Content and Style Aware Generation of Text-line Images for Handwriting Recognition

Lei Kang, Pau Riba, Marçal Rusiñol, Alicia Fornés, Mauricio Villegas

Abstract—Handwritten Text Recognition has achieved an impressive performance in public benchmarks. However, due to the high inter- and intra-class variability between handwriting styles, such recognizers need to be trained using huge volumes of manually labeled training data. To alleviate this labor-consuming problem, synthetic data produced with TrueType fonts has been often used in the training loop to gain volume and augment the handwriting style variability. However, there is a significant style bias between synthetic and real data which hinders the improvement of recognition performance. To deal with such limitations, we propose a generative method for handwritten text-line images, which is conditioned on both visual appearance and textual content. Our method is able to produce long text-line samples with diverse handwriting styles. Once properly trained, our method can also be adapted to new target data by only accessing unlabeled text-line images to mimic handwritten styles and produce images with any textual content. Extensive experiments have been done on making use of the generated samples to boost Handwritten Text Recognition performance. Both qualitative and quantitative results demonstrate that the proposed approach outperforms the current state of the art.

Index Terms—Handwritten Text Recognition, Transformers, Generative Adversarial Networks, Synthetic Data Generation.

1 Introduction

Document analysis and recognition is challenging because of the varied document types, ranging from historical documents to modern administrative ones. In the case of documents containing handwritten text, the inter- and intra- writer variability of handwriting styles hinder the recognition performance of Handwritten Text Recognition (HTR) methods. Since manually labeling lots of real handwritten text images is labor-consuming, the utilization of data augmentation and synthetic data generation using TrueType fonts is a common practice to boost the HTR performance [1]. However, the style bias between the synthetic and real data hinders the improvement of the recognition performance.

Since Generative Adversarial Networks (GANs) [2] were firstly introduced in 2014, we have witnessed a remarkable success in generating natural scene images, which are even indiscernible from real ones by humans [3]. Conditional Generative Adversarial Networks (cGANs) [4] were proposed to condition the generation process with a class label. Thus, controllable samples can be generated from different given types [5]. However, these conditioned class labels have to be predefined and hard-coded in the model before the training process, so that it lacks the flexibility to generate images from unseen classes at inference time.

Concerning the specific case of generating samples of handwritten text, there are two different approaches to the problem. Since handwritten text is a sequential signal in nature, the same as natural language strings [6], sketch drawings [7], [8], audio signals [9] or video streams [10], it is natural that the first attempts at generating handwritten data [11] were based on Recurrent Neural Networks (RNNs) [12]. Such approaches generate a sequence of strokes in vectorial format that are used to render images. On the contrary, some more recent approaches propose to directly generate images instead of sequences of strokes. By producing images directly, long-range dependency and gradient vanishing problems of recurrencies are avoided, while achieving a better efficiency. Furthermore, such approaches are able to produce richer results in the sense that they go beyond producing just nib locations, but also provide visual appearance such as the calligraphic styles, such as slant, glyph shapes, stroke width, darkness, character roundness, ligatures, etc., and background paper features like texture, opacity, show-through effects, etc.

Current state-of-the-art methods that directly generate handwriting images work at different levels. First, some approaches are focused on producing isolated characters or ideograms [13], [14]. Such approaches often work over a set of predefined classes, so that they can only generate a reduced set of contents. Second, some approaches are able to generate handwritten words [15], [16], allowing not to be restricted to a closed vocabulary. Finally, some works like [17], [18] go beyond isolated words and produce full text-lines. The generation on text-line level is difficult because not only the handwritten text should be readable and realistic, but also the writing flow should be natural and smooth.

Since we aim to boost the HTR performance at text-line level, in this work we propose a method for generating handwritten text-line images. By conditioning on

- *L. Kang is with Computer Science Dept., Shantou University, China. E-mail: lkang@stu.edu.cn*
- *P. Riba is with Helsing AI, Munich, Germany. E-mail: pau.riba@helsing.ai*
- *M. Rusiñol is with AllRead MLT, Barcelona, Spain. E-mail: marcal@allread.ai*
- *A. Fornés is with Computer Vision Center, Computer Science Dept., Universitat Autonoma de Barcelona, Spain. E-mails: afornes@cvc.uab.es*
- *M. Villegas is with omni:us, Berlin, Germany. E-mail: mauricio@omnius.com*

Manuscript received November 2, 2020; revised Jun 7, 2021.

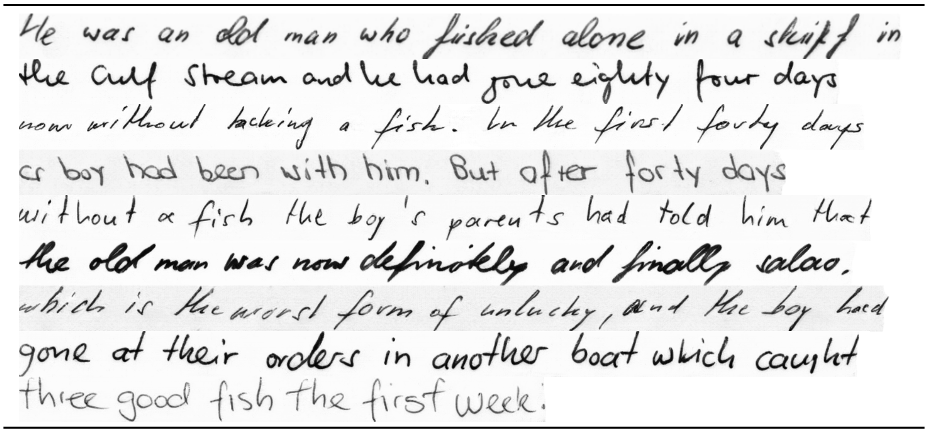

He was an old man who fished alone in a skiff in
the Culf Stream and he had gone eighty four days
now without tacking a fish. In the first forty days
cr boy had been with him. But after forty days
without a fish the boy's parents had told him that
the old man was now definiokly and finally salao,
which is the worst form of unlucky, and the boy had
gone at their orders in another boat which caught
three good fish the first week.

Fig. 1: Examples of generated text-lines, each one using a different handwriting style. The text corresponds to the first paragraph of the book “The Old Man and the Sea”.

both calligraphic style from handwritten images and textual content from an external text corpus, our proposed method is able to produce realistic, writer agnostic and readable samples for handwritten text-lines (see Figure 1), which can be effectively used in order to train and improve the final HTR performance.

This work supposes a significantly extended version of our previous conference paper [16]. In this work, we have enhanced our previous generative architecture in order to generate whole sentences rather than single words, where Periodic Padding and Transformer-based Recognizer are newly proposed. In addition, we propose a novel version of the Fréchet Inception Distance (FID) metric to guide the method to choose the best hyper-parameters specifically for variable-length samples like handwritten text images. In the training step, we make use of curriculum learning strategy to help the proposed method to generalize from short text-lines to longer ones. More importantly, and contrary to our previous work (in which the only goal was to generate realistic text images), the proposed method is particularly focused on improving the HTR performance, demonstrating that the use of realistic synthetic generated text at training time is indeed useful for improving HTR.

To summarize, the main contributions of this paper are the following:

- We propose a novel method for handwritten text-line image generation conditioning on textual content and visual appearance information, which is capable of generating open vocabulary text and visual appearance.
- We introduce an improved version of the FID measure, namely vFID, as a novel metric to evaluate the quality of the generated handwritten image. It is more robust to variable-length images and particularly suited for the handwriting case.
- We conduct extensive experiments to demonstrate, on the one hand, the realism of the generated handwritten text images and, on the other hand, the boost in HTR performance avoiding the manual labeling effort.

The rest of the paper is organized as follows. In Section 2 we introduce the state-of-the-art approaches related to handwriting generation. In Section 3 we explain our proposed method in details with different modules. In Section 4 the proposed novel vFID metric is introduced. In Section 5 extensive qualitative and quantitative experiments are presented and discussed. Finally, Section 6 draws the conclusions of this work.

2 Related Work

Traditional methods [19], [20], [21], [22] approached the generation of word samples by manually segmenting individual characters or glyphs and then tune a deformation to match the target writing style. Recently, based on these rendering methods, Haines *et al.* [23] succeeded in generating indistinguishable historical manuscripts of Sir Arthur Conan Doyle, Abraham Lincoln and Frida Kahlo with new textual contents, but these impressive results are obtained at the cost of a high manual intervention.

Editing text in the natural scene images aims to replace a word in the source image with a new one while maintaining the original style. Wu *et al.* [24] proposed an end-to-end trainable style retention network (SRNet) for the text editing task, which is the first work to edit text image in the word-level. Roy *et al.* [25] developed both Font Adaptive Neural Network (FANnet) and Colornet architectures to modify text in an natural scene image at character-level. Yang *et al.* [26] also succeeded in manipulating the texts from the natural scene images even with severe geometric distortion. However, the texts in the natural scene images are often to be typed fonts, especially in the datasets evaluated with these methods. The lack of cursive styles of the scene texts makes these approaches hard to work in handwriting scenarios.

The generation of sequential handwritten data consists of producing stroke sequences in vector form with nib locations and sometimes velocity records. With the coming of deep learning era, Graves [11] utilized Long Short-Term Memory (LSTM) to predict point by point at each time step to generate stroke sequences conditioned on a given writing style and a certain text string. Zhang *et al.* [27] investigated RNN as both discriminative and generative models for recognizing and drawing cursive handwritten Chinese characters. Following this sequential-based idea, some recent works [7], [8], [28], [29] have reached an impressive performance on text or sketch generation. Online handwritten data preserves rich dynamic information such as trajectory, velocity and pressure, which is a big advantage over the offline data. However, the offline data maintains richer visual appearance information such as stroke thickness, ink shading and paper textures. In this paper, we focus on the problems on generating realistic handwritten images at pixel-level, so we only make use of offline handwritten data.

Different levels of offline handwritten data can be processed: characters/glyphs, words and text-lines. Based on the ideas of variational auto-encoders [30] or GANs [2], some works achieve impressive performance on synthesizing Chinese ideograms [14], [31], [32], [33], [34] and glyphs [35]. However, these methods are restricted to a predefined set of content classes and the input images have fixed size. To overcome the limitation of incapability of generating out of vocabulary (OOV) texts, Alonso *et al.* [15] proposed a cGAN-based method to generate handwritten word samples, which is conditioned on RNN-embedded text information. However, this proposed approach suffers from the mode collapse problem so that it learns the general writing style of the training set and does not offer variability of the generated samples. Our previous work, GANwriting [16], works only at word level, so that it cannot render long text strings into a handwritten image. To keep the consistency in a text-line image, the extension from word to text-line level is needed. Fogel *et al.* [17] equip a style-promoting discriminator to be able to generate diverse styles for handwritten image samples. However, the generated characters have the same receptive field width, which can make the generated samples look unrealistic. Davis *et al.* [18] takes advantage of CTC activations [36] to produce spaced text, which helps the generator to achieve horizontal alignment with the input style image. The style information is the concatenation of both global style feature and character-wise style feature. However, the character-wise style feature highly depends on the performance of CTC, so mode collapse problem may happen when tackling the unseen style images from the target dataset.

