Training transformer architectures on few annotated data: an application to historical handwritten text recognition

Killian Barrere, Yann Soullard, Aurélie Lemaitre, Bertrand Coüasnon

▶ **To cite this version:**

Killian Barrere, Yann Soullard, Aurélie Lemaitre, Bertrand Coüasnon. Training transformer architectures on few annotated data: an application to historical handwritten text recognition. International Journal on Document Analysis and Recognition, In press, International Journal on Document Analysis and Recognition (IJDAR), 10.1007/s10032-023-00459-2 . hal-04447488

HAL Id: hal-04447488
https://hal.science/hal-04447488v1

Submitted on 9 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Training Transformer Architectures on Few Annotated Data: an Application to Historical Handwritten Text Recognition

Killian Barrere[1*], Yann Soullard[1,2], Aurélie Lemaitre[1] and Bertrand Coüasnon[1]

[1]Univ. Rennes, CNRS, IRISA, Rennes, France.
[2]Université de Rennes 2, CNRS, LETG, Rennes, France.

*Corresponding author(s). E-mail(s): killian.barrere@irisa.fr;
Contributing authors: yann.soullard@irisa.fr; aurelie.lemaitre@irisa.fr;
bertrand.couasnon@irisa.fr;

This is the accepted version of the article. The Version of Record of this article is published in IJDAR, and is available online at https://doi.org/10.1007/s10032-023-00459-2.

Abstract

Transformer-based architectures show excellent results on the task of handwritten text recognition, becoming the standard architecture for modern datasets. However, they require a significant amount of annotated data to achieve competitive results. They typically rely on synthetic data to solve this problem. Historical handwritten text recognition represents a challenging task due to degradations, specific handwritings for which few examples are available and ancient languages that vary over time. These limitations also make it difficult to generate realistic synthetic data. Given sufficient and appropriate data, Transformer-based architectures could alleviate these concerns, thanks to their ability to have a global view of textual images and their language modeling capabilities. In this paper, we propose the use of a lightweight Transformer model to tackle the task of historical handwritten text recognition. To train the architecture, we introduce realistic looking synthetic data reproducing the style of historical handwritings. We present a specific strategy, both for training and prediction, to deal with historical documents, where only a limited amount of training data is available. We evaluate our approach on the ICFHR 2018 READ dataset which is dedicated to handwriting recognition in specific historical documents. The results show that our Transformer-based approach is able to outperform existing methods.

Keywords: Lightweight Architecture, Few Annotated Data, Historical Documents, Transformer,Handwritten Text Recognition, Neural Networks

1 Introduction

The field of Handwritten Text Recognition (HTR) involves automatically recognizing handwritten text in images of text-lines, paragraphs, or pages. It typically follows a first step of document layout analysis. HTR is valuable for many libraries and archives digitizing vast document collections,

offering cost-effective automated transcription. Recent advances in the field of deep learning and computer vision allow methods from the state of the art to obtain low error rates on modern documents [1–6].

Recent methods use convolutional layers to automatically extract features from text-line images, combined with Transformer layers [7] for global views across features and language modeling. These approaches benefits from the massive parallelization offered by modern GPUs and show excellent results on modern datasets [8–16].

Nevertheless, recognizing historical handwritten texts is challenging. Historical documents often feature complex and hard-to-read handwritings, even for human experts. They are prone to degradations from time that may affect the paper quality or the ink, leading to faded handwritings. Additionally, few historical documents are annotated due to costly transcriptions processes. It therefore results in increased difficulties to learn visual writing styles and specific language models. Hence, traditional approaches for modern documents, and especially those using large architectures, might not be adapted for historical documents.

Transformer-based architecture could enable significant advances to recognize historical handwritings, by utilizing attention to recognize characters while modeling the language. However, as traditional Transformer-based architecture have many parameters, they require a tremendous amount of annotated data to perform well. On modern datasets, 10,000 annotated text-line images might produce satisfactory results. However, most Transformer-based approaches use additional synthetic data to reach state-of-the-art performances [9–15]. This volume of labeled data roughly equals 300–400 pages of a collection. Human experts cannot be reasonably considered to annotate that number of pages, while most collections typically contain far fewer pages. Hence, applying Transformer-based architectures to historical documents represents a hard task due to limited labeled data.

In this paper, we propose an approach to train a lightweight Transformer-based architecture for the task of historical HTR when the number of labeled data is substantially limited. To the best of our knowledge, this is the first time a text-line level Transformer-based approach is applied to specific historical documents. The contributions are:

- A lightweight Transformer architecture for very limited amount of labelled data;
- A Synthetic data generator reproducing the style of historical handwritings;
- Training and prediction strategies designed for adaptations on specific documents.

Our code[1] and our synthetic data generator[2] are openly available online.

We introduce this article by presenting related works on the task of HTR and provide an insight on strategies to recognize documents with few labeled examples. In Section 3, we present our Very Lightweight Transformer-based architecture (VLT), our synthetic data generation pipeline and our strategies. Extensive experiments and results are detailed in Section 4.

2 Related Works

Neural networks combining convolutional and recurrent layers trained with the Connectionist Temporal Classification (CTC) loss [17] used to be the prominent solution for the task of Handwritten Text Recognition (HTR) [1–4, 18]. Fully convolutional networks proposed to remove the usage of recurrent layers by instead using more complex convolutional backbones [5, 6]. Yet, Transformer-based architectures achieve better results and become the standard architecture nowadays.

2.1 Transformer Models for HTR

Vaswani et al. proposed the Transformer architecture [7] in the field of natural language processing to tackle sequence-to-sequence tasks. It is an efficient alternative to recurrent layers as transformer layers are able to learn large contextual dependencies while taking advantage of the strong parallelization offered by modern GPUs.

In the field of HTR, researchers have started including Transformer layers in existing pipelines. Thanks to the advantages offered by such architectures, the majority of the best performing architectures nowadays are based on Transformer layers. They enable global views on the image

[1] https://gitlab.inria.fr/intuidoc-public/vlt
[2] https://gitlab.inria.fr/intuidoc-public/synthetic-handwriting-generation

as well as some language modeling abilities. They obtain low error rates on line-level modern handwritings [8, 9, 11–13, 15, 16]. Additionally, the Transformer-based architectures proposed by Singh et al. [10] and Coquenet et al. [14] show groundbreaking results for page-level HTR.

Most architectures follow the encoder-decoder paradigm, offering both performing optical recognition and language modeling capabilities [8–12, 14–16]. These architectures are typically trained using a Cross-Entropy (CE) loss function.

