Faster DAN: Multi-target Queries with Document Positional Encoding for End-to-End Handwritten Document Recognition

Denis Coquenet[1(✉)], Clément Chatelain[2,3], and Thierry Paquet[2,4]

[1] Conservatoire National des Arts et Métiers, CEDRIC, Paris, France
denis.coquenet@lecnam.net
[2] LITIS Laboratory, EA 4108, Rouen Cedex, France
{clement.chatelain,thierry.paquet}@litislab.eu
[3] Rouen University, Mont-Saint-Aignan, France
[4] INSA of Rouen, Saint-Etienne-du-Rouvray, France

Abstract. Recent advances in handwritten text recognition enabled to recognize whole documents in an end-to-end way: the Document Attention Network (DAN) [9] recognizes the characters one after the other through an attention-based prediction process until reaching the end of the document. However, this autoregressive process leads to inference that cannot benefit from any parallelization optimization. In this paper, we propose Faster DAN, a two-step strategy to speed up the recognition process at prediction time: the model predicts the first character of each text line in the document, and then completes all the text lines in parallel through multi-target queries and a specific document positional encoding scheme. Faster DAN reaches competitive results compared to standard DAN, while being at least 4 times faster on whole single-page and double-page images of the RIMES 2009, READ 2016 and MAURDOR datasets. Source code and trained model weights are available at https://github.com/FactoDeepLearning/FasterDAN.

Keywords: Handwritten Document Recognition · Document Layout Analysis · Handwritten Text Recognition · Transformer

1 Introduction

Unconstrained offline handwritten text recognition has been studied for decades now. Until recently, all the proposed approaches were focused on recognizing the text from cropped parts (text regions) of the original document, leading to a sequential multistep approach, namely text region segmentation, ordering and recognition. Numerous advances enabled to extend the recognition stage to handle increasingly complex inputs. In the 90's, the use of Hidden Markov Model (HMM) enabled to go from isolated character recognition [19] to multi-character (word or line) recognition [12,14]. Thereafter, the democratization of

G. A. Fink et al. (Eds.): ICDAR 2023, LNCS 14190, pp. 182–199, 2023.
https://doi.org/10.1007/978-3-031-41685-9_12

deep neural networks, combined with the Connectionist Temporal Classification (CTC) loss [15], made the line-level approach the standard framework to handle handwritten document recognition [7,11,16,22,23,30,33].

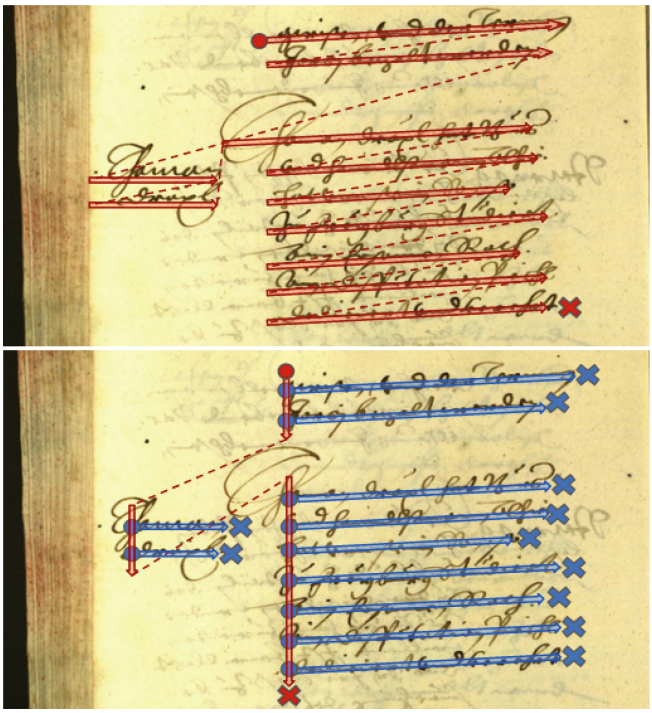


Fig. 1. Reading order comparison between DAN (top) and Faster DAN (bottom). Circles and crosses represent the start and the end of a pass, respectively. The first pass is showed in red, and the second one in blue. The DAN (top) sequentially predicts the characters of the whole documents in a single pass. The Faster DAN first predicts the first character of each line (as well as the layout tokens), and then predicts the remaining of all the text lines in parallel, in a second pass. (Color figure online)

More recently, few advanced works focused on text recognition at paragraph level [1,2,8,32], reaching similar performance compared to line-level recognition [10]. However, whether it is at character, word, line or paragraph level, this three-step paradigm has many drawbacks: the errors accumulate from one step to another, additional physical segmentation annotations are required to train the segmentation step, the use of a rule-based ordering stage is limited for documents with a complex layout, and the stages are performed independently, so they cannot benefit from one another.

Based on these observations, we proposed in [9] a new end-to-end paradigm named Handwritten Document Recognition (HDR). It aims at serializing documents in an XML-way, combining both Handwritten Text Recognition (HTR)

and Document Layout Analysis (DLA), through layout XML-markups. We proposed the Document Attention Network (DAN) [9] to tackle HDR.

It is made up of a Fully Convolutional Network (FCN) encoder to extract features from the input image, and a transformer [29] decoder to recurrently predict the different character and layout tokens. The DAN reached competitive results, recognizing both text and layout at page or double-page levels, compared to state-of-the-art line-level or paragraph-level approaches. The main drawback of the DAN is about its autoregressive character-level prediction process, which leads to high prediction times (a few seconds per document image).

In this paper, we propose Faster DAN, a novel approach to significantly reduce the prediction time of end-to-end HDR, without impacting the training time. This approach is based on a new document positional encoding whose aim is to inject the line membership information to each predicted character. In this way, we can parallelize the recognition of the text lines still using a single model, while reducing the total number of iterations. The Faster DAN relies on a two-step prediction process: a first step is dedicated to the prediction of the layout tokens, as well as the first character of each text line; all the text lines are then recognized in parallel in the second stage through multi-target transformer queries. This is illustrated in Fig. 1.