In summary, all the above described methods are not robust enough to produce high quality handwritten samples with a huge diversity in handwriting styles, especially when producing longer text-lines. In Section 3, we describe our generative method for handwritten text-line images with carefully designed modules.

3 Handwritten Text-line Synthesizer

3.1 Problem Formulation

Let $\{\mathcal{X}, \mathcal{Y}, \mathcal{W}\} = \{(x_i, y_i, w_i)\}_{i=1}^{N}$ be a multi-writer handwritten text-line dataset, containing gray-scale text-line images $\mathcal{X}$, their corresponding transcription strings $\mathcal{Y}$ and their writer identifiers $\mathcal{W}$. In this work, the handwriting calligraphic style is considered as an inherent feature for each of the different writers, and we also hypothesize that the background paper features are consistent within each writer. Thus, the visual appearance is identified with $w_i \in \mathcal{W}$. Therefore, let $X_i = \{x_{w_i,j}\}_{j=1}^{K} \subset \mathcal{X}$ be a subset of K real text-line images with the same style defined by writer $w_i \in \mathcal{W}$. Besides, $\mathcal{A}$ denotes the alphabet containing all the supported characters such as lower and upper case letters, digits and punctuation signs that the generator will be able to produce.

In this setting, the realistic handwritten text-line generation problem is formulated in terms of few-shot learning. Two inputs are given to the model: 1) a set of images X_i as a support example of the visual appearance attributes of a particular writer w_i; and 2) a textual content provided by any text string t where $t_n \in \mathcal{A}$. The proposed conditioned handwritten text generation model is able to combine both sources of information in order to yield realistic handwritten text-line images, which share the visual appearance attributes of writer w_i and the textual content provided by the string t. Finally, our objective model H, able to generate handwritten text, is formally defined as

$$\bar{x} = H\left(t, X_i\right) = H\left(t, \{x_1, \ldots, x_K\}\right), \tag{1}$$

where $\bar{x}$ is the artificially generated handwritten text-line image with the desired properties. From now on, we denote $\bar{\mathcal{X}}$ as the output distribution of the generative network H.

Figure 2 shows a detailed overview of the proposed architecture. The proposed model consists of four main components: the Visual Appearance Encoder, the Textual Content Encoder, the Generator and the learning objectives. On the one hand, the generator, which is conditioned by a combination of visual appearance attributes and textual content information, is able to produce human-readable handwritten text-line images. On the other hand, three learning objectives are proposed to guide the learning process towards generating realistic images, which are classified within a particular visual appearance while sharing the specified textual content.

3.2 Visual Appearance Encoder

The visual appearance encoder receives as input a given set X_i of handwritten text line images from a particular writer w_i. We assume that these given images share some visual appearance features that are inherent to each writer. These visual appearance attributes consist of properties such as slant, glyph shapes, stroke width, character roundness, ligatures, etc. In our proposed approach, the visual appearance encoder aims at extracting the handwriting style attributes from the set of images X_i. With this aim, the textual content information from those images is ignored and totally disentangled from the stylistic visual attributes.

The proposed visual appearance encoder consists of two modules: first, a periodic padding module which ensures that all the images share the same size and, second, a style blending module in charge of extracting the visual appearance features. In general, the visual appearance encoding process is denoted as $F_s = S\left(X_i\right)$.

3.2.1 *Periodic Padding Module*

As the style image samples X_i have varied shapes, they are firstly resized to the same 64 pixels height while keeping the aspect ratio. Let L be the maximum length of both input and output images. To mimic the background color with our padding, the input image $x_j \in X_i$ is first normalized within the range $[0, 1]$ and then their intensities are inverted $1 - I/255$. Thus, the writing strokes have values close to 1 whereas the background has values close to 0. In the HTR

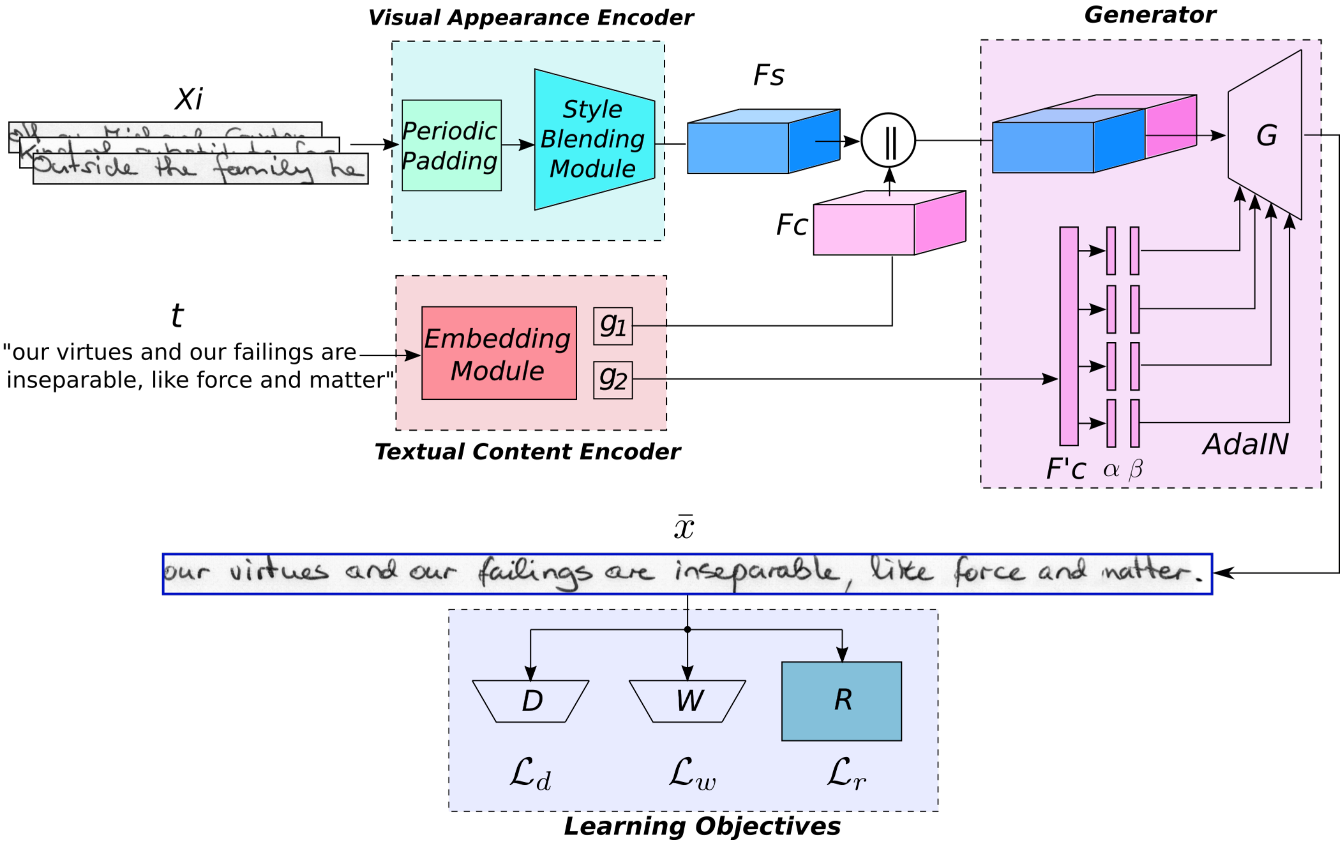


Fig. 2: Architecture of the proposed handwriting synthesis model. It consists of a Visual Appearance Encoder (cyan box), a Textual Content Encoder (red box), a Generator (magenta box) and learning objectives (blue box). X_i and t are the images and text string input, respectively. The $\bar{x}$ is the generated sample that shares the visual appearance with X_i and contains the textual information with t.

literature, the usual technique to align all the images to have the same length in a mini-batch is to add 0-padding to the right of each image until reaching the maximum length L. We have experimentally observed that 0-padding has a severe impact on the handwritten text-line image generation process, which can easily collapse in terms of style in the padded regions. This is especially important when there is a huge difference in the length of input images. The style representations F_s contain not only the visual appearance attributes, but also the spatial information. Thus, the padding would make longer texts to loose the handwritten style consistency in the generated output. To overcome this problem, we introduce a simple periodic padding module, which consists in repeating the input image several times to the right until the length fits the maximum width L. An example is shown in Figure 3. Thus, the periodic padding can deal with the visual appearance vanishing problem, which is especially useful to generate long text handwritten samples with short input images. Bear in mind that the style images X_i are only used to extract style features, which are completely independent from the textual content in the image.

3.2.2 *Style Blending Module*

The K images from X_i, now all having the same size are channel-wise concatenated and given as input for the subsequent stylistic feature extractor. The style blending module is a sequence of convolutional layers. It is in charge of producing the visual appearance features F_s from the set of images X_i, which is represented as a $64 \times L \times K$ input tensor. The choice of the convolutional architecture is detailed in Section 5.

3.3 Textual Content Encoder

The textual content encoding process transits an input text string t into textual content features F_c and F'_c, as shown in Figure 4, and denoted as $F_c, F'_c = C(t)$. Observe that the input text string t is firstly padded with the empty symbol ε to a fixed maximum string length T. Then, it goes through two pipelines: the character-wise embedding produces character-wise features F_c, and the global string encoding produces a global string description F'_c. These two pipelines have the textual content information that will be later combined with the visual appearance features during the generating process.

3.3.1 *Embedding Module*

As our method aims to generate any input sentence, including OOV words created from the predefined characters of alphabet $\mathcal{A}$, an embedding layer is applied on the input text string to extract character-level embedding features. Thus, each character $t_i \in t$ is mapped into a vector of size n by means of the embedding function:

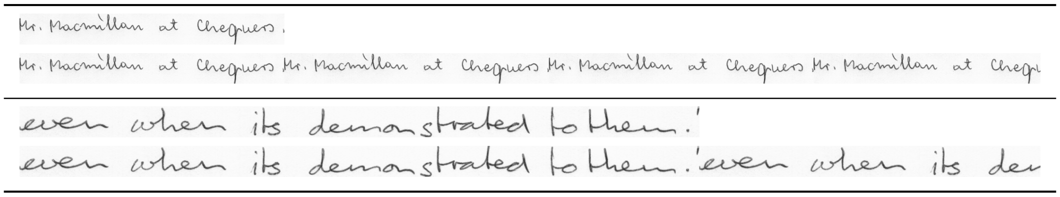

Mr. Macmillan at Chequers.
Mr. Macmillan at Chequers Mr. Macmillan at Chequers Mr. Macmillan at Chequers Mr. Macmillan at Chequ
even when its demonstrated to them.'
even when its demonstrated to them.'even when its dem

Fig. 3: Periodic padding example. Given a real image, periodic padding to the right is applied several times until the maximum image width L is reached.

