

This is the peer reviewed version of the following article:

Boosting Modern and Historical Handwritten Text Recognition with Deformable Convolutions / Cascianelli, Silvia; Cornia, Marcella; Baraldi, Lorenzo; Cucchiara, Rita. - In: INTERNATIONAL JOURNAL ON DOCUMENT ANALYSIS AND RECOGNITION. - ISSN 1433-2833. - 25:3(2022), pp. 207-217. [10.1007/s10032-022-00401-y]

*Terms of use:*

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

16/05/2024 12:00

(Article begins on next page)

# Boosting Modern and Historical Handwritten Text Recognition with Deformable Convolutions

Silvia Cascianelli, Marcella Cornia, Lorenzo Baraldi and Rita Cucchiara

Department of Engineering “Enzo Ferrari”, University of Modena and Reggio Emilia,  
Via Pietro Vivarelli, 10, Modena, 41125, Italy.

\*Corresponding author(s). E-mail(s): [silvia.cascianelli@unimore.it](mailto:silvia.cascianelli@unimore.it);

Contributing authors: [marcella.cornia@unimore.it](mailto:marcella.cornia@unimore.it); [lorenzo.baraldi@unimore.it](mailto:lorenzo.baraldi@unimore.it);  
[rita.cucchiara@unimore.it](mailto:rita.cucchiara@unimore.it);

## Abstract

Handwritten Text Recognition (HTR) in free-layout pages is a challenging image understanding task that can provide a relevant boost to the digitization of handwritten documents and reuse of their content. The task becomes even more challenging when dealing with historical documents due to the variability of the writing style and degradation of the page quality. State-of-the-art HTR approaches typically couple recurrent structures for sequence modeling with Convolutional Neural Networks for visual feature extraction. Since convolutional kernels are defined on fixed grids and focus on all input pixels independently while moving over the input image, this strategy disregards the fact that handwritten characters can vary in shape, scale, and orientation even within the same document and that the ink pixels are more relevant than the background ones. To cope with these specific HTR difficulties, we propose to adopt deformable convolutions, which can deform depending on the input at hand and better adapt to the geometric variations of the text. We design two deformable architectures and conduct extensive experiments on both modern and historical datasets. Experimental results confirm the suitability of deformable convolutions for the HTR task.

**Keywords:** Handwritten Text Recognition, Deformable Convolutions, Historical Manuscripts

## 1 Introduction

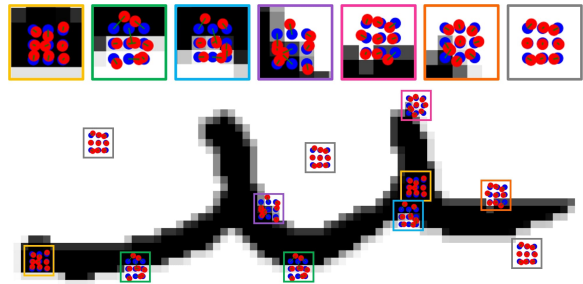
Handwritten Text Recognition (HTR) aims at automatically understanding the content of a handwritten document by providing a natural language transcription of its textual content. Because of the key role it can play in the automatization of the digitization of documents, it can be applied in automated services and any modern document processing pipeline. On the other hand, HTR is also applied in the field of Digital Humanities [20] for the transcription of historical documents and to enable search and retrieval

applications in ancient corpora, which would not be easily accessible otherwise [42].

Despite the advancements in Optical Character Recognition (OCR), HTR remains a challenging task. The task, originally tackled via Hidden Markov Models built upon heuristic visual features [26, 31], is currently performed with Deep Neural Network-based approaches [38, 39, 43]. The most common strategy entails employing a Convolutional Neural Network (ConvNet) and a Recurrent Neural Network (RNN) to represent the

input manuscript image and a Connectionist Temporal Classifier (CTC) to generate the output text sequence [38]. The visual feature extraction phase of these approaches relies on standard convolutional layers in which features from the input image are extracted by sliding kernels with a fixed regular grid and constant parameters. As a consequence of the constancy of convolutional weights, pixels in an image neighborhood are encoded according to their relative positions rather than the content of the neighborhood itself. This can be sub-optimal if considering the characteristics of HTR images, which contain handwritten characters and words. In fact, in these images, only the ink pixels are relevant for recognition, while the background ones can potentially contain misleading nuisances, especially in ancient documents. Moreover, handwritten documents typically feature highly varying characters, which cannot be effectively modeled with standard fixed-shape convolution kernels without performing ad hoc pre-processing and data augmentation.

Motivated by the above considerations, in this paper, we propose to employ deformable convolution operators [18] in HTR architectures. In deformable convolutions, the regular kernel grid is altered by adding a translation vector to the location of each kernel element. Translations vectors are computed in a content-dependent manner so that the kernel grid can be deformed depending on the input of the convolutional layer itself. As a consequence, deformable convolutions can adapt to the input geometric variations and part deformations, making them potentially more suitable for dealing with HTR images compared to standard convolutions. This kind of convolution has been originally proposed to tackle the object recognition task and, to the best of our knowledge, its usage in HTR has been explored only in our preliminary work [16], where we claim that its kernel adaptability (see Figure 1) can help to improve the efficiency and the performance in the task. In this work, we deepen our analysis and extend it to HTR on historical manuscripts. To demonstrate the effectiveness of using deformable kernels, we design two different deformable architectures and conduct extensive experiments on both modern and historical datasets for handwritten text recognition. Experimental results will demonstrate that deformable convolutions are a suitable operator



**Fig. 1:** Sampling grid of a standard convolutional kernel (in blue) and a deformable convolutional kernel (in red) when applied over a handwritten character. Deformable convolutions apply translation vectors to the kernel grid and adapt better to handwritten strokes (best seen in color).

for HTR networks and provide consistent improvements over standard convolutions.

The rest of this paper is organized as follows. In Section 2, we provide a brief overview of the HTR literature. Then, in Section 3, we describe the proposed approach and how deformable convolutions are applied to existing HTR networks. Finally, in Section 4, we outline our experimental setup and present qualitative and quantitative results.

## 2 Related Work

HTR can be tackled by considering different textual elements, *i.e.* characters [14, 15, 22], words [5, 21, 43, 45], lines [7, 8, 13, 34, 36, 38, 50], paragraphs [6, 7, 53], or pages [15, 34]. Line-level HTR is among the most popular variants, which can be performed on pre-segmented text [8, 13, 36, 38, 50], or used in combination with layout analysis and line-level segmentation to obtain a paragraph-level or page-level HTR system [7, 34, 54]. In this paper, we focus on pre-segmented line-level HTR.

