Pay Attention to What You Read: Non-recurrent Handwritten Text-Line Recognition

Lei Kang[*§], Pau Riba[†§], Marçal Rusiñol[‡§], Alicia Fornés[§], Mauricio Villegas[£]
[*]Computer Science Dept., Shantou University, China
lkang@stu.edu.cn
[†]Helsing AI, Munich, Germany
pau.riba@helsing.ai
[‡]AllRead MLT, Barcelona, Spain
marcal@allread.ai
[§]Computer Vision Center, Barcelona, Spain
{lkang, priba, marcal, afornes}@cvc.uab.es
[£]omni:us, Berlin, Germany
mauricio@omnius.com

Abstract

The advent of recurrent neural networks for handwriting recognition marked an important milestone reaching impressive recognition accuracies despite the great variability that we observe across different writing styles. Sequential architectures are a perfect fit to model text lines, not only because of the inherent temporal aspect of text, but also to learn probability distributions over sequences of characters and words. However, using such recurrent paradigms comes at a cost at training stage, since their sequential pipelines prevent parallelization. In this work, we introduce a novel method that bypasses any recurrence during the training process with the use of transformer models. By using multi-head self-attention layers both at the visual and textual stages, we are able to tackle character recognition as well as

to learn language-related dependencies of the character sequences to be decoded. Our model is unconstrained to any predefined vocabulary, being able to recognize out-of-vocabulary words, *i.e.* words that do not appear in the training vocabulary. We significantly advance over prior art and demonstrate that satisfactory recognition accuracies are yielded even in few-shot learning scenarios.
Keywords: Handwriting Text Recognition, Transformers, Self-Attention, Implicit Language Model

---

1. Introduction

Handwritten Text Recognition (HTR) frameworks aim to provide machines with the ability to read and understand human calligraphy. From the applications perspective, HTR is relevant both to digitize the textual contents from ancient document images in historic archives as well as contemporary administrative documentation such as cheques, forms, etc. Even though research in HTR began in the early sixties [1], it is still considered as an unsolved problem. The main challenge is the huge variability and ambiguity of the strokes composing words encountered across different writers. Fortunately, in most cases, the words to decipher do follow a well defined set of language rules that should be also modelled and taken into account in order to discard gibberish hypotheses and yield higher recognition accuracies. As a result, HTR is often approached by combining technologies from both computer vision and natural language processing communities.

Handwritten text is a sequential signal in nature, which is usually a sequence of characters from left to right in Latin languages. Thus, HTR approaches usually adopted temporal pattern recognition techniques to address it. The early approaches based on Hidden Markov Models (HMM) [2] evolved towards the use of Deep Learning techniques, in which Bidirectional Long Short-Term Memory (BLSTM) networks [3] became the standard solution. Recently, inspired by their success in the applications such as automatic translation or speech-to-text, Sequence-to-Sequence (Seq2Seq) approaches, conformed by encoder-decoder networks led by attention mechanisms have started to be applied for HTR [4]. All the above methods are not only a good fit to process images sequentially, but also have, in principle, the inherent power of language modelling, *i.e.* to learn which character is more probable to be found after another in their respective decoding steps. Nonetheless, this ability of language modelling has proven to be limited, since recognition performances are in most cases still enhanced when using a separate statistical language model as a post-processing step [5].

Despite the fact that attention-based encoder-decoder architectures have started to be used for HTR with impressive results, one major drawback still remains. In all of those cases, such attention mechanisms are still used in conjunction with a recurrent network, either BLSTMs or Gated Recurrent Unit (GRU) networks. The use of such sequential processing deters parallelization at training stage, and severely affects the effectiveness when processing longer sequence lengths by imposing substantial memory limitations.

Motivated by the above observations, Vaswani *et al.* proposed in [6] the seminal work on the Transformer architecture. Transformers rely entirely on attention mechanisms, relinquishing any recurrent designs. Stimulated by such advantage, we propose to address the HTR problem by an architecture inspired on transformers, which dispenses of any recurrent network. By using multi-head self-attention layers both at the visual and textual stages, we aim to tackle both the proper step of character recognition from images, as well as to learn language-related dependencies of the character sequences to be decoded.

The use of transformers in different language and vision applications have shown higher performances than recurrent networks while having the edge over BLSTMs or GRUs by being more parallelizable and thus involving reduced training times. Our method is, to the best of our knowledge, the first non-recurrent approach for HTR. Moreover, the proposed transformer approach is designed to work at character level, instead at the commonly used wordpiece level [7] in translation or speech recognition applications. By using such design we are not restricted to any predefined fixed vocabulary, so we are able to recognize out-of-vocabulary (OOV) words, *i.e.* never seen during training. Competitive state-of-the-art results on the public IAM dataset are reached even when using a small portion of training data.

The main contributions of our work are summarized as follows. ***i***) For the first time, we explore the use of transformers for the HTR task, bypassing any recurrent architecture. We attempt to learn, with a single unified architecture, to recognize character sequences from images as well as to model language, providing context to distinguish between characters or words that might look similar. The proposed architecture works at character level, waiving the use of predefined lexicons. ***ii***) By using a pre-training step using synthetic data, the proposed approach is able to yield competitive results with a limited amount of real annotated training data. ***iii***) Extensive ablation and comparative experiments are conducted in order to validate the effectiveness of our approach. Our proposed HTR system achieves new state-of-the-art performance on the public IAM dataset.

2. Related Work

The recognition of handwritten text has been commonly approached by the use of sequential pattern recognition techniques. Text lines are processed along a temporal sequence by learning models that leverage their sequence of internal states as memory cells, in order to be able to tackle variable length input signals. Whether we analyze the former approaches based on HMMs [2, 8, 9] or the architectures based on deep neural networks such as BLSTMs [3], Multidimensional LSTMs [10, 11] (MDLSTM) or encoder-decoder networks [12, 13, 14, 15, 4], they all follow the same paradigm. Although all those approaches use recurrent architectures to properly conceal and learn serial information, visually, but also from the language modelling perspective, they all suffer of the lack of parallelization during the training stage. Moreover, in order to efficiently train deep learning based approaches, a huge amount of labeled training data is required. Some approaches like [16, 17, 18] alleviate the cost and effort of collecting such amount of real annotated training data by using synthetically generated cursive data with electronic true-type fonts. Which, in turn, having unlimited annotated data for free and training models that are less prone to overfit to a set of specific writing styles, exaggerate even more the computational costs during the training process.

Vaswani *et al.* presented in [6] the Transformer architecture. Their proposal relies entirely on the use of attention mechanisms, avoiding any recurrent steps. Since the original publication, the use of transformers has been popularized in many different computer vision and natural language processing tasks such as automatic translation [19], speech-to-text applications [20] and emotion recognition [21]. Its use has started to eclipse recurrent architectures such as BLSTMs or GRUs for such tasks, both by being more parallelizable, facilitating training, and by having the ability to learn powerful language modelling rules of the symbol sequences to be decoded.