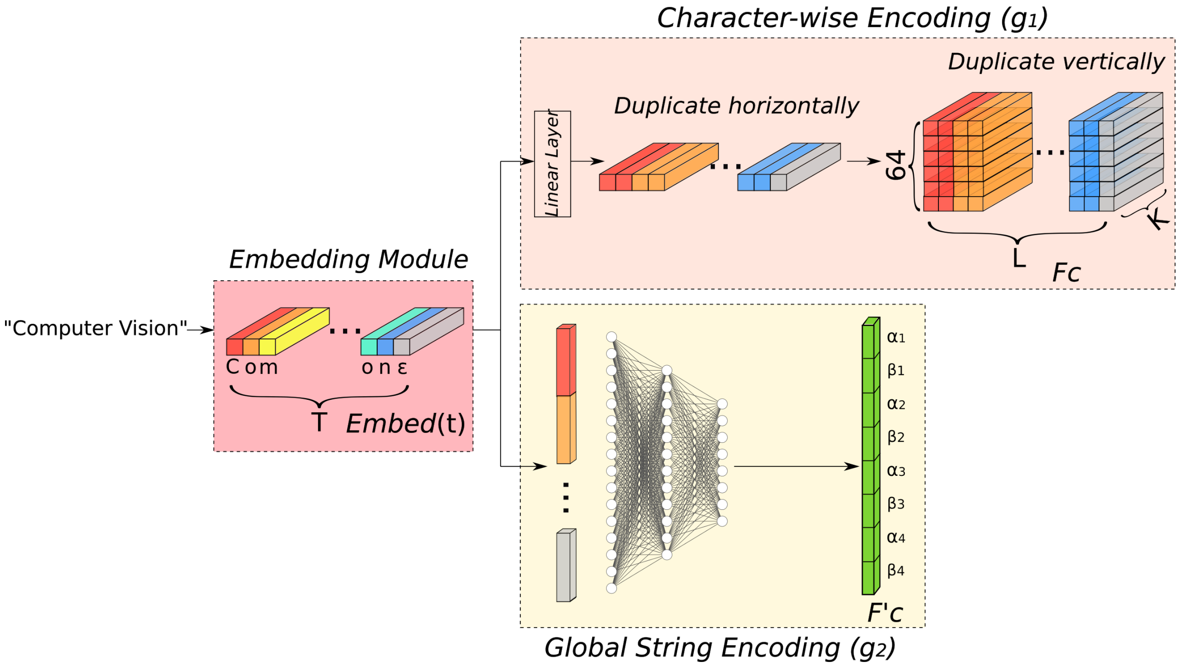


Fig. 4: Architecture of the textual content encoder. It consists of an Embedding Module (red box), a Character-wise Encoding (orange box) and a Global String Encoding (yellow box). In the sequence of character embeddings, each vector is represented by a specific color.

$$\begin{aligned}\mathrm{Embed}\colon \mathcal{A} &\to \mathbb{R}^n \\ t_i &\mapsto \mathrm{Embed}(t_i),\end{aligned}$$

For the sake of simplicity and with abuse of notation, we will denote $\mathrm{Embed}(t)$ as the embedding of the whole string applied character by character.

3.3.2 Character-wise Encoding

In order to properly combine the textual information with the style feature F_s in the next step, the length T should be aligned with the width of F_s. To achieve the alignment, each character embedding is repeated several times separately, and then all the chunks of repeated character embeddings are concatenated back together horizontally. To directly concatenate textual content feature F_c and visual appearance feature F_s in the next step, we also duplicate the horizontal character embeddings vertically. So the textual content feature F_c ends up with the shape of (64, L, K) as shown in the upper part of Figure 4 and denoted as g_1. Thus, we obtain the textual content feature F_c, which represents the local textual information and is obtained as $F_c = g_1(\mathrm{Embed}(t))$. Then both the content feature F_c and the style feature F_s are concatenated channel-wise.

3.3.3 Global String Encoding

Apart from injecting the character-wise information, an overall string information is helpful to guide the generating process as it gives a global coherence to the generation process. The character embedding $\mathrm{Embed}(t)$ is reshaped into a large one-dimensional vector of size $T \cdot n$. Then, a Multi-Layer Perceptron (MLP) g_2 is used to produce the global textual feature F'_c, as shown in the lower part of Figure 4. Thus, the global features are computed as $F'_c = g_2(\mathrm{Embed}(t))$, which is a one-dimensional vector. We first split it into 8 equally sized pieces, and then we use them as 4 pairs of α and β parameters orderly, which will be used in AdaIN (refer to Equation 2).

3.4 Generator

The generator G is in charge of combining the two sources of information: the visual appearance encoder and the textual content encoder. It consists of two residual blocks with AdaIN [37] as the normalization layer, 4 convolutional modules with nearest neighbor up-sampling and ReLU activations, and a final $\tanh(\cdot)$ activation layer. The global string information is equipped with the generator via AdaIN, which is formally defined as

$$\mathrm{AdaIN}\left(z,\alpha,\beta\right)=\alpha\left(\frac{z-\mu\left(z\right)}{\sigma\left(z\right)}\right)+\beta, \tag{2}$$

where $z \in F$, μ and σ are the channel-wise mean and standard deviations. The parameters α and β are assigned with the splitting of $F_c^{'}$. Hence, the generative process is defined as

$$\begin{aligned}\bar{x}&=H\left(t,X_i\right)=G\left(C\left(t\right),S\left(X_i\right)\right)=G\left(F_c,F_c',F_s\right)\\&=G\left(g_1\left(Embed(t)\right),g_2\left(Embed(t)\right),S\left(X_i\right)\right),\end{aligned} \tag{3}$$

where C is the textual content encoder and S is the visual appearance encoder.

3.5 Learning Objectives

Three learning objectives are proposed to enforce different properties on the generated images $\bar{x} \in \bar{\mathcal{X}}$, where $\bar{\mathcal{X}}$ is the generated image space mimicking the visual appearance of real images $\mathcal{X}$. First, a discriminative loss $\mathcal{L}_d$ is in charge to ensure a realistic global appearance. Second, a writer classification loss $\mathcal{L}_w$ forces the generated samples to follow a specific appearance style. Finally a recognition loss $\mathcal{L}_r$ ensures the preservation of the textual content.

3.5.1 Discriminative Loss

Following the traditional GAN paradigm [2], we propose a discriminative model D, which consists of one convolutional layer, six residual blocks, each of them with LeakyReLU activations and average poolings, and a final binary classification layer. Thus, given an input image, D produces a binary output that is classified either as real (1) or fake (0). It does not take into consideration neither the visual appearance provided by X_i nor the textual contents provided by t, and only focuses on the general visual appearance of the generated image $\bar{x}$ to look realistic. The discriminative loss $\mathcal{L}_d$ is formally defined as

$$\mathcal{L}_d\left(H,D\right)=\mathbb{E}_{x\sim\mathcal{X}}\left[\log\left(D\left(x\right)\right)\right]+\mathbb{E}_{\bar{x}\sim\bar{\mathcal{X}}}\left[\log\left(1-D\left(\bar{x}\right)\right)\right]. \tag{4}$$

3.5.2 Visual Appearance Loss

Assuming that each writer has an inherent writing style, we propose a writer classifier W that follows the same architecture of D by replacing the final binary classification layer with an N-classification layer, where N is the number of writers in the training dataset. The writer classifier W is optimized with real samples drawn from $\mathcal{X}$. This loss guides the generation of synthetic samples to align their styles with the given writer. Thus, the writer classifier acts as a style loss to provide diversity on the generated samples. The style loss $\mathcal{L}_w$ is formally defined as

$$\mathcal{L}_w\left(H,W\right)=-\mathbb{E}_{x\sim\{\mathcal{X},\bar{\mathcal{X}}\}}\left[\sum_{i=1}^{|\mathcal{W}|}w_i\log\left(\hat{w}_i\right)\right], \tag{5}$$

where $\hat{w}=W(x)$ is the predicted probability distribution over writers in $\mathcal{W}$ and w_i is the real writer distribution. Therefore, the generated samples should be classified as the same writer w_i used to construct the input style conditioning image set X_i.

3.5.3 Content Loss

A handwritten text recognizer R is used to ensure that the generated sample has the specific textual content, indicated by the input string t. Given that we generate text-lines, a robust recognizer for long sequences is needed. We adopt a Transformer-based recognizer [38] that has recently shown good performance on full handwritten text-lines.

The architecture of our Transformer-based HTR approach is shown in Figure 5. It also follows the encoder (upper part) and decoder (lower part) structure as proposed in [39]. The encoder extracts high-level features from the input handwritten images, which consists of a ResNet and 4 blocks of self-attention module and linear module with layer normalization and dropout. The decoder takes masked text strings as input [39] so that the decoding only depends on predictions produced prior to the current character. In addition, the processing with characters is done in parallel, avoiding the recurrency of sequence-to-sequence models. Such a parallel processing of what used to be different time steps in sequence-to-sequence approaches drastically reduces the training time. The decoder consists of 4 blocks of self-attention modules and 4 blocks of mutual-attention modules, which provide an even more powerful ability to handle long sequence inputs than sequence-to-sequence approaches. The comparison between both of the sequence-to-sequence-based and Transformer-based methods is detailed in Section 5.

The Kullback-Leibler divergence loss is used as the recognition loss at each time step. It is formally defined as:

$$\mathcal{L}_r\left(H,R\right)=-\mathbb{E}_{x\sim\{\mathcal{X},\bar{\mathcal{X}}\}}\left[\sum_{i=0}^{L}\sum_{j=0}^{|\mathcal{A}|}t_{i,j}\log\left(\frac{t_{i,j}}{\hat{t}_{i,j}}\right)\right], \tag{6}$$

where $\hat{t}=R(x)$; $\hat{t}_i$ being the i-th decoded character probability distribution by the recognizer, $\hat{t}_{i,j}$ being the probability of j-th symbol in $\mathcal{A}$ for $\hat{t}_i$, and $t_{i,j}$ being the real probability corresponding to $\hat{t}_{i,j}$. The empty symbol ε is ignored in the loss computation.