2.1.1 Encoders

The encoder automatically extracts features from the image thanks to convolutional layers. Depending on desired complexity, the convolutional backbone ranges from a few convolutional layers [8, 11–13, 16] to more complex Residual Networks [9, 10]. For line-level approaches, multi-head self-attention layers [7] follow to enhance the extracted features. Page-level approaches however do not include self-attention layers while they take advantage of more complex and carefully designed convolutional backbones. Wick et al. [12] use an encoder with convolutional and recurrent layers. Despite being more simple, the usage of recurrent layers might slow down the training. Opposed to that architecture, Li et al. [15] propose a significantly larger Transformer architecture. Instead of an encoder composed of a convolutional backbone combined with self-attention layers, they use a more complex Transformer for vision, using image patches instead of isolated text-lines. Although they reach very low error rates on modern datasets by taking advantage of pre-trained models, their heavy architecture requires extensive training data.

2.1.2 Decoders

Transformer-based decoders aims at producing the subsequent characters based on the sequence of characters that have already been predicted. They apply causal self-attention on the sequence of characters already predicted, as well as mutual attention that additionally includes the features produced by the encoder. Altogether, these layers enable a global view on both optical features and on the character sequence. It results in a reduction of the error rates, as Transformer-based decoders act as a language model to some extent. Wick et al. [12] show that their Transformer-based architecture benefits from the addition of a separate language model. Kang et al. [9] instead obtain little or no improvement using an additional language modeling and demonstrate their heavier decoder has reliable language modeling abilities. This difference might be explained by the complexity of the used decoders or even the amount of labeled data. However, compared to traditional language models, the decoder is trained in an end-to-end fashion with the encoder.

In addition, beam search decoding might be used to obtain predictions [12, 13, 15, 16]. Nevertheless, Singh et al. [10] observe no gain by using beam search decoding algorithm instead of a faster and more simple greedy decoding.

2.1.3 Discussion

Even though Transformer-based architectures allow many benefits, they typically rely on a substantial number of training weights. Heaviest architectures might use hundreds of millions of parameters to achieve low error rates [9, 15]. Using that many parameters leads to a few downsides. Heavy architectures require additional data (and consequently increase training times and costs) to be trained. This is especially challenging for the task of historical HTR, since datasets generally contain very few annotated data.

2.2 Data-Oriented Strategies

To counter the lack of annotated data, augmentation techniques are used by most traditional approaches for HTR [1–3, 18, 19] and Transformer-based architectures [8–15]. Typical augmentations pipelines might include elastic distortions, skewing or random rotations among others. Additionally, synthetic data for modern datasets is frequently used alongside real data, improving generalization and reducing error rates in Transformer-based models [9–15]. The impact on error rates varies based on model complexity, ranging from 6% to 39% for text-line level approaches [9–13]. Heavier architectures demonstrate a considerable improvement by training with synthetic data [9] compared to lighter architectures [11]. Nonetheless, they also require a greater number of examples and training time. For instance the TrOCR approach proposed by Li et

al. [15] relies on hundreds of millions of text-lines generated from existing PDF files.

2.2.1 Modern Synthetic Data

The standard approaches to generate synthetic data consist of the following steps: selection of textual content; generation of a text-line image using fonts (printed or cursive); and transformations applied to the text image. This straightforward process can lead to the generation of realistic synthetic modern handwritings [9, 11–13]. For the textual content, the approaches from the state of the art use either dedicated datasets [10, 13], ebooks [9], articles from Wikipedia [10, 11, 15] or even the ground truth from existing HTR datasets [14]. Following, a given textual content is generated using specific fonts.

Printed fonts can be used to produce synthetic modern printed text-lines images [10, 14, 15]. Yet, it is far from the visual style of cursive handwritings and their complexity (i.e. the usage of ligatures or overlapping lines). HTR methods instead consider generating synthetic modern handwritten images [9, 11–13, 15]. Such fonts produce realistic synthetic handwritten data but might differ from specific handwritings.

Neural networks might be considered to generate handwritten text-lines. Generative approaches show realistic generated handwritten text-line images on modern texts. Such methods typically rely on Generative Adversarial Networks (GAN) [20–22] while Bhunia et al. [23] instead uses a Transformer architecture. Most GAN [20–24] require annotated text images to train a text recognizer, ensuring a readable text is produced. GAN are therefore suitable to generate realistic images, but not to train a separate text recognizer due to their annotated data requirement.

2.2.2 Historical Synthetic Data

Generation of modern synthetic data represent a relatively straightforward task with most documents using black characters on top of a simple white background. Generating realistic historical synthetic data includes increased difficulties in comparison. The ink might use different color variants on top of diverse backgrounds, and documents might include additional degradations like ink fading. This makes approaches generating historical documents limited. Journet et al. [25] use isolated characters automatically segmented from real documents to generate text. However, their approach might be impractical to cursive handwritings that contain ligatures. Vögtlin et al. [24] as well as Madi et al. [26] instead propose generative approaches using GANs. They show their methods generate patches of images with the visual style of historical documents. However, patch-based approaches are not adapted for HTR since consecutive patches do not produce seamless images [24].

2.2.3 Discussion

As far as we are aware, GAN-based approaches are not robust enough yet to generate text-line historical handwritings. Synthetic data generation using cursive fonts represents a more straightforward process to virtually increase the amount of available data at a reduced cost. On modern datasets, they are suitable to generate realistic looking handwritten images leading to increased performance. To the best of our knowledge, synthetic data using handwriting fonts have not yet been proposed for historical documents, which comes with specific increased difficulties.

2.3 Architectural Designs for Few Annotated Data

In addition, some solutions related to architecture designs have been proposed to ease training. Residual connections help gradient propagation and proved to be useful for many networks in computer vision tasks and for HTR. They are used inside Transformer layers, but also in convolutional backbones [9, 10, 14].

Following that idea, loss shortcuts include auxiliary losses in the middle of the architectures. For HTR, Michael et al. [4] propose to use a CTC loss at the end of the encoder in addition to a Cross-Entropy (CE) loss function. Such loss shortcut is considered as a way to ease the training of Transformer-based approaches [11, 12, 14].

Despite the trend for Transformer-based architectures to develop heavier architectures, on the opposite, a few works instead focused on using lightweight architectures [11, 13]. In a previous work [11], we designed a light architecture that uses only 7.7M parameters in contrast to heavier architectures that use hundreds of millions of

parameters. Our light Transformer-based architecture takes advantage of architectural designs choices. It uses a simple convolutional backbone, a reduced number of parameters in Transformer layers and a loss shortcut. Compared to other Transformer-based architecture, it obtains state-of-the-art error rates when trained on only real data, while achieving better results with synthetic data on the modern English dataset IAM [27]. D'Arce et al. [13] proposed to go further and use a Transformer-based architecture composed solely of an encoder. It results in a lightweight architecture with only 1.7M parameters. Such architecture obtains acceptable error rates, however far from those of encoder-decoder based approaches.

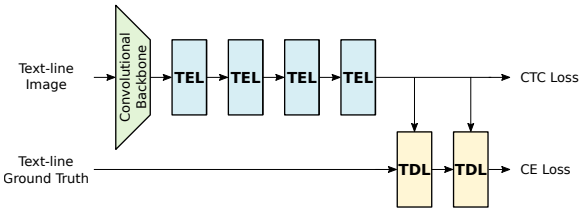
3 Our Approach to Train on Few Annotated data

We propose a new procedure to train Transformer-based architectures for the task of historical HTR. As the number of annotated data is very limited, we take advantage of a Transformer-based architecture with a reduced number of parameters. We design synthetic data specifically developed for historical handwritings to help in the training of Transformer-based architectures. Lastly, we propose a training strategy and a prediction algorithm combining multiple predictions to further reduce the error rates.

