Handwritten Text Generation from Visual Archetypes

Vittorio Pippi, Silvia Cascianelli, Rita Cucchiara
University of Modena and Reggio Emilia
Via Pietro Vivarelli, 10, Modena (Italy)
{name.surname}@unimore.it

Abstract

Generating synthetic images of handwritten text in a writer-specific style is a challenging task, especially in the case of unseen styles and new words, and even more when these latter contain characters that are rarely encountered during training. While emulating a writer's style has been recently addressed by generative models, the generalization towards rare characters has been disregarded. In this work, we devise a Transformer-based model for Few-Shot styled handwritten text generation and focus on obtaining a robust and informative representation of both the text and the style. In particular, we propose a novel representation of the textual content as a sequence of dense vectors obtained from images of symbols written as standard GNU Unifont glyphs, which can be considered their visual *archetypes. This strategy is more suitable for generating characters that, despite having been seen rarely during training, possibly share visual details with the frequently observed ones. As for the style, we obtain a robust representation of unseen writers' calligraphy by exploiting specific pre-training on a large synthetic dataset. Quantitative and qualitative results demonstrate the effectiveness of our proposal in generating words in unseen styles and with rare characters more faithfully than existing approaches relying on independent one-hot encodings of the characters.*

1. Introduction

Styled handwritten text generation (HTG) is an emerging research area aimed at producing writer-specific handwritten text images mimicking their calligraphic style [7, 14, 29]. The practical applications of this research topic range from the synthesis of high-quality training data for personalized Handwritten Text Recognition (HTR) models [5, 6, 8, 27, 28, 48] to the automatic generation of handwritten notes for physically impaired people. Moreover, the writer-specific style representations that can be obtained as a by-product of models designed for this task can be applied to other tasks such as writer identification, signature verification, and handwriting style manipulation. When focusing on styled handwriting generation, simply adopting style transfer is limiting. In fact, imitating a specific writer's calligraphy does not only concern texture (*e.g.*, the color and texture of background and ink), nor just stroke thickness, slant, skew, and roundness, but also single characters shape and ligatures. Moreover, these visual aspects must be handled properly to avoid artifacts that might result in content change (*e.g.*, even small additional or missing strokes).

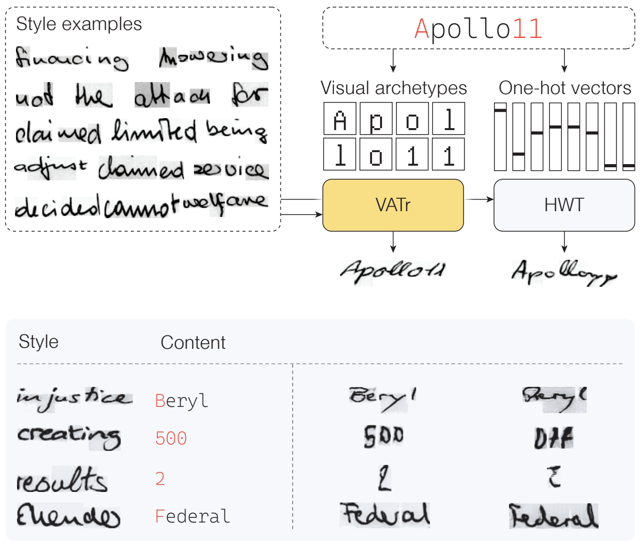

Figure 1. Different from previous approaches that use independent one-hot vectors as input text tokens (*e.g.*, the State-of-the-Art HWT [7]), we exploit *visual archetypes*, *i.e.*, geometrically-related binary images of characters. By resorting to similarities between the archetypes, we are able to generate both characters that are rarely seen during training (highlighted in red) and frequently observed ones more faithfully.

In sight of this, specific approaches have been designed for HTG. The handwriting can be handled in the form of a trajectory (made of the underlying strokes), as done in [1, 2, 19, 24, 31], or of an image that captures its appearance, as done in [3, 7, 11, 14–16, 20, 29, 32, 35, 36, 39, 43, 45]. The former approaches adopt online HTG strategies that entail predicting the pen trajectory point-by-point, while the latter ones are offline HTG models that output entire text images directly. We follow the offline HTG paradigm since it has the advantage, over the online one, of not requiring costly pen-recording training data, and thus, being applicable also to scenarios where the information on online handwriting is not available for a specific author (*e.g.*, in the case of historical data) and being easier to train for not suffering of vanishing gradient and being parallelizable.

Specifically, in this work, we focus on the *Few-Shot styled offline HTG* task, in which we have just a few example images of the writer's style to mimic. State-of-the-Art (SotA) approaches tackling this scenario feature an encoder that extracts writer-specific style features and a generative component, which is fed with the style features and the content representations, and produces styled text images conditioned on the desired content. These approaches usually exploit Generative Adversarial Networks (GANs [18, 40]), for example [3, 11, 14–16, 29, 32, 36]. A more recent approach [7] is based on an encoder-decoder generative Transformer model [44] that captures character-level style variations better than previous GAN-based strategies thanks to the cross-attention mechanism between style representation and content tokens. In the approaches mentioned above, the encoding of the text content is obtained by starting from one-hot vectors, each representing a different character in a fixed charset. In this way, the characters are all independent by design. Thus, possible geometric and visual similarity among them cannot be modeled nor exploited for generation, which might result in a quality gap between the images generated by these approaches for characters that are highly represented in the training set and rare ones (*i.e.*, long-tail characters). Moreover, for computational tractability, the fixed charset that the approaches relying on a one-hot representation of text tokens can handle is relatively small.

Contribution. Our proposed approach entails representing characters as *continuous variables* and using them as query content vectors of a Transformer decoder for generation. In this way, the generation of characters appearing rarely in the training set (such as numbers, capital letters, and punctuation) is eased by exploiting the low distance in the latent space between rare symbols and more frequent ones (see Figure 1). In particular, we start from the GNU Unifont font and render each character as a 16×16 binary image, which can be considered as the *visual archetype* of that character. Then, we learn a dense encoding of the character images and feed such encodings to a Transformer decoder as queries to attend the style vectors extracted by a Transformed encoder. Note that, by resorting to character images rendered in the richer GNU Unifont, which is the most complete in terms of contained Unicode characters, we can handle a huge charset (more than 55k characters) seamlessly, *i.e.*, without the need for additional parameters, as it is the case for the commonly-adopted one-hot encoding. Moreover, as for the style encoding part, we exploit a backbone to represent the style example images that has been pre-trained on a large synthetic dataset specifically built to focus on the calligraphic style attributes. This strategy, widely adopted for other tasks, is usually disregarded in HTG. Nonetheless, we demonstrate its effectiveness in leading to strong style representations, especially for unseen styles. We validate our proposal with extensive experimental comparison against recent generative SotA approaches, both quantitatively and qualitatively, and demonstrate the effectiveness of our proposal in generating words with both common and rare characters and in both seen and unseen styles. We call our approach VATr: Visual Archetypes-based Transformer. The code and trained models are available at https://github.com/aimagelab/VATr.