3.5.4 Joint Training Process

The whole architecture is trained with the three proposed loss functions jointly in an end-to-end fashion as follows.

$$\mathcal{L}(H,D,W,R)=\mathcal{L}_d(H,D)+\mathcal{L}_w(H,W)+\mathcal{L}_r(H,R), \tag{7}$$

$$\min_{H,W,R}\max_{D}\mathcal{L}(H,D,W,R). \tag{8}$$

The training strategy is further explained in Algorithm 1, where $\Theta=\{\Theta_H,\Theta_D,\Theta_W,\Theta_R\}$ represents the related network parameters and $\Gamma(\cdot)$ denotes the optimizer function. Even though the training process is end-to-end, the optimization process is performed in two steps. Firstly, we feed both real and generated samples together to the discriminator D, so that the discriminative loss can be obtained (line 3). Secondly, we only make use of real data to train the writer classifier W and the text recognizer R, so that the visual appearance and content losses are obtained (line 4). As W and R are optimized with only real data, they could be pre-trained independently as an initialization apart from the generative network H. However, the good performance

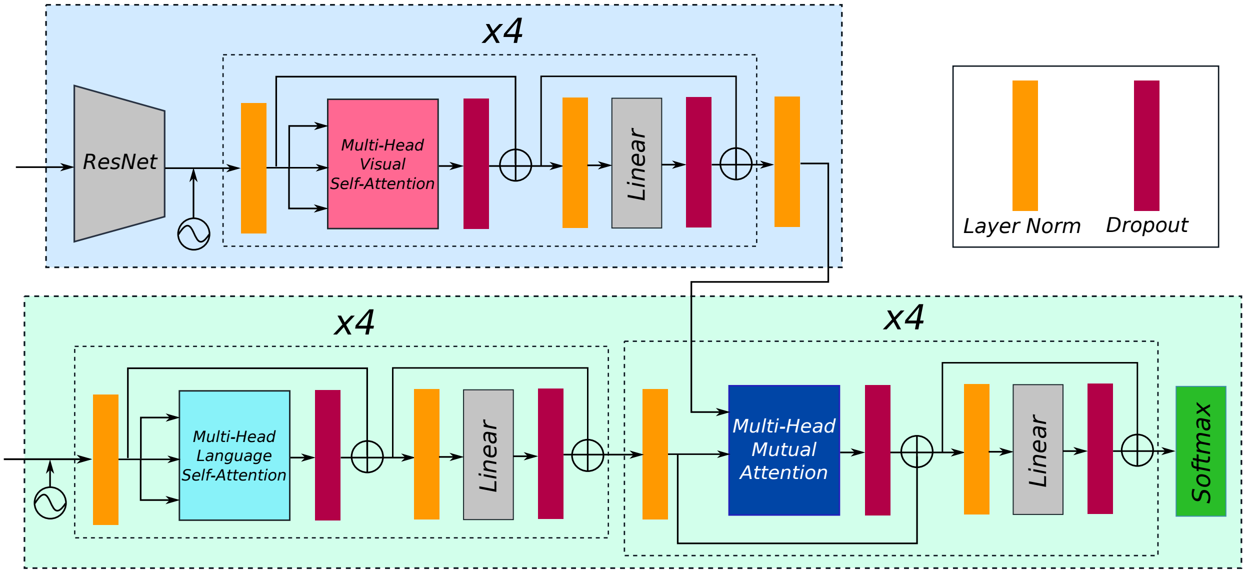

x4
ResNet
Multi-Head Visual Self-Attention
Linear
Layer Norm
Dropout
x4
Multi-Head Language Self-Attention
Linear
x4
Multi-Head Mutual Attention
Linear
Softmax

Fig. 5: Architecture of the Transformer-based handwritten text recognizer. The upper part is the Encoder (blue color) and the lower part is the Decoder (green color).

Algorithm 1 Training algorithm for the proposed model.

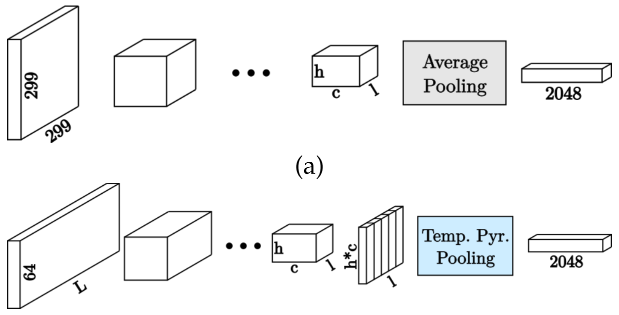
```
Input: Input data $\{\mathcal{X}, \mathcal{Y}, \mathcal{W}\}$; alphabet $\mathcal{A}$; max training iterations $Itr$
Output: Networks parameters $\Theta = \{\Theta_H, \Theta_D, \Theta_W, \Theta_R\}$.
1: repeat
2:     Get style and content mini-batches $\{X_i, w_i\}_{i=1}^{N_{Batch}}$ and $\{t^i\}_{i=1}^{N_{Batch}}$
3:     $\mathcal{L}_d \leftarrow$ Eq. 4        ▷ Real and generated samples $x \sim \{\mathcal{X}, \bar{\mathcal{X}}\}$
4:     $\mathcal{L}_{w,r} \leftarrow$ Eq. 5 + Eq. 6        ▷ Real samples $x \sim \mathcal{X}$
5:     $\Theta_D \leftarrow \Theta_D + \Gamma(\nabla_{\Theta_D} \mathcal{L}_d)$
6:     $\Theta_{W,R} \leftarrow \Theta_{W,R} - \Gamma(\nabla_{\Theta_{W,R}} \mathcal{L}_{w,d})$
7:     $\mathcal{L} \leftarrow$ Eq. 7        ▷ Generated samples $x \sim \bar{\mathcal{X}}$
8:     $\Theta_H \leftarrow \Theta_H - \Gamma(\nabla_{\Theta_H} \mathcal{L})$
9: until Max training iterations $Itr$
```

of W and R may unbalance the three losses in the early training steps, which could make the whole network hard to train. Thus, we initialize all the network parameters from scratch and jointly train them altogether. The network parameters Θ_D are optimized by gradient ascent following the GAN paradigm whereas the parameters Θ_W and Θ_R are optimized by gradient descent. Finally, the overall generator loss is computed following Equation 7 where only the generator parameters Θ_H are optimized (line 8).

4 Variable-length Fréchet Inception Distance (vFID)

In order to evaluate the quality of the generated images by GANs, there is a commonly used metric, namely Fréchet Inception Distance (FID) [40]. FID is used to evaluate the similarity between the generated images and the real ones. This is achieved by calculating the distance between two feature vectors, which are obtained from two image sets, the generated images and the real ones, respectively. FID has been widely used in evaluating the performance of GANs at image generation. This metric follows two steps: first, it extracts features from an InceptionV3 network [41] while keeping activations of the last pooling layer, which is pretrained on the ImageNet dataset [42]; then, it calculates the distance between the feature vectors. Even though FID has been widely used for the evaluation of the generated natural scene images, it is not well suited for handwritten image data. The main drawbacks in such case are (i) the ImageNet dataset consists of natural scene image samples that have very few common features with handwritten text images; (ii) the InceptionV3 model used by the FID requires a fixed size input, which could not handle the variable-length scenario of handwritten text images. Thus, we introduce a novel version of FID, namely vFID (Variable-length Fréchet Inception Distance), specially suited for such variable-length images such as handwritten text images. Similarly to the original FID, the proposed metric vFID share the same InceptionV3 network as the convolutional backbone. However, instead of the average pooling used by the FID, we first reshape the convolutional feature into a 2-dimensional feature map which is then fed into a Temporal Pyramid Pooling (TPP) layer [43] as shown in Figure 6. TPP is especially useful when the input is a variable-length sequence of features, which is the case for handwritten text-line images. Based on the pretrained InceptionV3, we fine-tune the vFID model with the IAM dataset by fitting a writer classifier. When applying the vFID metric, the input images should be resized to have 64 pixels height while preserving the aspect ratio. Thus, the variable resulting width is denoted by L. We calculate the vFID values for each input image without adding paddings. Thus, vFID is not affected by batching and different image widths.

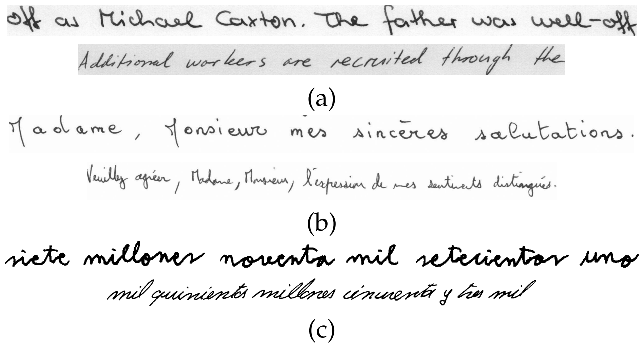
To achieve a valid metric performance and a fair comparison for both vFID and FID, we first reuse the Inception V3 network that is pretrained on ImageNet dataset, and then, we fine-tune both vFID and FID models on the IAM datasets towards a writer classification problem. Once they are properly trained, we can then evaluate the generated image quality through the metrics. The performance comparison of FID and vFID for the IAM dataset is shown in Figure 7.



(a)

(b)

Fig. 6: (a) Inception module of FID with Average Pooling, (b) Updated Inception module of vFID with Temporal Pyramid Pooling.


off as Michael Carton. The father was well-off
Additional workers are recruited through the
Madame, Monsieur mes sincères salutations.
Veuilley agréer, Madame, Monsieur, l'expression de mes sentiments distingués.
siete millones noventa mil setecientos uno
mil quinientos millones cincuenta y tres mil

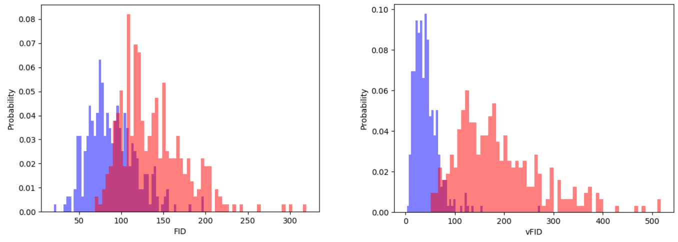
(a)

(b)

(c)

Fig. 8: Examples of the IAM, Rimes and Spanish Numbers datasets are shown in (a), (b) and (c), respectively.

The blue distribution indicates the performance for the same writer, while the red one indicates the performance for a different writer pair. The lower value of the FID/vFID, the better similarity is obtained. Therefore, we aim to achieve a robust metric that produces a lower value for the same writer (blue) and a higher value for the different writers (red). In Figure 7(a), we observe that the performance of FID do not have a good behaviour since it has a big overlapping area, so that it cannot provide a reasonable judgement on the performance of the generated text. Contrary, our proposed vFID in Figure 7(b) could provide a more trustful measure.