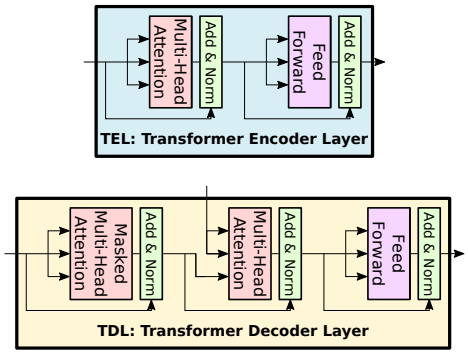
3.1 Very Lightweight Transformer

To tackle historical HTR, where a limited number of annotated data are available, we design a lightweight architecture, benefiting from advantages such as faster prediction times. Inspired by the Light Transformer-based architecture (LT) we introduced in our previous works [11], we now present an even lighter version, the *Very Lightweight Transformer* (VLT), as depicted in Fig. 1.

The VLT uses a convolutional backbone, composed of 5 convolutional layers. These layers include LeakyReLU activation functions, layer normalization, max pooling in the first three layers and dropout with a probability of 0.1. Next, 4 Transformer encoder layers with multi-head self-attention are used with a hidden size of 256, 4 heads and a dropout value of 0.2.



(a) Global view of the architecture.



(b) Details of the Transformer layers.

Fig. 1: Our proposed VLT architecture.

The VLT uses a Transformer decoder composed of 2 Transformer decoder layers. They combine masked multi-head self-attention that process the sequence of already predicted characters, as well as multi-headed encoder-decoder attention. Similar to the Transformer encoder layers, it uses a hidden size of 256 and 4 heads. Compared to the LT, the VLT only uses 2 decoder layers since historical documents have few annotated data. As in the original architecture, we combine a CTC loss with CE loss. It results in the following combined loss:

$$L = \lambda \cdot L_{CTC} + (1 - \lambda) \cdot L_{CE} \quad (1)$$

This leads to an encoder-decoder architecture with fewer parameters (5.6M compared to 7.7M for LT [11] and up to hundreds of millions for heavier architecture). It makes the VLT more efficient for adaptation tasks on few annotated data. As the VLT uses Transformer layers, it is well suited for efficient optical recognition and language modeling.

3.2 Low-Cost Additional Data

In the context of historical HTR, where annotated data are limited, we introduce data strategies for training Transformer-based architectures. Like prior approaches, we employ data augmentation to deform real images. Additionally, we introduce novel and realistic synthetic historical handwritten text-lines created with cursive fonts.

3.2.1 Data Augmentation

We use data augmentations to introduce more variability into the training set. Each training sample is randomly augmented, with a probability of 0.2 to apply each augmentation. Augmentations include dilation and erosion for varying handwriting thickness, elastic distortions for local deformations, perspective transformations to alter text-line perspective, and the addition of Gaussian noise, followed by random padding.

3.2.2 Synthetic Data

To produce realistic synthetic historical handwritten text-lines, we combine cursive fonts with random augmentations, significantly reducing costs compared to real data annotation.

We begin by selecting random text from diverse sources, like Wikipedia, online historical books, or training datasets' ground truth. We select textual content to match the language and time period of the real dataset. We use manually reviewed cursive fonts to generate handwritings matching the visual style of historical documents.

We generate historical data following the process outlined in Fig. 2. With a given text and cursive font, we create a synthetic paragraph (a). We then apply random augmentations (b) and degradations (c) to obtain a realistic synthetic paragraph. See Table 1 for transformations details.

We extract separate text-lines by tracking their positions through each transformation. A visualization of a few synthetic text-lines images is available in Fig. 3. Since we generate entire paragraphs, overlapping lines are generated, which frequently appear in real historical documents.

This process generates realistic historical text-lines, providing extra annotated data without manual labeling. We randomly generate synthetic text-lines each epoch, allowing our network to learn from a variety of examples and increasing its capabilities to generalize. It might also help our architecture to learn to model the language since we can offer additional sentences to the network.

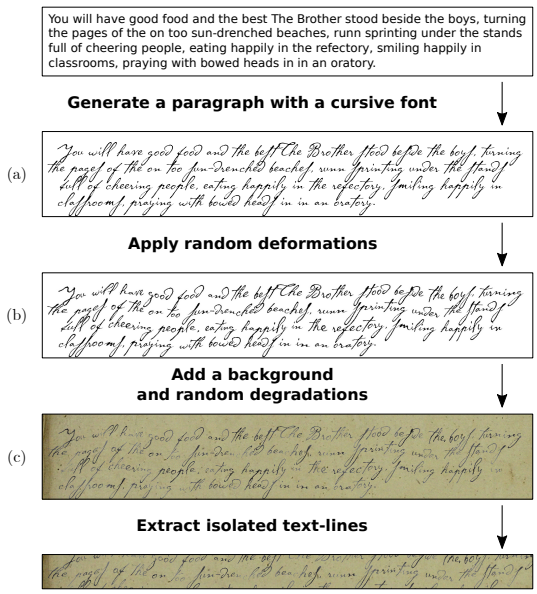


Fig. 2: Generation of synthetic historical data.

3.3 Adaptive Training Approach

We aim to transcribe a specific corpus where annotated data is limited to a few examples. To overcome the lack of labeled data, we pre-train our model on a generic dataset composed of various documents. Even if the documents differ (language or time period) from the specific corpus, we expect our model to perform better [1, 18]. Hence, to enable the training of Transformer-based architectures on low resource scenarios, we design a three-steps training strategy. They are focused on: the usage of synthetic data, an adaptation to the visual style of specific authors, and retaining generic language modeling abilities. Table 2 summarizes the three steps.

In step 1, we train a generic model on a generic labeled dataset composed of various corpora. In addition to real text-lines, we generate synthetic text-lines on-the-fly during each training epoch. 10% of the training data are reserved for validation, and we use early stopping to select the model with the lowest character error rate.

Step 2 extends training, using all generic data (including validation) and synthetic data. It allows us to train on a maximum amount of labeled data

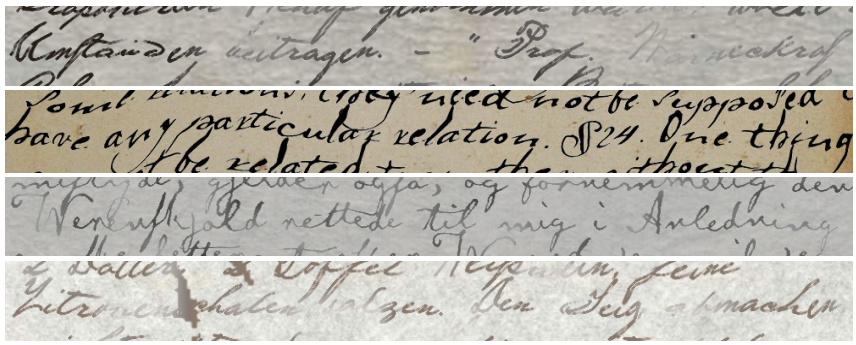


Fig. 3: Historical synthetic text-lines in diverse languages generated with our method.