2. Related Work

HTG is related to the Font Synthesis task, where the desired style must be represented and used to render characters consistently [4, 9, 33, 41, 46]. However, Font Synthesis approaches just need to generate single characters, thus, are more closely related to HTG for ideogrammatic languages [10, 17, 25, 47]. In general, both for ideogrammatic and non-ideogrammatic languages, handwriting can be treated either as a trajectory capturing the shape of the strokes making up the characters or as a static image capturing their overall appearance. Depending on this conception, online or offline approaches to HTG can be applied.

Online HTG. Approaches to online HTG exploit sequential models, such as LSTMs [19], Conditional Variational RNNs [2], or Stochastic Temporal CNNs [1], to predict the pen position point-by-point based on its current position and the input text to be rendered. The first approach following this strategy was proposed in [19] and did not have control over the style. This limitation was then addressed by following works by decoupling and then recombining content and writer's style [1, 2, 31]. Further improvements to online HTG approaches can be obtained by training the sequential model alongside a discriminator [24], which is a philosophy similar to SotA GAN-based offline HTG approaches. The main drawbacks of approaches following the online HTG strategy are that they struggle to learn long-range dependencies and that they require training data consisting of digital pen recordings, which are difficult to collect or even impossible to obtain for application scenarios such as historical manuscripts. In the sight of these limitations, in this work, we follow the offline HTG paradigm.

Offline HTG. Traditional offline HTG solutions [20, 35, 43, 45] resort to heavy human intervention for glyphs segmentation and then apply handcrafted geometric static-based feature extraction before combining those glyphs with appropriate ligatures and rendering them with texture and

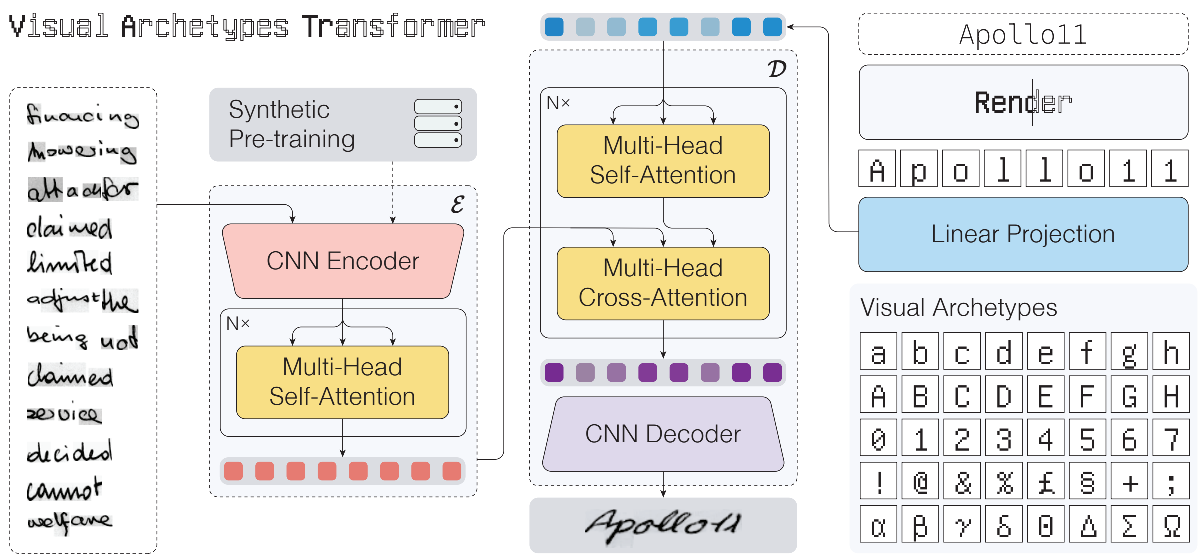

Figure 2. Overview of our Visual Archetypes-based Transformer for HTG (VATr). Few-shot learning is provided by a few images of the desired calligraphic style, encoded via a convolutional backbone pre-trained on a large synthetic dataset; the output vector is passed through a Transformer encoder for creating a latent space with robust style vectors (the *Style Encoder* $\mathcal{E}$ on the left). The text to generate is rendered as a sequence of GNU Unifont binary images, representing the visual archetypes of the characters. These are the queries of a Transformer decoder to perform cross-attention with the style vectors. The resulting content-style representation is then fed to a convolutional decoder that outputs the styled handwritten text image. These last two components are our *Content-Guided Encoder* $\mathcal{D}$.

background blending. Other than the costly human intervention that these approaches entail, their main limitation is that they can only render the glyphs and ligatures observed for each style. More recent deep learning-based approaches, instead, are able to infer styled glyphs even if not directly observed in the style examples. Learning-based solutions rely on GANs, either unconditioned (for non-styled HTG) or conditioned on a varying number of handwriting style samples (for styled HTG). In this latter case, style samples can be entire paragraphs or lines [11], a few words [7, 29], or a single word [15, 16, 36]. Collecting a few handwriting samples from a writer is not much more costly than one and generally results in better performance [32]. The first of these approaches was proposed in [3] and was able to generate fixed-sized images conditioned on the content embedding but with no control over the calligraphic style. Note that, different from natural image generation, generating handwritten text images should entail producing variable-sized images. Thus, the approach presented in [14] aims at overcoming this limitation by concatenating character images, still not being able to imitate handwriting style.

