Quo Vadis Handwritten Text Generation for Handwritten Text Recognition?

Anonymous SynData4CV submission

Paper ID 49

Abstract

The digitization of historical manuscripts presents significant challenges for Handwritten Text Recognition (HTR) models, particularly when dealing with small, author-specific collections that diverge from the training data distributions. Handwritten Text Generation (HTG) techniques, which generate synthetic data tailored to specific handwriting styles, offer a promising solution to address these challenges. However, the effectiveness of various HTG models in enhancing HTR performance, especially in low-resource transcription settings, has not been thoroughly evaluated. In this work, we systematically compare three state-of-the-art styled HTG models (representing the generative adversarial, diffusion, and autoregressive paradigms for HTG) to assess their impact on HTR fine-tuning. We analyze how visual and linguistic characteristics of synthetic data influence fine-tuning outcomes and provide quantitative guidelines for selecting the most effective HTG model. The results of our analysis provide insights into the current capabilities of HTG methods and highlight key areas for further improvement in their application to low-resource HTR.

1. Introduction

The process of digitizing documents is becoming essential across both cultural and industrial sectors for their effective management, preservation, and enhancement. As a result, Document Analysis (DA) methods, particularly those focused on handwritten text, have been attracting great attention from the research community. Modern Handwritten Text Recognition (HTR) systems, which are typically trained on large publicly available datasets, perform well when applied to documents that resemble the data used for training. However, their performance significantly declines when tested on documents that differ substantially from the training data. A key challenge arises with historical manuscripts held in archives. These are usually small but valuable collections that often feature limited pages written by specific, historically important authors. These manuscripts display unique stylistic and linguistic features that pose difficulties for current HTR systems. To address this challenge, developing strategies that optimize HTR performance for such materials is critical for their efficient digitization. A common approach to tackle this issue involves pretraining HTR models on large-scale datasets, either real or synthetic, followed by fine-tuning them on a small set of real data from the target domain.

Some research work has already been devoted to exploring the use of synthetic datasets for pretraining HTR systems [1, 24, 30, 33, 55]. The effectiveness of these strategies largely depends on the extent to which the synthetic data mirrors real-world data [8, 41]. In response, Handwritten Text Generation (HTG) techniques, particularly styled HTG, have emerged as promising tools [3, 38, 40, 43, 52]. These models allow for the generation of training data tailored to specific domains by synthesizing images of text in a desired handwriting style by using just a few sample images as a reference. Styled HTG models typically include an encoder to extract the style features from the examples and a generator that combines these features with a desired text representation to produce text images with control over both style and content.

Recent years have witnessed the development of various HTG paradigms, and great improvements in HTG performance in terms of reference style fidelity, making them potentially very useful for generating tailored training data for HTR models. Nonetheless, a systematic evaluation of such usefulness in low-resource HTR scenarios is still missing in the literature. In light of this, in this paper, we explore a pretraining plus fine-tuning strategy based on automatically generated, author-specific synthetic datasets for comparing three state-of-the-art styled HTG networks, each one being the best of its category: a generative adversarial model [52], a diffusion model [38], and an autoregressive model [43]. Via extensive evaluation, we assess the effectiveness of these HTG approaches when generating pretraining data for HTR scenarios spanning multiple languages, various authors, and different historical periods. The results of our analysis give insights into the current capabilities of HTG models and suggest key areas for future research to improve their applicability in low-resource HTR scenarios.