(a)

(b)

Fig. 7: Histogram of FID (a) and vFID (b). The x-axis indicates the FID/vFID values, and the y-axis indicates the counts. The FID/vFID between subsets of samples in the same writer is shown in blue, and between different writers in red. The distribution of blue and red should be apart as far as possible. Both histograms are normalized to sum up to one.

5 Experiments

In this section, we present the extensive evaluation of our proposed approach. First, we perform several ablation studies on the key modules to find the best balance between performance and efficiency. Then, we demonstrate qualitative and quantitative results on synthetically generated images. Finally, we make use of the generated samples to boost the HTR performance in different experimental settings.

5.1 Datasets and Metrics

The IAM offline dataset [44], the Rimes dataset [45] and the Spanish Numbers dataset [46] are utilized in our experiments as shown in Table 1. However, our proposed generative method is only trained with IAM dataset. All the three datasets are utilized in the HTR experiments. Examples of the three datasets are shown in Figure 8.

TABLE 1: Overview of the datasets used in our HTR experiments: Number of text-lines used for training, validation and test sets, and number of writers.

Dataset	Train	Val.	Test	Writers	Language
IAM [44]	6482	976	2914	657	English
Rimes [45]	11333	–	778	1300	French
Spanish Num. [46]	298	–	187	30	Spanish

WikiText-103 [47] is chosen to be our external text corpus when selecting random text strings as textual input. As in the case of images, we select texts in WikiText-103 from one word to N_t words to create sentences. We end up with 3.6 million text-lines with number of characters varying from 1 to 88.

The *Character Error Rate* (CER) and the *Word Error Rate* (WER) [48] are the performance measures. The CER is computed as the Levenshtein distance, which is the sum of the character substitutions (S_c), insertions (I_c) and deletions (D_c) that are needed to transform one string into the other, divided by the total number of characters in the groundtruth (N_c). Formally,

$$CER = \frac{S_c + I_c + D_c}{N_c} \tag{9}$$

Similarly, the WER is computed as the sum of the word substitutions (S_w), insertions (I_w) and deletions (D_w) that are required to transform one string into the other, divided by the total number of words in the groundtruth (N_w). Formally,

$$WER = \frac{S_w + I_w + D_w}{N_w} \tag{10}$$

5.2 Curriculum Learning Strategy

The IAM dataset is used to train our generative method. It is a multi-writer dataset in English, which consists of $1,539$ scanned pages written by 657 writers, as detailed in Table 1. Since we can access to the groundtruth of the training data, including the bounding-boxes at word level, we could enlarge the training set using the N-gram cropping strategy. For example, given a sequence of words, we can

TABLE 2: Three categories of the IAM offline dataset, from short to long text-lines.

Category	Num. of chars.	Image length
1	$1-24$	$64-600$
2	$24-48$	$600-1200$
3	$48-88$	$1200-2160$

iteratively crop out one-word, two-words, and so on until N-word sub-lines, where N is the maximum number of words in the given text-line. Thus, given the normalized height of 64 pixels, we end up with $598,489$ images with variable lengths from 64 to 2160 pixels and the number of characters from 1 to 88, where 2160 and 88 are the maximum image length and text length for IAM dataset, respectively.

To achieve a better handwriting generation with fine-grained details, we make use of curriculum learning strategy by splitting training data into 3 categories as shown in Table 2, from shorter to longer sentences. We start the training with data of Category 1 from scratch, then we fine-tune with data of Category 2, and finally we fine-tune with data of Category 3. The training is done step by step with increasing difficulty in the sense of image and text length. Note that the data used in the previous training step does not appear in the next fine-tuning step considering the training speed. In practice, the second and third steps just need to be fine-tuned for few epochs.

5.3 Implementation Details

The experiments were run using PyTorch [49] on a single NVIDIA RTX6000 GPU. As there are three objective modules, we set the learning rate of the discriminator and the generator to be $1 \cdot 10^{-4}$, whereas the ones of writer classifier and recognizer are $1 \cdot 10^{-5}$. The training process is optimized with Adam optimizer and a batch size of 4.

5.4 Ablation Study

As explained in Section 4, the vFID can measure the similarity of both real and generated images in a better way than the original FID, so we make use of vFID to choose the convolutional architecture and the recognizer.

First, we compare the generated samples with and without Periodic Padding Module as shown in Figure 9. The style input is randomly selected from a specific writer, and it may be a shorter image than what we expect to generate. In the upper part of Figure 9, the style input is padded with 0 to the maximum width, so that the generated image suffers from the style collapse problem in the corresponding padded area. Contrary, in the lower part of Figure 9, with the periodic padding process, the style image has been extended to the maximum width that is sure of covering all the possibly generated area. Thus, the generated sample keeps the consistency in the visual appearance from the first character until the end.

Second, we modify the convolutional layers from VGG19 to ResNet34, and study the performance and training speed. The performance is evaluated on the generated samples based on the style information from IAM test set and content information from a subset of WikiText-103. The models are trained until 500 epochs. Speed is the total time for a forward and backward pass. From Table 3, we observe that ResNet34 achieves a higher training speed while obtaining a slightly better performance.

TABLE 3: Ablation study for Convolutional layers on the IAM test set.

Conv.	vFID	Speed (ms)
VGG19	115.46	144.13
ResNet34	**114.39**	**136.80**

TABLE 4: vFID performance on generating different length of images for the sequence-to-sequence and Transformer-based HTR methods. The lower the value, the better the performance.

Method	**Num. of chars to be generated**	
	1-10 (words)	1-90 (lines)
Seq2Seq	**136.51**	249.94
Transformer	146.74	**114.39**

Third, we analyze the effect of replacing the sequence-to-sequence recognizer with the Transformer-based recognizer. Based on the number of characters to be rendered in the generated samples, we have two categories: words with 1 up to 10 characters and text-lines with 1 up to 90 characters, as shown in Table 4. From the Table we observe that sequence-to-sequence-based method performs well at word level but it significantly degrades when extending to text-lines. Contrary, the Transformer-based HTR method achieves a better performance when dealing with longer text sequences. Since the transformer network has the ability of dealing with long-term dependencies, it becomes more powerful to control the textual content of the generated samples.

Finally, we analyze the two schemes, either character-wise encoding (g_1) or global string encoding (g_2), to merge with the visual appearance feature as condition for the generation process. As shown in Table 5, the best performance is achieved with the use of both local and global encodings. Thus, the two schemes are utilized altogether in our model.

TABLE 5: Ablation study on the use of character-wise encoding (local feature) and global string encoding (global feature). vFID values are calculated on the IAM at word-level (1-10 characters).

Local (g_1)	Global (g_2)	vFID
✓		162.38
	✓	149.07
✓	✓	**146.74**

5.5 Latent Space Interpolation

Once the system is trained, the generator G has learned a map in the handwriting style latent space. Each writer corresponds to a point in this latent space and different writers are connected in a continuous way. Thus, we can explore it by randomly choosing two writers and try to traverse between the corresponded two points in the style latent

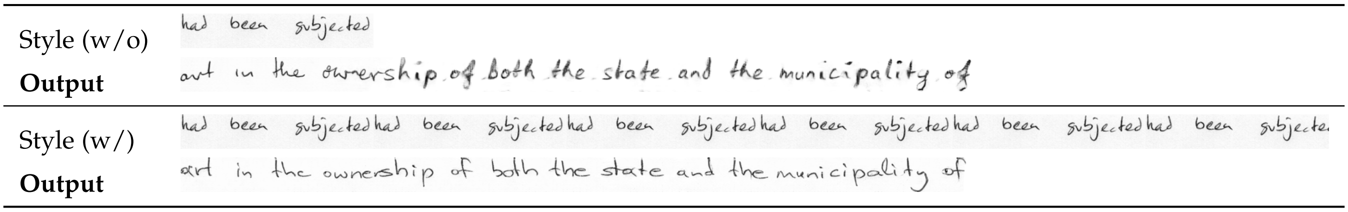

Style (w/o)
Output
had been subjected
art in the ownership of both the state and the municipality of
Style (w/)
Output
had been subjectedhad been subjectedhad been subjectedhad been subjectedhad been subjectedhad been subjecte
art in the ownership of both the state and the municipality of

Fig. 9: Comparison of the generated results for the same text string "art in the ownership of both the state and the municipality of" without (upper) and with (lower) the periodic padding process.

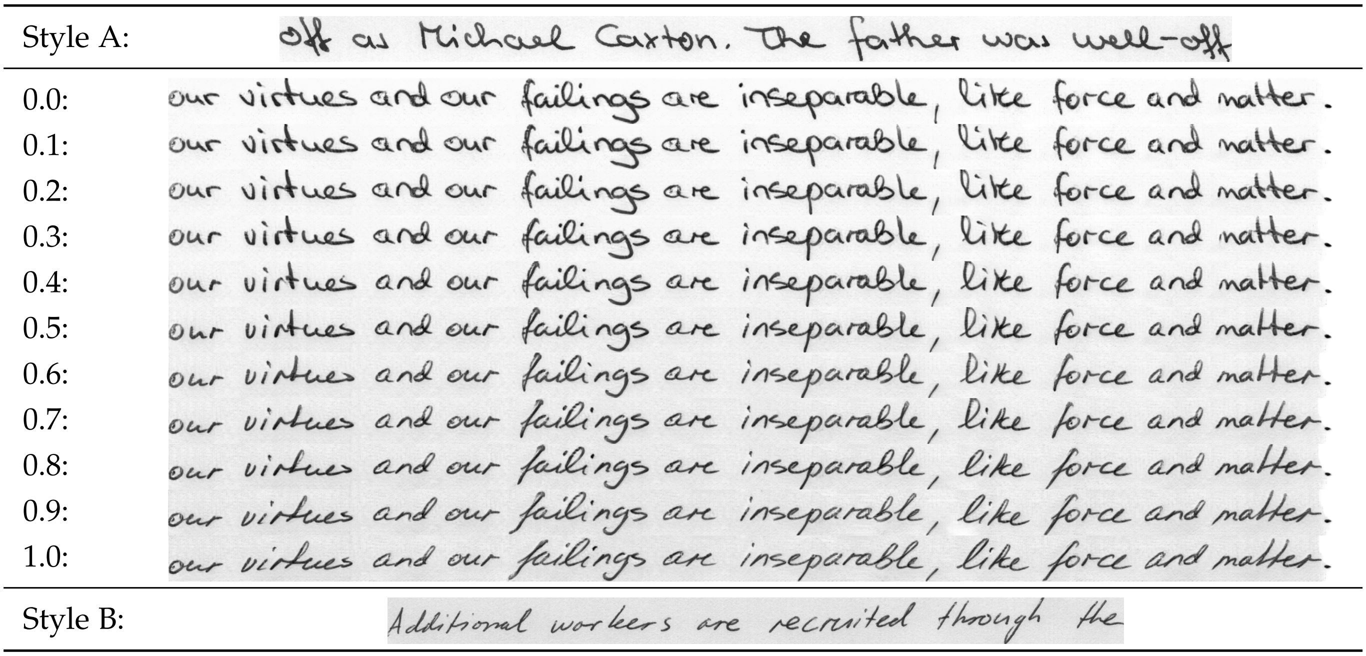

Style A:
Off as Michael Caxton. The father was well-off
0.0:
0.1:
0.2:
0.3:
0.4:
0.5:
0.6:
0.7:
0.8:
0.9:
1.0:
our virtues and our failings are inseparable, like force and matter.
Style B:
Additional workers are recruited through the

Fig. 10: Example of interpolations in the style latent space.

space as shown in Figures 10 and 11. The first and last rows show the real samples from writer A and B, respectively. The samples in between are synthetically generated ones that try to traverse from writer A to B. The rendered text is the quote of "our virtues and our failings are inseparable, like force and matter" from Nikola Tesla, which has not been seen during training.

