen-up or pen-down stroke, and whether it indicates the start of a new stroke.

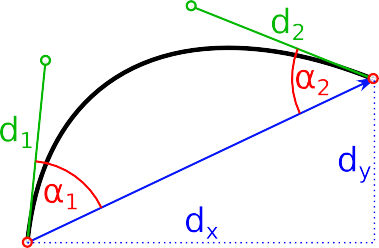
The document continues to discuss the flexibility of the system with respect to differences in the writing surface, such as area shape, size, spatial resolution, and sampling rate. Some minimal preprocessing is performed, including normalization of xi and yi coordinates, and equidistant linear resampling along the strokes.

The document also discusses the drawbacks of raw input data, such as resolution, length, and model complexity. To address these issues, the document proposes to represent a sequence of input points as a sequence of parametric cubic polynomials, or Bézier curves, and to use these as inputs to the recognition model.

Bézier curves for x, y, and t are cubic polynomials in s ∈ [0, 1]. The y values are within the range [0, 1], similar to how it is processed for raw points. The time values are scaled linearly to match the length of the ink.

For each stroke in an ink, the coefficients α, β, and γ are computed by minimizing the sum of squared errors (SSE) between each observed point i and its corresponding closest point (defined by si) on the Bézier curve. Given a set of coordinates si, computing the coefficients corresponds to solving a linear system of equations. This sets the time difference between the first and last points of the stroke to be equal to the total spatial length of the stroke.

The text describes an iterative curve fitting process. It alternates between minimizing error and updating coordinates until convergence. If the curve doesn’t fit well or is too bent, it’s split into two. The split point is determined by the smallest triplet of consecutive points.

 The figure shows the Bézier curve parameters used in a network: the vector between endpoints, distances and angles between control points and endpoints.

**The text discusses a method for data representation and recognition using Bézier curves and Long Short-Term Memory (LSTM) recurrent neural networks.**

**The process begins with the representation of data using Bézier curves. These curves are created by stitching together consecutive curves that can be represented by a single curve, resulting in a compact set of Bézier curves. Each curve is represented as a 10-dimensional vector, which includes various parameters such as the vector between the endpoints, the distance between the control points and the endpoints, the angles between control points and endpoints, the time coefficients, and a Boolean value indicating whether this is a pen-up or pen-down curve. This representation is roughly 4 times shorter than the raw representation, leading to faster recognition and better latency.**

**The sequences of curve representations are then processed using bidirectional LSTM recurrent neural networks. LSTMs are chosen due to their ease of training and good results. The input sequence is processed both forward and backward, and the output states of each layer are merged before being fed to the next layer. The number of layers and nodes is determined empirically for each script.**

**The output of the LSTM layers at each timestep is fed into a softmax layer to get a probability distribution over the possible characters in the script, including spaces, punctuation marks, numbers, or other special characters, plus the blank label required by the CTC loss and decoder.**

**The output of the softmax layer is decoded using CTC decoding. The logits from the softmax layer are combined with language-specific prior knowledge. For each of these additional knowledge sources, a weight is learned and combined linearly to guide the beam search during decoding. This combination of different knowledge sources allows the system to train one recognition model per script and then use it to serve multiple languages.**

**In most cases, the curve representations do not have a big impact on accuracy but contribute to the faster speed of the models. The exact configurations for several scripts in the production system are determined empirically and are subject to change.**

The text discusses the use of feature functions in a language recognition system. These functions introduce prior knowledge about the underlying language into the system. The system uses three feature functions:

1. **Character language models**: For each supported language, a 7-gram language model over Unicode codepoints is built from a large web-mined text corpus. The language model size has a smaller impact on the recognition accuracy due to the capability of recurrent neural networks to capture dependencies between consecutive characters.
2. **Word language models**: For languages that use spaces to separate words, a word-based language model is used, trained on a similar corpus as the character language models.
3. **Character classes**: A scoring heuristic is added which boosts the score of characters from the language’s alphabet. This provides a strong signal for rare characters that may not be recognized confidently by the LSTM.

The system is trained in two stages, on two different datasets. The first stage involves end-to-end training of the neural network model using the CTC loss on a large training dataset. The second stage involves tuning of the decoder weights using Bayesian optimization through Gaussian Processes in Vizier, using a much smaller and distinct dataset. The training data do not contain frame-aligned labels, so the CTC loss is used for training which treats the alignment between inputs and labels as a hidden variable. The system is trained until the error rate on the evaluation dataset no longer improves for 5 million steps.

**Fig. 4** CER of models trained on the IAM-OnDB dataset with different numbers of LSTM layers and LSTM nodes using raw (left) and curve (right) inputs. Solid lines indicate results without any language models or feature functions in decoding, and dashed lines indicate results with the fully tuned system



The text discusses the use of Bayesian optimization, specifically batched Gaussian process bandits and expected improvement as the acquisition function, for tuning decoder weights in a recognizer training. This process involves starting 7 Vizier studies, each performing 500 individual trials, and selecting the best configuration from these trials. The weights are trained on a subset of languages with sufficient data and then transferred to other languages.

The text also mentions an experimental evaluation on public datasets in both closed and open data scenarios, and on internal datasets for major languages. The IAM-OnDB dataset is used for online handwriting recognition, with a study conducted on the number of layers and nodes per layer for both raw and curve input formats. The study found that using 3 or 5 layers outperforms more shallow networks, and using 64 nodes per layer is sufficient. There was no significant difference in accuracy between the raw and curve representation.

The text discusses improvements in handwriting recognition systems. The new system, compared to the old one, has better input preprocessing, feature extraction, neural network architecture, and training methodology. This has led to state-of-the-art results on the IAM-OnDB dataset. The system also uses the IBM-UB-1 dataset, a less commonly used English-language dataset, for further testing and evaluation.

The text discusses the evaluation of a production system on public datasets, specifically the ICDAR-2013 for Chinese characters and the ICFHR2018 for Vietnamese text. The system was not specifically tuned for these tasks but still achieved competitive results. Differences between this system and others include input preprocessing, feature extraction, and neural network architecture. The text suggests that with specific tuning and training on competition datasets, the system could perform comparably to others.

The text discusses the tuning of neural network parameters on internal datasets, which are more heterogeneous than academic datasets. The best configuration was identified through multiple experiments. The system uses Bézier curve inputs, which are faster to train and evaluate due to shorter sequence lengths.

The system’s performance is evaluated through an ablation study, showing improvements in the network architecture stack, character language model, and other feature functions. The new architecture performs better over almost all languages.

The text also discusses the differences between the IAM-OnDB, IBM-UB-1, and internal datasets. Recognizers were trained on each of the three training sets separately, then evaluated on all three test sets.

The text discusses the performance of a handwriting recognition system on different datasets. The system uses Bézier curve inputs and a neural network architecture with 5 layers of bidirectional LSTMs. It performs well on all datasets, but there are discrepancies due to differences in the datasets’ characteristics. The system improves recognition accuracy by 20-40% while using smaller and faster models. It also outperforms previous results on public datasets like IAM-OnDB. The system is used at Google for 102 languages in 26 scripts.