2. Related Work

HTR is a well-established area of research due to its wide range of applications in both industrial and cultural sectors. Despite its progress, HTR remains a complex and challenging problem. The task can be tackled at different levels of granularity, ranging from individual characters, often used for idiomatic languages [11], to entire words [2, 51], lines [44, 48], paragraphs, and full pages [5, 7, 12, 36, 58]. Line-level recognition is particularly common for non-idiomatic languages, where it can be applied as a standalone method or integrated into a broader page-level system [7, 36, 59]. The majority of modern HTR systems employ learning-based approaches, relying on Multi-Dimensional Long Short-Term Memory networks (MD-LSTMs) [6, 39, 44, 48, 54] for feature extraction. These methods typically use the Connectionist Temporal Classification (CTC) decoding strategy to produce text transcriptions [5, 25]. Recently, alternative models based on fully convolutional networks [15, 59] and Transformer encoder-decoder architectures [30, 33, 56] have also been proposed for HTR tasks [53]. To improve transcription quality, explicit language models or lexicons can be employed. However, the effectiveness of these models depends on the consistency and regularity of the transcribed language, especially regarding the presence of rare words, proper nouns, or errors. This makes language models less reliable, particularly when working with historical manuscripts where the language can be highly variable or archaic. In this work, we focus on line-level HTR and on historical data, thus, we do not rely on any lexicon or explicit language model.

A significant challenge in HTR is the scarcity of training data, particularly for single-author documents or ancient manuscripts with unique characteristics. One solution to address this limitation is data augmentation, which can involve general image manipulations such as color changes and geometric transformations [44, 54, 57] or more targeted modifications specifically designed to reflect the characteristics of the target data [10]. Another widely adopted method is pretraining the HTR model on large, diverse datasets followed by fine-tuning on a smaller set of target-specific data [24, 27, 50]. This approach has been demonstrated to outperform basic data augmentation when applied to historical manuscripts [1]. The pretraining data can consist of real handwritten text (*e.g.*, publicly available benchmark datasets) or synthetic data, often generated by rendering text in calligraphic fonts [8, 30, 47]. For single-author scenarios, [41] highlights the importance of considering both the overall appearance (such as the type of paper, writing instrument, and average character width) and the language (including the time period and the topic) when selecting real datasets or generating synthetic ones for pretraining. Additionally, they demonstrate that an HTR model trained on text images with a wide range of handwriting styles tends to be more adaptable and robust compared to one trained on a single handwriting style. However, when synthetic data closely mimic the actual handwriting in the real data, achieving satisfactory performance becomes feasible.

Recent research has explored using HTG models to generate synthetic data for training HTR models, aiming to enhance their performance on real-world datasets [29, 37, 41, 49]. HTG entails generating realistic handwritten text images. In its styled variant, which is the primary focus of this work, the goal is to produce writer-specific handwritten text by using just a few example images to capture and replicate the writer unique style [3, 20, 28]. Three main paradigms have been proposed to tackle this task. The predominant technique is the use of generative-adversarial models [3, 21, 22, 28, 32, 40, 52]. Some research has explored HTG using Diffusion Models [17, 34, 37, 60], which led to impressive performance. Finally, a recent study introduced an autoregressive approach to HTG [43]. In this study, we consider HTG models representative of each of these paradigms and evaluate them in the context of low-resource HTR applications.

3. Method

In this work, we aim to evaluate the performance of state-of-the-art HTG models when used to generate synthetic pertaining datasets for HTR on small collections of documents with distinctive characteristics (*e.g.*, unique handwriting styles, language variations, paper supports). Moreover, we investigate some strategies to maximize the benefit of using such models. To these ends, we devise a pipeline that entails pretraining the HTR model on a large synthetic dataset, obtained from HTG models to imitate the style of a real target dataset, followed by filtering strategies based either on the readability or style fidelity of the generated samples. Finally, the pipeline entails fine-tuning on a limited number of real samples from the target collection. To create the synthetic datasets, we require exemplar style images, which can be easily extracted from digitized manuscripts within the target collection. Additionally, the textual content to be rendered in the desired handwriting style must be specified. We consider the scenario where the author and the language of the manuscript is known. In this case, if there exist transcribed texts written by the author of interest, we can use the HTG model to generate synthetic samples of these texts. Otherwise, if only the manuscript's language is known (or if no existing texts by the same author are available), the HTG model can generate text in the same language as the target collection. In both cases, the HTG model outputs handwritten lines images of varying lengths. Note that the quality of some of the generated lines can be non-ideal, limiting their usefulness for HTR training. To limit this risk, a possible solution is to filter out synthetic images that do not meet certain quality criteria. In