5.6 Handwritten Text-line Generation

For the qualitative experiments, we show the results in two cases. First, given a same writing style, we try to generate samples with different text strings. Second, given a specific text string, we try to generate samples in different writing styles. The first case is shown in Figure 12. The text string is the quote of "the progressive development of man is vitally dependent on invention." from Nikola Tesla. We translate it into German, French and Spanish while replacing special characters with the corresponding letters (e.g. "é" to "e"). In Figure 12, the first row is a sample of the style input, and the following rows are (text string, synthetically generated sample) pairs. By showing the generation on different languages, our method proves to be not restricted to a training corpus nor a language model. Thus, this method can be applied to generate any OOV words and sentences. The second case is shown in Figure 13. The first row is the text string input, and the following rows are (handwriting style sample, synthetically generated sample) pairs. From the results we observe that our method has the ability to generate text-line samples with diverse writing styles.

Furthermore, we show a comparative with the state-of-the-art methods on handwritten text generation in Table 6. In our previous work [16], we have conducted a human evaluation study to show that the generated samples are indistinguishable by humans. However, in this paper we focus on the improvement of HTR performance, so the interested reader is referred to our previous publication for details on the human evaluation study.

5.7 HTR Performance Improvement

As discussed before, our method has achieved good performance on generating realistic handwritten text-line images with varied styles. These generated data could indeed be used as training data in order to improve the HTR performance at text-line level. For this purpose, we define three settings: first, a conventional supervised learning on the IAM dataset; second, transfer learning from the IAM to the Rimes dataset; and third, a realistic few-shot setting on the Spanish Numbers dataset. To be fairly comparable, we do not use any data augmentation techniques nor pretrained modules.

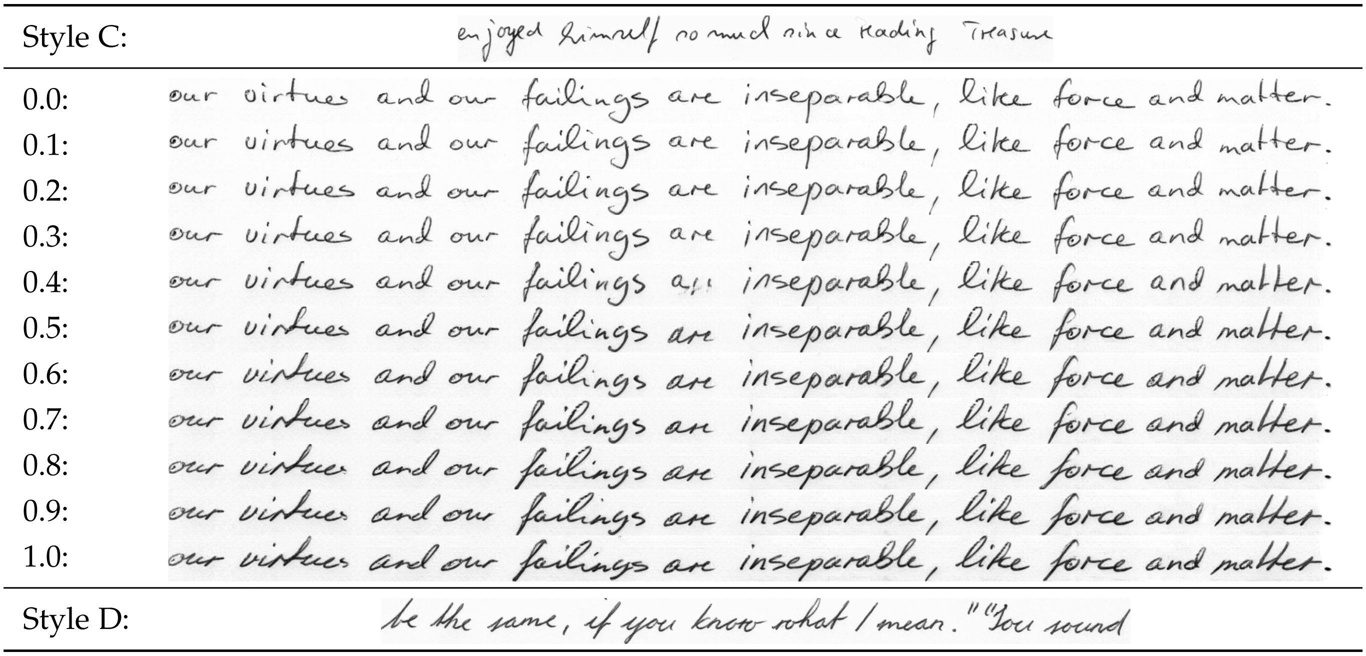

Style C:
enjoyed himself so much since Reading Treasure
0.0:
0.1:
0.2:
0.3:
0.4:
0.5:
0.6:
0.7:
0.8:
0.9:
1.0:
our virtues and our failings are inseparable, like force and matter.
Style D:
be the same, if you know what I mean." "You sound

Fig. 11: Example of interpolations in the style latent space.

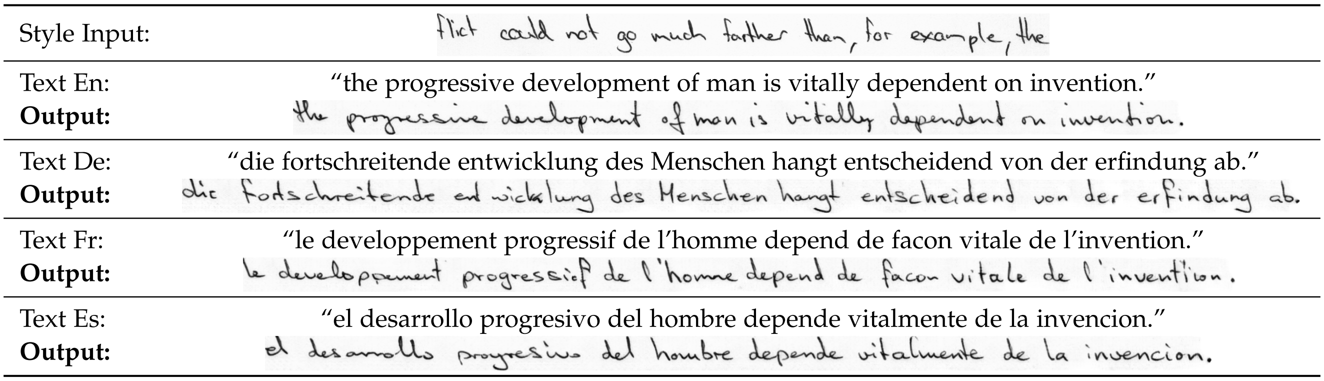

Style Input:
flict could not go much farther than, for example, the
Text En:
"the progressive development of man is vitally dependent on invention."
Output:
Text De:
"die fortschreitende entwicklung des Menschen hangt entscheidend von der erfindung ab."
Output:
Text Fr:
"le developpement progressif de l'homme depend de facon vitale de l'invention."
Output:
Text Es:
"el desarrollo progresivo del hombre depende vitalmente de la invencion."
Output:

Fig. 12: Generation on varied multi-lingual texts.

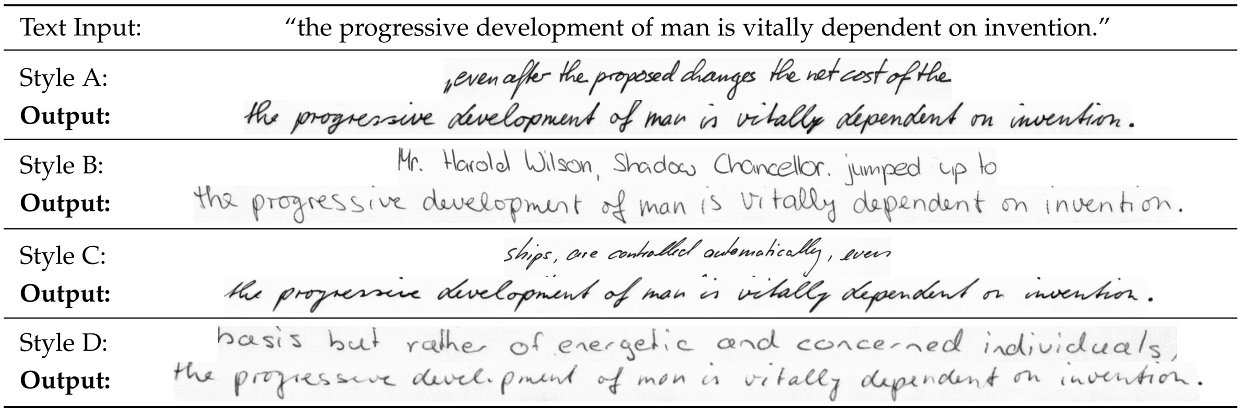
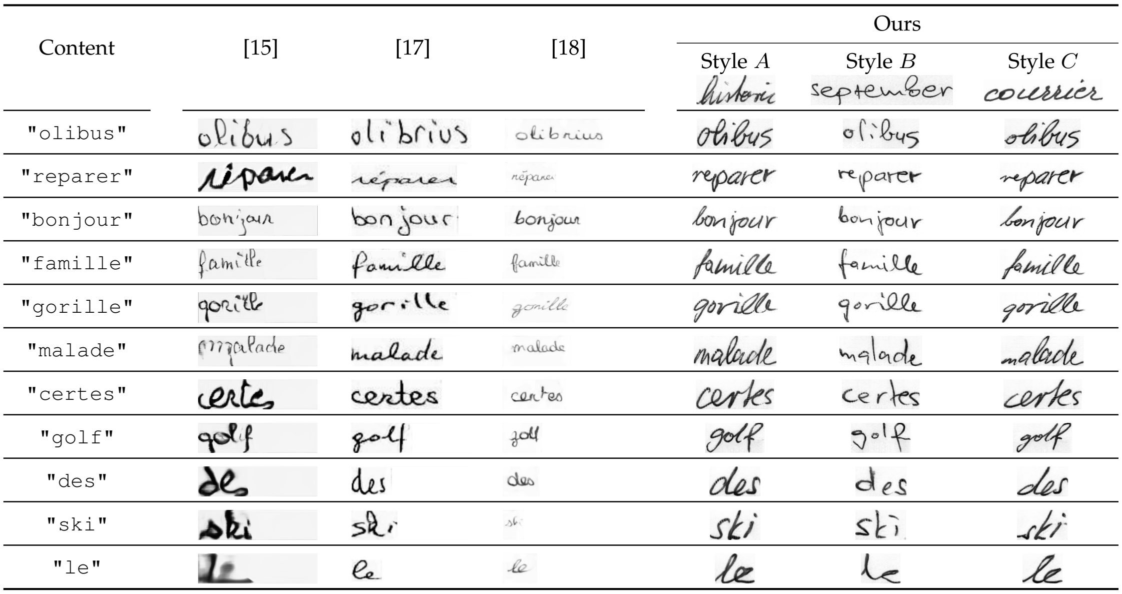