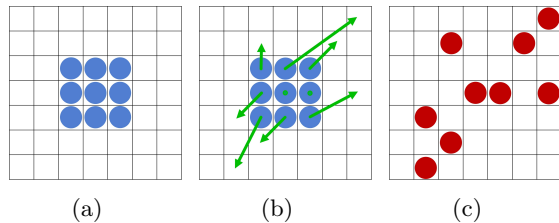
Originally, HTR was tackled by applying Hidden Markov Models for image representation, and  $n$ -gram based language models for textual output prediction [47, 48]. The first deep learning-based solution to HTR was proposed in [21], where multi-dimensional Long Short-Term Memory networks (MDLSTM-RNNs) are used to build a 2D representation of the textual image, which is then collapsed in a sequence of vectors used for CTC decoding. This strategy was the standard one [6, 19, 35] until simpler alternatives to MDLSTM-RNNs were proposed [38, 43]. These consist of a

ConvNet to extract a sequence of feature vectors from the text image and 1D-LSTMs to output character probabilities for the CTC decoding and became commonly used as a backbone for HTR systems [8, 13, 16, 36, 50] due to its performance and faster training compared to MDLSTMs.

A recent trend entails treating HTR as a sequence-to-sequence problem, from a sequence of text image slices encoded via convolutional and recurrent blocks to a sequence of transcribed text generated by a separate recurrent block [1, 33, 46, 56]. Networks implementing this approach can be trained either by optimizing the cross-entropy loss alone or combined with the CTC loss. A variant of this approach entails using Transformers [49] in place of RNNs [17, 54], often requiring pre-training on either real or synthetic data [25, 27, 51] to obtain performance comparable to RNN-based solutions. Finally, it is worth mentioning approaches that avoid using RNNs and are instead fully-convolutional [17, 54]. In these approaches, convolutional layers are combined with GateBlocks layers [55] that operate a selection mechanism to model dependency similarly to LSTM cells.

In the above-mentioned variants, recognition performance can be increased by integrating a language model, either a word-level or a character-level one. However, this strategy is effective only for those languages for which sufficient textual data are available. Thus, it is not always feasible for historical documents that are usually written in underrepresented languages. These can either be ancient versions of a modern language that has evolved over time or a language which is no longer spoken at all. In this work, we do not use any language model, both to assess the performance of the proposed HTR models in historical documents and to better enhance the benefits of deformable convolutions over standard convolutions.

Compared to OCR, HTR features the challenge related to the high variability of characters in shape and size. A common strategy is performing specific data augmentation and preprocessing [4, 9, 23, 28, 37, 38, 50, 52], while few works have faced this issue at the architectural design level. For example, in [57] a Spatial Transformer Network [22] is employed for character-level HTR, while in [5] an adversarial deformation module is used to warp intermediate convolutional features in a word-level HTR model. In this paper,



**Fig. 2:** The regular sampling grid of a standard convolutional kernel (a) is deformed by applying a set of offsets (b), obtaining the deformed sampling grid of the deformable convolution (c).

we propose to apply deformable convolutional kernels in the convolutional part of HTR models with the aim to tackle characters non-idealities without relying on data augmentation or specific preprocessing.

## 3 Proposed Method

In this section, we introduce our proposed approach. We first review deformable convolutions, and then introduce the convolutional-recurrent architectures we employ.

### 3.1 Deformable Convolutions

ConvNets owe their success to their representational power and capacity to extract position-independent local features from images. Their key element is the convolutional operator  $*$ , which performs a learned weighted sum of elements sampled over a regular grid  $\mathcal{N}$ . Formally, for a pixel  $p$  of an input feature map  $\mathbf{I}$ , given a kernel  $\mathbf{k}$  of learnable weights, the convolutional operator can be defined as follows:

$$(\mathbf{I} * \mathbf{k})(p) = \sum_{d \in \mathcal{N}} \mathbf{k}(d) \cdot \mathbf{I}(p + d), \quad (1)$$

where  $\cdot$  is the inner product between channel-wise feature vectors and  $d$  is a displacement vector. The size and structure of the sampling grid  $\mathcal{N}$  depend on the kernel size and dilation.

Conversely, the deformable convolution operator  $\circledast$  [18] relies on an irregular sampling grid. The shape and geometry of the grid is learned as a function of the processed input context, which allows a content-dependent, non-regular feature extraction. To obtain the deformed grid, a regular

sampling grid, such as that of a standard convolutional operator, is deformed by adding a learned 2D offset to each of its elements (as depicted in Figure 2).

In practice, a deformable convolution layer is obtained by applying two standard convolutional layers: one is in charge of computing the offsets, the other of computing the kernel weights. Both are applied over the input feature map to ensure content-dependent deformation. Formally, for deformable convolutions Eq. 1 becomes

$$(\mathbf{I} \circledast \mathbf{k})(p) = \sum_{d \in \mathcal{N}} \mathbf{k}(d) \cdot \mathbf{I}(p + d + \delta(d)). \quad (2)$$

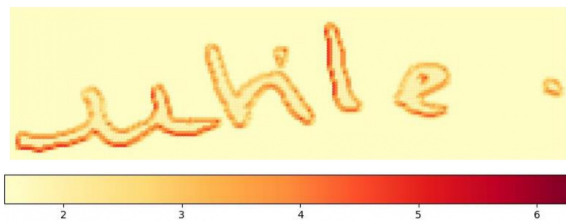
Thus, the points included in the deformed kernel are those in the set  $\{d + \delta(d)\}_{d \in \mathcal{N}}$ . Note that, in general, the computed offsets can be fractional numbers – and simply quantizing them would harm the training phase. To overcome this limitation, Eq. 2 is implemented by including a 2D bilinear interpolation kernel  $B(\cdot, \cdot)$ , *i.e.*

$$(\mathbf{I} \circledast \mathbf{k})(p) = \sum_{d \in \mathcal{N}} \mathbf{k}(d) \cdot \sum_{s \in \mathcal{S}} B(s, p + d + \delta(d)) \cdot \mathbf{I}(s), \quad (3)$$

where  $\mathcal{S}$  is the set of points in  $\mathbf{I}$  that are in the neighborhood of the sampling locations  $\{p + d + \delta(d)\}_{d \in \mathcal{N}}$ .

To appreciate the effects of the deformable convolution on handwriting images, in Figure 1 we represent some deformed sampling grids obtained from a  $3 \times 3$  deformable kernel applied to the image of a character, in comparison with those of a standard convolutional kernel. It can be noticed that when the kernel is applied to stroke edges it is considerably deformed to adapt to its input. On the other hand, on uniform regions such as background and solid ink, the deformation is less evident. The same behavior can be observed from Figure 3, which represents the cumulative magnitudes of the offsets of a  $3 \times 3$  deformable kernel applied to all the pixels of a short text line. When processing pixels from uniform areas, the learned offsets are small, thus the kernel does not deform much, while are larger when processing pixels on the edges.

Given a deformable convolutional layer with square kernels, kernel size  $k$ ,  $c_{in}$  input channels, and  $c_{out}$  output channels, the number of weights



**Fig. 3:** Cumulative magnitude of the offsets applied to a  $3 \times 3$  kernel grid on points of an image of a word (first convolutional layer of CRNN). Grids sampling in uniform regions are less deformed than those sampling on edges.

