SPAN: a Simple Predict & Align Network for Handwritten Paragraph Recognition

Denis Coquenet[1,2,3] (✉)[0000−0001−5203−9423], Clément Chatelain[1,4][0000−0001−8377−0630], and Thierry Paquet[1,2][0000−0002−2044−7542]

[1] LITIS Laboratory - EA 4108, France
[2] Rouen University, France
[3] Normandie University, France
[4] INSA of Rouen, France
{denis.coquenet, clement.chatelain, thierry.paquet}@litislab.eu

Abstract. Unconstrained handwriting recognition is an essential task in document analysis. It is usually carried out in two steps. First, the document is segmented into text lines. Second, an Optical Character Recognition model is applied on these line images. We propose the Simple Predict & Align Network: an end-to-end recurrence-free Fully Convolutional Network performing OCR at paragraph level without any prior segmentation stage. The framework is as simple as the one used for the recognition of isolated lines and we achieve competitive results on three popular datasets: RIMES, IAM and READ 2016. The proposed model does not require any dataset adaptation and can be trained without line breaks in the transcription labels. Our code and trained model weights are available at `https://github.com/FactoDeepLearning/SPAN`.

Keywords: Handwritten paragraph recognition · Fully Convolutional Network · Recurrence-free model

1 Introduction

Offline handwritten text recognition consists in recognizing the text from a scanned document. This task is usually performed in two steps, by two different neural networks. In a first step, the document image is cut into text regions: this is the segmentation step. Then, Optical Character Recognition (OCR) is applied on each text region images. Following the advances of the recognition process over time, segmentation was performed on larger and larger entities, from the character in the early ages, to text lines more recently, gradually decreasing the amount of segmentation labels required to train the system.

As a matter of fact, producing segmentation labels by hand, in addition to transcription labels, is costly. Moreover, the use of a two-step process requires to clearly define what a line should be in a non-latent pivot format, *i.e.* a line of text, to generate target labels. However, the definition of a text line raises several questions that prevent to optimize its detection in order to maximize the recognition performance: is a text line a bounding box, a polygon, a set of pixels or a baseline? How should it be measured? Which loss should it be trained with? Given all these open questions, we claim that segmentation and recognition should be trained in an end-to-end fashion using a latent space between both stages. Indeed, this allows to circumvent any text line definition, while leveraging the annotation needs.

In this paper, we propose the Simple Predict & Align Network (SPAN), an end-to-end recurrence-free Fully Convolutional Network (FCN) model free of those issues, further reducing the needs for labels in two ways. First, the proposed model performs OCR at paragraph level, so it does not need line-level segmentation labels. Second, it does not even require line breaks in the transcription labels. The proposed model totally circumvents the line segmentation problem using a very straightforward approach. The input paragraph image is analysed in a 2D fashion using a classical fully convolutional architecture, leading to a 2D latent space that is reshaped into a 1D sequential latent space. Finally, the CTC loss is simply used to align the 1D character prediction sequence with the paragraph transcription.

This paper is organized as follows. Related Works are presented in Section 2. The proposed SPAN architecture is described in Section 3. Section 4 presents the experimental environment and the results. We draw conclusion in Section 5.

2 Related Works

In the literature, multi-line text recognition is mainly carried out in two steps. First, a text region (line/word) segmentation is performed [17,16,12]; then, an OCR is applied on the extracted text regions images [8,7,23,14] thanks to the Connectionist Temporal Classification (CTC) [10]. As shown in [9], deep neural networks perform well on both task separately but, when put together, errors in the segmentation stage leads to errors in the OCR stage, leading to higher Character Error Rate (CER).

Recently, one can notice a trend towards the use of unified models. We can classify them into two categories: those performing a text region segmentation prior to the recognition and those without explicit segmentation.

2.1 Segmentation-based approaches

Segmentation-based approaches, by definition, require line or word segmentation labels, in addition to the associated transcription label; so the line breaks must be annotated.

Among these approaches, [4,3,6] are based on object-detection methods: a Region Proposal Network (RPN), followed by a non-maximal suppression process and Region Of Interest (ROI), generates line or word bounding boxes. An OCR is then applied on these bounding boxes.

Other approaches are based on predicting the start-of-line coordinates. While in [15] the line is considered horizontal, in [21,20], lines are normalized, recurrently predicting coordinates. Finally, an OCR is applied on these lines.

2.2 Segmentation-free approaches

Since they do not explicitly segment the input image, segmentation-free approaches do not require any segmentation labels; the models can be trained using transcription labels only.

In [1,9], the proposed models incorporate an attention mechanism to recurrently generate line features, performing a kind of implicit line segmentation. Indeed, an encoder generates features from the input image; then, the attention process sequentially selects features to focus on. Finally, a decoder predicts characters from these features. [2] proposed a similar approach with an implicit character segmentation.

