A Light Transformer-Based Architecture for Handwritten Text Recognition

Killian Barrere, Yann Soullard, Aurélie Lemaitre, Bertrand B. Coüasnon

A Light Transformer-Based Architecture for Handwritten Text Recognition

Killian Barrere, Yann Soullard, Aurélie Lemaitre, and Bertrand Coüasnon

Univ. Rennes, CNRS, IRISA, Rennes, France

This article has been accepted in the 15th IAPR International Workshop on Document Analysys Systems (DAS 2022). The final authenticated version is available at https://doi.org/10.1007/978-3-031-06555-2_19.

Abstract. Transformer models have been showing ground-breaking results in the domain of natural language processing. More recently, they started to gain interest in many others fields as in computer vision. Traditional Transformer models typically require a significant amount of training data to achieve satisfactory results. However, in the domain of handwritten text recognition, annotated data acquisition remains costly resulting in small datasets compared to those commonly used to train a Transformer-based model. Hence, training Transformer models able to transcribe handwritten text from images remains challenging. We propose a light encoder-decoder Transformer-based architecture for handwriting text recognition, containing a small number of parameters compared to traditional Transformer architectures. We trained our architecture using a hybrid loss, combining the well-known connectionist temporal classification with the cross-entropy. Experiments are conducted on the well-known IAM dataset with and without the use of additional synthetic data. We show that our network reaches state-of-the-art results in both cases, compared with other larger Transformer-based models.

Keywords: Light Network · Hybrid Loss · Transformer · Handwritten Text Recognition · Neural Networks

1 Introduction

Handwritten Text Recognition (HTR) refers to the process of automatically recognizing the text written inside an image of a text-line, a paragraph or even whole pages, after a first step of document layout analysis. This task is valuable nowadays for a growing number of people as it enables the text to be available in a digitized format and resist in time. There is still a tremendous amount of document collections waiting to be processed, and HTR models have shown acceptable error rates on specific documents [16]. This is mostly thanks to the recent advances and growing interest in deep learning approaches. However, HTR remains challenging for a variety of reasons. The variability of writing styles, degraded documents, the need for data matching documents and the lack of annotated records limit the abilities of current deep learning approaches.

Popular network architectures are based on both convolutional layers and long short-term memory layers, trained using the Connectionist Temporal Classification (CTC) loss function [8]. These layers result in an optical model, typically followed by a language model which aims to correct recognition errors. Fully Convolutional Networks have been investigated recently with the goal to significantly reduce training time by removing recurrent layers. Lately, few works based on Transformers [17] have been proposed for HTR [10,14,18]. Based on multi-head attention layers, a Transformer network proves to be an efficient alternative to recurrent layers. It enables a sequential analysis of the sequence thanks to positional encoding and efficient parallelism. Transformer-based models offer a great potential, but those architectures generally require a tremendous amount of annotated training data to be robust. This limits the efficiency of such networks in HTR tasks where annotated data are expensive.

In this paper, we propose a light encoder-decoder Transformer-based architecture for handwriting recognition. The network presented in this work is significantly smaller than traditional Transformer networks, which makes it lighter in the number of weights and easier to train on small datasets. Our architecture is trained using a hybrid loss combining both the CTC loss and the Cross-Entropy (CE) loss, which seems crucial.

The article is organized as follows. After reviewing related works in Section 2, our network architecture is presented with details in Section 3. Afterward, we present results obtained on the popular IAM dataset. As opposed to traditional Transformer approaches, we show that our approach reaches state-of-art results without any additional data nor transfer learning. Nevertheless, it is capable to outperform the other types of network architectures by using synthetic data.

2 Related Works

2.1 Standard approaches for HTR

Popular network architectures used for Handwriting Text Recognition combine both convolutional layers and recurrent layers. A number of convolutional layers are stacked at the beginning of the network to extract local features from text-line images. Then, recurrent layers, and more specifically Bi-directional Long Short-Term Memory (BLSTM) layers are stacked to process the features sequentially and output character probabilities based on contextual dependencies. Such an architecture is frequently called a Convolutional Recurrent Neural Networks (CRNN) [3,13,15,16]. Models are generally trained using the well-known Connectionist Temporal Classification (CTC) loss [7]. It enables to deal with label sequences of shortest length than predicted sequences, without any knowledge about character segmentation.

Encoder-decoder based architectures have also been investigated for HTR. They typically rely on an attention mechanism and a decoder based on LSTM to sequentially predict the characters [2,6,12]. Michael et al. [12] propose to use a hybrid loss combining both a CTC loss applied to the encoder and a cross-entropy

loss for the decoder. Such models can obtain low error rates on common datasets. However, they suffer from the lack of computation parallelization inherently due to recurrent layers, which impacts both training and inference time.