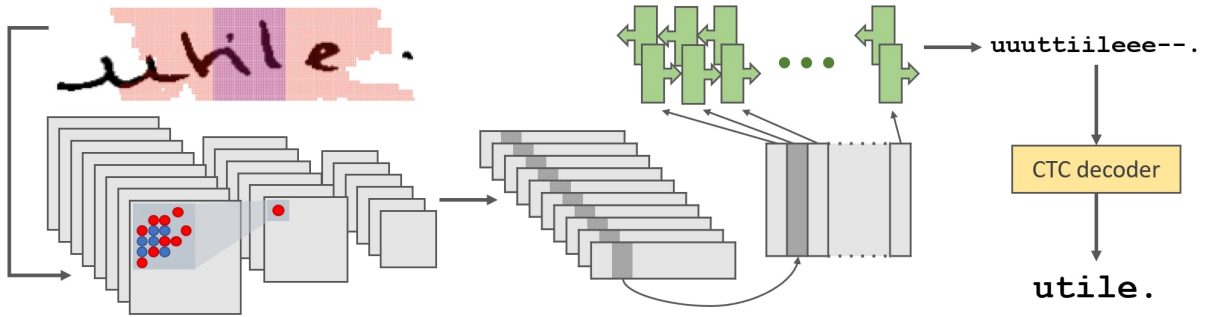
needed for computing the deformable convolution is  $k^2 \cdot c_{in} \cdot c_{out}$ <sup>1</sup>, which is comparable to those needed by a standard convolutional layer. If the convolutional layer computing the offsets is also created with the same kernel size, it will add  $k^2 \cdot c_{in} \cdot (2k^2)$  parameters, for a total of  $k^2 \cdot c_{in} \cdot (2k^2 + c_{out})$  weights. From the point of view of the memory footprint, therefore, replacing a standard convolutional layer with a deformable one implies the same cost of adding  $2k^2$  output feature maps.

### 3.2 Deformable Convolutional-Recurrent Network

Given the capability of deformable convolutions to adapt to geometric transformations, we propose to employ them instead of regular convolutions in HTR architectures. To this end, we build upon two commonly used backbone models for HTR: the sequence recognition network proposed in [43] and the model presented in [38]. In the following, we refer to these models as CRNN and 1D-LSTM, respectively.

Both models take as input an image representing a row of handwritten text and consist of three main components: a ConvNet to extract visual features from the input image, an RNN which treats the visual feature map as a sequence and outputs character probabilities, and a decoding block to output the final transcription. For training the models, we maximize the CTC probability of the output sequence. Thus, additional to textual

<sup>1</sup>For the sake of simplicity, we do not include biases in this analysis.



**Fig. 4:** General scheme of a convolutional-recurrent HTR system featuring deformable convolutions.

characters, the RNN scores a special *blank* character that means “no other character”. An overview of the main components of these HTR systems is shown in Figure 4.

The convolutional part of CRNN has the same architecture as of VGG-11 [44] up to the sixth convolutional block, to which we add another convolutional block with a  $2 \times 2$  kernel. Additionally, the receptive field of the 3<sup>rd</sup> and 4<sup>th</sup> max-pooling layers are changed from squared  $2 \times 2$  into rectangular  $2 \times 1$  in order to obtain wider feature maps reflecting the common height-width ratio of images of text lines. Note that in this architecture, all the convolutional layers are deformable.

For the convolutional part of 1D-LSTM, we stack five blocks containing a deformable convolution layer with  $3 \times 3$  kernels, a Batch Normalization layer, and a LeakyReLU activation function. The deformable convolution layer of the first block has 16 filters, and for the others, we increase the number of filters by 16 at each block. We apply  $2 \times 2$  max-pooling to the output of the first three blocks and leave the output of the last two blocks as it is.

In both the proposed variants, the  $H \times W \times C$  feature map of the last convolutional layer is used to obtain a sequence of  $W$  ( $H \cdot C$ )-elements feature vectors that serve as input for the recurrent part. These feature vectors are obtained by concatenating the  $C$ -dimensional vectors on the  $H$  rows of the map and represent regions of the original image, *i.e.* the receptive field. Due to the deformable kernels, such receptive fields have irregular shape and can possibly be non-connected, which allows them to cover a wider portion of the input and better

adapt to its patterns (see some examples reported in Figure 5).

For the recurrent part of both the CRNN and 1D-LSTM variants we use Bidirectional LSTMs (BLSTMs). We stack two BLSTMs layers in the CRNN variant and five in the 1D-LSTM. At each timestep, the recurrent part takes as input a feature vector in the sequence obtained from the last convolutional feature map, from left to right, and outputs the probabilities of each character to be in the corresponding image region.

Finally, in the decoding block, the transcription is obtained via greedy decoding, *i.e.* by concatenating the labels with highest probability at each timestep. Then, all the duplicate labels that are not separated by the *blank* token are collapsed in a single character and all the *blank* tokens are removed. The detailed architectures of the presented CRNN and 1D-LSTM models are reported in the Appendix.

## 4 Experimental Evaluation

In this section, we evaluate the suitability of the proposed method for the HTR task based on deformable convolutions (DefConvs) when compared to baselines that feature standard convolutions (StdConvs). For clarity, in this section, we refer to our proposed models as DefConv CRNN and DefConv 1D-LSTM, and we refer to their counterparts containing only standard convolutions as StdConv CRNN and StdConv 1D-LSTM, respectively.



**Fig. 5:** Some receptive fields of the HTR network using standard convolutions (in transparent blue) and deformable convolutions (in transparent red) on modern (top) and historical (bottom) text line images. Deformable convolutions lead to areas of irregular shape that better adapt to handwritten strokes and cover a wider portion of the image thanks to the limited amount of additional offsets parameters (best seen in color).

## 4.1 Datasets

In our experiments, we first consider the IAM [32] and the RIMES [3] datasets, which are commonly used as benchmark for line-level HTR. Both datasets feature different handwriting styles due to the presence of multiple writers and contain samples with curved text lines. Moreover, to validate the ability to deal with uneven background and other nuisances typical of aged manuscripts, we also use two benchmark line-level historical datasets, namely the ICFHR14 [40] and ICFHR16 [41] datasets. Finally, to test the benefits of DefConvs in a pre-training plus fine-tuning setting for HTR on small historical manuscripts [2], we consider the recently proposed Leopardi dataset [10].

**IAM.** The IAM Handwriting dataset contains free-layout modern English text lines from the Lancaster-Oslo/Bergen (LOB) corpus [24], handwritten by multiple users. Following the commonly adopted Aachen University splitting<sup>2</sup>, the dataset consists of 6 482 training lines, 976 validation lines,

and 2915 test lines. Non-*blank* characters are 95, and the line images are  $1698 \pm 292$  pixels wide and  $124 \pm 34$  pixels high.

**RIMES.** The RIMES dataset contains unconstrained modern French letters handwritten by multiple users. In our experiments, we consider the official train/test splitting consisting of 11 333 training lines and 778 test lines and obtain the validation set by retaining the lines contained in the 10% of the training documents. The total number of non-*blank* characters in this dataset is 79, and the line images width and height are  $1637 \pm 555$  pixels and  $130 \pm 36$  pixels, respectively.