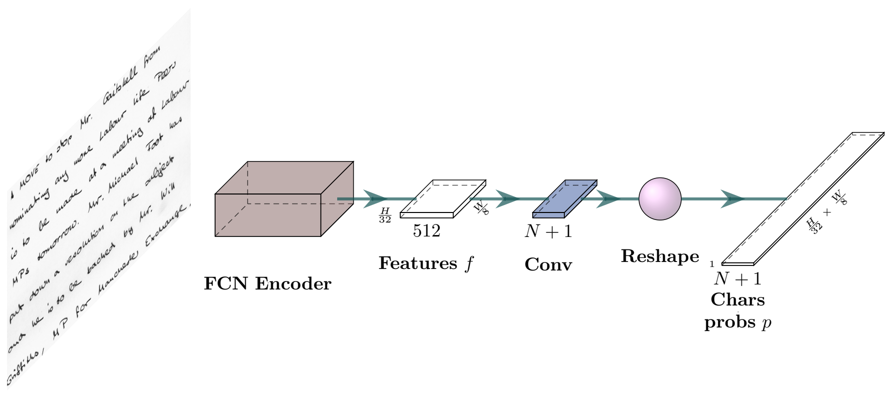
To our knowledge, only two other works have proposed segmentation-free approaches for multi-line text recognition. [18] focuses on the loss to tackle the two-dimensional aspect of the task, providing a Multi-Dimensional Connectionist Classification (MDCC). Ground truth transcription labels are converted to a two-dimensional model, using a Conditional Random Field (CRF). This model enables to jump from one line to the next one, adding a new line separator label in addition to the standard CTC blank label. In [22], the model is trained to unfold the input multi-line text image into a sequence of lines, thus forming a single large line. Thus, the task is reduced to a one-dimensional problem and the model can be trained with the standard CTC loss.

[18] and [2] are part of the first works proposed for multi-line text recognition, but they remain below the state of the art. While [1] requires pretraining on line-level images, [7] requires line breaks in the transcription labels. [22] is the only model that can be trained from scratch, without any segmentation labels nor line breaks in the transcription labels; but, as a counterpart, it requires some hyperparameters to be adapted for each dataset. Indeed, it requires input images of fixed sizes and includes intermediate bilinear interpolations with fixed dimensions which are specific to each dataset.

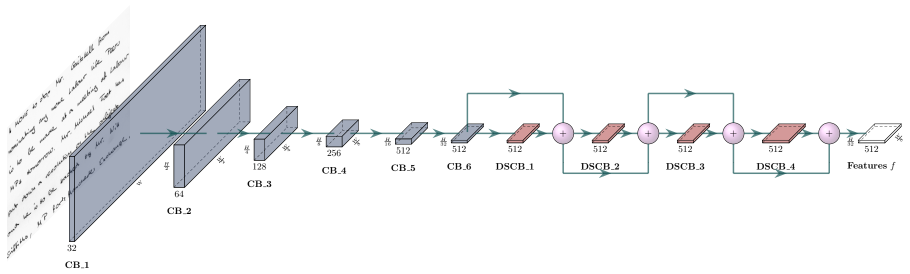
In this work, we propose an end-to-end model trained in the same conditions, *i.e.* with paragraph transcription as the only used label, without any line breaks. Instead of unfolding the input image as in [22], we propose to train a model to both predict and align characters so as to get vertical separation between lines, preserving the two-dimensional nature of the task. Contrary to the work presented in [22], the proposed model is able to handle variable size input images, making it flexible enough to be used on multiple datasets without modifying any hyperparameter.

3 SPAN Architecture

We propose an end-to-end model to perform the optical character recognition of paragraphs. We wanted to keep the original shapes of the input images in order to preserve both their ratio and their details as well as to be flexible enough to adapt to a large variety of datasets. To this end, we use a Fully Convolutional Network as the encoder to analyse the 2D paragraph images. An implicit line segmentation is performed by reducing the vertical axis through row concatenation, reshaping the 2D latent space into a 1D latent space, acting as a collapse operator for this dimension. The training process is based on the standard CTC loss that aligns the label sequence with the data in the 1D latent space without any need for line breaks in the annotation. Figure 1a shows an overview of the model architecture: it consists of an FCN encoder, which extracts the features, followed by a convolutional layer, which predicts the character probabilities. Finally, the rows of predictions are concatenated to obtain one single large row of predictions. This brings us back to a one-dimensional sequence alignment problem which is handled with the standard CTC loss.



(a) Global architecture overview.



(b) FCN Encoder overview. CB: Convolution Block, DSCB: Depthwise Separable Convolution Block.