Recently, Fully Convolutional Networks (FCN) have been proposed for HTR. They refer to deep architectures composed of many convolutional layers and no recurrent layer. They often include the most recent innovations in deep learning, like gating mechanism or residual connections to obtain state of the art results [5,4,9,19]. These architectures benefit from the computational parallelism offered by convolutional layers, and hence can be trained faster than traditional architectures based on recurrent layers. However, they may require tremendous work to be optimized well and require data augmentation to attain state-of-the-art results. By removing recurrent layers, Fully Convolutional Networks might struggle to learn long-range contextual dependencies, which can be useful in HTR.

CRNN architectures dominate in the field of HTR, thanks to their ability to learn local and long-range features. However, they highly suffer from the notable training time of recurrent layers. The last few years, Fully Convolutional Networks obtained state-of-the-art results but they might experience difficulties related to long-range contextual dependencies.

2.2 Transformer-Based Architectures

In natural language processing, multi-head attention have been proposed by Vaswani et al. [17] inside the so-called Transformer model. They propose an effective alternative to recurrent layers, capable of handling broad context in a constant amount of operation while enabling efficient parallelism. Transformer models also started to gain interest in the field of HTR. Kang et al. [10] propose an end-to-end Transformer that aims for both recognizing handwritten text and modeling the language. They obtain very low error rates by implementing a big network architecture that requires synthetic data to be trained efficiently. Singh et al. [14] propose a Transformer-based architecture to address the problem of full-page handwriting recognition. They obtain promising results on full-page recognition thanks to the use of synthetic data again. Wick et al. [18] use a bi-directional Transformer architecture coupled with a voting mechanism and show that their architecture outperforms a standard Transformer-based architecture.

Transformer-based layers propose an efficient trade-off between CRNN and Fully Convolutional Networks. Multi-head attention layers offer indeed both parallelism and the ability to learn long-range contextual dependencies. However, to perform well, a Transformer-based architecture, methods from the state of the art rely on synthetic data. Such additional data require to be designed as a complement to the training data which might remain a challenging task.

3 Our Light Encoder-Decoder Transformer-Based Model

Most Transformer-based architectures are based on large and deep models using many parameters and requiring a large amount of training data to perform well.

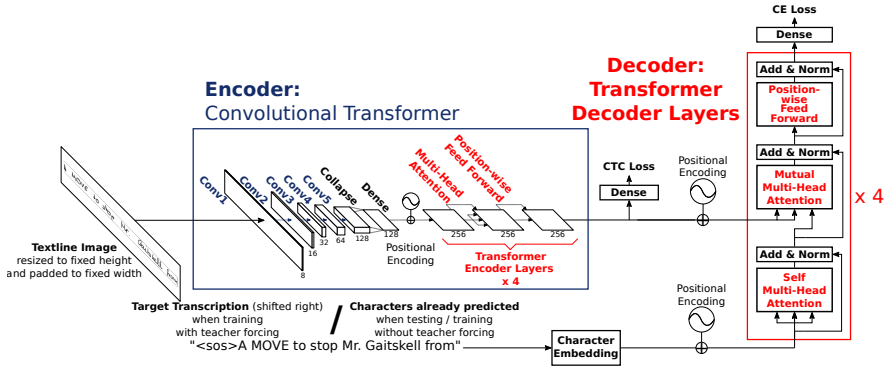

Fig. 1. Our encoder-decoder Transformer-based architecture. Our architecture is composed of an encoder combining convolutional layers and Transformer layers (Section 3.2), and of a Transformer-based decoder (Section 3.3).

Handwritten datasets typically contains too few examples for a Transformer-based model to perform well [10]. Synthetic data seem to be an efficient solution but designing valuable synthetic data might prove to be a complex task. While it takes efforts to design synthetic data, they may result in deteriorated performance, especially for difficult datasets like historical documents.

In this work, by contrast, we aim for a Transformer-based architecture capable of obtaining state-of-the-art results without the need of additional data, while still benefiting from the usage of synthetic data. We propose a light encoder-decoder Transformer-based architecture that can be trained efficiently on datasets of limited size, without the need for additional data[1]. This section provides details about our network architecture.

3.1 Summary of the architecture

We propose an end-to-end trainable Transformer-based architecture for HTR. Our network architecture is illustrated in Fig. 1. Our Transformer-based architecture remains low in the number of parameters, with 6.9M parameters, compared with traditional Transformer networks that might use up to 100M parameters. The architecture follows the principle of encoder-decoder models as it is composed of two key parts: an encoder and a decoder.