**ICFHR14.** The ICFHR14 dataset features legal forms and drafts from the Bentham Papers collection [11], handwritten by the English philosopher and renovator Jeremy Bentham and his collaborators from mid-18<sup>th</sup> century to mid-19<sup>th</sup> century. The dataset was used in a competition for the ICFHR conference in 2014, from which we use the indicated splitting<sup>3</sup> consisting of 9 198 lines for training, 1 415 for validation, and 860 for test.

<sup>2</sup>[www.tbluche.com/resources.html](http://www.tbluche.com/resources.html)

<sup>3</sup><http://doi.org/10.5281/zenodo.44519>

The total number of non-*blank* characters in this dataset is 88, and the line images width and height are  $1401 \pm 573$  pixels and  $121 \pm 38$  pixels, respectively.

**ICFHR16.** The ICFHR16 dataset features minutes of council meetings from the Ratsprotokolle collection, handwritten by multiple writers from 1470 to 1805 in old German [41]. The dataset was used in a competition for the ICFHR conference in 2016<sup>4</sup>. In this dataset, there are 8367 lines for training, 1043 for validation, and 1140 for test. Non-*blank* characters in this dataset are 93, and the images contained are  $963 \pm 318$  pixels wide, and  $123 \pm 39$  pixels high.

**Leopardi.** The Leopardi dataset consists of a small collection of early 19<sup>th</sup> Century letters written in Italian by Giacomo Leopardi [10]. It contains 1303 training lines, 587 validation lines, and 569 test lines. The total number of non-*blank* characters in this dataset is 77, and the images contained are  $1597 \pm 593$  pixels wide, and  $124 \pm 30$  pixels high. This dataset comes with synthetic data that can be used for pre-training plus fine-tuning. The synthetic data are divided in two sets: in one the handwriting resembles that of the original author on historical manuscripts, in the other, the handwriting is modern. The text is obtained from some Leopardi’s proses, so that the language is contemporary to that used in the letters contained in the original Leopardi dataset. Both synthetic sets consists of 89068 training lines and 22397 validation lines, and non-*blank* characters are 114.

## 4.2 Implementation Details

As the sole pre-processing steps, we normalize the text line images between  $-1$  and  $1$  and rescale them in height, keeping the original aspect ratio. In particular, we rescale them to become 60 pixels high for the CRNN model, 128 for the 1D-LSTM model. The output of the convolutional component is the feature map of its last layer, which is a  $2 \times W \times 512$  tensor in the CRNN model and a  $16 \times W \times 80$  tensor in the 1D-LSTM model. These are collapsed in a sequence of  $W$  vectors of 1024 and 1280 elements, respectively. The BLSTMs that constitute the recurrent part of the models have 512 hidden units each in the CRNN variant

**Table 1:** Results on the IAM dataset and the RIMES dataset. Note that <sup>†</sup> and <sup>‡</sup> indicate results of re-implemented method as from [35] and [38], respectively, without language model.

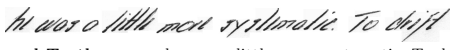
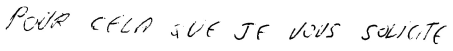
Method	IAM		RIMES	
	CER	WER	CER	WER
de Buy Wenniger <i>et al.</i> [19]	12.9	40.8	-	-
Chen <i>et al.</i> [12]	11.2	34.6	8.3	30.5
Pham <i>et al.</i> [36]	10.8	35.1	6.8	28.5
Bluche and Messina [8] <sup>†</sup>	10.2	32.9	5.8	19.7
Moysset and Messina [35]	8.9	29.3	4.8	16.4
Zhang <i>et al.</i> [56]	8.5	22.2	-	-
Voigtlaender <i>et al.</i> [50] <sup>‡</sup>	8.3	27.5	4.0	17.7
Chowdhury and Vig [13]	8.1	16.7	3.5	9.6
Coquenot <i>et al.</i> [17]	8.0	28.6	4.4	18.0
Bluche [6]	7.9	24.6	2.9	12.6
Markou <i>et al.</i> [30]	7.9	23.9	3.9	13.4
Kang <i>et al.</i> [25]	7.6	24.5	-	-
Ly <i>et al.</i> [29]	7.2	22.9	-	-
StdConv 1D-LSTM	7.7	<b>26.3</b>	5.8	25.5
DefConv 1D-LSTM	<b>7.5</b>	26.9	<b>5.2</b>	<b>23.7</b>
StdConv CRNN	7.8	27.8	4.4	16.0
DefConv CRNN	<b>6.8</b>	<b>24.7</b>	<b>4.0</b>	<b>13.7</b>

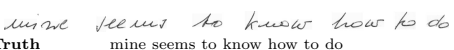
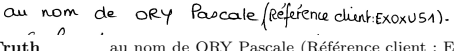
and 256 in the 1D-LSTM variant. In both cases, the recurrent layers are separated by a dropout layer with dropout probability equal to 0.5. The proposed models have been trained with batch size equal to 8 for the CRNN variant and 2 for the 1D-LSTM variant using Adam as optimizer with  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ , and learning rate equal to 0.0001 for the CRNN variant and to 0.003 for the 1D-LSTM variant. We train the models until the Character Error Rate (CER) on the validation set stops improving for 20 epochs. Further details on the models architecture can be found in the Appendix.

For the pretraining plus fine-tuning experiment on the Leopardi dataset, we use the two available synthetic sets separately for pre-training the StdConv and DefConv-based variants. In this phase, we apply random distortions to alter the lines appearance and shape, as done in [10]. The batch size is set to 16 and the learning rate to 0.0001. Also in this case, we stop the training when the validation CER does not improve for 20 epochs. After pre-training, we fine-tune on subsets of decreasing number of training lines from the original Leopardi dataset.

<sup>4</sup><http://doi.org/10.5281/zenodo.1164045>



IAM dataset	
	<b>Ground Truth</b> he was a little more systematic. To drift
<b>StdConv 1D-LSTM</b>	he was o little more syshmlolica. To chif
<b>DefConv 1D-LSTM</b>	he was a little more syslmalics. To diff
<b>StdConv CRNN</b>	hl wos o titll morl sysmalics. To chift
<b>DefConv CRNN</b>	he war o little more systmatie. To drift
RIMES dataset	
	<b>Ground Truth</b> POUR CELA QUE JE VOUS SOLICITE
<b>StdConv 1D-LSTM</b>	Pair CLA IUE JE VUS SOUETE
<b>DefConv 1D-LSTM</b>	POULR CELD SUE JE VOUS SOLIETE
<b>StdConv CRNN</b>	PoIR CELA IVE JE VOUS SOUCITE
<b>DefConv CRNN</b>	POUR CFLA IUE JE VOUS SOLITE

IAM dataset	
	<b>Ground Truth</b> mine seems to know how to do
<b>StdConv 1D-LSTM</b>	iave teems to krmour hour to do
<b>DefConv 1D-LSTM</b>	uiove seems to know how to do
<b>StdConv CRNN</b>	miare Jeeems tro know howr to do
<b>DefConv CRNN</b>	miare seemns tro know how to do
RIMES dataset	
	<b>Ground Truth</b> au nom de ORY Pascale (référence client: EXOXU51).
<b>StdConv 1D-LSTM</b>	au nom de OPy Pascale (référence clientGrOnSN).
<b>DefConv 1D-LSTM</b>	au nom de ORy Pascale (référence clientEXONUS1).
<b>StdConv CRNN</b>	au nom de ORy Poscale (référence clientFoXUS1).
<b>DefConv CRNN</b>	au nom de Ory Pascale (référence client : EXOXU51).