The transformer architecture has been used lately to recognize text in natural scenes [22]. In such works, the original transformers architecture, often applied to one-dimensional signals (*i.e.* text, speech, etc.), has been adapted to tackle two-dimensional input images. Image features are extracted by the use of CNNs [23], two-dimensional positional encodings [24, 25] or additional segmentation modules [26] help the system locate textual information amidst background clutter. However, all such works present some limitations when dealing with handwritten text lines. On the one hand, all such architectures work with fixed image size whereas for handwriting recognition we have to face variable length inputs. On the other hand, they work at individual word level, whereas in handwriting recognition we have to face much longer sequences. Finally, despite also having its own great variability, scene text is often much legible than cursive handwriting, since in most of the cases words are formed by individual block letters, which, in turn, are easier to be mimicked by synthesized data. Recently, there are some contemporary works dealing with handwriting recognition problems. Zhao *et al.* [27] proposed a transformer-based mathematical expression recognizer. In addition, Mostafa *et al.* [28] and Tsochatzidis *et al.* [29] introduced transformer-based handwriting recognizers for Arabic and Greek languages, respectively.

Summarizing, state-of-the-art handwriting recognition based on deep recurrent networks have started to reach decent recognition results, but are too computationally demanding at training stage. Moreover, albeit they shall have the ability to model language-specific dependencies, they usually fall short of inferring adequate language models and need further post-processing steps. In this paper we propose, for the first time, the use of transformers for the HTR task, bypassing any recurrent architecture. A single unified architecture, both recognizes long character sequences from images as well as models language at character level, waiving the use of predefined lexicons.

3. Proposed Method

3.1. Problem Formulation

Let $\{\mathcal{X}, \mathcal{Y}\}$ be a handwritten text dataset, containing images $\mathcal{X}$ of handwritten text lines, and their corresponding transcription strings $\mathcal{Y}$. The alphabet defining all the possible characters of $\mathcal{Y}$ (letters, digits, punctuation signs, white spaces, etc.), is denoted as $\mathcal{A}$. Given pairs of images $x_i \in \mathcal{X}$ and their corresponding strings $y_i \in \mathcal{Y}$, the proposed recognizer has the ability to combine both sources of information, learning both to interpret visual information and to model language-specific rules.

The proposed method's architecture is shown in Figure 1. It consists of two main parts. On the one hand a visual feature encoder aimed at extracting the relevant features from text-line images and at focusing its attention at the different character locations. Subsequently, the text transcriber is devoted

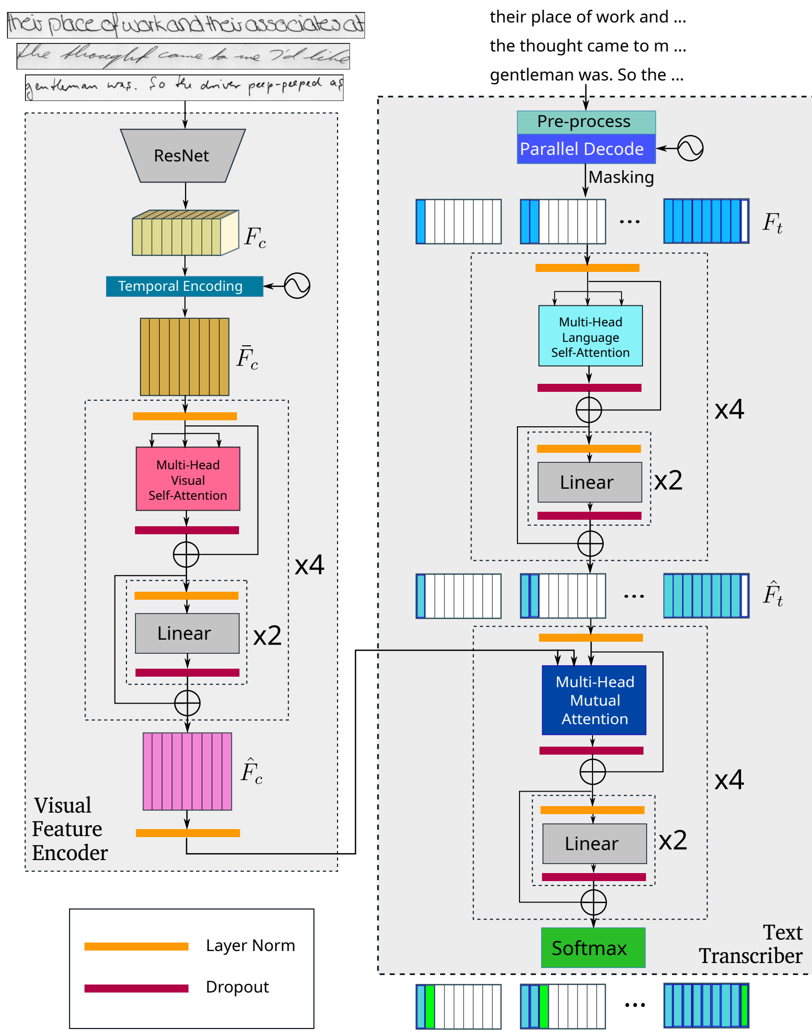


Figure 1: Overview of the architecture of the proposed method, which consists of Visual Feature Encoder (the left part) and Text Transcriber (the right part). The Visual Feature Encoder includes CNN nets (ResNet), temporal encoding and visual self-attention modules, while the Text Transcriber includes text encoding (pre-process and parallel decode), language self-attention modules and mutual-attention modules. Note that the mutual-attention modules bridge the information of both visual features and textual features.

to output the decoded characters by mutually attending both at the visual features as well as the language-related features. The whole system is trained in an end-to-end fashion, learning both to decipher handwritten images as well as modelling language.

3.2. Visual Feature Encoder

The role of the visual feature encoder is to extract high-level feature representations from an input handwritten image $x \in \mathcal{X}$. It will encode both visual content as well as sequential order information. This module is composed by the following three parts.

3.2.1. CNN Feature Encoder

Input images x of handwritten text-lines, which might have arbitrary lengths, are first processed by a Convolutional Neural Network. We obtain an intermediate visual feature representation F_c of size f. We use the ResNet50 [30] as our backbone convolutional architecture. Such visual feature representation has a contextualized global view of the whole input image while remaining compact.

3.2.2. Temporal Encoding

Handwritten text images are sequential signals in nature, to be read in order from left to right in Latin scripts. The temporal encoding steps are aimed to leverage and encode such important information bypassing any recurrency.

In a first step, the three-dimensional feature F_c is reshaped into a two-dimensional feature by keeping its width, *i.e.* obtaining a feature shape $(f \times h, w)$. This feature map is later fed into a fully connected layer in order to reduce $f \times h$ back to f. The obtained feature $F_c^{'}$, with the shape of (f, w), can be seen as a w-length sequence of visual vectors.

However, we desire that the same character appearing at different positions of the image has different feature representations, so that the attention mechanisms are effectively and unequivocally guided. That is, we want that the visual vectors $F_c^{'}$ loose their horizontal shift invariance. Following the proposal from Vaswani *et al.* [6], a one-dimensional positional encoding using sine and cosine functions is applied.

$$\begin{aligned} TE(pos, 2i) &= \sin\left(\frac{pos}{10000^{2i/f}}\right) \\ TE(pos, 2i+1) &= \cos\left(\frac{pos}{10000^{2i/f}}\right), \end{aligned} \tag{1}$$

where $pos \in \{0, 1, 2, \ldots, w-1\}$ and $i \in \{0, 1, 2, \ldots, f-1\}$.