Table 1: Details of transformations applied for synthetic data generation. We empirically assign a probability of 1.0 for transformations that we find beneficial for enhancing the realism of the synthetic image, and a probability of 0.2 to the others.

Type	Deformation	Probability	Details
Augmentations	Dilatation / Erosion	0.2	Up to ± 25% stroke width
	Elastic distortion	1.0	Similar to [6]. Both a local and a wide distortion
	Slant	1.0	Rotation ∈ $[0;\ \frac{\pi}{4}]$
	Rotation	0.2	Rotation up to ± $\frac{\pi}{60}$
	Perspective	0.2	Same as in [5]
	Brightness	0.2	Up to ± 25%
	Contrast	0.2	Up to ± 25%
	Sharpness	0.2	Up to ± 25%
Degradations	Ink stains	0.2	2 to 5 stains per paragraph
	Ink color	1.0	brownish ink
	Ink fading	1.0	Random fading using a Perlin noise
	Background	1.0	Empty backgrounds from historical documents

while they are limited. We limit the number of epochs to mitigate overfitting, as no validation set is used. This step offers a sound basis before adapting for a specific document.

Lastly, in step 3, we specialize our model on a specific document, typically with limited annotated data (500 or fewer text-lines from a single author). We split the available specific data into 90% for the training set and 10% for the validation set. A k-fold cross-validation is also considered due to the limited amount of annotated data in validation. The encoder is adapted to the specific handwriting style of the author and document degradations. Decoder weights remain fixed to preserve generic language modeling capabilities and prevent overfitting. We do not generate synthetic data since it would result in visually different styles. Early stopping is performed, and we select the model that achieves the lowest error rate on validation.

Table 2: Summary of our training procedure. For each training step, the table shows data used during training (G stands for Generic dataset and S for Specific dataset), parts of the network trained and the stopping criterion.

Step	Training data	Validation data	Synth. data	Encoder training	Decoder training	Stopping criterion
1	Train. G	Val. G	✓	✓	✓	early stopping
2	Train. G + Val. G	-	✓	✓	✓	few epochs
3	Train. S	Val. S	×	✓	fixed	early stopping

3.4 Prediction Algorithm

Despite the proposed strategies, the lack of annotated data still induces insufficient performance. Soullard et al. [19] showed that neural networks are sensitive to variations of text-line images. They address this issue by combining multiple predictions obtained from test-time augmentations. In this article, we propose an enhanced version of Soullard et al.'s prediction algorithm to reduce errors.

Fig. 4 illustrates our process. Each real image is randomly augmented $n - 1$ times, resulting in n images with one instance being the

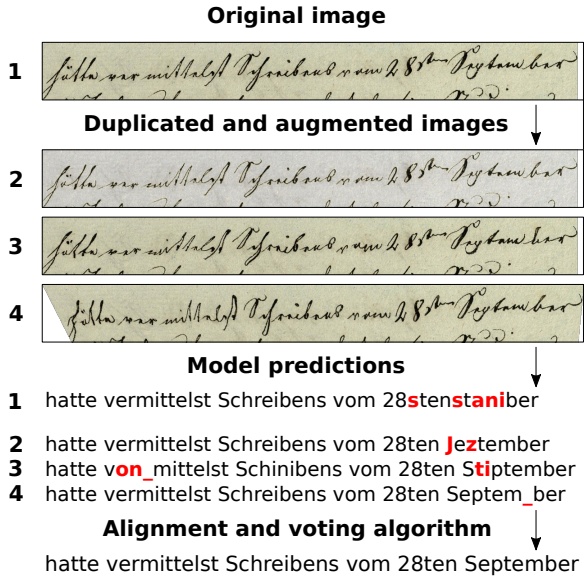


Fig. 4: Prediction algorithm: predictions obtained from the initial image and randomly augmented images (containing different errors) are merged to obtain a correct sequence.

Algorithm 1 Prediction algorithm

Input: P, τ
Output: v

$E \leftarrow \text{AVERAGEED}(P)$
$P, E \leftarrow \text{FILTER}(P, E, \tau)$
Sort P based on E
$A \leftarrow \emptyset$
for $p_i \in P$ **do**
    $A \leftarrow \text{INSERTANDALIGN}(p_i, A)$
end for
$v \leftarrow \text{VOTE}(A)$

unchanged real image. Our model is then used to obtain predictions. It results in n predictions $P = \{p_i \mid i \in [1, n]\}$, likely close to each other. The process is described in Algorithm 1.

As a first step, function AVERAGEED() computes for each prediction p_i in P, the average normalized edit distance e_i between that prediction and every other prediction in P:

$$E = \{e_i \mid i \in [1, n];\ e_i = \text{AVERAGEED}(p_i, P \setminus \{p_i\})\}$$

$$\text{AVERAGEED}(p_i, P') = \frac{\sum_{p_j \in P'} \text{ED}(p_i, p_j)}{|P'|}$$

We subsequently use a threshold τ to filter predictions P based on the value from E. If a prediction p_i obtains an average edit distance e_i superior to τ, we discard it from the set of predictions P (and E). Intuitively, if a prediction obtains a high average normalized edit distance, it considerably differs from the other ones. Therefore, discarding it from the predictions P allows our algorithm to predict a more reliable final prediction and to be resistant to outliers.

Using function INSERTANDALIGN(p_i, A), the set A of aligned predictions is constructed by inserting and realigning predictions in A with the given prediction p_i. As we sort P based on the values from E, we start with the predictions that are the closest to the others. The edit distance algorithm is used to provide the fewest changes between sequences with a wildcard token '$*$' denoting an inserted character. When all the predictions are aligned (i.e. of same length), VOTE() selects characters that obtain the most votes for each frame in A. It results in the voted prediction v after removing the wildcard token '$*$'.

4 Experiments

In this section, we demonstrate the benefits of the proposed approach for historical documents and the task of adaptation with few examples: (i) the interest of using synthetic data on a generic dataset; (ii) the interest of using a light architecture; (iii) the benefits of using specific training and prediction strategies. We validate our contributions by presenting results and a comparison with existing methods on the ICFHR 2018 READ historical handwritten dataset. We also provide results on modern datasets, IAM [27] and RIMES [28].

4.1 ICFHR 2018 READ Dataset

The ICFHR 2018 READ historical handwritten dataset [18] proposes a challenging task to evaluate the performance of one model trained with few annotated data. It includes a generic dataset of 11,903 annotated text-line images spanning 17 documents in German, English, Swedish, and Danish to pre-train a model.