**Fig. 6:** Qualitative results on the benchmark modern datasets considered.

**Table 2:** Results on the ICFHR14, ICFHR16, and Leopardi datasets.

Method	ICFHR14		ICFHR16		Leopardi	
	CER	WER	CER	WER	CER	WER
<b>StdConv 1D-LSTM</b>	4.8	15.3	5.8	25.5	3.8	13.8
<b>DefConv 1D-LSTM</b>	<b>3.6</b>	<b>14.3</b>	<b>5.2</b>	<b>23.7</b>	<b>3.4</b>	<b>12.6</b>
<b>StdConv CRNN</b>	3.9	15.3	5.5	25.9	3.4	13.4
<b>DefConv CRNN</b>	<b>3.6</b>	<b>13.9</b>	<b>4.5</b>	<b>21.7</b>	<b>2.8</b>	<b>10.8</b>

### 4.3 Compared Approaches

We perform a direct comparison of our models and the corresponding baselines not featuring DefConvs on all the considered datasets. For the baselines, we apply the same design choices and training strategies as for our proposed DefConv-based models. Moreover, for the IAM and RIMES datasets, we report the results of other approaches in literature that exploit standard convolution and not perform any data augmentation or language model correction. This way, we can analyse more clearly the effects of the deformable convolution with respect to the standard one. In particular, we consider the approaches proposed by Pham *et al.* [36], Voigtlaender *et al.* [50], Chen *et al.* [12], de Buy Wenniger *et al.* [19], Moysset and Messina [35], and Bluche [6], which are based on the MDLSTM-RNN network proposed in [21]. Moreover, we include in the analysis approaches employing 1D-LSTMs, such as those presented by Ly *et al.* [29], which also exploits self-attention in the convolutional block, and Markou *et al.* [30], which features a fully-connected layer after the recurrent block. We also consider the fully-convolutional architecture by Bluche and Messina [8], and Coquenat *et al.* [17].

Finally, we compare against approaches following the sequence-to-sequence paradigm, such as those proposed by Zhang *et al.* [56], Chowdhury and Vig [13], and Kang *et al.* [25], this latter based on Transformers.

### 4.4 Results


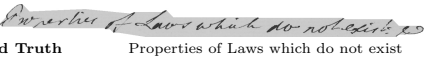
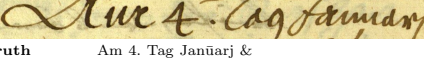
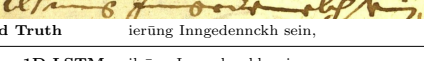
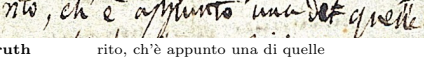
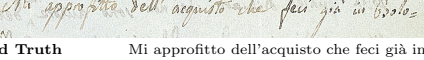
The obtained results are summarized in Table 1 for the modern datasets and Table 2 for the historical datasets. The performance is expressed in terms of the commonly used metrics, CER and the Word Error Rate (WER).

With respect to the considered state-of-the-art approaches, on the IAM and RIMES datasets, the DefConv-based models perform competitively. Compared to the StandardConv-based baselines, the proposed models allow decreasing both the CER and the WER. Note that the improvement in terms of WER is more evident. This suggests that, by capturing more context, DefConv-based models make character-level errors that are more concentrated in fewer difficult words. The improvement is even more evident on the historical datasets. This indicates that our approach is more robust to background non-idealities, which can be observed also from the qualitative results reported in Figure 6 and Figure 7. This confirms that DefConvs are more robust to the nuisances present in the input image since their activations are more concentrated on the writing.

The two proposed variants share the general HTR paradigm of convolutional-recurrent text image representation and CTC-based decoding of the transcription. The main difference between the two is in the number of convolutional and recurrent layers. While 1D-LSTM has a deeper

**Table 3:** Fine-tuning results at varying number of fine-tuning lines.

Method	Pre-training	100%		50%		25%		10%		5%		2,5%	
		CER	WER	CER	WER	CER	WER	CER	WER	CER	WER	CER	WER
StdConv 1D-LSTM	modern	2.9	10.9	4.4	16.1	7.6	25.5	13.6	40.0	18.1	49.6	90.8	99.4
DefConv 1D-LSTM	modern	<b>2.5</b>	<b>9.3</b>	<b>4.2</b>	<b>15.5</b>	<b>5.7</b>	<b>20.6</b>	<b>13.0</b>	<b>38.0</b>	<b>17.1</b>	<b>48.5</b>	<b>26.6</b>	<b>65.5</b>
StdConv CRNN	modern	3.1	12.3	4.9	19.0	7.6	27.4	11.8	38.0	14.8	45.1	20.0	56.4
DefConv CRNN	modern	<b>2.6</b>	<b>10.0</b>	<b>3.5</b>	<b>13.7</b>	<b>5.9</b>	<b>21.7</b>	<b>9.8</b>	<b>32.6</b>	<b>12.8</b>	<b>41.0</b>	<b>17.7</b>	<b>52.0</b>
StdConv 1D-LSTM	historical	2.5	9.2	4.3	16.1	6.4	22.4	9.9	30.7	12.1	37.5	16.8	47.4
DefConv 1D-LSTM	historical	<b>2.3</b>	<b>8.7</b>	<b>3.5</b>	<b>13.4</b>	<b>5.0</b>	<b>18.5</b>	<b>8.7</b>	<b>27.6</b>	<b>9.6</b>	<b>31.4</b>	<b>15.0</b>	<b>44.9</b>
StdConv CRNN	historical	2.7	10.8	4.2	16.2	6.2	22.6	9.3	31.8	11.7	38.0	14.4	44.9
DefConv CRNN	historical	<b>2.3</b>	<b>8.9</b>	<b>3.5</b>	<b>13.8</b>	<b>5.1</b>	<b>19.2</b>	<b>7.8</b>	<b>27.7</b>	<b>9.4</b>	<b>32.4</b>	<b>13.5</b>	<b>44.0</b>