We show that the Faster DAN reaches competitive results compared to the original DAN, while being at least 4 times faster on three public datasets: READ 2016, RIMES 2009 and MAURDOR.

This paper is organized as follows. Section 2 is dedicated to the related works. DAN background is presented in Sect. 3. We detail the proposed approach in Sect. 4. Section 5 presents the experimental environment and the results. We draw the conclusion in Sect. 6.

2 Related Works

Nowadays, the most popular HTR framework is made up of three stages: the input document image is segmented into text line crops, which are then ordered and recognized. Indeed, the concept of text line is widely used as a building block in many works, and has been studied from different angles.

The text line has mostly been studied from the physical point of view: the majority of the works focused on predicting text line bounding boxes, either through a pixel-by-pixel classification task [3,21,24] or through an object-detection approach [5,6]. Detecting the start-of-line information was also studied as part of the segmentation stage. In [20], a model is trained to predict the coordinates of the bottom-left corner of each text line, as well as their height. Similarly, in [28,31], the authors considered the prediction of the start-of-line coordinates as an object detection task, using a region proposal network. Scale and rotation values are also associated to each text line to handle monotonic slanted lines. Contrary to these works, the Faster DAN strategy we propose only relies on language supervision: we do not need any additional physical annotations.

Recent works proposed to perform the recognition step at paragraph level [1,2,8,33]. Although not relying on raw physical text line annotations, most

paragraph-level text recognition works take advantage of the physical properties of text lines in single-column layout: the whole horizontal axis is associated to a text line, no matter its length. The authors of [32] and [8] concatenate the representation of the different text lines, or the text line predictions, respectively, to get back to a one-dimensional alignment problem. In [1,10], the text lines are processed recurrently through a line-level attention mechanism.

Another approach to deal with multi-line images consists in relying on an autoregressive character-level prediction process [2,9,25,27]. This time, the notion of text line is limited to the use of a dedicated line break token, used as any other character token. This way, this approach is no longer limited to single-column document. This strategy is also used in [18] for visual question-answering, information extraction or classification of documents, the OCR task being reduced to pretraining. In [9], we proposed the Document Attention Network to tackle Handwritten Document Recognition, by predicting opening and closing layout markup tokens in an XML way: all the character and layout tokens are sequentially and indifferently predicted, leading to hundreds or even thousands of iterations for single-page or double-page document images. It results in long prediction times: approximately one second for 100 characters on a single GPU V100.

In this paper, we propose to speed up the prediction of this latter approach by reading text lines in parallel. This way, we take the best of both worlds: we can deal with documents with complex layout through this character-level attention, and we use the concept of text line more directly through the prediction of the first character of each line and by using a dedicated document positional encoding scheme, but without using any additional physical annotations.

3 DAN Background

We proposed the Document Attention Network (DAN) in [9] for the task of Handwritten Document Recognition. It takes an input image of a whole document $\boldsymbol{X} \in \mathbb{R}^{H_\mathrm{i} \times W_\mathrm{i} \times C_\mathrm{i}}$, where H_i, W_i, C_i are the height, the width and the number of channels, respectively. It outputs the associated XML-like serialized representation $\hat{\boldsymbol{y}}$, *i.e.*, a sequence of tokens, each token $\hat{\boldsymbol{y}}_i$ representing either a layout markup or a character among an alphabet $\mathcal{A}^*$. For an input document represented by N tokens, we can note the expected output sequence as $\boldsymbol{y}^* \in {\mathcal{A}^*}^N$. For instance, a three-line document, split into two paragraphs, could be represented as:

<D><P>Line 1\ nLine 2</P><P>Line 3</P></D>

where <D> and <P> corresponds to document and paragraph markups, respectively.

The DAN is made up of two main components. An FCN encoder is used to extract 2D features $\boldsymbol{f}^{\mathrm{2D}} \in \mathbb{R}^{H \times W \times d}$ from the input image $\boldsymbol{X}$, with $H = \frac{H_\mathrm{i}}{32}$, $W = \frac{W_\mathrm{i}}{8}$ and $d = 256$. A Transformer decoder is used to iteratively predict the tokens $\hat{\boldsymbol{y}}_i$. To this aim, a special start-of-transcription token is used to initialize

the prediction ($\hat{\boldsymbol{y}}_0 = <\text{sot}>$) and a special end-of-transcription token is added to the ground truth to stop it. This way, the new target sequence is $\boldsymbol{y} \in \mathcal{A}^{N+1}$ with $\boldsymbol{y}_{N+1} = <\text{eot}>$ and $\mathcal{A} = \mathcal{A}^* \cup \{<\text{eot}>\}$. During inference, a maximum number of iterations $N_{\max} = 3,000$ is fixed in case of the <eot> token is not predicted.

The transformer attention mechanism being invariant to the order of its input sequences, positional encoding is added to inject the positional information: 2D positional encoding $\boldsymbol{P}^{2\text{D}} \in \mathbb{R}^{H \times W \times d}$ for the 2D features of the image, and 1D positional encoding $\boldsymbol{P}^{1\text{D}} \in \mathbb{R}^{N_{\max} \times d}$ for the previously predicted tokens. Both positional encodings are defined as a fixed encoding based on sine and cosine functions with different frequencies, as proposed in the original Transformer paper [29]. The image features are flattened afterward, for transformer needs:

$$\boldsymbol{f}^{1\text{D}} = \text{flatten}(\boldsymbol{f}^{2\text{D}} + \boldsymbol{P}^{2\text{D}}). \tag{1}$$

The DAN can be seen under the prism of the question-answering paradigm. At iteration t, the question corresponds to the previously predicted tokens $\hat{\boldsymbol{y}}^{\boldsymbol{t}} = [\hat{\boldsymbol{y}}_0, ..., \hat{\boldsymbol{y}}_{t-1}]$, referred to as *context* in this work, and the answer is the next token $\hat{\boldsymbol{y}}_t$. Formally, the tokens are first embedded through a learnable matrix $\boldsymbol{E} \in \mathbb{R}^{(|\mathcal{A}|+1) \times d}$ (+1 for the <sot> token), leading to $\boldsymbol{e}^{\boldsymbol{t}} = [\boldsymbol{e}_0, ..., \boldsymbol{e}_{t-1}]$, with $\boldsymbol{e}_i = \boldsymbol{E}_{\hat{\boldsymbol{y}}_i}$ ($\in \mathbb{R}^d$). Positional embedding is then added to get the transformer input query $\boldsymbol{q}^{\boldsymbol{t}} = [\boldsymbol{q}_0, ..., \boldsymbol{q}_{t-1}]$ with $\boldsymbol{q}_i = \boldsymbol{e}_i + \boldsymbol{P}^{1\text{D}}_i$.