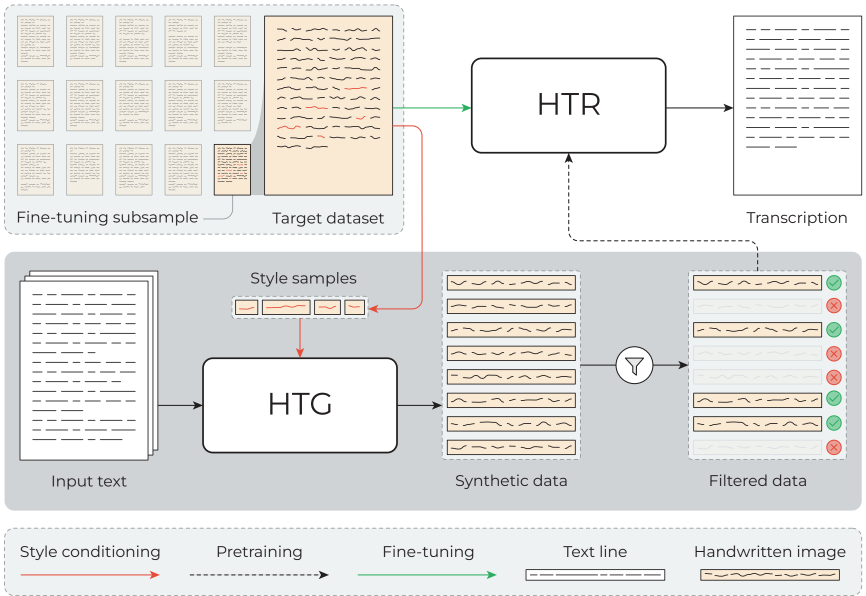

Figure 1. Overview of our pipeline for synthetic data generation from collection-specific handwritten lines. The generation process renders line images from a given text conditioned by a few style samples from the target dataset. Then, the synthetic dataset is filtered based on readability or fidelity criteria and is used to pretrain the recognition model before possible fine-tuning on the real data.

this work, we consider two alternatives: the lines readibility, expressed in terms of Character Error Rate (CER) of an off-the-shelf, language-agnostic HTR model, and their style fidelity *w.r.t.* the reference, expressed in terms of Handwriting Distance (HWD) [42].

In the following sections, we describe the HTR model used for transcription (dubbed **DefCRNN** [9]) and the HTG models considered to generate synthetic pretraining data. These are the generative-adversarial Transformer VATr++ [52], the diffusion model-based **DiffPen** [38], and the autoregressive generative Transformer **Emuru** [43]. Finally, we give the details of the proposed **HWD**-based and **CER**-based filtering strategies. An overview of our complete pipeline is illustrated in Figure 1.

3.1. HTR Approach

The combination of convolutional and recurrent neural networks has long been a standard approach for HTR, and it is widely used due to its effectiveness and efficiency. In this study, we employ a model based on one-dimensional LSTM networks, which offer comparable or even superior performance compared to MD-LSTMs [44]. Our model is based on a variant of the CRNN method [48], referred to as DefCRNN [9]. The convolutional part of the architecture consists of seven convolutional blocks. The first six blocks follow the VGG-11 structure, with modifications to the final two max-pooling layers to incorporate rectangular pooling, which helps preserve the aspect ratio of text line images. The seventh convolutional block utilizes a 2×2 kernel. The variant we exploit contains Deformable Convolutions [16] as proposed in [8, 9, 14], which enhance model performance by allowing for more flexible feature extraction. The output of the final convolutional layer is a feature map of size $2 \times W \times 512$, where W is determined by the width of the input image. This feature map is then collapsed along the channel dimension, resulting in a sequence of W feature vectors, each with a size of 1024. These vectors are passed to the recurrent module, which consists of two Bidirectional LSTM layers with 512 hidden units each, with a dropout layer (probability 0.5) in between. The output of the recurrent module is a sequence of probability distributions over character classes for each feature vector. As is typical in HTR, the model is trained using the CTC loss function, which includes a special *blank* character. Notably, we do not use any external language model to ensuring that the model is adaptable across different languages.