Next the goal is to adapt this model on specific documents and transcribe them using few labeled data. The specific dataset contains five

documents: Konzilprotokolle C, Schiller, Ricordi, Patzig and Schwerin. They are all written in German, except Ricordi which is instead written in Italian. This is another challenge since the generic set does not include any Italian scripts. The competition assesses adaptation capabilities with different numbers of annotated pages available for each specific document: 0 (meaning that no adaptation is possible), 1, 4 or 16. Nevertheless, having 16 pages is however difficult, with 500 labeled text-lines on average. Finally, each model is evaluated on a test set corresponding to new data from the specific document (250–700 text-lines). Results are presented by pages (average across all documents), by documents (average across all pages numbers), or as a global average of error rates (total error in [18]).

4.2 Experimental Settings

4.2.1 Training Details

We first use the generic set to train a generic model, before the adaptation task. The competition did not provide data splits for the generic data nor for specific datasets. Hence, we use custom data splits with 10% for validation. Our model takes RGB images as input to capture backgrounds and ink colors. In step 1 of our training strategy (see Table 2), we use an early stopping of 200 epochs. In step 2, we train our model for 10 additional epochs, including the validation set as training data. Lastly, our model is trained on specific documents using an early stopping of 50 epochs. Specific documents contain few text-lines due to the limited number of available training examples (i.e. a total of 20–70 text-lines for one page). Therefore, we perform a 4-fold cross-validation.

To train our architecture, we apply Vaswani's learning rate policy [7]. The learning rate linearly increase to a maximum value of 0.01 after 4,000 steps, followed by an exponential decay. We use the *Adam* algorithm to train our models. We set $\lambda = 0.5$ (Eq. 1) for the two loss functions, as they are of similar order of magnitude. Teacher forcing is used during training, with the decoder receiving the shifted-right target sequence. Masked causal attention in decoder layers ensures access to only past characters for prediction. At test time, character prediction continues until an end-of-sequence token or a maximum sequence length of 128 is reached.

Given the insufficient amount of annotated data to train a Transformer-based architecture, we use synthetic data during the training on the generic dataset. We use textual contents coming from 13,000 articles from Wikipedia (15M text-lines), 9,000 publicly available historical books[3] (16M text-lines). We also use the ground truth at the paragraph level of the ICFHR 2018 READ generic data (11,903 text-lines) to generate text-lines with textual content from consecutive lines. Textual contents from Wikipedia and historical books are selected to match the languages of the generic dataset (German, English, Swedish and Danish), and we use a similar language distribution. 21 manually selected historical handwritten fonts are used[4]. Each training epoch includes 50% of real annotated data and 50% of generated synthetic data.

Note that the 2018 competition protocol did not allow for additional data to train models. While most experiments include all available textual content, we also report results without supplementary textual content to align with the competition protocol.

4.2.2 Prediction Details

We outline our prediction algorithm and evaluation process. Each image is augmented 59 times, which results in $n = 60$ different images, including the original. We picked $n = 60$ as it offers a good trade-off between accuracy and prediction speed. We use best-path decoding to generate individual predictions, which are then merged using our prediction algorithm with a threshold of $\tau = 0.75$. As we perform a 4-fold cross-validation, we obtain 4 distinct models for each scenario. Therefore, we average the results obtained by these 4 four models to obtain the final performance. We present results using Character Error Rate (CER) and Word Error Rate (WER).

Table 3: Impact of using various textual data for synthetic data when pre-training models on the generic dataset. Results are presented on the specific ICFHR 2018 READ dataset, without any adaptation (0 page), or on an average of all tasks (0–16 pages).

Textual Content for Synthetic Data Generation	0 page for adaptation		Average error rates (0–16 pages)	
	CER (%) ↓	WER (%) ↓	CER (%) ↓	WER (%) ↓
No synthetic data	31.75	77.38	20.97 ± 1.51	53.24 ± 7.14
ICFHR 2018 READ	24.38	66.07	13.11 ± 0.51	40.68 ± 2.04
Wikipedia	26.38	69.51	14.42 ± 0.73	42.68 ± 1.17
Historical books	25.07	67.50	13.97 ± 0.43	42.07 ± 0.88
All synthetic data	**24.28**	**65.38**	**12.96** ± 0.42	**40.33** ± 1.33

4.3 Historical Synthetic Data

4.3.1 Impact of the Textual Content

In this section, we evaluate the benefits of using synthetic data during the pre-training on generic data, and first, the impact of different textual contents. Table 3 shows the results these models obtained on the specific data. No matter the textual content used, the number of synthetic lines used for each epoch remains the same (∼10,000).

Compared to not using synthetic data (Table 3, first row), any approach using synthetic data shows a lower CER. In particular, synthetic data limited to the textual content from the ICFHR 2018 READ dataset demonstrates that our model benefits from new visual styles as the vocabulary remains the same. In comparison, using Wikipedia articles or historical books results in slightly worse CERs. Such textual data might not provide an appropriate vocabulary for the given specific documents. We observe a broader difference when using articles from Wikipedia as the vocabulary is modern. Yet, when all the textual contents are combined, we reach the lowest error rates. It therefore demonstrates the impact of new textual contents, with a more diverse vocabulary.

4.3.2 Impact of the Visual Style

Then, we investigate the impact of the generated visual style for synthetic data. In Table 4, we therefore compare results obtained with synthetic data using raw fonts (as in Fig. 2a), augmentations only (Fig. 2b) or the complete pipeline using degradations in addition to augmentations (Fig. 2c). However, the visual style of synthetic data might have a reduced impact when there is annotated data available to adapt the encoder of the model to the visual style of a specific document.

Table 4: Impact of using augmentations (aug.) and degradations (degrad.) for synthetic data on the ICFHR READ 2018 dataset.

	0 page CER (%) ↓	Average CER (%) ↓ (0–16 pages) per test document					Average CER (%) ↓
		Konzil. C	Schiller	Ricordi	Patzig	Schwerin	
Raw fonts	25.12	6.42	14.15	17.61	18.20	11.15	13.67 ± 0.80
+ aug.	24.68	6.22	**13.86**	16.49	16.74	**10.85**	**12.95** ± 0.55
+ degrad.	**24.28**	**6.16**	13.91	**15.21**	**16.66**	11.71	12.96 ± 0.42

Compared to using raw fonts only, augmentations bring a reduction of the CER in every scenario thanks to an increase variability in the writing styles.

The impact of degradations varies across scenarios. For the specific document Ricordi, degradations significantly improve results as the associated test data contains many degraded handwritings. We also note a sharp gap when no adaptation is possible (0 page). However, for the Schwerin document, we observe worse CERs by using degradations as this specific document contains well-preserved ink and few degradations. On the remaining documents, as well as on the average over all scenarios, we observe no significant differences in the obtained CERs. Even with no adaptation (0 page), degradations reduce CERs, showing their value in generating diverse synthetic data for improving the model generalization. Hence, we use synthetic data with both augmentations and degradations as it offers a relevant solution to handle the most difficult historical documents.