The transformer's self-attention and cross-attention mechanisms compute an output $\boldsymbol{o}_i \in \mathbb{R}^d$ for each query input $\boldsymbol{q}_i$ by comparing them with the other query tokens, and with the image features $\boldsymbol{f}^{1\text{D}}$, respectively. Formally,

$$\boldsymbol{o}^{\boldsymbol{t}} = [\boldsymbol{o}_0, ..., \boldsymbol{o}_{t-1}] = \text{decoder}(\boldsymbol{q}^{\boldsymbol{t}}, \boldsymbol{f}^{1\text{D}}), \tag{2}$$

where the decoder corresponds to a stack of 8 standard transformer decoder layers [29]. This process being autoregressive, the query at position i can only attend to positions from 0 to i. In addition, the intermediate computations are preserved for each layer from one iteration to another in order to avoid computing the same output multiple times.

A score $\boldsymbol{s}^t_i$ is computed for each token i of the alphabet $\mathcal{A}$ using a single densely-connected layer of weights $\boldsymbol{W}_p$ ($\boldsymbol{s}^t \in \mathbb{R}^{|\mathcal{A}|}$):

$$\boldsymbol{s}^t = \boldsymbol{W}_p \cdot \boldsymbol{o}_{t-1}. \tag{3}$$

Probabilities are obtained through softmax activation: $\boldsymbol{p}^t_i = \frac{\exp \boldsymbol{s}^t_i}{\sum_j \exp \boldsymbol{s}^t_j}$. The predicted token at iteration t is the one whose probability is maximum:

$$\hat{\boldsymbol{y}}_t = \arg\max(\boldsymbol{p}^t). \tag{4}$$

The model is trained in an end-to-end fashion using the cross-entropy loss over the sequence of tokens:

$$\mathcal{L}_{\text{DAN}} = \sum_{t=1}^{N+1} \mathcal{L}_{\text{CE}}(\boldsymbol{y}_t, \boldsymbol{p}^t). \tag{5}$$

This autoregressive process can be parallelized during training through teacher forcing, but this is not possible during inference. That is why we propose the Faster DAN strategy.

4 Faster DAN

The standard character-level attention-based approach for HTR is to sequentially recognize all the characters $\boldsymbol{y}_i$ of the whole input image $\boldsymbol{X}$. This way the number of iterations, thus the prediction time, grows linearly with the number of characters in the document. This may be negligible for isolated text line images, for which the image feature extraction stage is predominant, but this becomes significant for whole page images (around one second for 100 characters on a GPU V100).

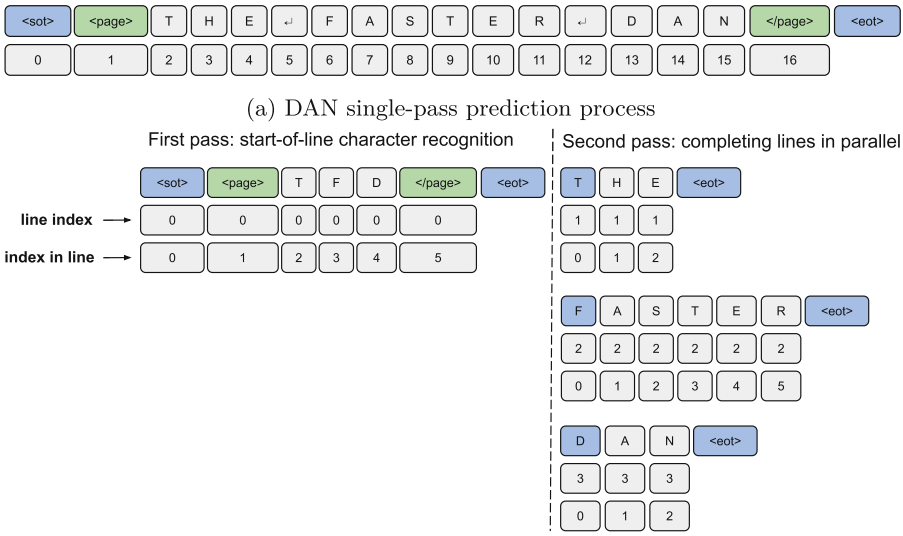
We propose the Faster DAN, a novel approach for Handwritten Document Recognition, to noticeably reduce the prediction time. The goal is to take advantage of the line-based structure of documents to parallelize the recognition of the text lines. Considering the layout markups and the <eot> tokens as lines by themselves (of unit length), we can rewrite the target sequence as $\boldsymbol{y} = \text{concatenate}(\boldsymbol{y}^1, ..., \boldsymbol{y}^L)$ where L is the number of lines in the document and $\boldsymbol{y}^j \in \mathcal{A}^{n_j}$ represent the different text lines ($\boldsymbol{y}_i^j$ is the character i of line j).

Using one model per line is prohibitive in terms of GPU memory consumption. Instead, the parallelization is carried out among a single model which processes multi-target queries through masking in the second pass. This is feasible thanks to the dedicated document positional encoding scheme we propose. It is important to note that the proposed approach is not specific to the DAN architecture. It could be used with any attention-based HDR model. However, to our knowledge, the only available end-to-end HDR model is the DAN.

Reading Lines in Parallel. Parallelizing the recognition faces two main challenges: detecting all the text lines, and recognizing them in parallel through transformer queries without mixing them. Moreover, since our goal is to perform HDR, and not only HTR, we also need to recognize the layout entities.