Approaches tackling styled HTG condition the generation not only on the text content but also on a vector representation of the style [11,15,16,29,39]. In such approaches, the style and the content representations are obtained separately and then combined in a later stage for generation. This prevents those approaches from effectively capturing local writing style and patterns. This limitation is addressed by the Transformer-based approach proposed in [7], which is able to better capture content-style entanglement by exploiting the cross-attention mechanism between the style vector representation and the content text representation. In this work, we follow the Transformer-based paradigm for its superior capability of rendering local style patterns.

Content Representation. The content tokens used in the approaches mentioned above [3,7,11,14–16,29,39] are usually independent one-hot vectors, each representing a character in a finite and generally small charset. This strategy is thus limiting due to the relatively small charset that can be handled with reasonable computational cost and is inefficient for hindering the possibility of leveraging similarities between characters. Our approach, instead, is to exploit character images as text content inputs. Note that the approaches proposed in [32, 36] feed the textual input as a whole image containing the desired text written in a typeface font. These images are then rendered in the desired style in a style-transfer fashion, also exploiting the geometry encoded in the typeface-written image for letter spacing and curvature. Different from these approaches, we input text tokens as sequences of character images rendered in the richer GNU Unifont, which is as modular as the approaches employing one-hot encodings and allows exploiting geometric similarities between characters for generation.

3. Proposed Approach

The few-shot offline HTG problem that we tackle in this work can be formulated as follows. Consider a writer

$\mathrm{w}\in\mathrm{W}$, for which we have P samples of handwritten word images at disposal, $\mathbf{X}_{\mathrm{w}}=\{\mathbf{x}_{\mathrm{w},i}\}_{i=0}^{P}$ (in this work, following [7, 29], we set $P=15$). Moreover, consider an arbitrarily long set of Q text words $\mathbf{C}=\{\mathbf{c}_i\}_{i=0}^{Q}$, each containing an arbitrary number n_i of characters. Our goal is to generate images $\mathbf{Y}_{\mathrm{w}}^{\mathbf{C}}$ of words with the content of the strings in $\mathbf{C}$ and the style of the writer w (see Figure 1, bottom).

Model Overview. We devise a Transformer encoder-decoder architecture, in combination with a pre-trained convolutional feature extractor for handling the style samples $\mathbf{X}_{\mathrm{w}}$, and rendered characters images for handling the content strings $\mathbf{C}$. First, a pre-trained convolutional feature extractor handles the style samples $\mathbf{X}_{\mathrm{w}}$ and feeds the resulting vectors to a Transformer encoder that enriches them with the long-range dependencies captured by the self-attention mechanism and outputs a sequence of style vectors $\mathbf{S}_{\mathrm{w}}$. The Transformer decoder performs cross-attention between $\mathbf{S}_{\mathrm{w}}$ and the content strings $\mathbf{C}$ to be rendered, which are represented as a sequence of their visual archetypes. The cross-attention mechanism brings to an entangled content-style representation that better captures local style patterns in addition to global word appearance. Finally, the obtained representation is fed into a convolutional decoder that generates the content and style conditioned word images $\mathbf{Y}_{\mathrm{w}}^{\mathbf{C}}$. We refer to this part of our architecture as the *Content-Guided Decoder* $\mathcal{D}$. A schematic overview of our VATr architecture is reported in Figure 2.

3.1. Style Encoder

The Style Encoder $\mathcal{E}$, which transforms the few sample images $\mathbf{X}_{\mathrm{w}}$ into the style features $\mathbf{S}_{\mathrm{w}}$, is a pipeline of a convolutional encoder and a Transformer encoder. This choice is motivated by the data efficiency of convolutional neural networks and their ability to extract representative features and the suitability of the multi-head self-attention mechanism to model long-range dependencies in the style images. The selected convolutional encoder backbone is a ResNet18 [21], which is a popular choice for approaches dealing with text images [7, 23, 37, 49]. A novel additional characteristic is a pre-training process to obtain robust features from the style sample images. For this, we exploit a specifically built large dataset of word images rendered in calligraphic fonts. The details on the pre-training dataset are given in § 3.1.1. Once pre-trained, we use the backbone to extract P feature maps $\mathbf{h}_{\mathrm{w},i}\in\mathbb{R}^{h\times w\times d}$ from the P style images $\mathbf{x}_{\mathrm{w},i}\in\mathbf{X}_{\mathrm{w}}$. These feature maps are then flattened along the spatial dimension to obtain a $(h\cdot w)$-long sequence of d-dimensional vectors. Note that while h and w depend on the input images shape, the embedding size d is fixed and set equal to 512 in this work. The elements of this sequence represent adjacent regions of the original images, corresponding to the receptive field of the convolutional backbone. The flattened feature maps of each style image are further concatenated to obtain the sequence $\mathbf{H}_{\mathrm{w}}\in\mathbb{R}^{N\times d}$, where $N=h\cdot w\cdot P$, which is fed into the first layer of the multi-layer multi-headed self-attention encoder. This encoder comprises $L=3$ layers, each with $J=8$ attention heads and a multilayer perceptron. The output of the last layer $\mathbf{H}^{L}=\mathbf{S}_{\mathrm{w}}\in\mathbb{R}^{N\times d}$ is the sequence of style features for writer w, which is fed to the Transformer decoder in $\mathcal{D}$.

3.1.1 Synthetic Pre-training

Large-scale pre-training is an effective strategy employed in a number of learning tasks. For HTG, the pre-training data should be abundant and should capture the shape variability of the glyphs and the texture characteristics of the ink and background. In the sight of these considerations, to build the dataset used for pre-training the convolutional backbone, we render 10 400 random words from the English vocabulary, each in 10 400 freely online available calligraphic fonts and on backgrounds randomly selected from a pool of paper-like images, thus obtaining more than 100M samples. To achieve better realism, we apply random transformations such as rotation and elastic deformation via the Thin Plate Spline transformation [13] to introduce shape variability, gaussian blur to avoid sharp borders and simulate handwriting strokes, and grayscale dilation and color jitter to simulate different ink types[1]. We use the so obtained dataset to train the backbone to recognize the style of the word images by minimizing a Cross-Entropy Loss. Note that by exposing the network to such variability, we force it to extract features that are representative of the calligraphic style rather than the overall image appearance (which is influenced by the textual content, the background, and the ink type or writing tool).