$F_c^{'}$ and TE, sharing the same shape, are added along the width axis. A final fully connected layer produces an abscissa-sensitive visual feature $\bar{F}_c$ with shape (f, w).

3.2.3. Visual Self-Attention Module

To further distill the visual features, self-attention modules are applied four times upon $\bar{F}_c$. The multi-head attention mechanism from [6] is applied using eight heads. This attention module takes three inputs, namely the query Q_c, key K_c and value V_c, where $Q_c = K_c = V_c = \bar{F}_c$. The correlation information is obtained by:

$$\hat{v}_c^i = \text{Softmax}\left(\frac{q_c^i K_c}{\sqrt{f}}\right) V_c, \tag{2}$$

where $q_c^i \in Q_c$ and $i \in \{0, 1, \ldots, w-1\}$. The final high-level visual feature is $\hat{F}_c = \{\hat{v}_c^0, \hat{v}_c^1, \ldots, \hat{v}_c^{w-1}\}$.

3.3. Text Transcriber

The text transcriber is the second part of the proposed method. It is in charge of outputting the decoded characters, attending to both the visual features as well as the language-specific knowledge learnt form the textual features. It is worth to note that unlike translation of speech-to-text transformer architectures, our text transcriber works at character level instead of word-level. It will thus learn n-gram like knowledge from the transcriptions, *i.e.* predicting the next most probable character after a sequence of decoded characters. The text transcriber consists of three steps, the text encoding, the language self-attention step and the mutual-attention module.

3.3.1. Text Encoding

Besides the different characters considered in alphabet $\mathcal{A}$, we require some symbols without textual content for the correct processing of the text-line string. Special character $\langle S \rangle$ denotes the start of the sequence, $\langle E \rangle$ the end of the sequence, and $\langle P \rangle$ is used for padding. The transcriptions $y \in \mathcal{Y}$ are extended to a maximum length of N characters in the prediction.

A character-level embedding is performed by means of a fully-connected layer that maps each character from the input string to an f-dimensional vector. The same temporal encoding introduced in eq. 1 is used here to obtain

$$F_t = \text{Embedding}\,(y) + TE, \tag{3}$$

where F_t has the shape of (f, N).

In the decoding step of Seq2Seq approaches [13, 4], every decoded character is iteratively fed again to the decoder, to predict the next character, thus inhibiting its parallelization. Contrary, in the transformer paradigm, all possible decoding steps are fed concurrently at once with a masking operation [6]. To decode the j-th character from y, all characters at positions greater than j are masked so that the decoding only depends on predictions produced prior to j. Such a parallel processing of what used to be different time steps in recurrent approaches drastically reduces training time.

3.3.2. Language Self-attention Module

This module follows the same architecture as in Section 3.2.3 and aims to further distill the text information and learn language-specific properties. $\hat{F}_t$ is obtained after the self-attention module implicitly delivers n-gram-like features, since to decode the j-th character from y all character features prior to j are visible.

3.3.3. Mutual-attention Module

A final mutual self-attention step is devoted to align and combine the learned features form the images as well as from the text strings. We follow again the same architecture from Section 3.2.3, but now the query Q_t comes from the textual representation $\hat{F}_t$ while the key K_c and value V_c are fed with the visual representations $\hat{F}_c$

$$\hat{v}_{ct}^{i} = \text{Softmax}\left(\frac{q_t^j K_c}{\sqrt{f}}\right) V_c, \tag{4}$$

where $q_t^j \in Q_t$ and $j \in \{0, 1, \ldots, N-1\}$. The final combined representation is $\hat{F}_{ct} = \{\hat{v}_{ct}^0, \hat{v}_{ct}^1, \ldots, \hat{v}_{ct}^{N-1}\}$.

The output $\hat{F}_{ct}$ is expected to be aligned with the transcription Y. Thus, by feeding the $\hat{F}_{ct}$ into a linear module followed by a softmax activation function, the final prediction is obtained.

3.4. Inference on Test Data

When evaluating on test data, the transcriptions $\mathcal{Y}$ are not available. The text pipeline is initialized by feeding the start indicator $\langle S \rangle$ and it predicts the first character by attending the related visual part on the input handwritten text image. With the strategy of greedy decoding, this first predicted character is fed back to the system, which outputs the second predicted character. This inference process is repeated in a loop until the end of sequence symbol $\langle E \rangle$ is produced or when the maximum output length N is reached.

4. Experimental Evaluation

4.1. Dataset and Performance Measures

We conduct our experiments on the popular IAM handwritten dataset [31], composed of modern handwritten English texts. We use the RWTH partition, which consists of 6482, 976 and 2914 lines for training, validation and test, respectively. The size of alphabet $|\mathcal{A}|$ is 83, including special symbols, and the maximum length of the output character sequence is set to 89. All the handwritten text images are resized to the same height of 64 pixels while keeping the aspect ratio, which means that the text line images have variable length. To pack images into mini-batches, we pad all the images to the width of 2227 pixels with blank pixels.

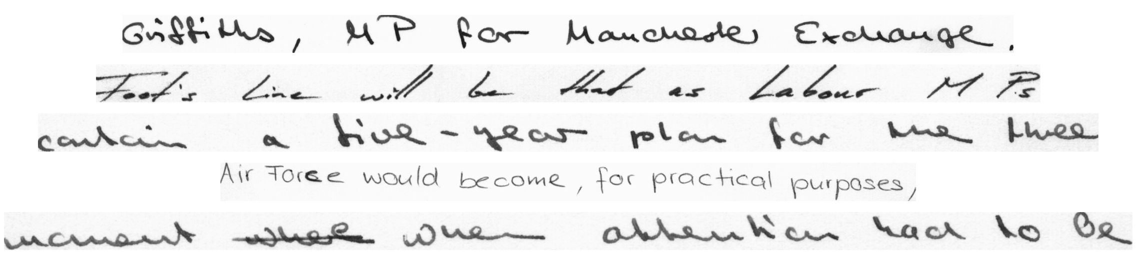
Character Error Rate (CER) and *Word Error Rate* (WER) [32] are used for the performance measures. The CER is computed as the Levenshtein distance which is the sum of the character substitutions (S_c), insertions (I_c) and deletions (D_c) that are needed to transform one string into the other, divided by the total number of characters in the groundtruth (N_c). Formally,