To tackle these issues, we opted for a two-pass process, as illustrated in Fig. 2b. In a first pass, the model sequentially predicts the layout tokens as well as the first character of each text lines, solving both layout recognition and text line detection. Then, the different text lines are completed in parallel based on their previously predicted first character. To this end, it is crucial to determine which token belongs to which line.

Document Positional Encoding. To parallelize the recognition of the text lines, we propose a new positional encoding scheme, as shown in Fig. 2. We associate to each predicted token $\hat{\boldsymbol{y}}_i^j$ (with $\hat{\boldsymbol{y}}_0^0 = <\text{sot}>$) two 1D positional embedding: one for the index of the line, and the other one for the index of the token in the line, leading to the global positional embedding $\boldsymbol{P}^{\text{doc}} \in \mathbb{R}^{l_{\max} \times n_{\max} \times d}$,



(a) DAN single-pass prediction process

(b) Faster DAN two-pass prediction process

Fig. 2. Comparison of the prediction process and positional encoding scheme between DAN and Faster DAN. This illustrates the example of a document input with three one-word text lines. The DAN associates a unique positional value for each token, which continues from one text line to the next. The Faster DAN uses two positional values: the index of the text line and the position of the token in this text line. Special (start and end) tokens are in blue and layouts tokens are in green. (Color figure online)

where $l_{\max}$ is the maximum number of line and $n_{\max}$ is the maximum number of characters per line. $\boldsymbol{y}_i^j$ is associated to:

$$\boldsymbol{P}_{j,i}^{\text{doc}} = \text{concatenate}(\boldsymbol{P}_j^{\text{1D'}}, \boldsymbol{P}_i^{\text{1D'}}), \tag{6}$$

where $\boldsymbol{P}^{\text{1D'}}$ is equivalent to $\boldsymbol{P}^{\text{1D}}$ but encoded on half channels ($\boldsymbol{P}_i^{\text{1D'}} \in \mathbb{R}^{d/2}$). The transformer input queries become $\boldsymbol{q}_i^j = \boldsymbol{E}_{\hat{y}_i^j} + \boldsymbol{P}_{j,i}^{\text{doc}}$. The idea of injecting the line information was already used in [27], but it was computed as a ratio with an arbitrary maximum number of lines, and concatenated to the token embedding directly. In addition, the position of the tokens was absolute, and not relative to the current text line, as for the standard DAN.

First Pass. The Faster DAN follows the standard autoregressive process to predict the first token $\hat{\boldsymbol{y}}_0^j$ of each line j based on Eqs. 2 to 4. At iteration t, $\boldsymbol{q}^t = [\boldsymbol{q}_0^0, ..., \boldsymbol{q}_0^{t-1}]$.

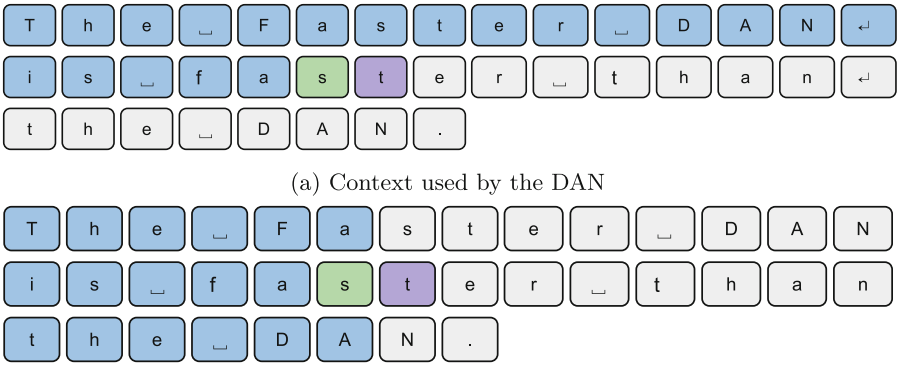
Second Pass. The standard Transformer decoding process is to give a sequence of query tokens $\boldsymbol{q}^t$ as input and keep the output corresponding to the last token only ($\boldsymbol{o}_{t-1}$), as single output for iteration t. Instead, the output of the

last token of each line $\boldsymbol{o}^j_{t-1}$ are kept in this second pass. We refer to this as multi-target queries. $\hat{\boldsymbol{y}}^j_0$ are duplicated into $\hat{\boldsymbol{y}}^j_1$ to initiate the second pass; the modification of the associated position in line (from 0 to 1) indicates to the model a change of expected behavior: from the prediction of the first token of the next line to the prediction of the next token of the current line. By setting $\boldsymbol{q^t} = [\boldsymbol{q}^0_0, ..., \boldsymbol{q}^0_{t-1}, ..., \boldsymbol{q}^L_0, ..., \boldsymbol{q}^L_{t-1}]$ (the t first tokens of all the lines), we obtain $\boldsymbol{o^t} = [\boldsymbol{o}^0_0, ..., \boldsymbol{o}^0_{t-1}, ..., \boldsymbol{o}^L_0, ..., \boldsymbol{o}^L_{t-1}]$ through Eq. 2. In this way, the t^{th} tokens of each line j are computed in a single iteration t:

$$\hat{\boldsymbol{y}}^j_t = \arg\max(\boldsymbol{W}_p \cdot \boldsymbol{o}^j_{t-1}). \tag{7}$$

Extra tokens ($\hat{\boldsymbol{y}}^j_i$ with $i > n_j$) are discarded through masking.

Context Exploitation. The naive approach to recognize the text lines in parallel would be to recognize them independently, by applying a mask to discard the tokens from all the other text lines. It means that $\boldsymbol{q}^j_i$ could only attend to line j (itself) and position 0 to i in that line. However, this would lead to an important loss of context. Instead, we propose to take advantage of all the partially predicted text lines: $\boldsymbol{q}^j_i$ can attend to all lines, from 0 to L, and from position 0 to i in those lines, this is illustrated in Fig. 3



(a) Context used by the DAN

(b) Context used by the Faster DAN

Fig. 3. Context comparison between DAN and Faster DAN. The colored cells represent the current character to predict (in purple), the previously predicted tokens *i.e.* the context (in blue and green), the token used for the prediction (in green), and the remaining characters to recognize (in gray). (Color figure online)

The major drawback of parallelizing the line recognition, compared to purely sequential recognition, is the loss of context. Indeed, the standard DAN benefits from all the past context during prediction: this is partially available for the Faster DAN since the past context is limited to the beginning of the text lines.