3.2. HTG Approaches

Styled HTG models efficiently create large volumes of synthetic text images in a specified handwriting style starting from a few real images from the target dataset. An overview of the considered HTG approaches, each representative of a distinct paradigm, is reported below (we refer the interested reader to the respective papers for more details). In this paper, we use them off-the-shelf, and none of them has been trained on the target datasets considered.

VATr++. VATr++ [52] employs a generator-discriminator framework [23, 35], complemented by an auxiliary HTR network for readability and a writer classification module to ensure stylistic fidelity. The model has been designed to address the generation of rare or out-of-charset characters. This is achieved by encoding target text as a sequence of Visual Archetypes (VAs) [40], which allow the model to exploit geometric similarities between glyphs, and by adopting specific training and data preparation stratiges. The architecture is a hybrid Convolutional-Transformer encoder-decoder. The encoder uses a synthetically pre-trained CNN to process the reference style samples, while the Transformer encoder aggregates them into a style vector using self-attention. The Transformer decoder aligns this representation with a sequence of VAs representing the desired text content through cross-attention, and a convolutional decoder synthesizes the final handwritten image. VATr++ accepts as input 15 word images, from which it extract the style, and generates word or text line images.

DiffPen. DiffPen [38] is a latent diffusion model that synthesizes images conditioned on a text prompt and style features in a few-shot setting. Similar to standard conditional latent diffusion models [45], the method utilizes a U-Net-based architecture [46] as the backbone denoising network and a pre-trained Variational Autoencoder (VAE) [31] to encode and decode images from pixel to latent space and vice-versa. Two auxiliary pre-trained encoders are used for the text and style conditions. To create the content embedding, an off-the-shelf pre-trained text encoder [13] that operates on character sequences. As the style encoder, the work proposes a CNN feature extractor that combines classification and metric learning to construct a continuous embedding space that supports diverse output and enables fine-grained control (*e.g.*, style interpolation and mixing). To create the style condition, DiffPen extracts features from 5 style examples of the same writer, and generates word images with the desired content. Although DiffPen is designed for word-level generation, the authors have proposed patching together subparts of text or words to obtain longer text or complete lines, which we also adopt in our work.

Emuru. Emuru [43] is a continuous-token autoregressive model for handwritten text generation, capable of producing text images of any length while preserving style fidelity and readability. It enhances generalization to novel styles and minimizes background artifacts. The architecture consists of a VAE [31] and an autoregressive Transformer Encoder-Decoder. The VAE maps style reference images into a continuous latent space, encoding only the writing style while removing background noise. The Transformer takes as input the style embeddings, the text present in the reference image, and the desired text, iteratively generating an image that preserves the target style. Both components of Emuru are trained on a large synthetic dataset. This ensures that the VAE learns to reconstruct text without background artifacts while providing a style representation that generalizes well to new handwriting styles and typefaces. The model generates text images in an autoregressive loop, where each iteration outputs visual embeddings that are then decoded by the VAE into a final styled image. This iterative process allows the model to determine its own stopping point, eliminating constraints on maximum text length. Emuru takes as input a text line image with its associated text content and is designed to generate entire text lines.

3.3. Filtering Approaches

We argue that not all the generated samples are equally useful and of high quality for HTR pretraining. To explore this aspect, we propose to analyze them based on two different criteria (*i.e.*, readability and style fidelity) and discard those that do not meet a predefined quality threshold. In the following, we describe our considered filtering strategies.