3.2. Content-Guided Decoder

The first block of the *Content-Guided Decoder* $\mathcal{D}$ is a multi-layer multi-head decoder with $L=3$ layers and $J=8$ heads as the encoder. The decoder performs self-attention between the content vectors representing the elements in $\mathbf{C}$, followed by cross-attention between the sequence of content vectors (treated as queries) and the style vectors $\mathbf{S}_{\mathrm{w}}$ (used as keys and values). In this way, the model can learn content-style entanglement since each query is forced to attend at the style vectors that are useful to render its final shape other than the general appearance.

Unlike existing approaches that represent the content queries as embeddings of independent one-hot-encoded characters, in this work, we propose to exploit a representation that captures similarities between characters. In particular, we obtain the content queries as follows. Each content string $\mathbf{c}_i\in\mathbf{C}$ is made of a variable number of characters k_i, *i.e.*, $\mathbf{c}_i=\{\mathbf{q}_j\}_{j=0}^{k_i}$. First, we render the characters in the

[1] Available at https://github.com/aimagelab/VATr

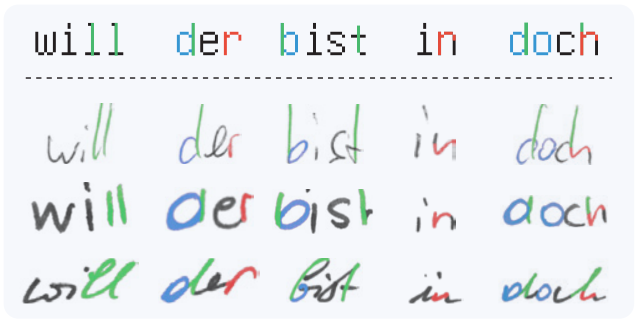

Figure 3. Comparison between the Unifont characters (top) and the same characters in different calligraphic styles (bottom). The geometric similarities between the characters are captured by the visual archetypes and thus can be exploited in generation.

GNU Unifont font, which, different from all other typeface fonts, contains all the Unicode characters. The rendering results in $16{\times}16$ binary images, which are then flattened and linearly projected to d-dimensional query embeddings, a strategy that is related to the direct use of image patches as input to Vision Transformer-like architectures [12].

In Figure 3, we show some exemplar visual archetypes (GNU Unifont characters) and the corresponding handwritten characters in different styles. It can be observed that the geometric similarities among the archetypes are reflected in styled characters. These similarities can be exploited for generating long-tail characters, *i.e.*, characters that are rarely seen during training. In fact, by being fed with independent tokens as content queries, the network is forced to simply memorize content-shape relations. Not being exposed to a sufficient number of such pairs does not allow the model to learn such relations and results in unsatisfactory generation capabilities of long-tail characters. Conversely, with our image-based input, the network can learn to exploit geometric attributes and similarities between highly-represented and long-tail characters for rendering those latter, and thus, can generate them more faithfully. It is also worth noting that this character representation makes our model more scalable than one-hot encoding-based solutions. In fact, the query embedding layer we use has $256{\times}d$ parameters and allows us to handle a charset containing up to $2^{(16\cdot16)}$ characters. For handling the same amount of characters represented as one-hot vectors, the query embedding layer would have $2^{(16\cdot16)}{\times}d$ parameters.

The output of the last Transformer decoder layer for the content string $\mathbf{c}_i$ is a tensor $\mathbf{F}_{c_i}{\in}\mathbb{R}^{k_i\times d}$. We add normal gaussian noise to $\mathbf{F}_{c_i}$, to enhance variability in the generated images, and project it into a $(k_i{\times}8192)$ matrix, which we then reshaped into a $512{\times}4{\times}4k_i$ tensor. This tensor is fed to a convolutional decoder consisting of four residual blocks and a $\tanh$ activation function that outputs the styled word images $\mathbf{Y}_{\mathrm{w}}^{\mathbf{C}}$.

3.3. Model Training

Formally, our complete VATr model is given by $\mathcal{G}_\theta = \mathcal{E}{\circ}\mathcal{D} : (\mathbf{X}_{\mathrm{w}}, \mathbf{C}){\rightarrow}\mathbf{Y}_{\mathrm{w}}^{\mathbf{C}}$. We train it alongside other modules used to calculate the overall loss for $\mathcal{G}_\theta$.

The first of those modules is a convolutional discriminator $\mathcal{D}_\eta$, which is trained to distinguish real images from images generated by $\mathcal{G}_\theta$, thus forcing the generator to produce realistic images. To optimize $\mathcal{G}_\theta$ and $\mathcal{D}_\eta$ we follow the adversarial paradigm with the hinge adversarial loss [34]

$$\begin{aligned}L_{adv} =&\mathbb{E}\left[\max(1-\mathcal{D}_\eta(\mathbf{X}_{\mathrm{w}}),0)\right]+\\ &\mathbb{E}\left[\max(1{+}\mathcal{D}_\eta(\mathcal{G}_\theta(\mathbf{X}_{\mathrm{w}},\mathbf{C})),0)\right].\end{aligned}$$

Additionally, we exploit an HTR model [42], $\mathcal{R}_\phi$, which is in charge of recognizing the text in the generated images, thus forcing the generator to reproduce the desired textual content other than rendering the style. The HTR model is trained with the real images $\mathbf{X}_{\mathrm{w}}$ and their ground truth transcription, while its loss value calculated on the generated images $\mathbf{Y}_{\mathrm{w}}^{\mathbf{C}}$ is propagated through the generator $\mathcal{G}_\theta$. The loss of the HTR model is obtained as

$$L_{HTR} = \mathbb{E}_{\mathbf{x}}\left[-\sum\log(p(t_{\mathbf{x}}|\mathcal{R}_\phi(\mathbf{x})))\right],$$

where $\mathbf{x}$ can be either a real or a generated image, and $t_{\mathbf{x}}$ is its transcription coming from the ground truth label in case $\mathbf{x}{\in}\mathbf{X}_{\mathrm{w}}$ or from $\mathbf{C}$ in case $\mathbf{x}{\in}\mathbf{Y}_{\mathrm{w}}^{\mathbf{C}}$.

Moreover, we employ a convolutional classifier $\mathcal{C}_\psi$ in charge of classifying the real and generated images based on their calligraphic style (*i.e.*, the style of writer w), thus forcing the generator $\mathcal{G}_\theta$ to render the correct style. As done for the $\mathcal{R}_\phi$ module, also this classifier is trained with the real images, and its loss value on the generated images is used to guide the generator. Formally, the loss for this module is

$$L_{class} = \mathbb{E}_{\mathbf{x}}\left[-\sum\log(p(\mathrm{w}|\mathcal{C}_\psi(\mathbf{x})))\right].$$

Also in this case, $\mathbf{x}{\in}\mathbf{X}_{\mathrm{w}}$ or $\mathbf{x}{\in}\mathbf{Y}_{\mathrm{w}}^{\mathbf{C}}$.