In this way, it becomes harder for the model to focus on the correct text part, especially for very short contexts. Indeed, a sequence of characters may appear several times in a document, especially if this sequence is short, *e.g.,* at the beginning of the recognition process. We counterbalance the loss of context from past by combining partial context from both past and future. We show the impact of this approach in Sect. 5.6.

Training and Inference. The model is trained over the target sequence using the cross-entropy loss:

$$\mathcal{L} = \sum_{j=1}^{L} \sum_{\substack{i=0 \\ i \neq 1}}^{n_j} \mathcal{L}_{\text{CE}}(\boldsymbol{y}_i^j, \boldsymbol{p}_i^j). \tag{8}$$

It has to be noted that the training time is not impacted by this two-step decoding strategy since the whole expected sequence prediction (from both passes) is trained in parallel through teacher forcing, with appropriate masks.

During inference, the Faster DAN reduces the number of iterations I from

$$I_{\text{DAN}} = \sum_{j=1}^{L} n_j = N + 1 \quad \text{to} \quad I_{\text{FasterDAN}} = L + \max_j(n_j),$$

by considering the line breaks as belonging to the lines. For example, 25 text lines of 50 characters, structured according to 3 layout entities, leads to 1,251 iterations for the DAN, and only 76 iterations for the proposed Faster DAN.

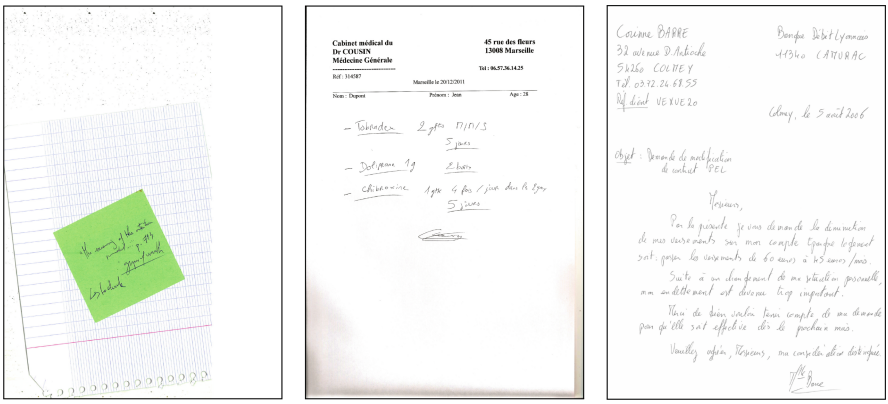
5 Experimental Study

5.1 Datasets

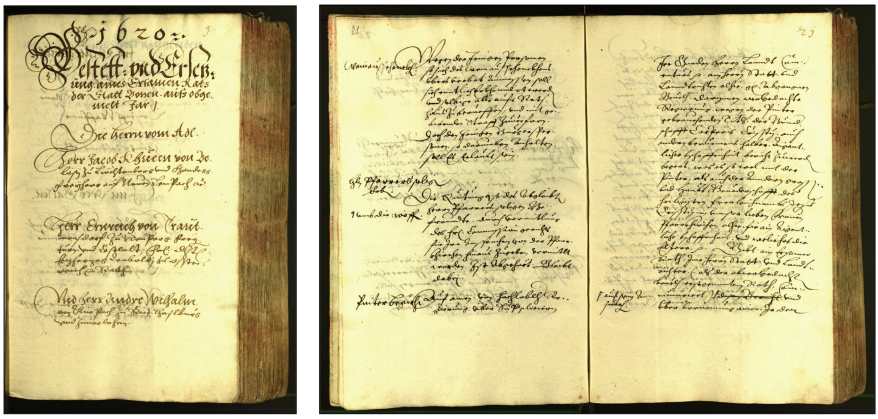
We used three document-level public datasets to evaluate the proposed approach: RIMES 2009 [17], READ 2016 [26] and MAURDOR [4]. Document image examples from these three datasets are showed in Fig. 4.

RIMES 2009. The RIMES 2009 dataset corresponds to French grayscale handwritten page images. These pages are letters produced in the context of writing mail scenarios. Text regions are classified among one of the following 7 classes: sender coordinates, recipient coordinates, object, date & location, opening, body and PS & attachment. We used these classes as layout tokens.

READ 2016. The READ 2016 dataset corresponds to Early Modern German handwritten pages from the Ratsprotokolle collection. Images are RGB encoded. We used two versions of this dataset: single-page images and double-page images. The layout classes are as follows: page, section, margin annotation and body.



(a) MAURDOR C3 (b) MAURDOR C4 (c) RIMES 2009



(d) READ 2016 (single-page) (e) READ 2016 (double-page)

Fig. 4. Document image examples from the RIMES 2009, READ 2016 and MAURDOR datasets.

MAURDOR. The MAURDOR dataset consists in a heterogeneous collection of documents. We used the same configuration as in [9] *i.e.* we only use the English and French documents, and we focus on the C3 and C4 subsets of this dataset, which corresponds to private or professional correspondences. The documents are either handwritten, printed, or a mix of both. There is no sufficient annotation to produce the layout tokens, so we only evaluate the HTR task on this dataset.

Table 1 details the splits in training, validation and test, as well as the number of characters in the alphabet and the number of layout tokens (2 by class, for opening and closing markups) for each dataset.

Table 1. Splits and number of character and layout tokens for each dataset.