$$CER = \frac{S_c + I_c + D_c}{N_c} \tag{5}$$

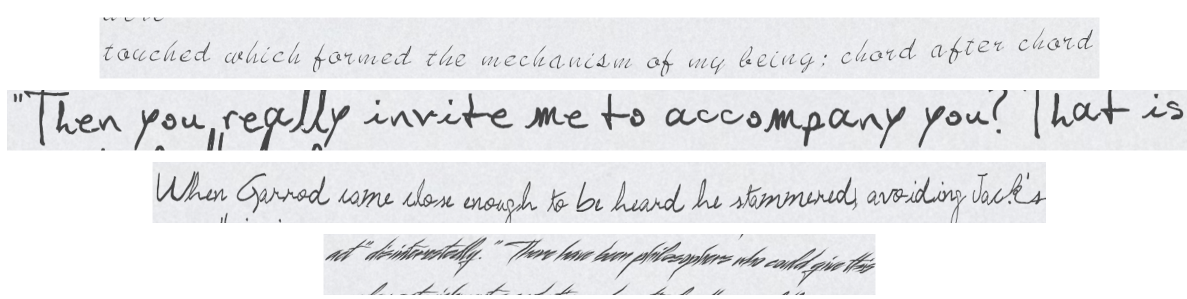
Similarly, the WER is computed as the sum of the word substitutions (S_w), insertions (I_w) and deletions (D_w) that are required to transform one string into the other, divided by the total number of words in the groundtruth (N_w). Formally,

$$WER = \frac{S_w + I_w + D_w}{N_w} \tag{6}$$

4.2. Implementation Details



a) Real data from IAM dataset



b) Synthetically rendered text-lines with truetype fonts

Figure 2: Examples of real and synthetic training handwritten text-line images.

4.2.1. Hyper-Parameters of Networks

In the proposed architecture, the feature size f is 1024. We use four blocks of visual and language self-attention modules, and each self-attention module has eight heads. We use 0.1 dropout setting for every dropout layer. In the text transcriber, all the transcriptions include the extended special symbols $\langle S \rangle$ and $\langle E \rangle$ at the beginning and at the end, respectively. Then, they are padded to 89 length with a special symbol $\langle P \rangle$ to the right, which is the maximum number of characters in the prediction N. The output size of the softmax is 83, which is the size of the alphabet $\mathcal{A}$, including upper/lower cased letters, punctuation marks, blank space and special symbols.

4.2.2. Optimization Strategy

We adopt label smoothing mechanism [33] to prevent the system from making over-confident predictions, which is also a way of regularization. As the ground-truth are one-hot vectors with binary values, label smoothing is done by replacing the 0 and 1 with $\frac{\varepsilon}{|\mathcal{A}|}$ and $1-\frac{|\mathcal{A}|-1}{|\mathcal{A}|}\varepsilon$, where ε is set to 0.4 in this paper. Based on the batch size of 48, we utilize Adam optimizer [34] for the training process with an initial learning rate of $2 \cdot 10^{-4}$, while reducing the learning rate by half every 20 epochs. The implementation of this system is based on PyTorch [35] and performed on a NVIDIA Cluster. The code will be publicly available.

4.3. Pre-training with Synthetic Data

Deep learning based methods need a large amount of labelled training data to obtain a well generalized model. Thus, synthetic data is widely used to compensate the scarcity of training data in the public datasets. There are some popular synthetically generated handwriting datasets available [36, 37], but they are at word level. For this reason we have created our own synthetic data at line level for pre-training. First, we collect a text corpus in English from online e-books and end up with over $130,000$ lines of text. Second, we select 387 freely available electronic cursive fonts and use them to randomly render text lines from the first step. Finally, by applying a set of random augmentation techniques (blurring/sharpening, elastic transforming, shearing, rotating, translating, scaling, gamma correcting and blending with synthetic background textures), we obtain a synthetic dataset with $138,000$ lines. The comparison between the synthetic data and the real data is shown in Figure 2.

4.4. Ablation Studies

In the ablation studies, all the experiments are trained with the IAM training set at line-level, and then early-stopped by the CER of the validation set, which is also utilized as an indicator to choose the hyper-parameters as shown in Table 1 2 3.

4.4.1. Architecture of CNN Feature Encoder

We have explored different popular Convolutional Neural Networks for the feature encoder detailed in Section 3.2.1. The best results were obtained with ResNet models. We provide a modified version of ResNet architecture, which has a stride value of 1 instead of 2 from the original ResNet at the last convolutional layer. Thus, the output features become 2-times bigger than that of the original ResNet method. From Table 1, the best performance is achieved with a modified version of ResNet50.

Table 1: Ablation study on Convolutional architectures. * indicates modified architectures. Results are shown on IAM validation set.

CNN	CER (%)	WER (%)
ResNet34	6.33	22.63
ResNet34*	5.44	20.13
ResNet50	5.49	20.93
ResNet50*	**4.86**	**18.65**

4.4.2. Function of Temporal Encoding

In both the visual feature encoder and the text transcriber, we have used temporal encoding in order to enforce an order information to both visual and textual features. Nonetheless we want to analyze its impact. In Table 2, it is clear that using temporal encoding at text level boosts the performance drastically from 7.72% to 4.86%, and from 6.33% to 5.52%, depending on whether we use it at image level or not. The best performance is reached when using the temporal encoding step both for image and text representations.

Table 2: Ablation study on the use of temporal encoding in image and text levels. Results are shown on IAM validation set.

Image level	Text level	CER (%)	WER (%)
–	–	6.33	21.64
✓	–	7.72	24.70
–	✓	5.52	20.72
✓	✓	**4.86**	**18.65**

4.4.3. Role of Self-Attention Modules

Self-attention modules have been applied in both image and text levels. In Table 3 we analyze their effect in our system. We observe that the language self-attention module does play an important role to improve the performance from 7.71% to 4.86%, and from 7.78% to 4.89%, with and without the visual self-attention module, respectively. Our intuition is that the language self-attention module actually does learn language-modelling information. This implicitly learned language model is at character level and takes advantage of the contextual information of the whole text-line, which not only boosts the recognition performance but also keep the capability to predict out-of-vocabulary (OOV) words (showcases in Table 5). However, the visual self-attention module barely improves the performance comparing the pairs of first two rows or the last two rows in Table 3. The intuition is that the former CNN module has already extracted good visual features for the handwritten input, so there is very little opportunity for the visual self-attention module to further boost the performance.

Furthermore, the "Time" column in Table 3 illustrates how much extra time costs when equipping either visual or textual self-attention modules. We see that the visual self-attention module costs extra 21.61% time for the system training, while the language self-attention module only spends 8.85% more time to do the same job. Considering the little improvement and huge training time cost of the visual self-attention module, it is suggested to use only the language self-attention module in real-world use cases as a trade-off. But in this paper, we would like to exploit all the modules to achieve the best performance. Thus, both of the visual and textual self-attention modules are equipped in the system for the following experiments.

The image widths and text lengths on IAM text-line training set have been analyzed statistically in Table 4. Based on the median image width of 1751 pixels and median text length of 43 characters, we have monitored how the training speed changes in terms of different image widths and text lengths as shown in Figure 3. Usually, a long handwritten image always roughly refers to a long text groundtruth, we put the two scales of both image pixels and text characters together. Note that, the image widths and the text lengths are not perfectly aligned in the figure. From Figure 3, we appreciate that squeezing input images with a shorter width always lead to a faster training speed.

Table 3: Ablation study on visual and language self-attention modules, where results are shown on IAM validation set. The "Time" column represents the training time in percentage based on the baseline (the first row) without neither visual nor textual self-attention modules.