To further enforce the generation of images in the desired style, we use an additional regularization loss, namely the cycle consistency loss given by:

$$L_{cycle} = \mathbb{E}\left[\left\|\mathcal{E}(\mathbf{X}_{\mathrm{w}}) - \mathcal{E}(\mathbf{Y}_{\mathrm{w}}^{\mathbf{C}})\right\|_1\right].$$

The rationale is to force the generator to produce styled images for which the encoder $\mathcal{E}$ would extract the same style vectors. In other words, we want the style features of the input images to be preserved in the generated ones.

Overall, the complete objective function we use to train our model is given by combining the above loss terms equally weighed, *i.e.*,

$$L = L_{adv} + L_{HTR} + L_{class} + L_{cycle}.$$

For an analysis of the role of each loss term on the performance, we refer to the Supplementary material.

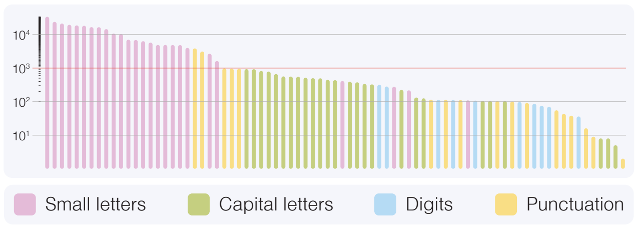

Figure 4. Distribution and classification of the characters in the training set of the IAM dataset (in logarithmic scale). We set a threshold on the frequency with which the characters appear equal to 1000 to identify the long-tail ones (indicated as a red line).

4. Experiments

In this section, in addition to analyzing the performance in the standard styled HTG scenarios, we aim to explore the capability of our approach and of SotA ones to generate characters that are long-tail-distributed in the dataset used for training. We believe that this is a relevant aspect to consider when evaluating the HTG performance, which has been so far neglected in the literature on this task. Additional analysis is reported in supplementary materials.

Implementation Details. All the experiments have been carried out on a single NVIDIA RTX 2080 Ti GPU. For pre-training the convolutional style encoder, we set the batch size to 32. We use the Adam optimizer with an initial learning rate equal to $2 \times 10^{-}5$ and apply exponential scheduling with a decay factor equal to $10^{-1/90000}$. We stop the training with an early stopping strategy with patience 30. Note that, due to the large amount of samples in the dataset, we are able to feed the convolutional backbone with almost always unseen samples before convergence. For this reason, we count an epoch every 1000 iterations. We employ the Adam optimizer also for training the complete HTG model on the real benchmark dataset considered in this work but fix the learning rate to $2 \cdot 10^{-4}$ and batch size equal to 8. In this case, the training is stopped after 7k epochs.

Evaluation Protocol. For validating our proposal and comparing it as fairly as possible against SotA HTG approaches, we consider the widely-used IAM dataset [38], with images rescaled to have 64 pixels in height and proportional width. Moreover, we adopt the same evaluation procedure as in [7, 29]. In particular, the dataset contains 62 857 English words from the Lancaster-Oslo/Bergen (LOB) corpus [26], handwritten by multiple users. For this work, we consider the words written by 340 of those users for training and those written by the remaining 160 for testing. The words in the dataset are composed starting from an alphabet of 79 characters distributed in the training set as shown in Figure 4. It can be noticed that these characters appear in the dataset in a long-tail distribution: small letters are the most represented (note that ‘e’, ‘t’, ‘a’, and ‘o’ are the most common letters, which reflects the frequency in the English vocabulary), while almost all the capital letters, digits, and punctuation are rare characters in the dataset. In our experiments, we consider long-tail characters those appearing less than 1000 times in the training set. We compare the proposed VATr model against the following learning-based methods for HTG. When available, we use the official implementation and weights released by the authors and evaluate all the models in the same setups. In particular, we consider the non-styled HTG ScrabbleGAN [14], the one-shot styled HTG model HiGAN [15], the approach proposed in [11] (which we refer to as TS-GAN), and the few-shot styled methods GANwriting [29] and HWT [7]. These two latter approaches use the same number of style examples as we use. Notably, HWT is the most closely related to our proposal for following a Transformer encoder-decoder paradigm for HTG. For performance evaluation, we consider the Fréchet Inception Distance (FID) [22] and the Geometry Score (GS) [30] for measuring the visual quality of the generated images. Moreover, to further evaluate the capability of the considered methods to generate rare characters faithfully, we calculate the Character Error Rate (CER) of an HTR network trained on the IAM dataset [42] when recognizing the text in the generated images. Note that, for all the scores, the lower the value, the better.

Table 1. Ablation analysis on the components of VATr.

Pre-training	Content input	FID (All)	FID (Long-Tail)
None	One-hot vectors	18.48	24.93
Synthetic	One-hot vectors	19.19	23.71
None	Visual archetypes	17.91	22.15
IAM	Visual archetypes	18.93	21.88
Synthetic	Visual archetypes	17.79	21.36

4.1. Ablation Analysis

First, we validate the benefits of using visual archetypes instead of one-hot vectors by replacing those latter as input to $\mathcal{D}$. The results of this analysis are reported in Table 1, both when generating styled words in the whole IAM test set and when generating words containing long-tail characters. It can be observed that our approach is superior to the baseline, especially for the generation of long-tail words. Moreover, we study the effect of the proposed synthetic pre-training strategy by comparing the performance obtained when training on real images only, and when pre-training to recognize the writers in the IAM dataset. Also these results are reported in Table 1 and show that the proposed synthetic pre-training brings more gain than training on real data, especially when used in combination with visual archetypes.