The encoder takes as input text-line images and aims to extract and process visual features. Our encoder is principally based on convolutional layers and Transformer encoder layers (Fig. 2). More details about the encoder are disclosed in Section 3.2.

The decoder subsequently uses the output of the encoder to sequentially predict the character sequence written inside the image (character by character).

[1] By additional data, we mean data from another dataset than the one which is studied and synthetic data. We nevertheless use data augmentation techniques to improve our models.

In addition, it also has access to the sequence of previously predicted characters. Thus the decoder might act as a language model at character level together with optical features from the input image. Section 3.3 provides more specific explanations about the decoder.

We train our models with a hybrid loss combining both the Connectionist Temporal Classification (CTC) loss [7] and the Cross-Entropy (CE) loss. This is discussed further in Section 3.4.

3.2 Network Encoder

The encoder is inspired by traditional CRNN and by Transformer layers [17]. We propose replacing the recurrent layers from the CRNN architecture by Transformer encoder layers. These layers refer to the encoder block of the Transformer architecture [17]. They are based on multi-head attention followed by a position-wise feed forward layer. Transformer encoder layers have the advantage of being more parallelizable than recurrent layers on GPU while being able to handle long-range contextual dependencies in a constant number of operations. Our convolutional Transformer encoder is illustrated in Fig. 2.

The first part of the encoder is composed of 5 convolutional blocks used to extract visual features from the image. Except the last one, each convolutional block is composed of a 2D convolutional layer with a kernel of size 3×3, a stride of 1 and no padding. The last convolutional block uses a kernel size of 4×2 to better match the shape of a character [3,13]. The number of filters in the convolutional layers are respectively equal to 8, 16, 32, 64 and 128. Each convolution layer is then followed by a LeakyReLU activation function. Following the activation function, we apply a layer normalization to ease the network training capabilities and increase the regularization capacities of the network. A 2×2 max pooling is used inside the first three convolutional blocks to decrease the size of intermediate feature maps. It also focuses the training process on the most impacting features and reduces the number of network parameters. Lastly, a dropout is applied with a probability of 0.2 at the end of each block.

Following the last convolutional block, a collapse layer is used to flatten the vertical dimension of the feature maps, therefore enabling us to easily work

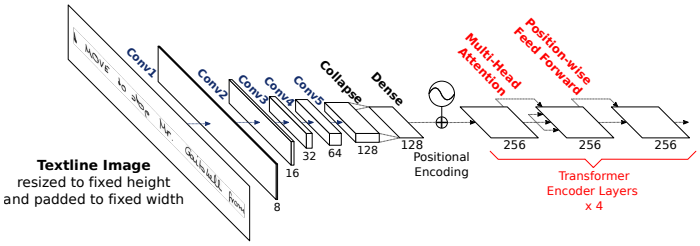

Fig. 2. Our Convolutional Transformer Encoder. This encoder is composed of a first stack of convolutional layers followed by Transformer Encoder layers.

with Transformer layers. It is composed of a convolution layer with a kernel size of width 1 and height similar to the height of the input feature maps. We subsequently apply a LeakyReLU activation function followed by a layer normalization.

Following that layer, we use a dense layer to increase the hidden size from 128 to 256. Before the Transformer encoder layers, sinusoidal positional encoding [17] is added to the output of the dense layer. We then use 4 stacked Transformer encoder layers. Each Transformer layer uses a hidden dimension of 256 and is composed of self-attention layers based on multi-head attention with 4 heads. They are then followed by a position-wise feed-forward layer with an intermediate feature size of 1024. Residual connections are used between each sub-layer. A dropout with a probability of 0.2 is applied to each sub-layer output.

The encoder of our architecture remains light in its number thanks to a small convolutional backbone, while additionally using a few numbers of parameters inside of Transformer layers. Compared to other Transformer-based architectures that use heavier convolutional neural networks like ResNets [10,14], our architecture only uses 5 convolutional layers. It results in a total of only 237k parameters for the convolutional backbone, which is lower than most of the convolutional backbone used in other Transformer-based architectures [10,14]. In addition, we maintain the number of intermediate neurons small inside Transformer layers, with a hidden size of 256. Heavier Transformer-based architectures might use up to 1,024 neurons inside Transformer layers.

3.3 Transformer Decoder

Our encoder-decoder Transformer-based model then processes the output from the encoder along with the sequence of previously predicted characters (or the target sequence shifted right when training with teacher forcing). An illustration of the decoder is shown in Fig. 1.