Fig. 1: Model visualization. 1a presents an overview of the architecture and 1b focuses on the encoder.

3.1 Encoder

The purpose of the encoder is to extract features from the input images. It employs some convolutions with stride in order to reduce the memory consumption: it takes an input image $X \in \mathbb{R}^{H \times W \times C}$ and outputs some feature maps $f \in \mathbb{R}^{\frac{H}{32} \times \frac{W}{8} \times 512}$ where H, W and C are respectively the height, the width and the number of channels (C=1 for a grayscale image, C=3 for a RGB image). The encoder architecture is depicted in Figure 1b. It corresponds to the encoder proposed in [9], to which the number of channels has been modified going from 16-256 up to 32-512. It is made up of a succession of Convolution Blocks (CB) and Depthwise Convolution Blocks (DSCB).

CB is defined as two convolutional layers followed by instance normalization and a third convolutional layer. This third convolutional layer has a stride of 1×1 for CB_1, 2×2 for CB_2 to CB_4 and 2×1 for CB_5 to CB_6. The strides are chosen to progressively decrease the dimensions. It is a trade-off between memory consumption and performance.

DSCB follows the same structure as CB but the convolutional layers are superseded by Depthwise Separable Convolutions [5] in order to reduce the number of parameters at stake. Moreover, the third DSC has always a stride of 1×1. This enables to introduce residual connections with element-wise sum operator between the DSCB.

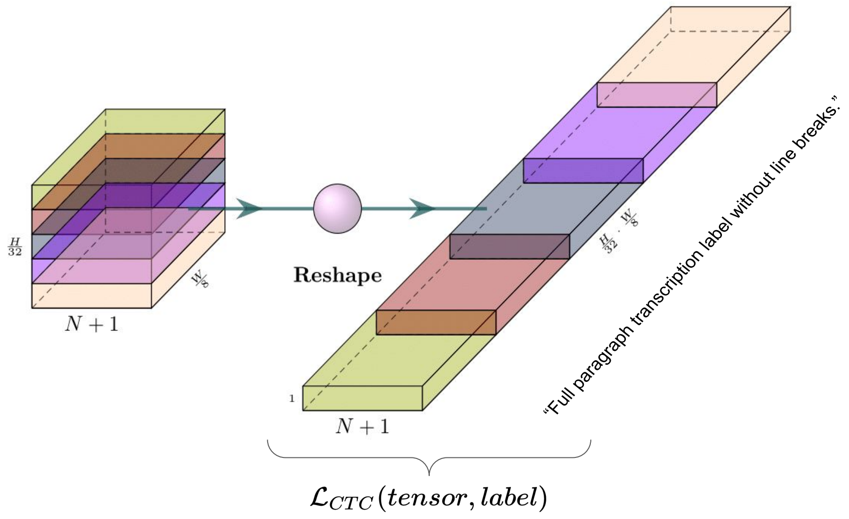
For both blocks, convolutional layers have a 3×3 kernel, 1×1 padding and are followed by ReLU activations. In addition, Diffused Mix Dropout (DMD)[9] is used with three potential locations inside each block to reduce overfiting.

3.2 Decoder

The decoder aims at predicting and aligning the probabilities of the characters and the CTC blank label for each 2D position of the features f. The decoder is made up of a single convolutional layer with kernel 5×5, stride 1×1 and padding 2×2. It outputs $N + 1$ channels, N being the size of the charset. Finally, the $\frac{H}{32}$ rows are concatenated to obtain the one-dimensional prediction sequence $p \in \mathbb{R}^{(\frac{H}{32} \cdot \frac{W}{8}) \times (N+1)}$ as depicted in Figure 2. The CTC loss is then computed between this one-dimensional prediction sequence and the paragraph transcription ground truth, without line breaks.

We can highlight some important aspects about the decoder:

- In this work, the CTC blank label has a new function. Indeed, in standard OCR applied to text lines, the CTC blank label enables to recognize two identical successive characters and to predict "nothing", acting like a joker. Here, it is also used to separate lines in a two-dimensional context, as it allows to label line spacing in the input image.
- One should notice that the prediction occurs before reshaping to 1D, which allows to take advantage of the two-dimensional context in the decision layer. This enables to localize the previous and next lines, and to align the predictions of the same text line on the same row *i.e.*, and to separate it from the other text line predictions.



$\frac{H}{32}$
$\frac{W}{8}$
$N+1$
Reshape
1
$N+1$
$\frac{H}{32} \cdot \frac{W}{8}$
"Full paragraph transcription label without line breaks."
$\mathcal{L}_{CTC}(tensor, label)$

Fig. 2: Reshape operation and loss visualization. No computations are performed in the reshape operation, both left and right tensors represent characters and CTC blank label probabilities. The CTC loss is computed between the one-dimensional probabilities sequence and the paragraph transcription.