Dataset	Training	Validation	Test	# char tokens	# layout tokens
RIMES 2009	1,050	100	100	108	14
READ 2016 (single-page)	350	50	50	89	10
READ 2016 (double-page)	169	24	24	89	10
MAURDOR (C3)	1,006	148	166	134	✗
MAURDOR (C4)	721	111	114	127	✗

5.2 Metrics

In addition to the standard Character Error Rate (CER) and Word Error Rate (WER) metrics used to evaluate the text recognition performance, we proposed two metrics in [9] to evaluate the specific layout recognition of the HDR task. The Layout Ordering Error Rate (LOER) consists in considering the document layout as a graph and computing the graph edit distance between the prediction and the ground truth. The LOER aims at evaluating the layout recognition only, considering the reading order between layout entities. Since LOER and CER/WER only evaluate the layout and text recognition independently, the mAP_{CER} is used to evaluate the recognition of the layout with respect to the text content. It is computed as the area under the precision/recall curve, as in object detection approaches [13] for instance, but it is based on a CER threshold instead of a IoU one. The mAP_{CER} does not dependent on the reading order between layout entities. That is why it is important to consider all these metrics altogether to evaluate the HDR task.

5.3 Training Details

In [9], we used some pretraining and curriculum training strategies to speed up the convergence of the DAN, and to not use any physical segmentation annotation during training. To be fairly comparable with this work, we follow the exact same training configuration, whose major points are as follows:

- The encoder is pretrained on synthetic isolated text line images using the CTC loss and a dedicated FCN line-level OCR model.
- The Faster DAN is trained on a mixture of real and synthetic documents. Using a curriculum strategy, the Faster DAN is trained on increasingly complex synthetic documents through the epochs. The complexity varies from two aspects: the number of lines contained in the document image, and the size of this image. The ratio between synthetic and real document also evolves during training, from 90%/10% to 20%/80%.
- A rule-based post-processing is used to make sure that the layout markups have the correct format (no unpaired markup, for instance).

– Whether it is for pretraining or training, input images are downsized to 150 dpi, normalized and data augmentation is performed 90% of the time.

We carried out 2-day pretraining and 4-day training on a single GPU V100 (32 Go), using automatic mixed-precision. We used the Adam optimizer with an initial learning rate of 10^{-4}. We do not use any external data, external language model nor lexicon constraints.

5.4 Comparison with the State of the Art

To our knowledge, the only work performing HDR is the DAN [9]. Tables 2, 3 and 4 provide an evaluation of the Faster DAN on the READ 2016, RIMES 2009 and MAURDOR datasets, respectively, as well as a comparison with the state of the art.

Table 2. Evaluation of the Faster DAN on the test set of the READ 2016 dataset and comparison with the state of the art. Metrics are expressed in percentages.

Architecture	READ 2016 (single-page)				READ 2016 (double-page)			
	CER ↓	WER ↓	LOER ↓	mAP_{CER} ↑	CER ↓	WER ↓	LOER ↓	mAP_{CER} ↑
DAN [9]	**3.43**	**13.05**	5.17	93.32	**3.70**	**14.15**	4.98	93.09
Faster DAN	3.95	14.06	**3.82**	**94.20**	3.88	14.97	**3.08**	**94.54**

Table 3. Evaluation of the Faster DAN on the test set of the RIMES 2009 dataset and comparison with the state of the art. Metrics are expressed in percentages.

Architecture	RIMES 2009			
	CER ↓	WER ↓	LOER ↓	mAP_{CER} ↑
DAN [9]	**4.54**	**11.85**	**3.82**	**93.74**
Faster DAN	6.38	13.69	4.48	91.00

Table 4. Evaluation of the Faster DAN on the test set of the MAURDOR dataset and comparison with the state of the art. Metrics are expressed in percentages.

Architecture	C3		C4		C3 & C4	
	CER ↓	WER ↓	CER ↓	WER ↓	CER ↓	WER ↓
DAN [9]	**8.62**	**18.94**	**8.02**	**14.57**	11.59	27.68
Faster DAN	8.93	19.00	9.88	16.52	**10.50**	**19.64**

The Faster DAN reaches competitive results compared to the DAN on the three datasets. For the READ 2016 dataset, it even reaches state-of-the-art results in terms of LOER and mAP_{CER} for both single-page and double-page versions, involving a better recognition of the layout. Results are not as good for the RIMES 2009 dataset, which includes more variability in terms of layout. We assume that this higher variation makes the first pass of the Faster DAN more

difficult. This is confirmed when measuring the CER for the first pass only: it is of 4.72% and 5.34% for READ 2016 at single-page and double-page levels, and of 9.10% for RIMES 2009. Concerning the MAURDOR dataset, the Faster DAN reaches competitive results on the C3 and C4 categories, taken separately, and it reaches new state-of-the art results when mixing both categories with 10.50% of CER, compared to 11.59% for the standard DAN.

Discussion. It has to be noted that it is more difficult to compare the text recognition performance at document level than at line level. Indeed, the reading order is far more complex for documents, to go from one paragraph to another, and to one line to the next, than for isolated lines. This way, even perfectly recognized, the CER can be severely impacted if the paragraphs are recognized in the wrong order. On the contrary, the mAP_{CER} is invariant to the order of the layout entities, but it is dependent to the well recognition of the layout.

Another point to emphasize is about the severity of the errors made. There are two types of errors to be distinguished. The first corresponds to standard character addition, removal, or substitution cases. During the first pass of the Faster DAN, this kind of error may have a great impact because a whole text line may be duplicated or discarded. However, during the second pass, we assume that the impact of such errors is rather equivalent for both DAN and Faster DAN. The second kind of errors is related to the end-of-transcription token prediction. Indeed, although rare, the model may not predict the end of the transcription and loop on the same text region again and again until reaching an arbitrary chosen iteration limit. For this later issue, the standard DAN is more impacted than the Faster DAN. Indeed, the DAN only have one iteration limit, which corresponds to the global number of tokens to predict for the whole document. For the Faster DAN, we used two iteration limits: one for the number of lines, and one for the number of characters per line. Given that the range of values for a line length is smaller than for the whole document, the impact is less important for the Faster DAN.