Synthetic data yield a major improvement and are therefore valuable for training Transformer-based models. They are especially useful to bring a new vocabulary that greatly reduce the error rates obtained by our models. The visual style of synthetic data also benefits to the error rates, but yet with less impact. However, training with synthetic data results in longer training time. We observe our model requires to see 6–8 more text-lines images before converging. This is caused by

[3]We use books available at https://www.gutenberg.org/.

[4]The fonts are available on https://fonts.google.com, https://www.dafont.com and https://www.p22.com.

Table 5: Comparison between LT [11] and our VLT on the specific data of the ICFHR 2018 READ dataset. We also provide the number of text-lines processed each second during prediction without the prediction strategy, using a single NVIDIA Tesla V100.

Model	Params.	Average error rates		Text-lines per second
		CER (%) ↓	WER (%) ↓	
LT [11]	7.7M	13.36 ± 0.45	41.24 ± 3.47	34.03 ± 1.23
VLT	5.6M	**12.96** ± 0.42	**40.33** ± 1.33	**60.00** ± 2.42

an unseen and wide content that is also responsible for a significant improvement.

4.4 Benefits of a Very Lightweight Architecture on Few Data

We now demonstrate the advantage of the *Very Lightweight Transformer* (VLT) architecture, designed for few annotated data. The VLT is compared to the previous Light Transformer (LT) [11], and both are trained with synthetic data. Table 5 presents results on the specific data of the ICFHR 2018 READ historical dataset. These results demonstrate the interest of the VLT for the adaptation to specific documents.

With a reduced number of parameters (5.6M instead of 7.7M), the VLT achieves a lower average CER than the LT. We observe a reduction in the empirical variance, especially for the WER. Therefore, the VLT is easier to train on the very limited number of annotated data. In addition, the VLT enables reduced prediction times, with the ability to produce predictions for almost twice as much text-line images per second. With such speed, the VLT could be used to process approximately 2 pages each second. However, as the prediction strategy process additional images, it results in less speeds (i.e. 60 times slower with 60 duplicated images). We do not find any significant difference in the training time between the architectures.

4.5 Training Strategy Effects

We assess our training strategy (see Section 3.3) that adapts the encoder to specific authors' handwriting while keeping the decoder generic. To assess this impact, we use the VLT that is first trained using synthetic data, and compare results using or not the strategy. Results are given inside table 6 by also using the prediction strategy.

Table 6: Interest of our training strategy to reduce the CER on the ICFHR 2018 READ dataset. The first four columns show CERs obtained with a given number of labeled pages, averaged over the five specific documents.

Training Strategy	CER (%) ↓ per specific training page				Average CER (%) ↓
	0	1	4	16	
no	24.28	14.67	10.18	6.50	13.91 ± 1.23
yes	**24.28**	**13.03**	**8.89**	**5.64**	**12.96** ± **0.42**

Table 7: Interest of our prediction strategy (test-time augmentations combined with a voting algorithm) to reduce the CER on the ICFHR 2018 READ dataset.

Prediction Strategy	CER (%) ↓ per specific training page				Average CER (%) ↓
	0	1	4	16	
no	25.07	14.18	9.61	6.20	13.77 ± 0.48
yes	**24.28**	**13.03**	**8.89**	**5.64**	**12.96** ± **0.42**

We observe lower CERs (6% relatively) by fixing the weights of the decoder, which proves the benefit of using the training strategy when adapting with few labeled data. We also note that the training strategy greatly reduces the empirical variance of the results. By fixing the decoder, the VLT is able to maintain generic language capabilities as the decoder is trained on lots of (real and synthetic) annotated data. The fine-tuning is focused on the task of adapting the encoder to the visual style of the writers, which bring lower CERs. Learning the writer's language instead would lead to overfitted models since there is not enough variability in the limited amount of annotated data.

4.6 Prediction Strategy Assessment

We now show the value of the prediction algorithm, that apply multiple test-time augmentations and combine them using a voting algorithm. Table 7 compares results with or without using the prediction strategy, by using the VLT trained with synthetic data and our training strategy.

Compared to not using the algorithm, the approach achieves a relative CER reduction of 5–6% globally. In addition, we observed lower CERs regardless of the scenario considered. The prediction strategy produces greater improvements on adapted models (8%), whereas we observe a smaller improvement when no specific pages is available (3%). When fine-tuning, our model might be sensitive to slight variations in the input images, which are smoothed out with the prediction strategy to improve recognition. It therefore demonstrates the benefits of using a prediction algorithm, in particular when fine-tuning.

Combined together, our training and prediction strategies enable a relative CER reduction of 12% on the VLT. We also observe a reduction in the empirical variance of the error rate when using these strategies. They therefore help to consolidate the training and evaluation of such models.

4.7 Comparison with State-of-the-art Methods

Table 8 compares our VLT architecture, trained with synthetic data as well as the proposed strategies, with the state-of-the-art methods on the ICFHR 2018 READ dataset. Meanwhile, Table 9 provides a comprehensive view of the results obtained on each scenario, including the empirical standard deviations.

We provide results using all textual content for synthetic data generation. Additionally, we provide "VLT (Competition)" results, strictly following the competition guidelines, where synthetic data are generated using only textual data provided by the competition. The VLT (Competition) obtains an average CER of 13.11% at the level of the best performing approach [5]. Using additional textual content, the VLT outperforms existing methods. Note that additional textual content might be unbeneficial for the documents Schiller and particularly Schwerin. This is particularly true for Schwerin that yet contains many annotated text-lines on each page (2–3 times more text-lines compared to other documents).

We now focus on the results obtained by the VLT using additional textual content. It obtains the best average CER with a value of 12.96%. Speaking of documents separately, it outperforms existing methods on two documents: Ricordi and Schwerin with an average CER of 15.21% and 11.71% respectively. For the Italian document Ricordi, we mind a significant difference of 2% with the second method. As the VLT is not trained on Italian data, the encoder is responsible for this gap. This difference may be explained by the use of synthetic data and handwriting fonts that closely match Ricordi's handwriting style. Therefore, our model can be adapted to data that differ from the generic set, here with an unseen language.

With 0 additional specific pages, the VLT obtains the lowest CER (24.28%) compared to other methods from the state of the art. This result demonstrates our approach, although trained with synthetic data, is capable of outperforming traditional methods on new corpora.

By adapting with 1 and 4 annotated pages, our architecture obtains the second lowest CER, close to ASIUT [5]. The fully convolutional network from ASIUT shows better adaptation abilities with very few annotated data. It has fewer parameters as well as an efficient design, making it more data-efficient. Although Transformer-based, the VLT shows excellent adaptation results even with the most limited amount of annotated data.