– Since the prediction rows are concatenated, they are processed sequentially; nothing prevents the model from predicting the beginning of the text line on one row and the end on the next one as long as there is enough space between this text line and the following one. In Section 4.7, we show that this allows us to process inclined lines.

4 Experimental study

4.1 Datasets

We evaluate our model on three popular datasets at paragraph level: RIMES [11], IAM [13] and READ 2016 [19].

RIMES We used the RIMES dataset which is made up of French handwritten paragraphs, produced in the context of writing mails scenarios. The images are gray-scaled and have a resolution of 300 dpi. In the official split, 1,500 paragraphs are dedicated to training and 100 paragraphs to evaluation. The last 100 training images are used for validation so as to be comparable with the state of the art.

IAM The IAM dataset corresponds to handwritten copy of English text passages extracted from the LOB corpus. The images are gray-scaled handwritten paragraph with a resolution of 300 dpi. In this work, we used the unofficial but commonly used split as detailed in Table 1.

READ 2016 The READ 2016 dataset corresponds to Early Modern German handwriting. It has been proposed in the ICFHR 2016 competition on handwritten text recognition. It is a subset of the Ratsprotokolle collection, used in the READ project. Images are in color and we used the paragraph level segmentation. We assume that the images have a resolution of around 300 dpi too.

In section 4, some experiments implies pretraining using the line level images of these three datasets. The corresponding splits are shown in Table 1.

Table 1: Datasets split in training, validation and test sets and associated charset size

Dataset	Level	Training	Validation	Test	Charset size
RIMES	Line	10,532	801	778	100
	Paragraph	1,400	100	100	
IAM	Line	6,482	976	2,915	79
	Paragraph	747	116	336	
READ 2016	Line	8,349	1,040	1,138	89
	Paragraph	1,584	179	197	

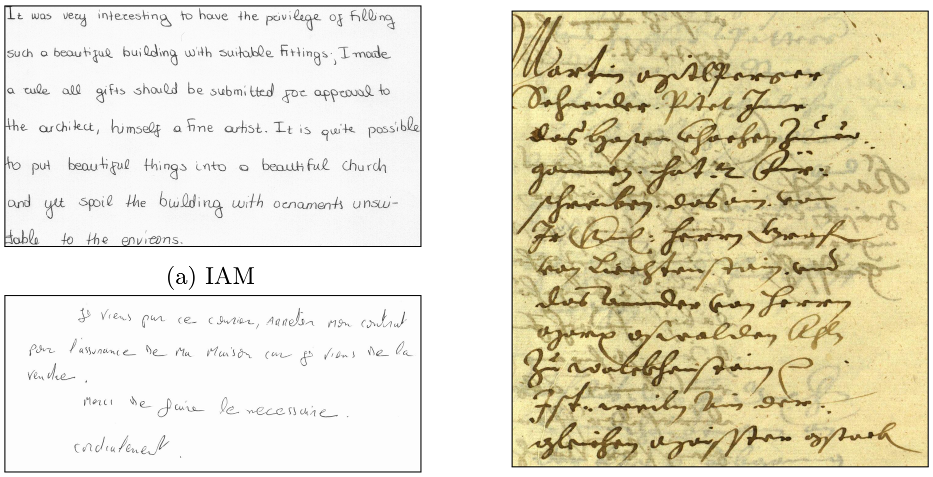
Paragraph image examples from these three datasets are depicted in Figure 3. IAM layout is the more structured and regular. RIMES brings some irregularities in terms of line spacing, text inclination and horizontal text alignment. Finally, the READ 2016 dataset is more complex in terms of noise, text line separation (due to ascents and descents) and size variety.

4.2 Preprocessing

Paragraph images are downscaled by a factor of 2 through a bilinear interpolation leading to a resolution of 150 dpi. Gray-scaled images are converted into RGB images concatenating the same values three times, for transfer learning purposes. They are then normalized (zero mean and unit variance) considering the channels independently.

4.3 Data Augmentation

Data augmentation is applied at training time to reduce over-fitting. The augmentation techniques are used in this order: resolution modification, perspective transformation, elastic distortion and random projective transformation (from [22]), dilation and erosion, brightness, and contrast adjustment and sign flipping. Each transformation has a probability of 0.2 to be applied. Except for perspective transformation, elastic distortion and random projective transformation which are mutually exclusive, each augmentation technique can be combined with the others.


It was very interesting to have the povilege of filling
such a beautiful building with suitable fittings; I made
a rule all gifts should be submitted for approval to
the architect, himself a fine artist. It is quite possible
to put beautiful things into a beautiful church
and yet spoil the building with ornaments unsui-
table to the environs.