Our Transformer-based decoder uses both the output from the encoder part for mutual (or encoder-decoder) attention and the sequence of predicted characters. Sinusoidal positional encoding [17] is added to the output of the encoder. Despite the fact that our encoder already includes a positional encoding inside, we nevertheless find it beneficial to add a second positional encoding to the output of the encoder. Similarly to including an additional loss in the middle of a dense network, we believe that re-adding the information of the position in the middle of the architecture might help our architecture to converge. In a similar way, we apply positional encoding to the sequence of predicted characters, after applying a character-level embedding and before feeding it to the decoder part.

The Decoder is composed of a stack of 4 Transformer decoder layers [17]. Each layer takes as input the output from the previous layer or the embedded character sequence for the first layer. It is composed of a self-attention layer, using multi-head attention. Following the first sub-layer, a multi-headed encoder-decoder attention is applied. It uses the sequence coming from the encoder and apply weighted attention based on the encoder output and the output from the previous decoder sub-layer. Each attention sub-layer uses a hidden size of 256

and 4 heads. Lastly, we apply a position-wise feed forward layer composed of two dense layers with an intermediate feature size of 1,024. As used in the encoder, a residual connection is applied to each sub-layer and dropout of probability 0.2 is used.

3.4 Hybrid Loss

To train the entire model, we use a hybrid loss combining both the Connectionist Temporal Classification (CTC) loss function [7] and a Cross-Entropy (CE) loss function. Such a hybrid loss has been introduced fist in the domain of HTR by Michael et al. [12] to train an encoder-decoder model. In this work, both losses are linearly combined as follow:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{CTC} + (1 - \lambda) \cdot \mathcal{L}_{CE} \tag{1}$$

where λ balances between the CTC loss and CE loss. For a given input observation $\mathbf{x} = (x_1, ..., x_T)$ of length T with label $\mathbf{y} = (y_1, ..., y_L)$ of length L, we have:

$$\mathcal{L}_{CTC} = -ln\big(\sum_{\pi \in \mathcal{B}^{-1}(\mathbf{y})} p(\pi|\mathbf{x})\big) \quad \text{and} \quad \mathcal{L}_{CE} = -\sum_{i=1}^{L} y_i \cdot log(\hat{y}_i^{dec}) \tag{2}$$

with $\hat{y}_i^{dec}$ the probability to predict y_i in output of the decoder and $\pi \in \mathcal{B}^{-1}(\mathbf{y})$ a path from the encoder output that produces the label sequence $\mathbf{y}$ by applying the function $\mathcal{B}$ that maps the output sequence[2] of length T' to a sequence of length L with $T' > L$. Note that the CTC loss requires the probability of each character per frame in input with an additional blank label that refers to predict no character. The function $\mathcal{B}$ removes all repeated labels in the path and then removes blank labels (see [7] for more details).

Thus, the CE loss is applied using the output from the decoder, whereas the CTC is used with the output of the encoder. The decoder is therefore only trained using gradients coming from the CE loss, while the network encoder is trained using gradients coming from both the CTC and CE losses after backpropagation through the decoder.

Both losses require the probabilities for each class which are the characters and the *CTC Blank* label for the CTC loss and the characters and an *end-of-sequence* token for CE loss. Therefore, we apply a dense layer followed by a softmax at the end of the encoder and the decoder. Unlike the architecture proposed by Michael et al. [12], the character probabilities in output of the encoder are unused in input of the decoder. As illustrated in Fig. 1, the character probabilities are only used for the CTC loss and the output of the last hidden layer of the encoder is given in input of the decoder.

[2] Note that T' is equal to T if no reduction of the input sequence length is applied inside the network.

4 Experiments and results

Handwritten datasets are generally too small to get traditional Transformer-based architectures that perform well. As stated before, additional data are generally used to improve the performance of such architectures. By contrast, we aimed for a relatively light Transformer-based architecture to perform well both with and without additional data.

In this section, we demonstrate that, compared to other Transformer-based architectures, our light Transformer architecture reaches state-of-the-art results without requiring any additional data. To prove the efficiency of our network, we conduct experiments on the well-known IAM dataset (1) without using additional data; (2) using synthetic data as often done in other works.

We start by introducing experimental settings as well as the process used to generate synthetic data. We investigate the impact on performance of the main components from our network. Later, we compare the results obtained with a larger version of our light architecture and show that a big architecture is unrequired for a task of HTR. We then show the interest of using a hybrid loss that include a CTC loss at the end of the encoder part. We conclude the section by comparing the performance of our architecture with the ones of other methods from state of the art.