Text Input:
"the progressive development of man is vitally dependent on invention."
Style A:
, even after the proposed changes the net cost of the
Output:
Style B:
Mr. Harold Wilson, Shadow Chancellor, jumped up to
Output:
Style C:
ships, are controlled automatically, even
Output:
Style D:
basis but rather of energetic and concerned individuals,
Output:

Fig. 13: Generation of varied styles.

In all the HTR experiments, we make use of an independently trained handwritten text recognizer, which shares the same architecture with R as detailed in Figure 5 and is trained using the IAM training set at text-line level. If we follow the same experimental setting along the first row (baseline) of Table 7 with the jointly trained recognizer, we achieve the CER of 16.73%. In contrast, in the first row of Table 7, we can achieve a CER of 10.46% with an independently trained recognizer. From the comparison, we notice that the jointly trained recognizer becomes overfitted

TABLE 6: Qualitative comparison with Alonso *et al.* [15], Fogel *et al.* [17], and Davis *et al.* [18]. These images are cropped from their papers. Three random writing styles are selected in our results, where Style A is IAM writer 583, Style B is IAM writer 281, and Style C is Rimes writer lot_13_01258.


Content
[15]
[17]
[18]
Ours
Style A
Style B
Style C
"olibus"
"reparer"
"bonjour"
"famille"
"gorille"
"malade"
"certes"
"golf"
"des"
"ski"
"le"

during the image generation process. This overfitting effect of the recognizer benefits the image generation, because the overfitted recognizer is sensitive to the data noise, which in return guides the generated image to be cleaner and more readable.

5.7.1 Enhance the training set

The most straightforward way to improve the HTR performance is to incorporate extra synthetically generated data to the training set. In Table 7, we evaluate the improvements in different cases. The first row shows the results when using the IAM training set only. To keep the training data balanced between real and generated samples, we generate $8,000$ text-line images based on the style of the IAM images and a lexicon. Concerning the lexicon, we have two choices: WikiText-103 or the groundtruth of IAM training set, shown in the second and third row, respectively. Note that the HTR performance is boosted when adding the $8,000$ synthetically generated samples. Furthermore, the choice of lexicon also matters because the performance is further boosted when using a similar lexicon to the target dataset. Finally, we even apply data augmentation techniques on both the real and generated training samples so that we end up with the best result as shown in the fourth row. Furthermore, our method shows a significant improvement over the performance achieved by ScrabbleGAN [17] in comparable settings. Thus, we can conclude that our proposed generative method generates useful samples that are useful to train HTR networks.

5.7.2 Transfer learning on a new dataset

Another useful setting is transfer learning, which consists of transferring a trained recognizer to an unknown dataset. In this case, the source data is the IAM dataset and the target is the Rimes dataset. Both datasets are at text-line level and share some characters in vocabularies such as English letters, space, punctuation marks and numbers. However, the IAM dataset is in English while the Rimes dataset is in French, so some special letters like “é” or “â” are exclusive from the Rimes dataset. This scenario may occur in real use cases in which there is a *general* recognizer, which has been properly trained, that is used to recognize a target dataset, containing some exclusive characters. Instead of manually labeling a subset of target data and training the recognizer again, we could provide a faster solution: to generate a set of synthetic samples mimicking the style of target dataset and then fine-tuning on it. In this way, the HTR performance on the target data is boosted to some extent with a manual-free effort, although it can not recognize those special characters.

TABLE 7: HTR experiments. Results are evaluated on the IAM test set at text-line level. The Error rate reduction is calculated taking the results of the first and last rows.

Aug.	GAN	Lexicon	ScrabbleGAN [17] CER	ScrabbleGAN [17] WER	**Proposed** CER	**Proposed** WER
–	–	IAM	13.82	25.10	10.46	33.40
–	✓	WikiText			9.66	31.87
–	✓	IAM			9.37	30.58
✓	✓	IAM	13.57	23.98	8.62	26.69
Error Rate Reduction (%)			1.8	4.5	**17.6**	**20.1**

Table 8 shows the transfer learning results. In the first row, considered as a lower bound, the recognizer is pre-trained on the IAM dataset and directly evaluated on the Rimes test set. As an upper bound, we train the recognizer from scratch using the Rimes dataset. Below these two baselines, we show the performance when using the IAM training set and $8,000$ synthetically generated samples using IAM handwriting styles and random text strings from WikiText-103. Secondly, we assume that we have access to images of the Rimes dataset (but not their labels), and we generate $8,000$ synthetic samples that mimic the style of the Rimes dataset while sampling text strings from WikiText-103. We observe that by incorporating these extra synthetically generated samples, the HTR performance for the unlabeled Rimes target dataset is boosted in a transfer learning setting (the CER decreases from 27.3% down to 18.19%).

TABLE 8: Transfer learning setting from IAM to Rimes. Results are evaluated on Rimes test set at text-line level. Only the second row has access to labeled Rimes data, while the Adaptation indicates the usage of unlabeled images and external text strings.

	Train set	Adaptation	CER	WER
Baselines	IAM	–	27.30	74.57
	Rimes	–	6.45	19.56
Transfer	IAM	IAM + WikiText (8K)	20.55	63.20
	IAM	Rimes + WikiText (8K)	18.19	54.83

5.7.3 Few-shot setting on target dataset

We are also interested in investigating how to make use of the generated images to improve the HTR performance in another realistic scenario: when the target dataset is very small, such as the Spanish Number dataset. Here, we take our baseline method, a recognizer pretrained on IAM dataset, and test it with the test set of Spanish Number data directly, so that we obtain the lower bound with CER 27.82% as shown as the dashed black line in Figure 14. We hypothesize that we have access to the whole labeled training set of Spanish Number data (298 images), thus we further fine-tune our pretrained recognizer and achieve a CER of 4.94%, which is the ideal case as shown as the dashed magenta line. Then, we select 5, 10, 20, 40, 80, 160 labeled real samples from the Spanish Number training set randomly to carry on the next experiments. Based on our baseline recognizer, we fine-tune on the selected subset of labeled real images to end up with the red curve. Ten individual experiments have been done for each subset of labeled real data, and the data sampling process is also randomly done for every experiment. From the red curve, we can see that the performance is significantly improved with few labeled real samples, while remaining steady when adding more data.

For comparisons, we carry on experiments with a sequence-to-sequence method that uses synthetic handwritten images based on TrueType fonts [50]. To avoid a large unbalance between synthetic and real data, we make use of 100, 300, 500, 700 and 900 synthetic data with a specific amount of real subset (indicated as x-axis) to fine-tune the recognizer. We carry on 10 individual experiments with randomly selected synthetic and real subsets, so that we obtain the green curve that behaves better than the red one. The results prove that using extra synthetic data along the training set boosts the HTR performance. However, the handwriting style diversity that the synthetic data provides is very limited to the chosen fonts, so the improvement is also limited.

Since we already have the generative model pretrained on the IAM dataset, we produce synthetic samples based on the unlabeled Spanish Number images and random text strings of WikiText-103. We follow the same experimental setting with the TrueType font based experiments, and generate the blue curve. We observe that our generated samples significantly boost the HTR performance over both the red and green curves. Even more, when using our generated samples with 160 labeled real ones, the recognizer performs better than when using the whole real training set (298 images).

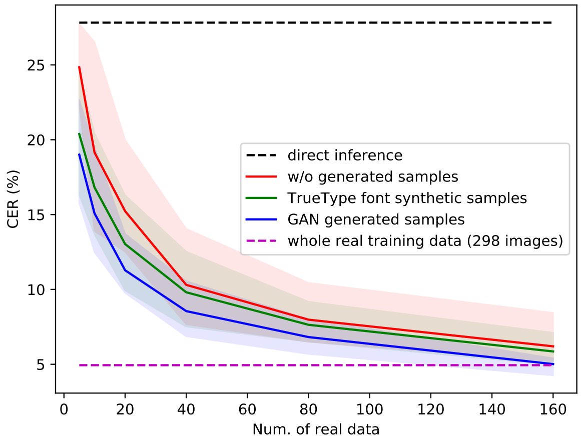


Fig. 14: HTR improvement in a real use case on Spanish Number dataset.

6 Conclusion and Future Work

In this work we have presented a generative method to produce realistic and varied artificially rendered samples of handwritten text-line images. With the usage of periodic padding module, the method is able to generate samples of any length disregarding the length of style input. By replacing the Seq2Seq-based recognizer with the Transformer-based one, the ability to generate longer text-line images is obtained. Higher quality results are achieved by training with curriculum learning. Extensive qualitative results have demonstrated the high capacity to generate realistic handwritten text-line images by conditioning the generative process with both visual appearance and textual content information. In addition, our method is able to generate any text-line sample without restriction to any predefined vocabulary, and even work on other languages (except special characters like accents). Also, and once properly trained, the inference can also run in a few-shot setup for the target handwriting style images.

Furthermore, comprehensive studies on making use of generated samples in both supervised and transfer learning settings have proven that our generated samples can effectively boost the HTR performance with almost no manual effort. Indeed, when comparing to other existing handwritten text generation methods, our method is the one that obtains the most significant HTR improvement (an error reduction of 17.6% in CER and 20.1% in WER).