(a) IAM

Je viens par ce courrier, arreter mon contrat
pour l'assurance de ma maison car je viens de la
vendre.
Merci de faire le necessaire.
cordialement.

(b) RIMES

[illegible]

(c) READ 2016

Fig. 3: Paragraph image examples from the RIMES, IAM and READ 2016 datasets.

4.4 Metrics

The Character Error Rate (CER) and the Word Error Rate (WER) are used to evaluate the quality of the text recognition. They are both computed with the Levenshtein distance between the ground truth text and the predicted text at paragraph level, without line breaks. Those edit distances are then normalized by the length of the ground truth. Other metrics are provided in the following experiments such as the number of parameters implied by the models.

4.5 Training details

We used the Pytorch framework to train and evaluate our models. In all experiments, the networks are trained with the Adam optimizer, with an initial learning rate of 10^{-4}. Trainings are performed on a single GPU Tesla V100 (32Gb), during 2 days, with a mini-batch size of 4 for paragraph images and 16 for text lines images. We used the original paragraph ground truth replacing line breaks by space characters.

4.6 Additional information

We do not use any post-processing *i.e.* we do not use any language model nor lexicon constraint. Moreover, we only use best path decoding to get the final predictions from the character probabilities lattice. We use exactly the same training configuration from one dataset to another, without model modification, except for the last layer which depends on the charset size.

4.7 Results

Comparison with state of the art In this section, we compare our approach to state-of-the-art models on the RIMES, IAM and READ 2016 datasets, at paragraph level and in the same conditions *i.e.* without language model nor lexicon constraint.

Prior to compare the obtained results to the state of the art, it is important to understand the experimental conditions of each of the methods. Table 2 shows model details that should be taken into account to fairly compare the following tables of results. Quantitative metrics are computed for the IAM dataset, without automatic mixed precision (for a fair comparison with respect to the memory usage). From left to right, the columns respectively denote the architecture, the number of trainable parameters, the maximum GPU memory usage during training (for a mini-batch size of 1, data augmentation included), the minimum transcription level required, the minimum segmentation level required, the use of PreTraining (PT) on subimages, the use of specific Curriculum Learning (CL) and finally the Hyperparameter Adaptation (HA) requirements from one dataset to another.

Table 2: Requirements comparison of the SPAN with the state-of-the-art approaches.

Architecture	# Param.	Max memory	Transcription label	Seg. label	PT	CL	HA
[4] RPN+CNN+BLSTM			Word	Word			
[6] RPN+CNN+BLSTM			Word	Word			
[21] RPN+CNN+BLSTM			Line	Line			
[9] FCN+LSTM*	2.7 M	2.2 Gb	Paragraph + line breaks	Paragraph	Line	✗	✗
[2] CNN+MDLSTM**			Paragraph	Paragraph	Line	✓	✗
[1] CNN+MDLSTM*			Paragraph	Paragraph	Line	✓	✗
[22] GFCN	16.4 M	8.8 Gb	Paragraph	Paragraph	✗	✗	✓
[This work - SPAN] FCN	19.2 M	5.1 Gb	Paragraph	Paragraph	Line	✗	✗

* with line-level attention
** with character-level attention

As one can see, models from [4,6,21] require transcription and segmentation labels at word or line levels to be trained, which implies more costly annotations. The models from [1,2,9] and the SPAN are pretrained on text line images to speed up convergence and to reach better results, thus also using line segmentation and transcription labels even if it is not strictly necessary. While the model from [9] needs line breaks in the transcription annotation, [1,2] used a specific curriculum learning method for training. In [22], some hyperparameters must be modified from one dataset to another in order to reach optimal performance, namely the fixed input dimension and two intermediate upsampling sizes which are crucial. We do not have such problem since we are working with input images of variable size and we focus on the resolution to be robust to the variety of datasets. Moreover, despite a larger number of parameters (+ 17% compared to [22]), the SPAN requires less GPU memory which is a critical point when training deep neural networks.

Table 3, 4 and 5 show the results of the SPAN compared to the state-of-the-art approaches, for the RIMES, IAM and READ 2016 datasets respectively. One can notice that we reach competitive results on those three datasets, each having its own complexities, without any hyperparameter adaptation. Results here includes model pretraining on line images but the model can be trained without pretraining *i.e.* without using any line-level annotation, while keeping competitive results, as shown in Section 4.7.

It has to be noted that the SPAN encoder extracts character representations and aligns them vertically as well, also recognizing interlines. This additional task explains the difference with the results of the other works, notably the VAN [9] which uses a similar encoder. Moreover, the SPAN aligns the prediction with the ground truth at paragraph level whereas the VAN uses the line breaks to make this alignment at line level.