4.1 Handwritten Text-line Data

IAM The IAM offline handwriting database [11] is probably the most popular dataset in the domain. It is composed of text-line images of modern English handwriting, produced by several writers that have been asked to write a specific text. It has been extensively used in the literature.

To conduct our ablation study and compare our network with state of the art methods, we use the *aachen* split. While this is not the original split, the *aachen* split, is commonly used in the field to compare results. This split provides 6,482 text-line images in training, 976 for validation and 2,915 images to evaluate the performance of the model.

Synthetic Data To further investigate our model, we use generated synthetic data in addition to the training dataset. Recent Transformer architectures in the fielduse synthetic data, and report significant gains by using such data [10,14].

We use textual data extracted from a collection of English Wikipedia articles. We then generate text-line images by using handwritten fonts[3]. Generated text-line images could be more realistic thanks to the recent advances on generative models [1]. However, this approach would also require training data, while we aim at performing well with limited amount of real training data. We generate various handwritings by setting different stroke widths and slant angles. Lastly, to simulate real handwritten variations, we apply some image transformations close to a usual data augmentation pipeline. We apply elastic distortions, vary

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

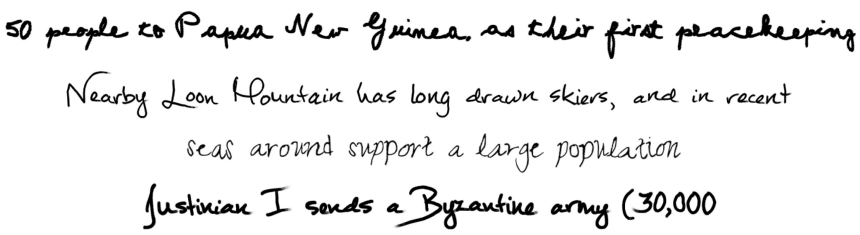

Fig. 3. Examples of generated English synthetic data, trying to match the IAM dataset.

the perspective of the image and add some noise to the generated image. Some examples of generated synthetic data are displayed in Fig. 3.

Synthetic text-line images are generated on the fly and are combined with the training dataset. In our experiments using synthetic data, we use 10,000 synthetic text-line images for each training epoch in addition to every real example from the IAM training dataset.

4.2 Experimental Settings

In this section, we describe the experimental settings we use for each of the proposed experiments, unless specified otherwise.

We train our model using teacher forcing with a probability of 1 by feeding the target sequence shifted right to the decoder. We employ masking inside the layers of self-attention in the decoder, therefore ensuring that each prediction of a character only depends on the sequence of characters before it. It follows the identical principle used by Vaswani et al. [17]. In addition to the usual gain in convergence speed, teacher forcing also allows our model to be trained in a parallel fashion, using a single decoder pass. Hence, teacher forcing helps in reducing training times. At test time, characters are predicted step by step, by applying one decoder pass for each character to be produced. The sequence of characters that have been previously predicted is provided in input of the decoder to predict the following character. We decode until an *end-of-sequence* token is predicted or up to 128 characters. Meanwhile, at training time, we use the length of the target sequence to reduce training times.

We use a custom learning rate policy as proposed by Vaswani et al. [17] with a first linear ramp for a warm-up phase of 4,000 steps. We then decay the learning rate following an invert square root function. When training with a hybrid loss, we use a value of $\lambda = 0.5$ to train our models as the two losses are of equivalent orders of magnitude.

Input images are resized to a fixed height of 128 pixels while keeping the aspect ratio. Following that, we apply data augmentation as usually done in the field to virtually increase the number of training examples. We then randomly apply elastic transform, random erosion or dilatation, random perspective and random padding both on the left and right size of the image. Images are then standardized and we add a gaussian noise. Each transformation is applied with

a probability of 0.2. Images are then padded according to the largest image in each batch, and a similar process is applied to the target sequence.

To compare results with others, we measure our models performance with both Character Error Rate (CER) and Word Error Rate (WER). It is computed using the Levenshtein distance (also called edit distance), by measuring the number of inserted, replaced and deleted characters (respectively words) between the predicted text and the ground-truth. That distance is then normalized by the length of the ground-truth.