ICFHR14 dataset			
	<b>Ground Truth</b>	reason of the concurrence of certain circumstances, it	
	StdConv 1D-LSTM	teason of the concnence of antans ciraummstances, to	
	DefConv 1D-LSTM	teason of the covnerence of certais circumstances, to	
	StdConv CRNN	reosn of the concnences of antrvie ciroumstances, to	
	DefConv CRNN	reosn of the conerrence of cantain circumntances, th	
	<b>Ground Truth</b>	Properties of Laws which do not exist	
	StdConv 1D-LSTM	Pwperlies of Laws which so nobeist: es	
	DefConv 1D-LSTM	Parertics or Laws which do rot exist a	
	StdConv CRNN	Pwrteries of Laws which do notedist:	
	DefConv CRNN	Pwperities of Laws which do notexist a	
ICFHR16 dataset			
	<b>Ground Truth</b>	Am 4. Tag Januarj &	
	StdConv 1D-LSTM	Am 4. Tag Janarj	
	DefConv 1D-LSTM	Am4. Tag Kamarj	
	StdConv CRNN	Am4. Tag Janmarj	
	DefConv CRNN	Am: Tag Jamarj	
	<b>Ground Truth</b>	ierung Inngedenckh sein,	
	StdConv 1D-LSTM	ilsung Inngedenckh sein.,	
	DefConv 1D-LSTM	ierung Inngedenckh sein,	
	StdConv CRNN	ilung Inngedenckh sein.	
	DefConv CRNN	ierung Inngedenckh sein.	
Leopardi dataset			
	<b>Ground Truth</b>	rito, ch' è appunto una di quelle	
	StdConv 1D-LSTM	rito, ch' è agplunto una det queti=	
	DefConv 1D-LSTM	rito, ch' è aggiunto una dil quelle	
	StdConv CRNN	rito, ch' è apunto unla dil quelle	
	DefConv CRNN	rito, ch' è appunto una di quelle	
	<b>Ground Truth</b>	Mi approfito dell'acquisto che feci già in Bolo=	
	StdConv 1D-LSTM	Mi oppofito dell'aspisto che fai già in biril=	
	DefConv 1D-LSTM	Mi pprofito dell'apisto che fei già in Bolo=	
	StdConv CRNN	Mi pogio dell'aluisto che fuai giù in dolo.=	
	DefConv CRNN	Mi gpro siteo dell'aguisto che feui giù in Bolo	

**Fig. 7:** Qualitative results on the historical datasets considered.

recurrent component, CRNN has a deeper convolutional component. The larger number of recurrent layers makes the 1D-LSTM comparable or better than the CRNN variant when both their convolutional components contain standard convolutions. On the other hand, due to the larger number of convolutional layers in CRNN, this variant outperforms the other when both are equipped with DefConvs. In general, the CRNN variant benefits the most from the introduction of DefConvs. In fact, on average, its CER decreases by 0.7 and the WER by 2.5, while for the 1D-LSTM variant, the CER decreases by 0.6 and the WER by 1.0. This is more evident on the historical datasets, where the background is noisier due to the aging of the page support. From this observation, we can

conclude that a larger number of DefConv layers improves the robustness to background noise.

The results of the pretraining plus fine-tuning experiments on the Leopardi dataset are reported in Table 3. It emerges that DefConvs allow better exploiting this paradigm, as confirmed by the comparable to smaller errors obtained with the variants featuring DefConvs in all settings. Moreover, the performance gap between pre-training on historical synthetic data and on modern synthetic data is smaller when using DefConv-based models. This suggests that models based on DefConvs can generalize better than their StdConv-based counterparts and would be more effective for HTR of small historical manuscript collections, even if pretrained on general, modern-looking data.

## 5 Conclusion

In this work, we investigated the suitability of deformable convolutions for the HTR task. We validated our approach on both modern and historical datasets of various languages and historical periods and demonstrated their superior performance with respect to standard convolutions. Due to the ability to adapt to highly distorted handwritten strokes and focus on ink pixels while still capturing more context, DefConv-based HTR models are effective when dealing with free-layout documents and allow better exploiting the pre-training plus fine-tuning paradigm for HTR of small collections of historical documents. Further performance improvements could be achieved by employing a language-specific language model, a direction which we leave for future investigation.

## Declarations

This work was supported by the “AI for Digital Humanities” project (Pratica Sime n.2018.0390), funded by “Fondazione di Modena”, and by the “DHMoRe Lab” project (CUP E94I19001060003), funded by “Regione Emilia Romagna”.

## References

- [1] Aberdam, A., R. Litman, S. Tsiper, O. Anshel, R. Slossberg, S. Mazor, R. Manmatha, and P. Perona 2021. Sequence-to-Sequence Contrastive Learning for Text Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15302–15312.
- [2] Aradillas, J.C., J.J. Murillo-Fuentes, and P.M. Olmos. 2021. Boosting offline handwritten text recognition in historical documents with few labeled lines. *IEEE Access* 9(9438636): 76674–76688 .
- [3] Augustin, E., M. Carré, E. Grosicki, J.M. Brodin, E. Geoffrois, and F. Prêteux 2006. RIMES evaluation campaign for handwritten mail processing. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, pp. 231–235.
- [4] Bera, S.K., A. Chakrabarti, S. Lahiri, E.H.B. Smith, and R. Sarkar. 2019. Normalization of unconstrained handwritten words in terms of Slope and Slant Correction. *Pattern Recognition Letters* 128: 488–495 .
- [5] Bhunia, A.K., A. Das, A.K. Bhunia, P.S.R. Kishore, and P.P. Roy 2019. Handwriting Recognition in Low-resource Scripts using Adversarial Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4767–4776.
- [6] Bluche, T. 2016. Joint Line Segmentation and Transcription for End-to-End Handwritten Paragraph Recognition. In *Advances in Neural Information Processing Systems*, Volume 29, pp. 838–846.
- [7] Bluche, T., J. Louradour, and R. Messina 2017. Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention. In *Proceedings of the International Conference on Document Analysis and Recognition*, Volume 1, pp. 1050–1055. IEEE.
- [8] Bluche, T. and R. Messina 2017. Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, Volume 1, pp. 646–651. IEEE.
- [9] Bouillon, M., R. Ingold, and M. Liwicki. 2019. Grayification: a meaningful grayscale conversion to improve handwritten historical documents analysis. *Pattern Recognition Letters* 121: 46–51 .
- [10] Cascianelli, S., M. Cornia, L. Baraldi, M.L. Piazzì, R. Schiuma, and R. Cucchiara 2021. Learning to Read L’Infinito: Handwritten Text Recognition with Synthetic Training Data. In *Proceedings of the International Conference on Computer Analysis of Images and Patterns*, pp. 340–350. Springer.
- [11] Causer, T. and V. Wallace. 2012. Building a volunteer community: results and findings from Transcribe Bentham. *Digital Humanities Quarterly* 6(2) .