Image level	Text level	CER (%)	WER (%)	Time (%)
–	–	7.78	29.78	–
✓	–	7.71	28.50	+21.61
–	✓	4.89	**18.57**	+8.85
✓	✓	**4.86**	18.65	+28.54

Table 4: The statistic of the image widths and text lengths on IAM text-line training set.

	Min	Max	Median
Image width (pixels)	104	2260	1751
Text length (num. of chars)	1	80	43

We showcase in Figure 5 some qualitative results on text-line recognition, where we visualize the attention maps as well. The attention maps are obtained by averaging the mini attention maps across different layers and

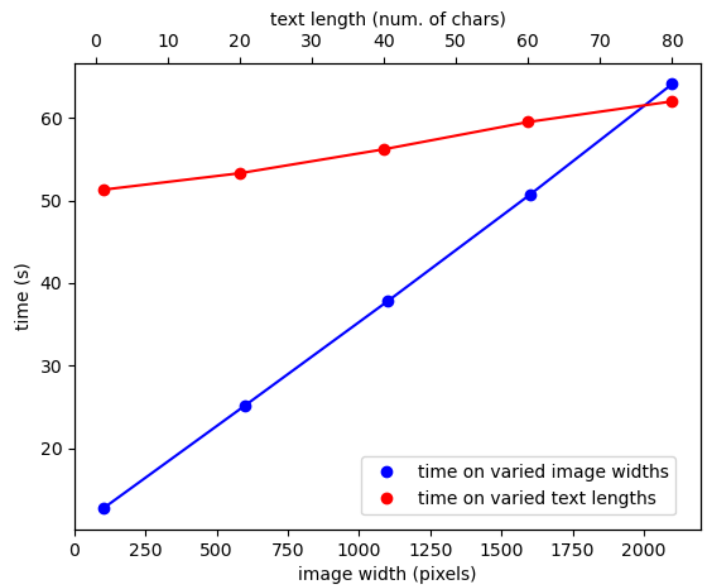


Figure 3: Training speed of different image widths and text lengths. The text length has been set to the median number of 43 characters for the experiments on image widths (blue curve), while the image width has been kept as the median width of 1751 pixels for the experiments on text lengths (red curve).

different heads. Those visualizations prove the successful alignment between decoded characters and images.

4.5. Detailed Comparison with Seq2Seq Model

In order to provide a fair comparison between the proposed architecture and recurrent-based solutions, we re-implemented a state-of-the-art recurrent handwriting recognition pipeline, and we train and evaluate those under the exact same circumstances. Following the methods proposed in [13, 4] we built a sequence-to-sequence recognizer composed of an encoder, a decoder and an attention mechanism. The encoder consists of a VGG19-BN [38] and a two-layer Bidirectional Gated Recurrent Units (BGRU) with feature size of 512.

Table 5: Examples of handwritten text-line recognition results with OOV words on IAM test set, where the OOV words are highlighted.

	been able to explain the textual movements by
Result	been able to explain the textual movements by
	This indicated how ephemeral in the Church had
Result	This indicated how ephemeral in the Church had
	Samuel in support of his thesis and quoted
Result	Samuel in support of his thesis and quoted
	of winds is needed for this effect of
Result	of winds is needed for this effect of
	fascinated by the way he looked when you
Result	fascinated by the way he looked when you

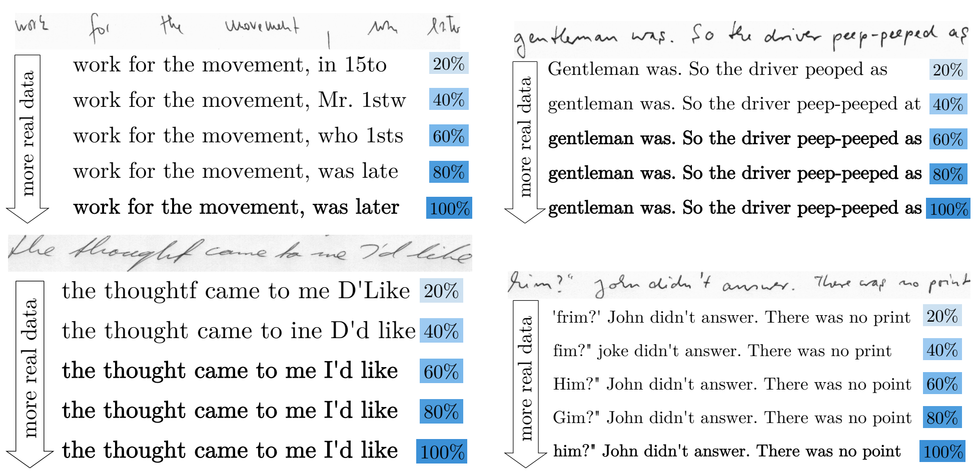


Figure 4: Performance of the transformer-based decodings for different amounts of real training data.

The decoder is a two-layer one directional GRU with feature size of 512, and we power the architecture with a location-based attention mechanism [39].

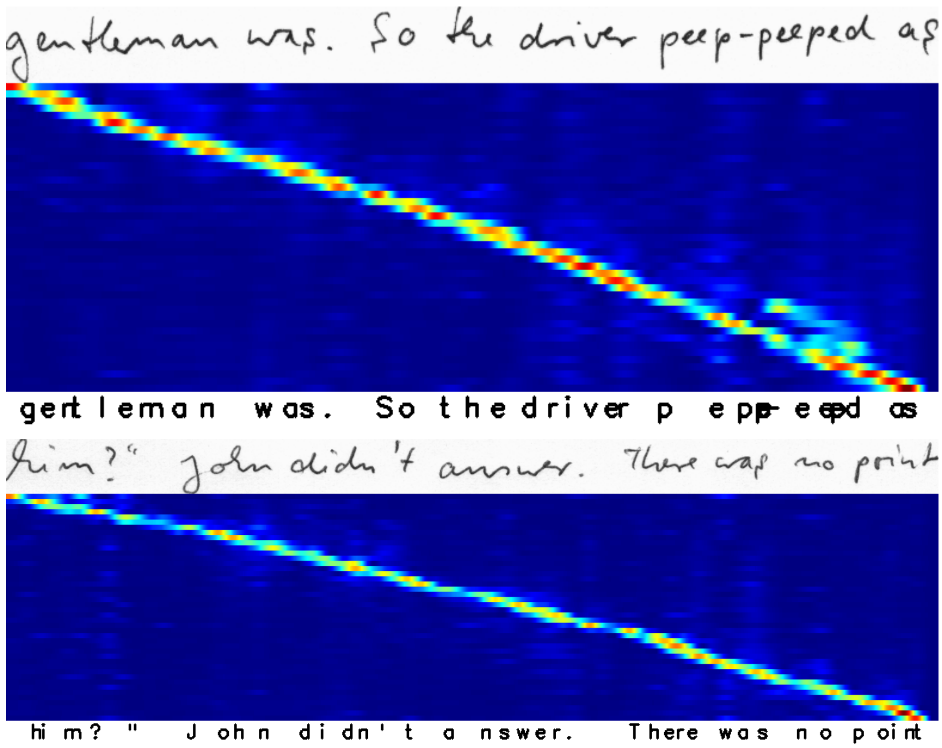


Figure 5: Qualitative results on text-line recognition and visualization of attention maps that coarsely align transcriptions and corresponding image characters.