Nonetheless, we obtain a CER of 5.64% with 16 pages and outperforms other methods from the state of the art. Table 10 details the results with 16 additional specific pages for each specific document. The VLT outperforms the other methods on two out of five documents: Schiller and Ricordi, with a notable difference of 1.3 points on Ricordi which is written in Italian. Nevertheless, on the others documents, the VLT obtains results close to the best performing architectures. Notably, our architecture obtains a CER lower than 3% on two specific documents and lower than 10% on each of the 5 documents. This is considered as an acceptable error rate for manual corrections [18]. It demonstrates that the VLT, trained with the proposed strategies, is well designed for the adaptation on limited annotated data.

4.8 Evaluation on Modern Data

Lastly, we evaluate the proposed VLT architecture on modern datasets: the IAM dataset [27] (10,373 text-lines) to assess performances on modern English handwritings and the RIMES dataset [28]

Table 8: Comparison of VLT with state-of-the-art methods on the ICFHR 2018 READ dataset.

Method	CER (%) ↓ per specific training page				CER (%) ↓ per test document					Average CER (%) ↓
	0	1	4	16	Konzil. C	Schiller	Ricordi	Patzig	Schwerin	
OSU [18]	31.39	17.73	13.27	9.02	9.39	21.10	23.27	23.17	12.98	17.86
ParisTech [18]	32.25	19.79	16.98	14.72	10.49	19.05	35.60	23.83	17.02	20.94
LITIS 2018 [18]	35.29	22.51	16.89	11.34	9.14	25.69	30.50	25.18	18.04	21.51
PRHLT [18]	32.79	22.15	17.90	13.33	8.65	18.39	35.07	26.26	18.65	21.54
RPPDI [18]	30.80	28.40	27.25	22.85	11.90	21.88	37.29	32.75	28.55	27.32
ASIUT [5]	25.35	**12.63**	**8.28**	5.82	6.49	**13.77**	17.33	**14.85**	12.33	13.02
LITIS 2019 [19]	26.57	15.47	10.00	5.82	**5.94**	14.81	21.62	18.08	11.73	14.46
VLT (Competition)	24.38	13.07	9.19	5.81	6.29	13.87	15.96	17.24	**11.21**	13.11
VLT	**24.28**	13.03	8.89	**5.64**	6.16	13.91	**15.21**	16.65	11.71	**12.96**

Table 9: Results of our approach on each scenario (a given document and a number of annotated pages) for the ICFHR 2018 READ test set: average CER and standard deviation from the 4-fold cross-validation.

Document	CER (%) ↓ for each scenario				
	0 page	1 page	4 pages	16 pages	Average
Konzil. C	10.54	6.81 ± 0.73	4.33 ± 0.32	2.95 ± 0.12	6.16 ± 0.46
Schiller	21.20	14.92 ± 0.12	11.86 ± 0.26	7.64 ± 0.31	13.91 ± 0.24
Ricordi	22.96	16.72 ± 0.89	12.77 ± 0.13	8.41 ± 0.29	15.21 ± 0.54
Patzig	29.37	17.46 ± 0.60	12.10 ± 0.07	7.67 ± 0.23	16.65 ± 0.37
Schwerin	30.09	9.37 ± 0.70	4.75 ± 0.13	2.63 ± 0.08	11.71 ± 0.41
Average	24.28	13.03 ±0.66	8.89 ± 0.18	5.64 ± 0.22	12.96 ± 0.42

Table 10: Comparison of VLT with the state of the art of the CER obtained on the ICFHR 2018 READ test set, with the maximum number of annotated data (16 pages) on each of the five documents.

Method	CER (%) ↓ per test document with 16 pages				
	Konzil. C	Schiller	Ricordi	Patzig	Schwerin
OSU [18]	3.79	12.45	15.04	12.54	3.50
ParisTech [18]	8.02	14.58	30.20	15.51	9.18
LITIS 2018 [18]	4.81	19.57	16.37	12.83	6.61
PRHLT [18]	4.98	12.55	28.52	16.35	7.12
RPPDI [18]	9.18	16.29	30.49	28.30	24.90
ASIUT [5]	2.83	8.17	11.44	**6.73**	**2.28**
LITIS 2019 [19]	**2.73**	8.41	9.72	7.19	2.74
VLT (Competition)	2.94	7.80	8.62	8.12	2.64
VLT	2.95	**7.64**	**8.41**	7.67	2.63

(12,058 text-lines) for modern French handwritings. We also compare VLT against other methods from the state of the art.

The VLT architecture is trained with synthetic data designed for modern (either English or French) handwritings [11]. Predictions are obtained by using the proposed decoding strategy.

Table 11: Comparison of the CER (%) obtained by our VLT architecture and a CRNN architecture, with restricted amount of annotated data from the IAM dataset. The training and prediction strategies are not used.

Method	Size of the dataset				
	10%	25%	50%	75%	100%
CRNN	14.07	9.75	8.01	7.68	6.35
CRNN + synth.	9.34	7.15	6.44	6.22	5.78
VLT (+ synth.)	**8.93**	**6.36**	**5.37**	**4.97**	**4.41**

We use the usual training, validation and test sets[5] to train our network from scratch and we therefore do not used the prediction strategy we proposed for historical documents in Section 3.4.

4.8.1 Training with Few Data

In Table 11, we evaluate the performance of our VLT architecture with restricted amount of annotated data from the IAM dataset. The results are also compared to a baseline CRNN architecture, inspired from Puigcerver [2]. We either use 10, 25, 50, 75 or 100% of the annotated data from both the training and validation set. The test set remained unchanged. We nevertheless use data augmentations and synthetic data to leverage low amounts of annotated data.

Compared to the baseline CRNN architecture, the VLT architecture obtains lower error rates. While we observe a small difference on the lowest amount of training data, the gap between the

[5]On the IAM dataset, the *Aachen* split is used.

Table 12: Comparison with state-of-the-art methods on the IAM dataset.

Method	# params.	IAM	
		CER (%) ↓	WER (%) ↓
CRNN [2]	9.6M	4.4	
CRNN [3]	-	5.7	17.82
CRNN + LSTM [4]	-	4.87	
VAN (line level) [6]	1.7M	4.97	16.31
FCN [5]	3.4M	4.9	
CNN-SAN [13]	1.7M	7.50	25.46
LT [11]	7.7M	4.33	14.75
CRNN-Transformer [12]	24M	3.13	**8.94**
Bidi. Transformer [8]	27M	5.67	
FPHR Transformer [10]	28M	6.5	
Transformer [9]	100M	4.67	15.45
$\text{TrOCR}_{\text{SMALL}}$ [15]	62M	4.22	
$\text{TrOCR}_{\text{LARGE}}$ [15]	558M	**2.89**	
VLT	5.6M	4.23	14.61

Table 13: Comparison with state-of-the-art methods on the RIMES dataset.