4.2. Few-Shot Styled HTG

In this section, we evaluate the capability of the proposed approach to generate realistic handwritten text images, re-

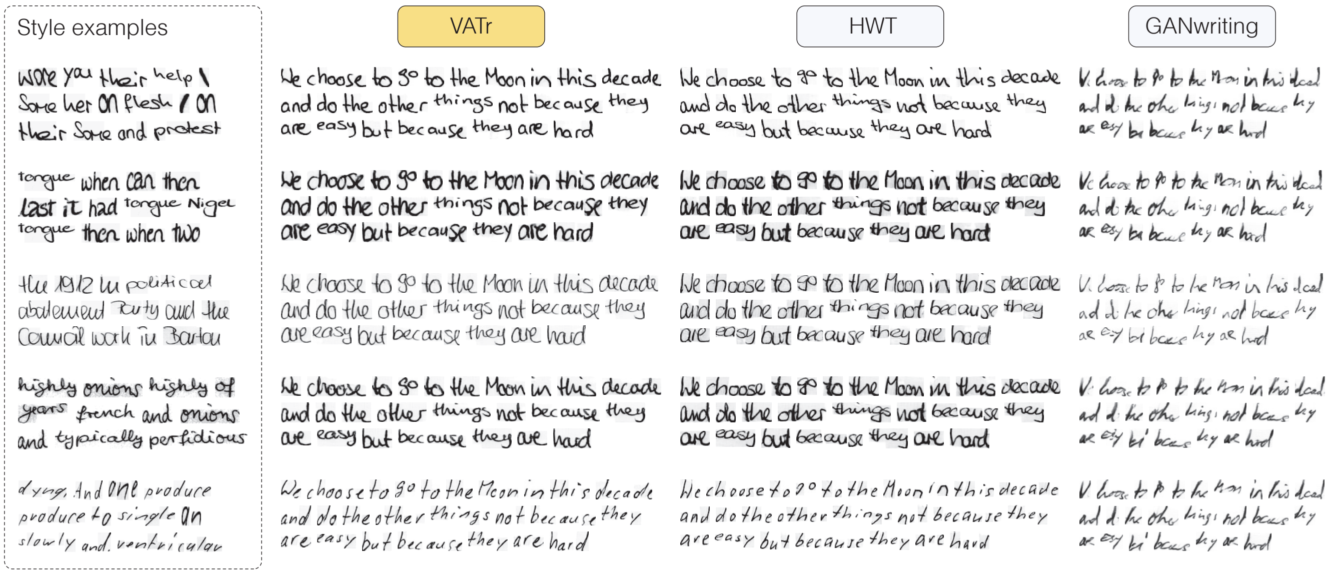

Figure 5. Qualitative comparison between our approach and the few-shot style HTG competitors in generating images with the desired textual content in the desired calligraphic style.

Table 2. Generated image quality evaluation on the IAM test set, regardless of the calligraphic style. Best results in bold.

	FID	GS
ScrabbleGAN [14]	20.72	2.56×10^{-2}
HiGAN [15]	24.90	3.19×10^{-2}
TS-GAN [11]	20.65	4.88×10^{-2}
HWT [7]	19.40	$\mathbf{1.01\times10^{-2}}$
VATr (Ours)	**17.79**	1.68×10^{-2}

gardless of the calligraphic style. To this end, we calculate the FID and GS on the IAM test set. The results of this study are reported in Table 2. It can be observed that our approach gives the best FID score and is second-best in terms of GS by a small margin, suggesting the realism of its generated images. As for the styled HTG evaluation, we follow the procedure proposed in [29]. In particular, we calculate the FID of the generated images in comparison to the real ones for each considered writer separately and then average the scores. We perform our analysis by distinguishing four increasingly challenging scenarios, namely: 1) the IV-S case, in which we generate in-vocabulary words in styles seen during training (*i.e.*, both style and content have been seen during training); 2) the IV-U case, in which the words to generate are in-vocabulary, but the style has never been observed; 3) the OOV-S case, in which the textual content consists of out-of-vocabulary words, but the style has been encountered during training; 4) the OOV-U case, in which both the desired style and words are unseen. The results of this analysis are reported in Table 3. It can be observed that our approach outperforms the competitors

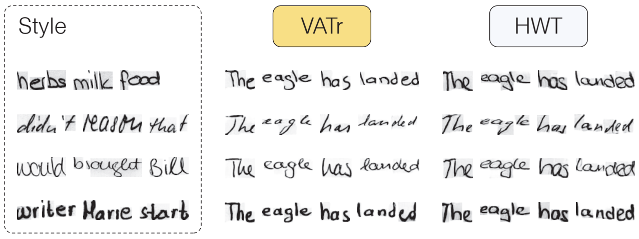

Figure 6. Exemplar generated images from style images with background artifacts.

in all four settings by a large margin. Some qualitative results are reported in Figure 5, which refer to the generation of a text with different unseen styles. It is also worth noting that, thanks to our large-scale synthetic pre-training strategy, VATr is able to focus on the shape attributes of the style to reproduce rather than on the background. This results in clearer generated images that reflect the handwriting of the reference ones rather than nuisances in the background (see Figure 6). Note that separating the handwriting from the background can negatively affect the FID score, which works on Inception v3 features, and even more the GS since it is calculated directly on the images. Nevertheless, this is an interesting capability for HTG models since it makes them suitable for generating styled text that can be easily superimposed to any desired background without artifacts.

4.3. Long-Tail Characters Generation

In this section, we focus on the capability of our visual archetypes-based approach to faithfully render rare characters. Note that the split of the IAM dataset used in training contains a total of 66 608 word images. Among those,

Table 3. Generated image quality evaluation by considering seen and unseen calligraphic style and in-vocabulary and out-of-vocabulary textual content. Best results in bold.

	IV-S	IV-U	OOV-S	OOV-U
TS-GAN [11]	118.56	128.75	127.11	136.67
GANwriting [29]	120.07	124.30	125.87	130.68
HWT [7]	106.97	108.84	109.45	114.10
VATr (Ours)	**88.20**	**91.11**	**98.57**	**102.22**

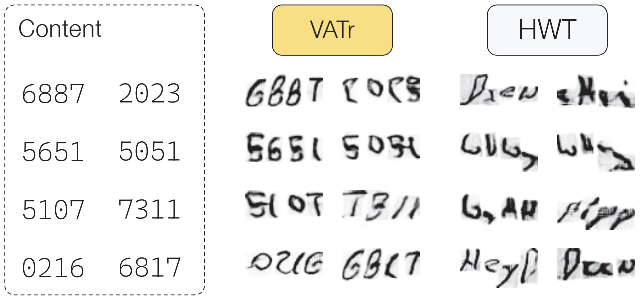

Figure 7. Comparison of the images of numbers generated by our approach and HWT.

only 13 064 contain at least one long-tail character. In Table 4, we present the performance on the generation of test strings that contain those characters, with a further evaluation on words made up of just digits. In particular, we evaluate both style and content preservation by measuring the FID and the CER. As can be observed from the values of the FID, SotA approaches relying on one-hot vector encodings of the content struggle to generate realistic images, especially when these contain only rare characters, as in the case of numbers. Our approach, instead, can handle such words more easily by exploiting shape similarity between the visual archetypes of the characters to render. This is confirmed by the qualitative results of the generation of numbers reported in Figure 7, where we compare our approach against HTW to better highlight the benefit of using the visual archetypes over one-hot encodings. It can be observed that the images generated by HWT do not resemble digits, while those of VATr better preserve the content.