All the dropout layers are set to 0.5. Label smoothing technique is also used during the training process. The maximum number of predicted characters is also set to 89. All the hyper-parameters in this sequence-to-sequence model are also exhaustively validated by ablation studies with validation data.

We first provide in Table 6, the CER and WER rates on the IAM test set both when training the networks from scratch and just using the IAM training data, and when pre-training the networks with synthetic data for a later fine-tuning step on real data. We also provide the model size and the time taken per epoch during training. From Table 6, we can see that the proposed transformer-based method achieves a better recognition performance than the sequence-to-sequence model, even with a heavy network architecture. We have tried to make the sequence-to-sequence model comparable with the transformer method, so that it ends up with 37M parameters. But normally the sequence-to-sequence model has much less parameters such as 725k parameters in [40]. Thus, some sequence-to-sequence models might advance in the training speed. But the heavier transformer architecture does further boost the HTR performance. We also observe that both models benefit from the use of synthetic pre-training, improving the final error rates quite noticeably for the transformers model, although such boost is not so drastic for the sequence-to-sequence approach.

Table 6: Comparison between Recurrent and Transformers.

Method	CER (%)	WER (%)	Time(s)	Param(M)
Seq2Seq	11.91	37.39	338.7	37
+ Synth	10.64	33.64	338.7	37
Ours	7.62	24.54	202.5	100
+ Synth	**4.67**	**15.45**	**202.5**	**100**

4.6. Few-shot Training

Due to the scarcity and the cost of producing large volumes of real annotated data, we provide an analysis on the performance of the proposed approach when dealing with a few-shot training setup, when compared again with the sequence-to-sequence approach. To mimic a real scenario in which only a small portion of real data is available, we randomly selected 20%, 40%, 60% and 80% of the IAM training set.

Table 7: Fine-tuning with different portions of real data (line-level test set with greedy decoding).

	20%		**40%**		**60%**		**80%**		**100%**	
	CER	WER	CER	WER	CER	WER	CER	WER	CER	WER
Seq2Seq	20.61	56.50	16.15	46.97	15.61	46.01	12.18	38.11	11.91	37.39
+ Synth	18.64	51.77	13.01	39.72	13.00	39.34	12.15	37.43	10.64	33.64
Ours	73.81	132.74	17.34	42.57	10.14	30.34	10.11	29.90	7.62	24.54
+ Synth	**6.51**	**20.53**	**6.20**	**19.69**	**5.54**	**17.71**	**4.90**	**16.44**	**4.67**	**15.45**

As shown in Table 7, both sequence-to-sequence and transformer-based approaches follow the same trend. The more real training data is available, the better the performance is. Overall, the transformer-based method performs better than the sequence-to-sequence, except for the extreme case of just having a 20% of real annotated training data available. The transformer approach, being a much larger model, struggles at such drastic data scarcity conditions. However, when considering the models that have been pre-trained with synthetic data, the transformer-based approach excels in few-shot setting conditions. We provide in Figure 4 some qualitative examples of the transcriptions provided by different models trained with reduced training sets. All of the models were pre-trained with synthetic data.

4.7. Language Modelling Abilities

We try to explore how would an external language model help the recognition process and how good is the inherent language modelling ability as shown in Table 8. Our baseline refers to our proposed method with training from scratch. In order to propose a fairly comparable result, the text corpus of IAM training set is the only data to be seen by the baseline model. Learning rate restart strategy [41] is applied to the baseline model, so that the inherent language model could be properly trained. The external language model is applied on two folds: first, a word-level unigram is trained on the text corpus of either IAM training set or WikiText-103 [42] training set; then, a fine-grained character-level bigram/trigram is trained on IAM training set. The word-level unigram is utilized to detect the error words and find the top 10 candidate words by edit-distance, while the character-level bigram or trigram is for ranking the candidate words by calculating the perplexity of each word. From Table 8, the WikiText trained language model advances over the IAM trained one, because the large WikiText-103 dataset covers more available words on the IAM test set. According to the WER, the two-step language model strategy do improve the recognition performance when training with a large text corpus. But the best CER is still achieved by our baseline model. Due to the complexity of a Transformer architecture that involves the context information into different stages of multi-head attention modules, it is difficult to evaluate the inherent language model ability. Thus, according to Table 8, as the external language model does not improve the recognition performance in a large margin, we would infer that the proposed method does incorporate a language-specific contextual information within the language self-attention module.

During the evaluation process, we also try to decode with beam search [6], and the comparison of both greedy decoding and beam search decoding with a beam size of 4 is shown in Table 9. From this table, we observe that the beam search strategy contribute quite little to the recognition performance. Considering the extra calculation cost, we adopt greedy decoding on all our

Table 8: Effect of using a post-processing language model. Word-level unigram is trained on the training set of IAM and WikiText-103, namely LM(IAM) and LM(WikiText), correspondingly. Character-level bigram and trigram, trained on IAM training set, are utilized to further rank and select the candidate words proposed by the word level unigram.

Method	CER (%)	WER (%)	PPL_{word}	PPL_{char}
Our Baseline	**7.97**	19.61	–	–
+LM(IAM)	14.70	26.01	453.62	–
+bigram	16.17	24.86	453.62	15.22
+trigram	15.31	24.70	453.62	20.12
+LM(WikiText)	9.24	17.27	10938.61	–
+bigram	8.66	15.49	10938.61	15.22
+trigram	8.59	**15.39**	10938.61	20.12

experiments.

Table 9: Comparison of the decoding strategies: greedy and beam search, where the beam size is 4.

Method	CER (%)	WER (%)
Greedy decoding	7.97	19.61
Beam search decoding	7.95	19.60

4.8. Error Analysis

We have applied an error analysis for the impact of greedy decoding as shown in Table 11. First, the text lines are split into 8 equally sized parts; then, three different probability analysis are computed. In the 1st row Table 11, we have witnessed that there are normally more errors at the beginning and the end. From the second row, it is clear that if there is an error in $Part_0$, then it would be very likely to occur an error in $Part_1$.

Table 10: Comparison with the State-Of-The-Art approaches on the test set of IAM line level dataset. "Ω" column represents the amount of lexicon that is utilized to train an external language model, where "–" means without applying external language modelling.