Table 3: Comparison of the SPAN results with the state-of-the-art approaches at the paragraph level on the RIMES dataset.

Architecture	CER (%) validation	WER (%) validation	CER (%) test	WER (%) test
[1]	2.5	12.0	2.9	12.6
[21]			2.1	9.3
[9]	**1.74**	**8.72**	**1.90**	**8.83**
This work - SPAN	3.56	14.29	4.17	15.61

Table 4: Comparison of the SPAN results with the state-of-the-art approaches at the paragraph level on the IAM dataset.

Architecture	CER (%) validation	WER (%) validation	CER (%) test	WER (%) test
[4]*	13.8		15.6	
[6]			8.5	
[21]			6.4	23.2
[9]	**3.04**	**12.69**	**4.32**	**16.24**
[2]			16.2	
[1]	4.9	17.1	7.9	24.6
[22]			4.7	
This work - SPAN	3.57	15.31	5.45	19.83

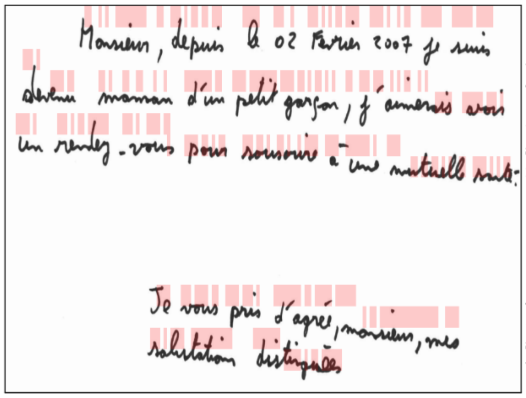
*Results are given for page level

SPAN prediction visualization Figure 4 presents a visualization of the SPAN prediction for an example of the RIMES test set. Character predictions are shown in red; they seem like rectangle since they are resized to fit the input image

Table 5: Comparison of the SPAN results with the state-of-the-art approaches at the paragraph level on the READ 2016 dataset.

Architecture	CER (%) validation	WER (%) validation	CER (%) test	WER (%) test
[9]	**3.75**	**18.61**	**3.63**	**16.75**
This work - SPAN	5.09	23.69	6.20	25.69

size (the features size is $\frac{H}{32} \times \frac{W}{8}$). Combined with the receptive field effect, this explains the shift that can occur between the prediction and the text. As one can notice, text line predictions are totally aligned, or aligned by blocks; the lines are well separated by blank labels, which act as line spacing labels. As one can see, this block alignment enables to handle downward inclined lines, especially for lines 3 and 4. Moreover, the model does not degrade in the presence of large line spacing.



Monsieur, depuis la 02 Ferries 2007 je suis

de
enu maman d'un petit garçon, j'aimer
ais avoir
un rendez-vous
s pour sousoire à une m
ntuelle sont

Je vous pris d'agrée, m
ansieurs, mes
salutation dis
tinguées

Monsieur, depuis l**a** 02 Fe**r**rie**s** 2007 je suis de***v***enu maman d'un petit garçon, j'aimerais avoir un rendez-vous**s** pour sous**o**ire à une m**n**tuelle s**o**nt***é.*** Je vous pris d'agrée, m**a**nsieurs, mes salutation***s*** distinguées

Fig. 4: SPAN predictions visualization for a RIMES test example. Left: 2D characters predictions are projected on the input image. Red color indicates a character prediction while transparency means blank label prediction. Right: row by row text prediction. Bottom: full text prediction where errors are shown in bold and missing letters are shown in italic.

Impact of pretraining In this experiment, we try to highlight the impact of pretraining on the SPAN results. To this end, we compare two pretraining methods at line level: one focusing only on the optical recognition task and the second one focusing on both recognition and prediction alignment. Let's define the following training approaches:

- SPAN-Line-R&A: the SPAN is trained with line-level images. Here, the network has to learn both the recognition and the alignment tasks.
- Pool-Line-R: a new model is trained with line-level images to only focus on the recognition task. This network consists in the previous defined SPAN encoder followed by an Adaptive MaxPooling to collapse the vertical axis; then, a convolutional layer predicts the characters and blank label probabilities. This is the standard way to process text line images, as in [7]. Since the prediction is already in one dimension, the network does not need to care about vertical alignment.
- SPAN-Scratch: the SPAN is trained directly on paragraph images without pretraining.
- SPAN-PT-R: SPAN weights are initialized with Pool-Line-R ones. It is then trained with paragraph images.
- SPAN-PT-R&A: SPAN weights are initialized with SPAN-Line-R&A ones. It is then trained with paragraph images.