4.3 Ablation Study of the Main Components of our Network

In this section, we investigate the impact on error rates of the main components of our architecture (the impact of the hybrid loss will be discussed later). We compare the following networks; each trained and evaluated on the IAM dataset with and without synthetic data:

- **Light Transformer** corresponds to our proposed architecture as detailed in Section 3.
- **CTNN** is our network encoder, trained and performing as a standalone. It is trained with CTC loss.
- **CRNN** is a common and popular network (it refers to a baseline architecture) composed of convolutional and recurrent layers. It uses the same convolutional layers than our Light Transformer (presented in Section 3.2) followed by a stack of four BLSTMs layers with a hidden size of 256. It is similar to CTNN, but uses recurrent layers instead of Transformer layers and is also trained using CTC loss.
- **CRNN + Decoder** combines both the CRNN and our network decoder. Compared to Light Transformer, we replaced transformer layers in the encoder by recurrent layers.

Results are available in Table 1. First, we discuss results without synthetic data. Our Light Transformer architecture is able to obtain better results than a baseline CRNN architecture on the IAM dataset. Compared to this baseline, our architecture is able to obtain a CER 22% better relatively without using any additional data, but with more parameters. When we remove the decoder

Table 1. Comparison between the results obtained by our Transformer-based architecture and a CRNN architecture on the test set of the IAM dataset (*aachen* split), with and without synthetic data added.

Architecture	# params.	IAM		IAM + Synth. Data	
		CER (%)	WER (%)	CER (%)	WER (%)
CRNN (Baseline)	1.7M	6.14	23.26	5.66	21.62
CRNN + Decoder	5.5M	6.92	21.16	5.36	18.01
CTNN (Encoder)	3.2M	5.93	22.82	6.15	24.02
Light Transformer	6.9M	**5.70**	**18.86**	**4.76**	**16.31**

from our Light Transformer, we observe that the resulting architecture (CTNN) performs worse. However, it is nevertheless able to obtain a better CER and WER than the baseline architecture. Besides, using a decoder in addition to a CRNN architecture seems to bring worse CER, even if the WER is improved. We believe that when using a Transformer-based decoder, the context from an architecture based on recurrent layers might not be good enough. Lastly, when we compare recurrent layers and Transformer layers in the encoder (CRNN + Decoder versus Light Transformer and CRNN versus CTNN), we observe a gain in the network performance.

Using synthetic data with our Light Transformer architecture, we observe a major improvement of the CER and WER. This gain seems to be far more beneficial for the decoder part, while it might even be unbeneficial for the encoder part performing as a standalone. On the one hand, synthetic data may not fit the real data well and this may affect more specifically the encoder part for which the images are given in input. On the other hand, the decoder trained to predict one character at a time might be able to act as a language model, benefiting from the amount of synthetic training data to learn the language. In addition, we also observe that the WER highly benefits from the usage of a decoder with, and without synthetic data.

As expected regarding a Transformer architecture, synthetic data bring a major improvement for our proposed architecture. Nevertheless, our architecture is able to perform well without any additional data, and we highlight the relevance of the different parts of our network.

4.4 Benefits of Using a Light Architecture

Traditional Transformer-based architecture are relatively heavier (up to 100M parameters) than our network (6,9M of parameters) and they generally perform poorly without additional data. To highlight the advantages of our light architecture, we propose a variant of it, resulting in a similar architecture with more parameters. Our scaled variant of the light architecture is mostly based on the same parameters described in Section 3. We use twice as many neurons inside Transformer layers for both the encoder and the decoder. While our light architecture uses a hidden size of 256, our scaled version uses 512 neurons instead. We also increase the number of attention heads to 8, therefore resulting in the identical number of neurons per head as in our light architecture. Lastly, intermediate feature size of position-wise feed-forward layers is also doubled, with 2,048 features instead of 1,024. It results in a scaled version of our architecture using 28M parameters instead of 6.9M parameters. We refer to this scaled version as Large Transformer. Table 2 shows the results obtained by both our light and large Transformer architectures.

We obtain slightly better results with our Light Transformer architecture, therefore indicating the heavy architecture might not prevail over our light architecture. This might be explained by the fact that the number of parameters is lower. Our light architecture might be easier to train on a scenario in which

Table 2. Comparison of the results obtained by our light Transformer-based architecture (Light Transformer) and a heavy version of it (Large Transformer) on the test set of the IAM dataset (*aachen* split).

Architecture	# params.	IAM		IAM + Synth. Data	
		CER (%)	WER (%)	CER (%)	WER (%)
Light Transformer	6.9M	**5.70**	**18.86**	**4.76**	**16.31**
Large Transformer	28M	5.79	19.67	4.87	17.67

we have a relatively low number of annotated training data for training traditional Transformer-based models. Even so, by adding synthetic data, we find that our light Transformer-based architecture performs better than Large Transformer version. Hence, we believe a big Transformer architecture is unnecessary to obtain good results on a specific type of documents.