System	Method	Ω (k)	CER (%)	WER (%)
HMM/ANN 2008 - now	Almazán *et al.* [43]	–	11.27	20.01
	España *et al.* [8]	–	9.80	22.40
	Dreuw *et al.* [44]	50	12.40	32.90
	Bertolami *et al.* [45]	20	–	32.83
	Dreuw *et al.* [46]	50	10.30	29.20
	Zamora *et al.* [47]	103	7.60	16.10
	Pastor *et al.* [48]	103	7.50	19.00
	España *et al.* [8]	5	6.90	15.50
	Kozielski *et al.* [49]	50	5.10	13.30
	Doetsch *et al.* [50]	50	4.70	12.20
RNN+CTC 2008 - now	Chen *et al.* [51]	–	11.15	34.55
	Pham *et al.* [52]	–	10.80	35.10
	Krishnan*et al.* [53]	–	9.78	32.89
	Wigington *et al.* [54]	–	6.40	23.20
	Puigcerver [11]	–	5.80	18.40
	Dutta *et al.* [55]	–	5.70	17.82
	Graves *et al.* [3]	20	18.20	25.90
	Pham *et al.* [52]	50	5.10	13.60
	Puigcerver [11]	50	4.40	12.20
	Bluche *et al.* [40]	**50**	**3.20**	**10.50**
Seq2Seq 2016 - now	Chowdhury [15]	–	8.10	16.70
	Bluche [12]	–	7.90	24.60
	Bluche [12]	50	5.50	16.40
Transf.	**Ours**	–	**4.67**	**15.45**

Finally, the 3rd row indicates that the multiple errors tend to be grouped together. Thus, we could conclude that the errors early in the sequence do not necessarily cause a tail of further errors. The recognizer has the ability to recover the text sequence from early errors with greedy decoding strategy.

Table 11: Error analysis for the impact of greedy decoding, where $i \in \{0, 1, ..., 7\}$ corresponds to the specific part of the text sequence.

	$Part_0$	$Part_1$	$Part_2$	$Part_3$	$Part_4$	$Part_5$	$Part_6$	$Part_7$
$P(err_i)$	0.24	0.19	0.22	0.21	0.22	0.22	0.24	0.26
$P(err_i \mid err_0)$	–	1	0.35	0.32	0.32	0.33	0.32	0.35
$P(err_{i..7} \mid err_{0..i})$	–	0.83	0.8	0.75	0.7	0.61	0.51	0.34

4.9. Comparison with the State-Of-The-Art

Finally, we provide in Table 10 and extensive performance comparison with the state of the art. Different approaches have been grouped into a taxonomy depending on whether they are based on HMMs or early neural network architectures, whether they use recurrent neural networks (usually different flavours of LSTMs) with a Connectionist Temporal Classification (CTC) loss function, or if they are based on encoder-decoder sequence-to-sequence architectures. Within each group, we differentiate results depending on whether applying an external language model or not. In the "Ω" column, "$-$" indicates that the method is an end-to-end recognizer without external language modelling, while the exact number ("k" represents "thousand") is the amount of lexicon that is utilized to train an external language model. Bluche *et al.* [40] achieves the best result among the methods with an external language model, while our proposed method obtains the best result among the methods without external language modelling, while still competing with the state of the arts.

5. Conclusion and Future Work

In this paper, we have proposed a novel non-recurrent and open-vocabulary method for handwritten text-line recognition. As far as we know, it is the first approach that adopts the transformer networks for the HTR task. We have performed a detailed analysis and evaluation on each module, demonstrating the suitability of the proposed approach. Indeed, the presented results prove that our method not only achieves the state-of-the-art performance, but also has the capability to deal with few-shot training scenarios, which further extends its applicability to real industrial use cases. Finally, since the proposed approach is designed to work at character level, we are not constrained to any closed-vocabulary setting, and transformers shine at combining visual and language-specific learned knowledge.

We will push forward two research lines for the future. On the one hand, we will carry on further research on the integration of an external language model. As the state-of-the-art language models are tremendously researched in the field of natural language processing, we would like to stand on the shoulders of these modern studies and propose a better solution for the language model integration. On the other hand, as the prosperous development of Transformer-based methods, we are aware of many novel variations of Transformer, such as Conformer [56], which adapt a Macaron-style block with a convolution module and a pair of feed-forward modules to be surrounded. We think it is worth to study these advanced Transformer variations and further apply them into our specific tasks for performance improvement, which would be especially important for industrial use cases. Furthermore, As the transformer based methods have heavy network architectures, they might advance on bigger and more complex tasks. So we would like to explore on real use cases with more authors and different languages, such as MAURDOR [57]. Thus, we will further tune our model for the varied realistic challenges.

Acknowledgments

This work has been partially supported by the grant 140/09421059 from Shantou University, the Spanish project RTI2018-095645-B-C21, the grant 2016-DI-087 from the Secretaria d'Universitats i Recerca del Departament d'Economia i Coneixement de la Generalitat de Catalunya, the grant FPU15/06264 from the Spanish Ministerio de Educación, Cultura y Deporte, the Ramon y Cajal Fellowship RYC-2014-16831 and the CERCA Program/ Generalitat de Catalunya. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] P. Mermelstein, M. Eyden, A system for automatic recognition of handwritten words, in: Proceedings of the Fall Joint Computer Conference, 1964.

[2] A.-L. Bianne-Bernard, F. Menasri, R. A.-H. Mohamad, C. Mokbel, C. Kermorvant, L. Likforman-Sulem, Dynamic and contextual information in HMM modeling for handwritten word recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (10) (2011) 2066–2080.

[3] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (5) (2008) 855–868.

[4] J. Michael, R. Labahn, T. Grüning, J. Zöllner, Evaluating sequence-to-sequence models for handwritten text recognition, in: Proceedings of the International Conference on Document Analysis and Recognition, 2019.

[5] C. Tensmeyer, C. Wigington, B. Davis, S. Stewart, T. Martinez, W. Barrett, Language model supervision for handwriting recognition model adaptation, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2018.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the Neural Information Processing Systems Conference, 2017.

[7] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., Google's neural machine translation system: Bridging the gap between human and machine translation, arXiv preprint arXiv:1609.08144 (2016).

[8] S. España-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, F. Zamora-Martínez, Improving offline handwritten text recognition with hybrid HMM/ANN models, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (4) (2010) 767–779.

[9] A. Giménez, I. Khoury, J. Andrés-Ferrer, A. Juan, Handwriting word recognition using windowed Bernoulli HMMs, Pattern Recognition Letters 35 (2014) 149–156.

[10] A. Graves, J. Schmidhuber, Offline handwriting recognition with multidimensional recurrent neural networks, in: Proceedings of the Neural Information Processing Systems Conference, 2009.

[11] J. Puigcerver, Are multidimensional recurrent layers really necessary for handwritten text recognition?, in: Proceedings of the International Conference on Document Analysis and Recognition, 2017.

[12] T. Bluche, Joint line segmentation and transcription for end-to-end handwritten paragraph recognition, in: Proceedings of the Neural Information Processing Systems Conference, 2016.

[13] L. Kang, J. I. Toledo, P. Riba, M. Villegas, A. Fornés, M. Rusiñol, Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition, in: Proceedings of the German Conference on Pattern Recognition, 2018.

[14] J. Sueiras, V. Ruiz, A. Sanchez, J. F. Velez, Offline continuous handwriting recognition using sequence to sequence neural networks, Neurocomputing 289 (2018) 119–128.

[15] A. Chowdhury, L. Vig, An efficient end-to-end neural model for handwritten text recognition, in: Proceedings of the British Machine Vision Conference, 2018.

[16] A. K. Bhunia, A. Das, A. K. Bhunia, P. S. R. Kishore, P. P. Roy, Handwriting recognition in low-resource scripts using adversarial learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[17] N. Gurjar, S. Sudholt, G. A. Fink, Learning deep representations for word spotting under weak supervision, in: Proceedings of the IAPR International Workshop on Document Analysis Systems, 2018.