Finally, it is worth mentioning that our approach comes with the machinery to generate, to some extent, also out-of-charset characters, *i.e.*, unseen symbols (*e.g.*, from other alphabets) in different handwriting styles. In particular, when those unseen symbols share visual details with the characters encountered during training (*e.g.*, as in the case of Greek letters 'δ' and 'ω' and Latin letters 's' and 'w'), our model can resort to the geometric patterns learned for the latter. Some qualitative examples of out-of-charset generation are given in Figure 8. Although the visual quality is inferior compared to that of the seen characters, our VATr strives to generate some out-of-charset symbols.

Table 4. Generated image quality evaluation by considering words containing at least one among the long-tail characters in the IAM dataset, and just numbers. The CER value calculated on real images is reported for reference. Best results in bold.

	All Long-Tail		Digits	
	FID	CER	FID	CER
Real Images	-	6.21	-	45.80
HiGAN [15]	26.08	**8.63**	129.61	101.53
HWT [7]	40.95	20.36	131.74	98.47
VATr (Ours)	**21.36**	11.85	**104.12**	**94.66**

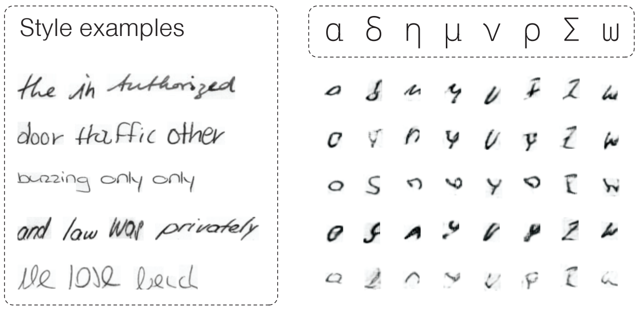

Figure 8. Generated images of some out-of-charset symbols (greek letters) in different styles.

5. Conclusion

In this work, we have proposed VATr, a few-shot styled HTG approach that is able to reproduce unseen calligraphic styles and generate characters rarely encountered in the training set. These capabilities are achieved by exploiting supervised pre-training on a large synthetic dataset of calligraphic fonts and by representing the textual content as a sequence of visual archetypes, *i.e.*, binary images of Unifont-rendered characters. Experimental results demonstrate that by pre-training, we are able to extract more representative style features that disregard the background and the ink texture. Moreover, by using the visual archetypes, we are able to exploit shape similarities among characters, which eases the generation of the rare ones.

Acknowledgement

This work was supported by the "AI for Digital Humanities" project (Pratica Sime n.2018.0390), funded by "Fondazione di Modena" and the PNRR project Italian Strengthening of Esfri RI Resilience (ITSERR) funded by the European Union – NextGenerationEU (CUP: B53C22001770006).

References

[1] Emre Aksan and Otmar Hilliges. STCN: Stochastic Temporal Convolutional Networks. In *ICLR*, 2018. 1, 2

[2] Emre Aksan, Fabrizio Pece, and Otmar Hilliges. DeepWriting: Making digital ink editable via deep generative modeling. In *CHI*. ACM, 2018. 1, 2

[3] Eloi Alonso, Bastien Moysset, and Ronaldo Messina. Adversarial Generation of Handwritten Text Images Conditioned on Sequences. In *ICDAR*. IEEE Computer Society, 2019. 1, 2, 3

[4] Samaneh Azadi, Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. Multi-content GAN for Few-Shot Font Style Transfer. In *CVPR*. IEEE, 2018. 2

[5] Ayan Kumar Bhunia, Abhirup Das, Ankan Kumar Bhunia, Perla Sai Raj Kishore, and Partha Pratim Roy. Handwriting Recognition in Low-Resource Scripts Using Adversarial Learning. In *CVPR*. IEEE, 2019. 1

[6] Ayan Kumar Bhunia, Shuvozit Ghose, Amandeep Kumar, Pinaki Nath Chowdhury, Aneeshan Sain, and Yi-Zhe Song. MetaHTR: Towards Writer-Adaptive Handwritten Text Recognition. In *CVPR*, 2021. 1

[7] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, and Mubarak Shah. Handwriting Transformers. In *ICCV*, 2021. 1, 2, 3, 4, 6, 7, 8

[8] Ayan Kumar Bhunia, Aneeshan Sain, Pinaki Nath Chowdhury, and Yi-Zhe Song. Text is Text, No Matter What: Unifying Text Recognition using Knowledge Distillation. In *ICCV*, 2021. 1

[9] Junbum Cha, Sanghyuk Chun, Gayoung Lee, Bado Lee, Seonghyeon Kim, and Hwalsuk Lee. Few-Shot Compositional Font Generation with Dual Memory. In *ECCV*, 2020. 2

[10] Bo Chang, Qiong Zhang, Shenyi Pan, and Lili Meng. Generating Handwritten Chinese Characters Using CycleGAN. In *WACV*. IEEE, 2018. 2

[11] Brian Davis, Chris Tensmeyer, Brian Price, Curtis Wigington, Bryan Morse, and Rajiv Jain. Text and Style Conditioned GAN for Generation of Offline Handwriting Lines. In *BMVC*, 2020. 1, 2, 3, 6, 7, 8

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 5

[13] J DUCHON. Splines minimizing rotation-invariant seminorms in sobolef spaces. *Lecture Notes in Mathematics*, pages 85–100, 1977. 4

[14] Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, and Roee Litman. ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation. In *CVPR*, 2020. 1, 2, 3, 6, 7