The method presented in this paper focuses on handwritten data, but in the future, we could further incorporate typed text data. The intuition will be that if the method could generate both cursive handwriting and non-cursive typed text data, the visual style transfer from any random handwritten images to unified typed text samples may achieve a good performance, which could drastically improve the HTR performance nowadays.

Acknowledgments

This work has been partially supported by the grant 140/09421059 from Shantou University, the Spanish project RTI2018-095645-B-C21, the grant 2016-DI-087 from the Secretaria d'Universitats i Recerca del Departament d'Economia i Coneixement de la Generalitat de Catalunya, the Ramon y Cajal Fellowship RYC-2014-16831 and the CERCA Program/ Generalitat de Catalunya.

References

[1] P. Krishnan and C. Jawahar, "Hwnet v2: An efficient word image representation for handwritten documents," *International Journal on Document Analysis and Recognition*, vol. 22, no. 4, pp. 387–405, 2019.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the Conference on Neural Information Processing Systems*, 2014.

[3] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8110–8119.

[4] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[5] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[6] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[7] D. Ha and D. Eck, "A neural representation of sketch drawings," in *Proceedings of the International Conference on Learning Representations*, 2018.

[8] N. Zheng, Y. Jiang, and D. Huang, "Strokenet: A neural painting environment," in *Proceedings of the International Conference on Learning Representations*, 2019.

[9] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[10] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "MoCoGAN: Decomposing motion and content for video generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[11] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[12] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[13] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "DRAW: A recurrent neural network for image generation," in *Proceedings of the International Conference on Machine Learning*, 2015.

[14] B. Chang, Q. Zhang, S. Pan, and L. Meng, "Generating handwritten Chinese characters using CycleGAN," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2018.

[15] E. Alonso, B. Moysset, and R. Messina, "Adversarial generation of handwritten text images conditioned on sequences," in *Proceedings of the International Conference on Document Analysis and Recognition*, 2019.

[16] L. Kang, P. Riba, Y. Wang, M. Rusiñol, A. Fornés, and M. Villegas, "Ganwriting: Content-conditioned generation of styled handwritten word images," in *Proceedings of the European Conference on Computer Vision*, 2020.

[17] S. Fogel, H. Averbuch-Elor, S. Cohen, S. Mazor, and R. Litman, "Scrabblegan: Semi-supervised varying length handwritten text generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4324–4333.

[18] B. Davis, C. Tensmeyer, B. Price, C. Wigington, B. Morse, and R. Jain, "Text and style conditioned gan for generation of offline handwriting lines," in *Proceedings of the British Machine Vision Conference*, 2020.

[19] J. Wang, C. Wu, Y.-Q. Xu, and H.-Y. Shum, "Combining shape and physical modelsfor online cursive handwriting synthesis," *International Journal on Document Analysis and Recognition*, vol. 7, no. 4, pp. 219–227, 2005.

[20] Z. Lin and L. Wan, "Style-preserving english handwriting synthesis," *Pattern Recognition*, vol. 40, no. 7, pp. 2097–2109, 2007.

[21] T. Konidaris, B. Gatos, K. Ntzios, I. Pratikakis, S. Theodoridis, and S. J. Perantonis, "Keyword-guided word spotting in historical printed documents using synthetic data and user feedback," *International Journal on Document Analysis and Recognition*, vol. 9, no. 2-4, pp. 167–177, 2007.

[22] A. O. Thomas, A. Rusu, and V. Govindaraju, "Synthetic handwritten captchas," *Pattern Recognition*, vol. 42, no. 12, pp. 3365–3373, 2009.

[23] T. S. Haines, O. Mac Aodha, and G. J. Brostow, "My text in your handwriting," *ACM Transactions on Graphics*, vol. 35, no. 3, pp. 1–18, 2016.

[24] L. Wu, C. Zhang, J. Liu, J. Han, J. Liu, E. Ding, and X. Bai, "Editing text in the wild," in *Proceedings of the 27th ACM international conference on multimedia*, 2019, pp. 1500–1508.

[25] P. Roy, S. Bhattacharya, S. Ghosh, and U. Pal, "Stefann: scene text editor using font adaptive neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 228–13 237.

[26] Q. Yang, J. Huang, and W. Lin, "Swaptext: Image based texts transfer in scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 700–14 709.

[27] X.-Y. Zhang, F. Yin, Y.-M. Zhang, C.-L. Liu, and Y. Bengio, "Drawing and recognizing chinese characters with recurrent neural network," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 849–862, 2017.

[28] Y. Ganin, T. Kulkarni, I. Babuschkin, S. Eslami, and O. Vinyals, "Synthesizing programs for images using reinforced adversarial learning," in *Proceedings of the International Conference on Machine Learning*, 2018.

[29] M. Mayr, M. Stumpf, A. Nikolaou, M. Seuret, A. Maier, and V. Christlein, "Spatio-temporal handwriting imitation," *arXiv preprint arXiv:2003.10593*, 2020.

[30] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proceedings of the International Conference on Learning Representations*, 2014.

[31] P. Lyu, X. Bai, C. Yao, Z. Zhu, T. Huang, and W. Liu, "Auto-encoder guided GAN for Chinese calligraphy synthesis," in *Proceedings of the International Conference on Document Analysis and Recognition*, 2017.

[32] Y. Tian, "zi2zi: Master chinese calligraphy with conditional adversarial networks," 2017. [Online]. Available: https://github.com/kaonashi-tyc/zi2zi

[33] H. Jiang, G. Yang, K. Huang, and R. Zhang, "W-net: one-shot arbitrary-style Chinese character generation with deep neural networks," in *Proceedings of the International Conference on Neural Information Processing*, 2018.

[34] S.-J. Wu, C.-Y. Yang, and J. Y.-j. Hsu, "Calligan: Style and structure-aware chinese calligraphy character generator," *arXiv preprint arXiv:2005.12500*, 2020.

[35] S. Azadi, M. Fisher, V. G. Kim, Z. Wang, E. Shechtman, and T. Darrell, "Multi-content gan for few-shot font style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7564–7573.

[36] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the International Conference on Machine Learning*, 2006, pp. 369–376.

[37] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 172–189.

[38] L. Kang, P. Riba, M. Rusiñol, A. Fornés, and M. Villegas, "Pay attention to what you read: Non-recurrent handwritten text-line recognition," *arXiv preprint arXiv:2005.13044*, 2020.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the Conference on Neural Information Processing Systems*, 2017, pp. 5998–6008.

[40] D. Dowson and B. Landau, "The fréchet distance between multivariate normal distributions," *Journal of multivariate analysis*, vol. 12, no. 3, pp. 450–455, 1982.

[41] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Proceedings of the Conference on Neural Information Processing Systems*, 2017, pp. 6626–6637.

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[43] P. Wang, Y. Cao, C. Shen, L. Liu, and H. T. Shen, "Temporal pyramid pooling-based convolutional neural network for action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 12, pp. 2613–2622, 2016.

[44] U.-V. Marti and H. Bunke, "The iam-database: an english sentence database for offline handwriting recognition," *International Journal on Document Analysis and Recognition*, vol. 5, no. 1, pp. 39–46, 2002.

[45] E. Augustin, M. Carré, E. Grosicki, J.-M. Brodin, E. Geoffrois, and F. Prêteux, "Rimes evaluation campaign for handwritten mail processing," in *International Workshop on Frontiers in Handwriting Recognition*, 2006, pp. 231–235.

[46] A. H. Toselli, A. Juan, J. González, I. Salvador, E. Vidal, F. Casacuberta, D. Keysers, and H. Ney, "Integrated handwriting recognition and interpretation using finite-state models," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 04, pp. 519–539, 2004.

[47] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.

[48] V. Frinken and H. Bunke, "Continuous handwritten script recognition," in *Handbook of Document Image Processing and Recognition*, 2014, pp. 391–425.

[49] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NeurIPS 2017 Autodiff Workshop*, 2017.

[50] L. Kang, P. Riba, M. Villegas, A. Fornés, and M. Rusiñol, "Candidate fusion: Integrating language modelling into a sequence-to-sequence handwritten word recognition architecture," *Pattern Recognition*, vol. 112, p. 107790, 2021.

Lei Kang received the B.Sc. degree from Jilin University, Changchun, China in 2012, M.Sc. degree from University of Science and Technology of China, Hefei, China in 2015, and Ph.D. degree from Computer Vision Center, Universitat Autònoma de Barcelona, Barcelona, Spain and omni:us, Berlin, Germany in 2020. He is currently a lecturer of Computer Science Dept. at Shantou University, Shantou, China. His main research interests include Transfer Learning, Domain Adaptation, Attention Mechanisms of Seq2Seq Model and GANs applied to the problem of Handwritten Text Recognition and Synthesis.

Pau Riba received the B.Sc. degrees in Mathematics and Computer Science, the M.Sc. and Ph.D. degrees in Computer Vision from the Universitat Autònoma de Barcelona, in 2015, 2016 and 2020, respectively. Currently, he worked as a postdoctoral research fellow in the Computer Vision Center. Currently, he works as an AI research engineer at Helsing AI. His main research interests revolve around Self-supervised Learning, Graph-based Representations and Machine Learning. P. Riba has actively participated in the organization of the GMPRDIA tutorial within the ICDAR 2019 and the GREC workshop within the ICDAR 2021. In addition, he has been awarded the ”Best paper award” in ICFHR 2020 and ICPR 2018.

Marçal Rusiñol received his B.Sc., M.Sc. and Ph.D. degrees in Computer Sciences from the Universitat Autònoma de Barcelona, in 2004, 2006, and 2009 respectively. In 2012 and 2014 he worked as a Marie Curie research fellow at Itesoft and Université de La Rochelle, France, in 2012 and 2014 respectively. In 2019 he co-founded the spinoff company AllRead MLT where he currently works.

Alicia Fornés is a senior research fellow at the Universitat Autonoma (UAB) de Barcelona and the Computer Vision Center. She obtained the Ph.D. degree in Computer Science from the UAB in 2009. She was the recipient of the AERFAI (Spanish brand of the IAPR, International Association for Pattern Recognition) best thesis award 20092010, and the IAPR/ICDAR Young Investigator Award in 2017. She has more than 100 publications related to document analysis and recognition. Her research interests include document image analysis, handwriting recognition, optical music recognition, writer identification and digital humanities.

Mauricio Villegas received M.Sc degree on Pattern Recognition and Ph.D. degree on Computer Science from the Universitat Politècnica de València, in 2008 and 2011, respectively. He is currently a Senior Data Scientist at omni:us, Berlin, Germany. He participated in two EU funded projects FP7 and H2020, both related to hand-written text recognition, and organized competitions related to automatic image annotation (ImageCLEF 2013-2016), handwritten text recognition (ICDAR 2017) and handwritten document retrieval (ImageCLEF 2016).