In addition, a light architecture would require less resource to be trained efficiently. The training cost could then be reduced both in training time and in the number of training examples.

4.5 Interest of the Hybrid Loss

We evaluate how important it is to train our model with a hybrid loss composed of both the Connectionist Temporal Classification (CTC) loss and a Cross-Entropy (CE) loss. To do so, we compare the results obtained with our light architecture trained with CE only (without the CTC loss at the end of the encoder) and with a hybrid loss. Results are available in Table 3.

Using a hybrid loss seems to be crucial in our architecture, as our light Transformer model trained with a hybrid loss is able to attain lower error rates when compared to a training achieved with CE loss only. Despite the fact that our Transformer architecture relies on residual connections (that help gradients to flow), this may not be enough to efficiently train our architecture. Using a hybrid loss improves this. This is especially accurate when the model is only trained with the given original dataset, as the amount of data is too small for traditional Transformer-based models. However, even by using additional synthetic data, we find it beneficial to include a hybrid loss inside our training procedure.

Table 3. Comparison of the results obtained with our model trained with or without a hybrid loss on the test set of the *aachen* split of the IAM dataset. We present results with and without synthetic data.

Loss Function(s)	IAM		IAM + Synth. Data	
	CER (%)	WER (%)	CER (%)	WER (%)
CE only	10.29	26.36	6.76	19.62
Hybrid (CTC + CE)	**5.70**	**18.86**	**4.76**	**16.31**

No matter the amount of training data, we believe a hybrid loss is essential four our light architecture to converge quickly and efficiently. Adding an intermediate loss at the end of the encoder seems to assist the model in efficiently training the first layers in a deep architecture. In addition, the CTC loss is dedicated to recognize characters from input sequence of longer length which seems to be useful in our application. From our point of view, using a hybrid loss might help training an encoder-decoder network and combining the CTC and CE losses seems to be efficient for HTR.

4.6 Comparison with the State of the Art

The performance of our architecture is compared with the main results from the state of the art on the IAM dataset (Table 4). To allow a proper comparison, we do not include methods that use an explicit Language Model to correct outputs from the optical model. This is a general tendency, as most results from the state of the art use neither language model nor lexicon. We only report the CER as most of the works do not present WER results. However, we have shown WER results obtained with our light Transformer in the previous tables.

Without using synthetic data, we obtain a CER on the IAM test set of 5.70%. Compared to models based on FCN proposed by Yousef et al. [19] and Coquenet et al. [5], our architecture obtains worse results without synthetic data, as other transformer-based models. To the best of our knowledge, there are no published results on FCN with synthetic data included to which we can compare fairly. An FCN model might obtain better results than our approach when dealing with limited real samples while benefiting from the addition of synthetic data. However, we believe a transformer-based model will benefit more than an FCN from the addition of synthetic data due to the fact that it includes a decoder, capable of learning to model the language to some extent.

Compared to other Transformer-based networks, our architecture is able to obtain state-of-the-art results. As opposed to the Transformer model proposed

Table 4. Results on the IAM dataset (*aachen* split). We compare our architecture to methods based on CRNN [12], FCN [19,5] and Transformer [10,14,18].

Model	# params.	IAM	IAM + Synth. Data
		CER (%)	CER (%)
CRNN + LSTM [12]		5.24	
FCN [19]	3.4M	**4.9**	
VAN (line level) [5]	1.7M	4.95	
Transformer [10]	100M	7.62	**4.67**
FPHR Transformer [14]	28M		6.5
Forward Transformer [18]	13M	6.03	
Bidi. Transformer [18]	27M	5.67	
Our Light Transformer	6.9M	5.70	4.76

by Kang et al. [10], our light architecture performs well even without additional data, while Singh et al. [14] do not present result without synthetic data. Still, by including synthetic data to the training set, our model obtains even lower error rates, reaching a CER of 4.76%. Our proposed architecture reaches results close to the best performing architecture, while using 14 times less parameters.

In a general manner, our light architecture remains low in its number of parameters. Our model only uses 6.9M parameters which is far lower than the Transformer architectures from the state of the art [10,14,18]. Hence, we expect our model to be trained much faster in comparison, while requiring less resource.

Wick et al. [18] propose a medium-sized Transformer (Forward Transformer) which has 13M parameters. In their work, they propose to duplicate the architecture to perform both a forward and backward scan of the text-line image, and reduce the error rates. The resulting bidirectional Transformer attains similar CER to the one obtained by our architecture. Furthermore, our contribution is compatible with their work by adding a backward Transformer which might improve the results obtained by our network.