Prediction Time. Table 5 shows a comparison of the Faster DAN with the DAN in terms of prediction times for the three datasets: RIMES 2009, READ 2016 and MAURDOR. To be fairly comparable, these times account for the whole prediction process, including the time dedicated to the encoder part and to formatting instructions. Additional details are given for each dataset such as the image sizes, the number of characters, lines, and layout tokens per image, and the number of characters per line. The values are given as average for the test set of each dataset. As one can note, the Faster DAN is significantly faster than the DAN for all the datasets, speeding up the prediction process by at least 4.

We showed that the Faster DAN reaches competitive results on three document-level datasets while being at least 4 times faster than the standard DAN at prediction time. We now evaluate the performance on heterogeneous documents, by mixing both RIMES 2009 and READ 2016 datasets.

Table 5. Prediction time comparison between the DAN and the Faster DAN. Times (in seconds) are averaged on the test set for a single document image, using a single GPU V100.

	RIMES 2009	READ 2016		MAURDOR		
		single-page	double-page	C3	C4	C3 & C4
Dataset details (averaged for a document on the test set)						
width (px)	1,235	1,190	2,380	1,336	1,240	1,297
height (px)	1,751	1,755	1,755	1,658	1,754	1,697
# chars	578	528	1,062	481	706	575
# lines	18	23	47	16	22	18
# chars/line	31	22	22	30	31	30
# layout tokens	11	15	30	0	0	0
Prediction times (in seconds)						
DAN [9]	5.6	4.6	8.5	5.8	7.7	6.6
Faster DAN	**1.4**	**0.9**	**1.9**	**1.0**	**1.6**	**1.3**
Speed factor	x4	x5.1	x4.5	x5.8	x4.8	x5.1

5.5 Evaluation on Heterogeneous Documents

In this experiment, we mixed both RIMES 2009 and READ 2016 datasets at single page level, for both training and evaluation. Examples from both datasets are balanced at training time, *i.e.*, the models have been trained on the same number of documents for both datasets. These are the first results for such an experiment; we also train the standard DAN for comparison purposes. Results are shown in Table 6. As one can note, results are rather similar when training on datasets separately or altogether, except for the DAN on the RIMES dataset whose CER increases from 4.54% up to 7.96%.

Table 6. Evaluation of the Faster DAN on heterogeneous data (mixing READ 2016 and RIMES 2009 for both training and evaluation) and comparison with the state of the art.

Architecture	RIMES 2009 (page)				READ 2016 (single-page)			
	CER ↓	WER ↓	LOER ↓	mAP_{CER} ↑	CER ↓	WER ↓	LOER ↓	mAP_{CER} ↑
DAN [9]	7.96	15.76	8.72	**91.59**	**3.50**	**13.36**	**3.86**	**94.23**
Faster DAN	**6.73**	**15.22**	**5.56**	90.10	3.81	14.30	4.32	93.57

Table 7. Ablation study of the Faster DAN and DAN. Results (in percentages) are given for the test set of the RIMES 2009 and READ 2016 datasets.

Architecture	RIMES 2009 (page)			READ 2016 (single-page)			READ 2016 (double-page)		
	CER	LOER	mAP_{CER}	CER	LOER	mAP_{CER}	CER	LOER	mAP_{CER}
Faster DAN	**6.38**	4.48	91.00	3.95	**3.82**	**94.20**	**3.88**	**3.08**	**94.54**
(1) No line encoding	79.39	6.21	0.00	75.08	11.81	0.29	75.01	10.79	5.44
(2) Single-line context	94.73	**4.30**	0.00	91.23	4.61	0.00	91.22	4.03	0.00
(3) First-pass context	8.27	4.90	90.73	6.68	4.50	88.37	6.87	5.22	87.93
(4) Sum PE	6.88	4.90	**91.06**	**3.82**	4.27	94.08	4.55	4.39	92.76

5.6 Ablation Study

In Table 7, we propose an ablation study of the proposed approach on the RIMES 2009 and READ 2016 datasets. The first line corresponds to the Faster DAN baseline. In experiment (1), the document positional encoding is replaced by standard 1d positional encoding, *i.e.*, a unique index is associated to each token. The model does not succeed to recognize the text, showing the necessity of injecting line positional information to parallelize the recognition. The model can only access to tokens of the text line to recognize in (2), also preventing the text recognition. Indeed, it is nearly impossible to predict the next character with only a one-character query (beginning of the second pass) since characters are not unique in a document. For both experiments, one can note that the LOER is nearly not impacted, this is because the layout recognition takes place in the first pass, before the parallelization.

In experiment (3), in addition to the tokens of the text line to recognize, the first character of all the text lines, as well as the layout markup tokens, are available. This leads to an increase of the CER of at least 1.89 points for RIMES 2009, and up to 2.99 points for READ 2016 at double-page level, compared to the baseline. This shows the efficiency of the text line detection performed in the first pass, since the text recognition is parallelized, but it also demonstrates that gathering the context from past and future lines helps to improve the performance. In experiment (4), the positional encoding of the line and of the index in the line are summed instead of being concatenated. As one can note, results are slightly in favor of the concatenation.

6 Conclusion

In this paper, we proposed the Faster DAN, a novel approach for end-to-end Handwritten Document Recognition. We evaluate this approach with the current state-of-the-art architecture and showed that this approach reaches competitive results on three document-level datasets while being at least 4 times faster. This way, we preserved the advantages of using a single end-to-end approach, while greatly mitigating the major drawback of prediction time. In this work, we focused on line-level multi-target queries to show the gain in prediction time.

However, it would also be possible to perform this parallelization at paragraph level in order to have a more important language modeling of the past: this would represent an in-between in terms of prediction time.

Acknowledgments. This work was granted access to the HPC resources of IDRIS under the allocation 2020-AD011012155.