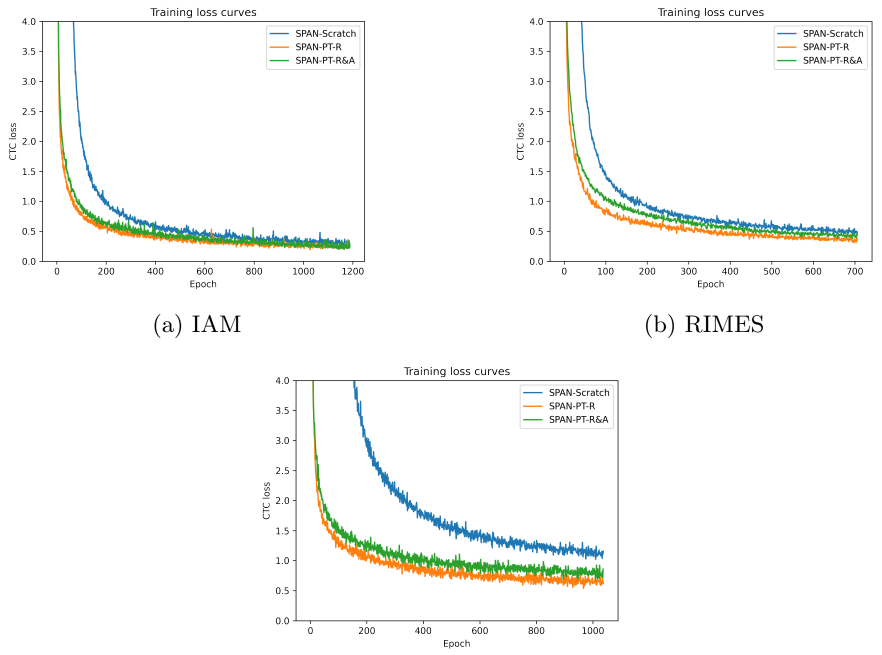
One can note that the vertical receptive field is bigger than line image heights. Thus, when the model switches from line images to paragraph images, the decision benefits from more context, which replaces part of the previously used padding.

Results are given in Table 6. Focusing on the line-level section, one can notice that, as expected, we reached better results on text lines when the task is reduced to optical recognition compared to the task of recognition and alignment, whatever the dataset. This leads to a CER improvement of 0.94 point for IAM, 0.79 point for RIMES and 0.38 point for READ 2016. Now, comparing the paragraph level approaches, one can notice that, except for the RIMES CER, pretraining leads to better results, and sometimes by far (-2.93 points of CER for READ 2016); moreover, pretraining on an easier task, *i.e.* only on the optical recognition, is even more efficient.

Table 6: Impact of pretraining the SPAN on line images for the IAM, RIMES and READ 2016 datasets. Results are given on the test sets.

Approach	IAM		RIMES		READ 2016	
	CER (%)	WER (%)	CER (%)	WER (%)	CER (%)	WER (%)
Line-level training						
Pool-Line-R	**4.82**	**18.17**	**3.02**	**10.73**	**4.56**	**21.07**
SPAN-Line-R&A	5.76	21.33	3.81	13.80	4.94	22.19
Paragraph-level training						
SPAN-Scratch	6.46	23.75	**4.15**	16.31	9.13	36.63
SPAN-PT-R	**5.45**	**19.83**	4.74	**15.55**	**6.20**	**25.69**
SPAN-PT-R&A	5.78	21.16	4.17	15.71	6.62	27.38

The RIMES CER value can be explained by the difference between the CTC loss and the Levenshtein distance which are not the same. As a matter of fact, generally, a lower CTC loss implies a lower CER but it is not always true. Indeed, Figure 5 shows the different loss CTC training curves for the three datasets. This time, we can clearly see that, even for RIMES, pretraining on the recognition task only is more beneficial. This figure also demonstrates the convergence speed up brought by these pretraining approaches.



(a) IAM

(b) RIMES

(c) READ 2016

Fig. 5: Training curves comparison between the different pretraining approaches, on the RIMES, IAM and READ 2016 datasets.

Discussion As we have seen in Figure 4, the 2D prediction keeps the spatial information. As such, we can assume that the SPAN could be used as a primary stage of a deeper end-to-end network that could handle more complex tasks such as word spotting in handwritten digitized document. Moreover, since we are using the standard CTC loss, one can easily add standard character or word language model to further improve the results. The SPAN could also adapt to full page documents since it is based on the image resolution regardless of its size. However, it has to be noticed that this model is limited to single-column multi-line text images due to the row concatenation operation. Moreover, the SPAN can easily handle downward sloping lines but cannot handle upward ones due to the fixed reshaping order.