- [12] Chen, Z., Y. Wu, F. Yin, and C.L. Liu 2017. 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*, Volume 1, pp. 525–530. IEEE.
- [13] Chowdhury, A. and L. Vig 2018. An Efficient End-to-End Neural Model for Handwritten Text Recognition. In *Proceedings of the British Machine Vision Conference*.
- [14] Cilia, N.D., C. De Stefano, F. Fontanella, and A.S. di Freca. 2019. A ranking-based feature selection approach for handwritten character recognition. *Pattern Recognition Letters* 121: 77–86 .
- [15] Clanuwat, T., A. Lamb, and A. Kitamoto 2019. KuroNet: Pre-Modern Japanese Kuzushiji Character Recognition with Deep Learning. In *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 607–614. IEEE.
- [16] Cojocaru, I., S. Cascianelli, L. Baraldi, M. Corsini, and R. Cucchiara 2021. Watch Your Strokes: Improving Handwritten Text Recognition with Deformable Convolutions. In *Proceedings of the International Conference on Pattern Recognition*, pp. 6096–6103. IEEE.
- [17] Coquenet, D., C. Chatelain, and T. Paquet 2020. Recurrence-free unconstrained handwritten text recognition using gated fully convolutional network. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pp. 19–24. IEEE.
- [18] Dai, J., H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei 2017. Deformable Convolutional Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 764–773.
- [19] de Buy Wenniger, G.M., L. Schomaker, and A. Way 2019. No Padding Please: Efficient Neural Handwriting Recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 355–362. IEEE.
- [20] Fontanella, F., F. Colace, M. Molinara, A. Scotto Di Freca, and S. Filippo. 2020. Pattern Recognition and Artificial Intelligence techniques for Cultural Heritage. *Pattern Recognition Letters* 138: 23–29 .
- [21] Graves, A. and J. Schmidhuber 2008. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, Volume 21, pp. 545–552.
- [22] Jaderberg, M., K. Simonyan, A. Zisserman, and K. Kavukcuoglu 2015. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, Volume 28, pp. 2017–2025.
- [23] Jayasundara, V., S. Jayasekara, H. Jayasekara, J. Rajasegaran, S. Seneviratne, and R. Rodrigo 2019. TextCap: Handwritten Character Recognition with Very Small Datasets. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pp. 254–262. IEEE.
- [24] Johansson, S., G.N. Leech, and H. Goodluck. 1978. *Manual of information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital computer*. Department of English, University of Oslo.
- [25] Kang, L., P. Riba, M. Rusiñol, A. Fornés, and M. Villegas. 2020. Pay Attention to What You Read: Non-recurrent Handwritten Text-Line Recognition. *arXiv preprint arXiv:2005.13044* .
- [26] Krevat, E. and E. Cuzzillo. 2006. Improving Off-line Handwritten Character Recognition with Hidden Markov Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 .
- [27] Li, M., T. Lv, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei. 2021. TrOCR: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282* .
- [28] Liu, C.L. and K. Marukawa. 2005. Pseudo two-dimensional shape normalization methods for handwritten Chinese character recognition. *Pattern Recognition* 38(12): 2242–2255 .

- [29] Ly, N.T., H.T. Nguyen, and M. Nakagawa 2021. 2D Self-attention Convolutional Recurrent Network for Offline Handwritten Text Recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 191–204. Springer.
- [30] Markou, K., L. Tsochatzidis, K. Zagoris, A. Papazoglou, B. Karagiannis, S. Symeonidis, and I. Pratikakis 2021. A Convolutional Recurrent Neural Network for the Handwritten Text Recognition of Historical Greek Manuscripts. In *Proceedings of the International Conference on Pattern Recognition Workshops and Challenges*, pp. 249–262. Springer.
- [31] Marti, U.V. and H. Bunke 2000. Handwritten Sentence Recognition. In *Proceedings of the International Conference on Pattern Recognition*, Volume 3, pp. 463–466. IEEE.
- [32] Marti, U.V. and H. Bunke. 2002. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition* 5(1): 39–46 .
- [33] Michael, J., R. Labahn, T. Grüning, and J. Zöllner 2019. Evaluating Sequence-to-Sequence Models for Handwritten Text Recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 1286–1293. IEEE.
- [34] Moysset, B., C. Kermorvant, and C. Wolf 2017. Full-Page Text Recognition: Learning Where to Start and When to Stop. In *Proceedings of the International Conference on Document Analysis and Recognition*, Volume 1, pp. 871–876. IEEE.
- [35] Moysset, B. and R. Messina. 2019. Are 2D-LSTM really dead for offline text recognition? *International Journal on Document Analysis and Recognition* 22(3): 193–208 .
- [36] Pham, V., T. Bluche, C. Kermorvant, and J. Louradour 2014. Dropout improves recurrent neural networks for handwriting recognition. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pp. 285–290. IEEE.
- [37] Poznanski, A. and L. Wolf 2016. CNN-N-Gram for Handwriting Word Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2305–2314.
- [38] Puigcerver, J. 2017. Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition? In *Proceedings of the International Conference on Document Analysis and Recognition*, Volume 1, pp. 67–72. IEEE.
- [39] Quirós, L., V. Bosch, L. Serrano, A.H. Toselli, and E. Vidal 2018. From HMMs to RNNs: computer-assisted transcription of a handwritten notarial records collection. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pp. 116–121. IEEE.
- [40] Sánchez, J.A., V. Romero, A.H. Toselli, and E. Vidal 2014. ICFHR2014 competition on handwritten text recognition on transcriptorium datasets (HTRtS). In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pp. 785–790. IEEE.
- [41] Sanchez, J.A., V. Romero, A.H. Toselli, and E. Vidal 2016. ICFHR2016 competition on handwritten text recognition on the READ dataset. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pp. 630–635. IEEE.
- [42] Santoro, A. and A. Marcelli. 2020. Using keyword spotting systems as tools for the transcription of historical handwritten documents: Models and procedures for performance evaluation. *Pattern Recognition Letters* 131: 329–335 .
- [43] Shi, B., X. Bai, and C. Yao. 2016. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39(11): 2298–2304 .
- [44] Simonyan, K. and A. Zisserman 2015. Very deep convolutional networks for large-scale

- image recognition. In *Proceedings of the International Conference on Learning Representations*.
- [45] Such, F.P., D. Peri, F. Brockler, H. Paul, and R. Ptucha 2018. Fully Convolutional Networks for Handwriting Recognition. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pp. 86–91. IEEE.
- [46] Sueiras, J., V. Ruiz, A. Sanchez, and J.F. Velez. 2018. Offline continuous handwriting recognition using sequence to sequence neural networks. *Neurocomputing* 289: 119–128 .
- [47] Toselli, A.H., A. Juan, J. González, I. Salvador, E. Vidal, F. Casacuberta, D. Keysers, and H. Ney. 2004. Integrated handwriting recognition and interpretation using finite-state models. *International Journal of Pattern Recognition and Artificial Intelligence* 18(04): 519–539 .
- [48] Toselli, A.H. and E. Vidal 2015. Handwritten text recognition results on the Bentham collection with improved classical n-gram-HMM methods. In *Proceedings of the International Workshop on Historical Document Imaging and Processing*, pp. 15–22.
- [49] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008.
- [50] Voigtlaender, P., P. Doetsch, and H. Ney 2016. Handwriting recognition with large multi-dimensional long short-term memory recurrent neural networks. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pp. 228–233. IEEE.
- [51] Wick, C., J. Zöllner, and T. Grüning. 2021. Rescoring Sequence-to-Sequence Models for Text Line Recognition with CTC-Prefixes. *arXiv preprint arXiv:2110.05909* .
- [52] Wigington, C., S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen 2017. Data augmentation for recognition of handwritten words and lines using a CNN-LSTM network. In *Proceedings of the International Conference on Document Analysis and Recognition*, Volume 1, pp. 639–645. IEEE.
- [53] Wigington, C., C. Tensmeyer, B. Davis, W. Barrett, B. Price, and S. Cohen 2018. Start, Follow, Read: End-to-End Full-Page Handwriting Recognition. In *Proceedings of the European Conference on Computer Vision*, pp. 367–383.
- [54] Yousef, M. and T.E. Bishop 2020. OrigamiNet: Weakly-Supervised, Segmentation-Free, One-Step, Full Page Text Recognition by learning to unfold. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14710–14719.
- [55] Yousef, M., K.F. Hussain, and U.S. Mohammed. 2020. Accurate, Data-Efficient, Unconstrained Text Recognition with Convolutional Neural Networks. *Pattern Recognition* 108: 107482 .
- [56] Zhang, Y., S. Nie, W. Liu, X. Xu, D. Zhang, and H.T. Shen 2019. Sequence-to-sequence domain adaptation network for robust text image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2740–2749.
- [57] Zhong, Z., X.Y. Zhang, F. Yin, and C.L. Liu 2016. Handwritten Chinese character recognition with spatial transformer and deep residual networks. In *Proceedings of the International Conference on Pattern Recognition*.