[18] P. Krishnan, C. Jawahar, HWNet v2: An efficient word image representation for handwritten documents, International Journal on Document Analysis and Recognition 22 (4) (2019) 387–405.

[19] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[20] L. Dong, S. Xu, B. Xu, Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2018.

[21] G. Tu, J. Wen, C. Liu, D. Jiang, E. Cambria, Context-and sentiment-aware networks for emotion recognition in conversation, IEEE Transactions on Artificial Intelligence (2022).

[22] N. Lu, W. Yu, X. Qi, Y. Chen, P. Gong, R. Xiao, MASTER: Multi-aspect non-local network for scene text recognition, Pattern Recognition 117 (2021) 107980.

[23] F. Sheng, Z. Chen, B. Xu, NRTR: A no-recurrence sequence-to-sequence model for scene text recognition, in: Proceedings of the International Conference on Document Analysis and Recognition, 2019.

[24] J. Lee, S. Park, J. Baek, S. J. Oh, S. Kim, H. Lee, On recognizing texts of arbitrary shapes with 2D self-attention, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020.

[25] M. Bleeker, M. de Rijke, Bidirectional scene text recognition with a single decoder, ECAI: Frontiers in Artificial Intelligence and Applications 325 (2020) 2664–2671.

[26] C. Bartz, J. Bethge, H. Yang, C. Meinel, KISS: Keeping it simple for scene text recognition, arXiv preprint arXiv:1911.08400 (2019).

[27] W. Zhao, L. Gao, Z. Yan, S. Peng, L. Du, Z. Zhang, Handwritten mathematical expression recognition with bidirectionally trained transformer, in: Proceedings of the International Conference on Document Analysis and Recognition, 2021.

[28] A. Mostafa, O. Mohamed, A. Ashraf, A. Elbehery, S. Jamal, G. Khoriba, A. S. Ghoneim, Ocformer: A transformer-based model for arabic handwritten text recognition, in: Proceedings of the International Mobile, Intelligent, and Ubiquitous Computing Conference, 2021.

[29] L. Tsochatzidis, S. Symeonidis, A. Papazoglou, I. Pratikakis, HTR for greek historical handwritten documents, Journal of Imaging 7 (12) (2021) 260.

[30] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[31] U.-V. Marti, H. Bunke, The IAM-database: an English sentence database for offline handwriting recognition, International Journal on Document Analysis and Recognition 5 (1) (2002) 39–46.

[32] V. Frinken, H. Bunke, Continuous handwritten script recognition, in: Handbook of Document Image Processing and Recognition, 2014, pp. 391–425.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[34] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the International Conference on Learning Representations, 2015.

[35] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch (2017).

[36] M. Yousef, K. F. Hussain, U. S. Mohammed, Accurate, data-efficient, unconstrained text recognition with convolutional neural networks, Pattern Recognition 108 (2020) 107482.

[37] L. Kang, P. Riba, M. Villegas, A. Fornés, M. Rusiñol, Candidate fusion:

Integrating language modelling into a sequence-to-sequence handwritten word recognition architecture, Pattern Recognition 112 (2021) 107790.

[38] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proceedings of the International Conference on Learning Representations, 2015.

[39] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio, Attention-based models for speech recognition, in: Proceedings of the Neural Information Processing Systems Conference, 2015.

[40] T. Bluche, R. Messina, Gated convolutional recurrent neural networks for multilingual handwriting recognition, in: Proceedings of the International Conference on Document Analysis and Recognition, 2017.

[41] A. Gotmare, N. S. Keskar, C. Xiong, R. Socher, A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation, in: Proceedings of the International Conference on Learning Representations, 2019.

[42] S. Merity, C. Xiong, J. Bradbury, R. Socher, Pointer sentinel mixture models, in: Proceedings of the International Conference on Learning Representations, 2017.

[43] J. Almazán, A. Gordo, A. Fornés, E. Valveny, Word spotting and recognition with embedded attributes, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (12) (2014) 2552–2566.

[44] P. Dreuw, P. Doetsch, C. Plahl, H. Ney, Hierarchical hybrid MLP/HMM or rather MLP features for a discriminatively trained Gaussian HMM:

a comparison for offline handwriting recognition, in: Proceedings of the IEEE International Conference on Image Processing, 2011, pp. 3541–3544.

[45] R. Bertolami, H. Bunke, Hidden Markov model-based ensemble methods for offline handwritten text line recognition, Pattern Recognition 41 (11) (2008) 3452–3460.

[46] P. Dreuw, G. Heigold, H. Ney, Confidence-and margin-based mmi/mpe discriminative training for off-line handwriting recognition, International Journal on Document Analysis and Recognition 14 (3) (2011) 273.

[47] F. Zamora-Martinez, V. Frinken, S. España-Boquera, M. J. Castro-Bleda, A. Fischer, H. Bunke, Neural network language models for off-line handwriting recognition, Pattern Recognition 47 (4) (2014) 1642–1652.

[48] J. Pastor-Pellicer, S. Espana-Boquera, M. J. Castro-Bleda, F. Zamora-Martinez, A combined convolutional neural network and dynamic programming approach for text line normalization, in: Proceedings of the International Conference on Document Analysis and Recognition, 2015.

[49] P. Doetsch, H. Ney, et al., Improvements in RWTH's system for off-line handwriting recognition, in: Proceedings of the International Conference on Document Analysis and Recognition, 2013.

[50] P. Doetsch, M. Kozielski, H. Ney, Fast and robust training of recurrent neural networks for offline handwriting recognition, in: Proceedings of

the International Conference on Frontiers in Handwriting Recognition, 2014.

[51] Z. Chen, Y. Wu, F. Yin, C.-L. Liu, Simultaneous script identification and handwriting recognition via multi-task learning of recurrent neural networks, in: Proceedings of the International Conference on Document Analysis and Recognition, 2017.

[52] V. Pham, T. Bluche, C. Kermorvant, J. Louradour, Dropout improves recurrent neural networks for handwriting recognition, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2014.

[53] P. Krishnan, K. Dutta, C. Jawahar, Word spotting and recognition using deep embedding, in: Proceedings of the IAPR International Workshop on Document Analysis Systems, 2018.

[54] C. Wigington, C. Tensmeyer, B. Davis, W. Barrett, B. Price, S. Cohen, Start, follow, read: End-to-end full-page handwriting recognition, in: Proceedings of the European Conference on Computer Vision, 2018.

[55] K. Dutta, P. Krishnan, M. Mathew, C. Jawahar, Improving CNN-RNN hybrid networks for handwriting recognition, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2018.

[56] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, R. Pang, Conformer: Convolution-augmented transformer for speech recognition, in: Proceedings of Interspeech, 2020.

[57] S. Brunessaux, P. Giroux, B. Grilhères, M. Manta, M. Bodin, K. Choukri, O. Galibert, J. Kahn, The maurdor project: improving automatic processing of digital documents, in: Proceedings of the IAPR International Workshop on Document Analysis Systems, 2014.