[15] Ji Gan and Weiqiang Wang. HiGAN: Handwriting Imitation Conditioned on Arbitrary-Length Texts and Disentangled Styles. In *AAAI*, 2021. 1, 2, 3, 6, 7, 8

[16] Ji Gan, Weiqiang Wang, Jiaxu Leng, and Xinbo Gao. HiGAN+: Handwriting Imitation GAN with Disentangled Representations. *ACM Trans. Graphics*, 42(1):1–17, 2022. 1, 2, 3

[17] Yue Gao, Yuan Guo, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. Artistic Glyph Image Synthesis via One-Stage Few-Shot Learning. *ACM Trans. Graphics*, 38(6), 2019. 2

[18] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative Adversarial Nets. In *NeurIPS*, 2014. 2

[19] Alex Graves. Generating Sequences with Recurrent Neural Networks. *arXiv preprint arXiv:1308.0850*, 2013. 1, 2

[20] TSF Haines, O Mac Aodha, and GJ Brostow. My Text in Your Handwriting. *ACM Trans. Graphics*, 35(3), 2016. 1, 2

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 4

[22] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. 2017. 6

[23] Malihe Javidi and Mahdi Jampour. A deep learning framework for text-independent writer identification. *Eng. Appl. Artif. Intell.*, 95:103912, 2020. 4

[24] Bo Ji and Tianyi Chen. Generative Adversarial Network for Handwritten Text. *arXiv preprint arXiv:1907.11845*, 2019. 1, 2

[25] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. SCFont: Structure-Guided Chinese Font Generation via Deep Stacked Networks. In *AAAI*, 2019. 2

[26] Stig Johansson, Geoffrey N Leech, and Helen Goodluck. *Manual of information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital computer*. Department of English, University of Oslo, 1978. 6

[27] Lei Kang, Pau Riba, Marcal Rusinol, Alicia Fornes, and Mauricio Villegas. Distilling content from style for handwritten word recognition. In *ICFHR*, 2020. 1

[28] Lei Kang, Pau Riba, Marcal Rusinol, Alicia Fornes, and Mauricio Villegas. Content and style aware generation of text-line images for handwriting recognition. *IEEE Trans. PAMI*, pages 1–1, 2021. 1

[29] Lei Kang, Pau Riba, Yaxing Wang, Marçal Rusiñol, Alicia Fornés, and Mauricio Villegas. GANwriting: Content-Conditioned Generation of Styled Handwritten Word Images. In *ECCV*, 2020. 1, 2, 3, 4, 6, 7, 8

[30] Valentin Khrulkov and Ivan Oseledets. Geometry Score: A Method For Comparing Generative Adversarial Networks. In *ICML*. PMLR, 2018. 6

[31] Atsunobu Kotani, Stefanie Tellex, and James Tompkin. Generating Handwriting via Decoupled Style Descriptors. In *ECCV*, 2020. 1, 2

[32] Praveen Krishnan, Rama Kovvuri, Guan Pang, Boris Vassilev, and Tal Hassner. TextStyleBrush: Transfer of Text Aesthetics from a Single Example. *arXiv e-prints*, pages arXiv–2106, 2021. 1, 2, 3

[33] Jeong-Sik Lee, Rock-Hyun Baek, and Hyun-Chul Choi. Arbitrary font generation by encoder learning of disentangled features. *Sensors*, 22(6):2374, 2022. 2

[34] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *arXiv preprint arXiv:1705.02894*, 2017. 5

[35] Zhouchen Lin and Liang Wan. Style-preserving english handwriting synthesis. *Pattern Recognition*, 40(7):2097–2109, 2007. 1, 2

[36] Canjie Luo, Yuanzhi Zhu, Lianwen Jin, Zhe Li, and Dezhi Peng. SLOGAN: Handwriting Style Synthesis for Arbitrary-Length and Out-of-Vocabulary Text. *IEEE Trans. Neural Netw. Learn. Syst.*, 2022. 1, 2, 3

[37] Siladittya Manna, Soumitri Chattopadhyay, Saumik Bhattacharya, and Umapada Pal. SWIS: Self-Supervised Representation Learning For Writer Independent Offline Signature Verification. *arXiv preprint arXiv:2202.13078*, 2022. 4

[38] U-V Marti and Horst Bunke. The IAM-database: an English sentence database for offline handwriting recognition. *IJDAR*, 5(1):39–46, 2002. 6

[39] Alexander Mattick, Martin Mayr, Mathias Seuret, Andreas Maier, and Vincent Christlein. SmartPatch: Improving Handwritten Word Imitation with Patch Discriminators. In *ICDAR*, 2021. 1, 3

[40] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[41] Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee, and Hyunjung Shim. Few-shot Font Generation with Localized Style Representations and Factorization. In *AAAI*, volume 35, 2021. 2

[42] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Trans. PAMI*, 39(11):2298–2304, 2016. 5, 6

[43] Achint Oommen Thomas, Amalia Rusu, and Venu Govindaraju. Synthetic Handwritten CAPTCHAs. *Pattern Recognition*, 42(12):3365–3373, 2009. 1, 2

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[45] Jue Wang, Chenyu Wu, Ying-Qing Xu, and Heung-Yeung Shum. Combining Shape and Physical Models for On-line Cursive Handwriting Synthesis. *IJDAR*, 7(4):219–227, 2005. 1, 2

[46] Yangchen Xie, Xinyuan Chen, Li Sun, and Yue Lu. DG-Font: Deformable Generative Networks for Unsupervised Font Generation. In *CVPR*, 2021. 2

[47] Shaozu Yuan, Ruixue Liu, Meng Chen, Baoyang Chen, Zhijie Qiu, and Xiaodong He. SE-GAN: Skeleton Enhanced GAN-based Model for Brush Handwriting Font Generation. *arXiv preprint arXiv:2204.10484*, 2022. 2

[48] Yaping Zhang, Shuai Nie, Wenju Liu, Xing Xu, Dongxiang Zhang, and Heng Tao Shen. Sequence-to-sequence domain adaptation network for robust text image recognition. In *CVPR*, 2019. 1

[49] Yecheng Zhu, Songxuan Lai, Zhe Li, and Lianwen Jin. Point-to-Set Similarity Based Deep Metric Learning for Offline Signature Verification. In *ICFHR*, 2020. 4