# Appendix

## Models Architectures

We provide the detailed architecture of the proposed DefConv-based HTR models: in Table 1 for CRNN, in Table 2 for 1D-LSTM. The offsets of the DefConvs layers are handled in a standard convolutional layer before the DefConv, which is in charge of learning two parameters for each kernel cell of the DefConv. Note that the output size of the final Linear layer,  $c$  depends on the charset size of each dataset (including the *blank* character). In particular:  $c = 96$  for the IAM dataset,  $c = 80$  for the RIMES,  $c = 89$  for the ICFHR14,  $c = 94$  for the ICFHR16, and  $c = 77$  for the Leopardi. Note that, from a practical standpoint, when the whole dataset is available,  $c$  can be calculated directly as the number of the characters appearing in the dataset (*i.e.* the charset), plus the *blank* character. For new or unknown datasets, the charset, and thus,  $c$ , can be estimated *e.g.*, from large corpora in the same language as the dataset of interest, but can potentially include as many characters as the designer wants. In this latter case, logits corresponding to characters included in the charset but not appearing in the dataset of interest will be assigned zero probability. In the StandardConv-based baselines we used in the experiments, each pair of offset Convolution layer and the DefConv layer is replaced by a standard convolution layer with the same characteristics as the DefConv layer.

Layer Type	Size	Kernel	Stride	Padding
Convolution	18	$3 \times 3$	(1, 1)	(1, 1)
DefConv	64	$3 \times 3$	(1, 1)	(1, 1)
Batch Normalization	–	–	–	–
ReLU	–	–	–	–
Max Pooling	–	$2 \times 2$	(2, 2)	(0, 0)
Dropout (p = 0.2)	–	–	–	–
Convolution	18	$3 \times 3$	(1, 1)	(1, 1)
DefConv	128	$3 \times 3$	(1, 1)	(1, 1)
Batch Normalization	–	–	–	–
ReLU	–	–	–	–
Max Pooling	–	$2 \times 2$	(2, 2)	(0, 0)
Dropout (p = 0.2)	–	–	–	–
Convolution	18	$3 \times 3$	(1, 1)	(1, 1)
DefConv	256	$3 \times 3$	(1, 1)	(1, 1)
Batch Normalization	–	–	–	–
ReLU	–	–	–	–
Convolution	18	$3 \times 3$	(1, 1)	(1, 1)
DefConv	256	$3 \times 3$	(1, 1)	(1, 1)
ReLU	–	–	–	–
Max Pooling	–	$2 \times 2$	(2, 1)	(0, 1)
Dropout (p = 0.2)	–	–	–	–
Convolution	18	$3 \times 3$	(1, 1)	(1, 1)
DefConv	512	$3 \times 3$	(1, 1)	(1, 1)
Batch Normalization	–	–	–	–
ReLU	–	–	–	–
Dropout (p = 0.2)	–	–	–	–
Convolution	18	$3 \times 3$	(1, 1)	(1, 1)
DefConv	512	$3 \times 3$	(1, 1)	(1, 1)
ReLU	–	–	–	–
Max Pooling	–	$2 \times 2$	(2, 1)	(0, 1)
Dropout (p = 0.2)	–	–	–	–
Convolution	8	$2 \times 2$	(1, 1)	(0, 0)
DefConv	512	$2 \times 2$	(1, 1)	(0, 0)
Batch Normalization	–	–	–	–
ReLU	–	–	–	–
BLSTM	512	–	–	–
Dropout (p = 0.5)	–	–	–	–
BLSTM	512	–	–	–
Linear	$c$	–	–	–

**Table 1:** Architecture details for the CRNN variant.

Layer Type	Size	Kernel	Stride	Padding
Convolution	18	$3 \times 3$	(1, 1)	(1, 1)
DefConv	16	$3 \times 3$	(1, 1)	(1, 1)
Batch Normalization	–	–	–	–
LeakyReLU	–	–	–	–
Max Pooling	–	$2 \times 2$	(2, 2)	(0, 0)
Convolution	18	$3 \times 3$	(1, 1)	(1, 1)
DefConv	32	$3 \times 3$	(1, 1)	(1, 1)
Batch Normalization	–	–	–	–
LeakyReLU	–	–	–	–
Max Pooling	–	$2 \times 2$	(2, 2)	(0, 0)
Dropout (p = 0.2)	–	–	–	–
Convolution	18	$3 \times 3$	(1, 1)	(1, 1)
DefConv	48	$3 \times 3$	(1, 1)	(1, 1)
Batch Normalization	–	–	–	–
LeakyReLU	–	–	–	–
Max Pooling	–	$2 \times 2$	(2, 2)	(0, 0)
Dropout (p = 0.2)	–	–	–	–
Convolution	18	$3 \times 3$	(1, 1)	(1, 1)
DefConv	64	$3 \times 3$	(1, 1)	(1, 1)
Batch Normalization	–	–	–	–
LeakyReLU	–	–	–	–
Dropout (p = 0.2)	–	–	–	–
Convolution	18	$3 \times 3$	(1, 1)	(1, 1)
DefConv	80	$3 \times 3$	(1, 1)	(1, 1)
Batch Normalization	–	–	–	–
LeakyReLU	–	–	–	–
BLSTM	256	–	–	–
Dropout (p = 0.5)	–	–	–	–
BLSTM	256	–	–	–
Dropout (p = 0.5)	–	–	–	–
BLSTM	256	–	–	–
Dropout (p = 0.5)	–	–	–	–
BLSTM	256	–	–	–
Dropout (p = 0.5)	–	–	–	–
BLSTM	256	–	–	–
Dropout (p = 0.5)	–	–	–	–
BLSTM	256	–	–	–
Dropout (p = 0.5)	–	–	–	–
Linear	$c$	–	–	–

**Table 2:** Architecture details for the 1D-LSTM variant.