Method	# params.	RIMES	
		CER (%) ↓	WER (%) ↓
CRNN [2]	-	**2.3**	
CRNN [3]	-	5.07	14.70
VAN (line level) [6]	1.7M	3.08	8.14
LT [11]	7.7M	2.51	**7.26**
CRNN-Transformer [12]	24M	3.19	
VLT	5.6M	2.82	8.47

two architectures gets bigger with more annotated data. With only 10% of the training data (650 text-lines), our approach is able to reach a CER lower than 10%. Additionally, 4,500 text-lines are enough to reach a CER below 5%. It therefore demonstrates that VLT is also able to be trained efficiently on limited amount of modern data. The addition of synthetic data has a significant impact on the lowest amount of annotated data.

4.8.2 Evaluation on IAM and RIMES datasets

In Tables 12 and 13, we present the results obtained by VLT on the full IAM and RIMES datasets respectively. We omit methods that use external annotated data to allow a proper comparison, with the exception of synthetic data. On the IAM dataset, our approach attains a CER of 4.23% and a WER of 14.61%, which is competitive with the state of the art. Notably, VLT allows lower error rates compared to many Transformer architectures while also using a reduced number of weights. Similar results are observed on the RIMES dataset, where our approach reaches a CER of 2.82% and a WER of 8.47%. These results suggest a large architecture is not required to reach competitive results on common datasets.

5 Conclusion

In this paper, we introduced a method to train Transformer-based architectures for historical handwritten document recognition focusing on specific document adaptation tasks. We took advantage of a lightweight Transformer-based architecture and its benefits for historical datasets with few annotated data and proposed various strategies. Specifically, we introduced: (i) a Very Lightweight Transformer-based (VLT) architecture; (ii) the design of historical synthetic data; (iii) a training strategy in which only the encoder is adapted to a specific document; (iv) a prediction algorithm based on a combination of multiple predictions. Altogether, our approach reaches the lowest average CER of 12.96% on the ICFHR 2018 READ dataset compared to state-of-the-art methods. In particular, our approach obtains the lowest CER on specific documents with no labelled data (0 page) for an adaptation task, and with 16 annotated pages. The obtained results show our approach is properly designed to generalize on new documents even if the language differs. It also demonstrates excellent results on modern documents such as IAM and RIMES.

Despite the complexity of training Transformer-based architectures with limited data, our strategies offer efficient solutions, leading to improvements in historical handwritten document recognition compared to existing methods. Our approach and its ideas could be extended to other types of documents and other architectural models when few labeled data are available. We also believe future works might be carried out toward the most challenging scenarios where the number of labeled data is heavily reduced or where the language is complex to learn.

Acknowledgments. This work was performed using HPC/AI resources from GENCI-IDRIS (Grant 2021-AD011012550). We would also like to

thanks Solène Tarride for her contribution on the degradations used to generate synthetic data.

References

[1] Bluche T, Messina R (2017) Gated convolutional recurrent neural networks for multilingual handwriting recognition. In: ICDAR, pp 646–651

[2] Puigcerver J (2017) Are multidimensional recurrent layers really necessary for handwritten text recognition? In: ICDAR, pp 67–72

[3] Dutta K, Krishnan P, Mathew M, et al (2018) Improving cnn-rnn hybrid networks for handwriting recognition. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, pp 80–85

[4] Michael J, Labahn R, Grüning T, et al (2019) Evaluating sequence-to-sequence models for handwritten text recognition. In: ICDAR, pp 1286–1293

[5] Yousef M, Hussain KF, Mohammed US (2020) Accurate, data-efficient, unconstrained text recognition with convolutional neural networks. PR 108:107,482

[6] Coquenet D, Chatelain C, Paquet T (2022) End-to-end handwritten paragraph text recognition using a vertical attention network. PAMI

[7] Vaswani A, Shazeer N, Parmar N, et al (2017) Attention is all you need. In: NIPS, pp 5998–6008

[8] Wick C, Zöllner J, Grüning T (2021) Transformer for handwritten text recognition using bidirectional post-decoding. In: ICDAR, pp 112–126

[9] Kang L, Riba P, Rusiñol M, et al (2022) Pay attention to what you read: non-recurrent handwritten text-line recognition. PR 129:108,766

[10] Singh SS, Karayev S (2021) Full page handwriting recognition via image to sequence extraction. In: ICDAR, pp 55–69

[11] Barrere K, Soullard Y, Lemaitre A, et al (2022) A light transformer-based architecture for handwritten text recognition. In: DAS, pp 275–290

[12] Wick C, Zöllner J, Grüning T (2022) Rescoring sequence-to-sequence models for text line recognition with ctc-prefixes. In: DAS, pp 260–274

[13] d'Arce R, Norton T, Hannuna S, et al (2022) Self-attention networks for non-recurrent handwritten text recognition. In: ICFHR, pp 389–403

[14] Coquenet D, Chatelain C, Paquet T (2023) Dan: a segmentation-free document attention network for handwritten document recognition. PAMI https://doi.org/10.1109/TPAMI.2023.3235826

[15] Li M, Lv T, Chen J, et al (2023) Trocr: Transformer-based optical character recognition with pre-trained models. In: AAAI

[16] Riaz N, Arbab H, Maqsood A, et al (2022) Conv-transformer architecture for unconstrained off-line urdu handwriting recognition. IJDAR https://doi.org/10.1007/s10032-022-00416-5

[17] Graves A, Fernández S, Gomez F, et al (2006) Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: ICML, pp 369–376

[18] Strauß T, Leifert G, Labahn R, et al (2018) Icfhr2018 competition on automated text recognition on a read dataset. In: ICFHR, pp 477–482

[19] Soullard Y, Swaileh W, Tranouez P, et al (2019) Improving text recognition using optical and language model writer adaptation. In: ICDAR, pp 1175–1180

[20] Fogel S, Averbuch-Elor H, Cohen S, et al (2020) Scrabblegan: Semi-supervised varying length handwritten text generation. In: ICCV, pp 4324–4333

[21] Davis B, Tensmeyer C, Price B, et al (2020) Text and style conditioned gan for generation of offline handwriting lines. In: BMVC

[22] Gan J, Wang W (2021) Higan: Handwriting imitation conditioned on arbitrary-length texts and disentangled styles. In: AAAI, pp 7484–7492

[23] Bhunia AK, Khan S, Cholakkal H, et al (2021) Handwriting transformers. In: ICCV, pp 1086–1094

[24] Vögtlin L, Drazyk M, Pondenkandath V, et al (2021) Generating synthetic handwritten historical documents with ocr constrained gans. In: ICDAR, pp 610–625

[25] Journet N, Visani M, Mansencal B, et al (2017) Doccreator: A new software for creating synthetic ground-truthed document images. Journal of imaging 3(4):62

[26] Madi B, Alaasam R, Droby A, et al (2022) Hst-gan: Historical style transfer gan for generating historical text images. In: DAS, pp 523–537

[27] Marti UV, Bunke H (2002) The iam-database: an english sentence database for offline handwriting recognition. IJDAR 5(1):39–46

[28] Augustin E, Carré M, Grosicki E, et al (2006) Rimes evaluation campaign for handwritten mail processing. In: IWFHR, pp 231–235