5 Conclusion

In this paper, we proposed the Simple Predict & Align Network, an end-to-end recurrence-free segmentation-free FCN model performing OCR at paragraph level. It reaches competitive results on the RIMES, IAM and READ 2016 datasets without any model architecture or training adaptation from one to another. It follows a new training approach bringing several other advantages. First, it only needs transcription label at paragraph level (without line breaks), leveraging the need for handmade annotation, which is a critical point for a deep learning system. Second, training this model is as simple as training a line-level OCR with the CTC loss. Finally, it can handle variable image input sizes, making it robust enough to adapt to multiple datasets; it is also able to deal with downward inclined text lines.

Acknowledgments The present work was performed using computing resources of CRIANN (Normandy, France) and HPC resources from GENCI-IDRIS (Grant 2020-AD011012155). This work was financially supported by the French Defense Innovation Agency and by the Normandy region.

References

1. Bluche, T.: Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In: Advances in Neural Information Processing Systems 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: ICDAR. pp. 1050–1055 (2017)
3. Carbonell, M., Fornés, A., Villegas, M., Lladós, J.: A neural model for text localization, transcription and named entity recognition in full pages. Pattern Recognit. Lett. **136**, 219–227 (2020)
4. Carbonell, M., Mas, J., Villegas, M., Fornés, A., Lladós, J.: End-to-end handwritten text detection and transcription in full pages. In: Workshop on Machine Learning, ICDAR. pp. 29–34 (2019)

5. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. CVPR (2017)
6. Chung, J., Delteil, T.: A computationally efficient pipeline approach to full page offline handwritten text recognition. In: Workshop on Machine Learning, ICDAR. pp. 35–40 (2019)
7. Coquenet, D., Chatelain, C., Paquet, T.: Recurrence-free unconstrained handwritten text recognition using gated fully convolutional network. In: ICFHR. pp. 19–24 (2020)
8. Coquenet, D., Soullard, Y., Chatelain, C., Paquet, T.: Have convolutions already made recurrence obsolete for unconstrained handwritten text recognition ? In: Workshop on Machine Learning, ICDAR. pp. 65–70 (2019)
9. Coquenet, D., Chatelain, C., Paquet, T.: End-to-end handwritten paragraph text recognition using a vertical attention network (2020)
10. Graves, A., Fernández, S., Gomez, F.J., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: ICML. vol. 148, pp. 369–376 (2006)
11. Grosicki, E., El Abed, H.: Icdar 2011-french handwriting recognition competition. pp. 1459–1463 (2011)
12. Grüning, T., Leifert, G., Strauß, T., Michael, J., Labahn, R.: A two-stage method for text line detection in historical documents. Int. J. Document Anal. Recognit. **22**(3), 285–302 (2019)
13. Marti, U.V., Bunke, H.: The iam-database: An english sentence database for offline handwriting recognition. Int. J. Document Anal. Recognit. **5**, 39–46 (2002)
14. Michael, J., Labahn, R., Grüning, T., Zöllner, J.: Evaluating sequence-to-sequence models for handwritten text recognition. In: ICDAR. pp. 1286–1293 (2019)
15. Moysset, B., Kermorvant, C., Wolf, C.: Full-page text recognition: Learning where to start and when to stop. In: ICDAR. pp. 871–876 (2017)
16. Oliveira, S.A., Seguin, B., Kaplan, F.: dhsegment: A generic deep-learning approach for document segmentation. In: ICFHR. pp. 7–12 (2018)
17. 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. Document Anal. Recognit. **21**(3), 177–186 (2018)
18. Schall, M., Schambach, M., Franz, M.O.: Multi-dimensional connectionist classification: Reading text in one step. In: 13th International Workshop on Document Analysis Systems. pp. 405–410 (2018)
19. Sánchez, J.A., Romero, V., Toselli, A., Vidal, E.: Icfhr2016 competition on handwritten text recognition on the read dataset. pp. 630–635 (2016)
20. Tensmeyer, C., Wigington, C.: Training full-page handwritten text recognition models without annotated line breaks. In: ICDAR. pp. 1–8 (2019)
21. Wigington, C., Tensmeyer, C., Davis, B.L., Barrett, W.A., Price, B.L., Cohen, S.: Start, follow, read: End-to-end full-page handwriting recognition. In: ECCV. Lecture Notes in Computer Science, vol. 11210, pp. 372–388 (2018)
22. Yousef, M., Bishop, T.E.: Origaminet: Weakly-supervised, segmentation-free, one-step, full page text recognition by learning to unfold. In: CVPR. pp. 14698–14707 (2020)
23. Yousef, M., Hussain, K.F., Mohammed, U.S.: Accurate, data-efficient, unconstrained text recognition with convolutional neural networks. Pattern Recognit. **108**, 107482 (2020)