To conclude, compared to other Transformer-based networks, our light Transformer architecture reaches state of the art results, no matter the amount of training data with significantly less parameters.

5 Conclusion and Future Works

We present a light encoder-decoder network based on Transformer-like architecture for handwritten text recognition. We highlight the relevance to use a hybrid loss combining linearly the connectionist temporal classification loss and the cross-entropy loss, which seems crucial for encoder-decoder architecture to be trained efficiently. Compared to other Transformer-based models, our architecture remains light in the number of parameters and does not require any additional data to be trained efficiently. Our network architecture reaches results at the level of state-of-the-art Transformer-based models, with a 5.70% CER on the IAM test set. Using synthetic data, our architecture is able to attain a 4.76% CER, close to the best performing network.

As future works, we would like to apply our architecture to historical documents in which the number of annotated data is even more critical. Using a light architecture might be beneficial considering the moderate amount of training data, while a Transformer-like architecture based on multi-head attention might be useful for even more difficult writings compared to modern texts. Synthetic data prove to be an efficient solution to the limited number of training data, but the context of historical documents will require a thorough design of the generation process of synthetic data.

Acknowledgments

This work was granted access to the HPC resources of IDRIS under the allocation 2021-AD011012550 made by GENCI.

References

1. Bhunia, A.K., Khan, S., Cholakkal, H., Anwer, R.M., Khan, F.S., Shah, M.: Handwriting transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1086–1094 (2021)
2. Bluche, T., Louradour, J., Messina, R.: Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention. In: 14th IAPR ICDAR. vol. 1, pp. 1050–1055. IEEE (2017)
3. Bluche, T., Messina, R.: Gated convolutional recurrent neural networks for multilingual handwriting recognition. In: 14th IAPR ICDAR. pp. 646–651. IEEE (2017)
4. Coquenet, D., Chatelain, C., Paquet, T.: Span: a simple predict & align network for handwritten paragraph recognition. In: ICDAR. pp. 70–84. Springer (2021)
5. Coquenet, D., Chatelain, C., Paquet, T.: End-to-end handwritten paragraph text recognition using a vertical attention network. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022)
6. Doetsch, P., Zeyer, A., Ney, H.: Bidirectional decoder networks for attention-based end-to-end offline handwriting recognition. In: 15th ICFHR. pp. 361–366 (2016)
7. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: 23rd international conference on Machine learning. pp. 369–376. ACM (2006)
8. Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for unconstrained handwriting recognition. IEEE transactions on pattern analysis and machine intelligence **31**(5), 855–868 (2008)
9. Ingle, R.R., Fujii, Y., Deselaers, T., Baccash, J., Popat, A.C.: A scalable handwritten text recognition system. In: 15th IAPR ICDAR. IEEE (2019)
10. Kang, L., Riba, P., Rusiñol, M., Fornés, A., Villegas, M.: Pay attention to what you read: Non-recurrent handwritten text-line recognition. arXiv preprint arXiv:2005.13044 (2020)
11. Marti, U.V., Bunke, H.: The iam-database: an english sentence database for offline handwriting recognition. IJDAR **5**(1), 39–46 (2002)
12. Michael, J., Labahn, R., Grüning, T., Zöllner, J.: Evaluating sequence-to-sequence models for handwritten text recognition. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 1286–1293. IEEE (2019)
13. Puigcerver, J.: Are multidimensional recurrent layers really necessary for handwritten text recognition? In: 14th IAPR ICDAR. vol. 1, pp. 67–72. IEEE (2017)
14. Singh, S.S., Karayev, S.: Full page handwriting recognition via image to sequence extraction. In: ICDAR. pp. 55–69. Springer (2021)
15. Soullard, Y., Swaileh, W., Tranouez, P., Paquet, T., Chatelain, C.: Improving text recognition using optical and language model writer adaptation. In: International Conference on Document Analysis and Recognition. pp. 1175–1180 (2019)
16. Strauß, T., Leifert, G., Labahn, R., Hodel, T., Mühlberger, G.: Icfhr2018 competition on automated text recognition on a read dataset. In: 216th ICFHR. pp. 477–482. IEEE (2018)
17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NIPS. pp. 5998–6008 (2017)
18. Wick, C., Zöllner, J., Grüning, T.: Transformer for handwritten text recognition using bidirectional post-decoding. In: ICDAR. pp. 112–126. Springer (2021)
19. Yousef, M., Hussain, K.F., Mohammed, U.S.: Accurate, data-efficient, unconstrained text recognition with convolutional neural networks. Pattern Recognition **108**, 107482 (2020)