References

1. Bluche, T.: Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In: Advances in Neural Information Processing Systems (NIPS), vol. 29, pp. 838–846 (2016)
2. Bluche, T., Louradour, J., Messina, R.O.: Scan, attend and read: end-to-end handwritten paragraph recognition with MDLSTM attention. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 1050–1055 (2017)
3. Boillet, M., Kermorvant, C., Paquet, T.: Robust text line detection in historical documents: learning and evaluation methods. Int. J. Doc. Anal. Recogn. (IJDAR) **25**, 95–114 (2022)
4. Brunessaux, S., et al.: The Maurdor project: improving automatic processing of digital documents. In: International Workshop on Document Analysis Systems (DAS), pp. 349–354 (2014)
5. Carbonell, M., Fornés, A., Villegas, M., Lladós, J.: A neural model for text localization, transcription and named entity recognition in full pages. Pattern Recogn. Lett. **136**, 219–227 (2020)
6. Chung, J., Delteil, T.: A computationally efficient pipeline approach to full page offline handwritten text recognition. In: Workshop on Machine Learning (WML@ICDAR), pp. 35–40 (2019)
7. Coquenet, D., Chatelain, C., Paquet, T.: Recurrence-free unconstrained handwritten text recognition using gated fully convolutional network. In: 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 19–24 (2020)
8. Coquenet, D., Chatelain, C., Paquet, T.: SPAN: a simple predict & align network for handwritten paragraph recognition. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR 2021. LNCS, vol. 12823, pp. 70–84. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86334-0_5
9. Coquenet, D., Chatelain, C., Paquet, T.: Dan: a segmentation-free document attention network for handwritten document recognition. IEEE Trans. Pattern Anal. Mach. Intell. (2023). https://doi.org/10.1109/TPAMI.2023.3235826
10. Coquenet, D., Chatelain, C., Paquet, T.: End-to-end handwritten paragraph text recognition using a vertical attention network. Trans. Pattern Anal. Mach. Intell. (TPAMI) **45**(1), 508–524 (2023)
11. Coquenet, D., Soullard, Y., Chatelain, C., Paquet, T.: Have convolutions already made recurrence obsolete for unconstrained handwritten text recognition ? In: Workshop on Machine Learning (WML@ICDAR), pp. 65–70 (2019)
12. El-Yacoubi, M.A., Gilloux, M., Sabourin, R., Suen, C.Y.: An hmm-based approach for off-line unconstrained handwritten word modeling and recognition. Trans. Pattern Anal. Mach. Intell. (TPAMI) **21**(8), 752–760 (1999)
13. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)

14. Gilloux, M., Lemarié, B., Leroux, M.: A hybrid radial basis function network/hidden markov model handwritten word recognition system. In: Third International Conference on Document Analysis and Recognition (ICDAR), pp. 394–397 (1995)
15. Graves, A., Fernández, S., Gomez, F.J., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: International Conference on Machine Learning (ICML), vol. 148, pp. 369–376 (2006)
16. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: Advances in Neural Information Processing Systems (NIPS), vol. 21, pp. 545–552 (2008)
17. Grosicki, E., Carré, M., Brodin, J., Geoffrois, E.: Results of the RIMES evaluation campaign for handwritten mail processing. In: 10th International Conference on Document Analysis and Recognition (ICDAR), pp. 941–945 (2009)
18. Kim, G., et al.: Ocr-free document understanding transformer. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) ECCV 2022. LNCS, vol. 13688, pp. 498–517. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19815-1_29
19. LeCun, Y., et al.: Handwritten digit recognition with a back-propagation network. In: Advances in Neural Information Processing Systems, vol. 2 (1989)
20. Moysset, B., Kermorvant, C., Wolf, C.: Full-page text recognition: learning where to start and when to stop. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 871–876 (2017)
21. Oliveira, S.A., Seguin, B., Kaplan, F.: dhSegment: a generic deep-learning approach for document segmentation. In: 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 7–12 (2018)
22. Ptucha, R.W., Such, F.P., Pillai, S., Brockler, F., Singh, V., Hutkowski, P.: Intelligent character recognition using fully convolutional neural networks. Pattern Recogn. **88**, 604–613 (2019)
23. Puigcerver, J.: Are multidimensional recurrent layers really necessary for handwritten text recognition? In: 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), pp. 67–72 (2017)
24. Renton, G., Soullard, Y., Chatelain, C., Adam, S., Kermorvant, C., Paquet, T.: Fully convolutional network with dilated convolutions for handwritten text line segmentation. Int. J. Doc. Anal. Recogn. (IJDAR) **21**(3), 177–186 (2018)
25. Rouhou, A.C., Dhiaf, M., Kessentini, Y., Salem, S.B.: Transformer-based approach for joint handwriting and named entity recognition in historical documents. Pattern Recogn. Lett. **155**, 128–134 (2022)
26. Sánchez, J., Romero, V., Toselli, A.H., Vidal, E.: ICFHR2016 competition on handwritten text recognition on the READ dataset. In: 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 630–635 (2016)
27. Singh, S.S., Karayev, S.: Full page handwriting recognition via image to sequence extraction. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR 2021. LNCS, vol. 12823, pp. 55–69. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86334-0_4
28. Tensmeyer, C., Wigington, C.: Training full-page handwritten text recognition models without annotated line breaks. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 1–8 (2019)
29. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems (NIPS), vol. 30, pp. 5998–6008 (2017)

30. Voigtlaender, P., Doetsch, P., Ney, H.: Handwriting recognition with large multi-dimensional long short-term memory recurrent neural networks. In: International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 228–233 (2016)
31. Wigington, C., Tensmeyer, C., Davis, B., Barrett, W., Price, B., Cohen, S.: Start, follow, read: end-to-end full-page handwriting recognition. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11210, pp. 372–388. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01231-1_23
32. Yousef, M., Bishop, T.E.: OrigamiNet: weakly-supervised, segmentation-free, one-step, full page text recognition by learning to unfold. In: Conference on Computer Vision and Pattern Recognition (CVPR), pp. 14698–14707 (2020)
33. Yousef, M., Hussain, K.F., Mohammed, U.S.: Accurate, data-efficient, unconstrained text recognition with convolutional neural networks. Pattern Recogn. **108**, 107482 (2020)