Readability. To evaluate how the readability of the generated samples affects the training of the HTR model, we measure the CER for each synthetic image by using a pretrained TrOCR network [33] and then filter out those for which the CER is above a certain threshold. We define four filtering thresholds (*i.e.*, CER<0.15, CER<0.30, CER<0.45, and CER<0.60), which progressively include samples based on transcription quality. Images that satisfy stricter thresholds are considered more readable, as they exhibit fewer transcription errors. Conversely, images that exceed higher CER values are discarded during the filtering process and will not be used to pretrain the DefCRNN.

Fidelity. To assess the stylistic fidelity of the generated samples, we quantify how closely each synthetic image matches the handwriting style of authentic manuscripts. We compute the HWD [42] between each generated image and a representative style embedding extracted from the real samples. This embedding is obtained by averaging the features of genuine handwriting samples, serving as a reference for style similarity. Since each generation model produces a distinct distribution of HWD values, we define filtering thresholds using the 25th, 50th, and 75th percentiles of its respective HWD distribution. Samples with HWD below a given percentile are considered more stylistically faithful, while those above the higher percentiles are progressively filtered out. This percentile-based approach ensures that style fidelity is evaluated fairly according to each model's intrinsic variability in generated handwriting.

4. Experiments

In this section, we present our experimental study. First, we provide additional information regarding how we train the DefCRNN HTR model. Next, we describe the small, single-author target datasets and we detail how the synthetic datasets for pretraining were constructed. Finally, we explain the evaluation protocol and discuss our results.

4.1. HTR Training Details

For training the DefCRNN model, all input images are rescaled to a height of 64 pixels while maintaining their original aspect ratio, followed by intensity normalization to the range $[-1, 1]$. During pretraining, we apply a series of augmentations to enhance robustness. Brightness is adjusted using a randomly sampled factor from $[0.5, 5]$, contrast from $[0.1, 10]$, saturation from $[0, 5]$, and hue from $[-0.1, 0.1]$. Additionally, Gaussian blur with a kernel size of 5 is applied, with a standard deviation randomly chosen from $[0.1, 2]$. To introduce geometric variability, we randomly apply one of the following transformations: a slight rotation between $-1°$ and $1°$, an affine transformation with rotation in the same range and shear between $-50°$ and $30°$, or a random tomography. Pretraining is conducted with a batch size of 16, which is reduced to 8 for fine-tuning and training from scratch. The model is optimized using Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and a learning rate of 10^{-4} across all experiments. A scheduler reduces the learning rate by 10% if the CER on the validation set plateaus. Early stopping is applied with a patience of 20 epochs, using CER as the criterion. When fine-tuning, optimal CER values are typically reached within the first few epochs.

4.2. Datasets

For our analysis, we consider three low-resources, line-level datasets as target collections. All of them are obtained from historical, single-author manuscripts. When generating synthetic data for pretraining, we consider the characteristics of each target dataset separately. The details of the datasets used in this work are given below.

Leopardi. The Leopardi dataset [8] comprises a collection of early 19th-century Italian manuscripts authored by Giacomo Leopardi, a prominent Romantic-era philologist, writer, and poet. It consists of 1303 training lines, 596 validation lines, and 587 test lines. All samples are RGB scans of ink-written texts on historical paper.

Washington. The George Washington dataset [19] includes 20 handwritten English letters from 1755, authored by George Washington, the first U.S. President, and a collaborator. It is structured into 526 training lines, 65 validation lines, and 65 test lines. The dataset consists of binarized images of these historical documents.

Saint Gall. The Saint Gall dataset [18] originates from a late 9th-century Latin manuscript written by a single scribe. It spans 60 pages and is divided into 468 training lines, 235 validation lines, and 707 test lines. All images are binarized scans of the original manuscript pages.

Synthetic Data. Following [41], we employ HTG models to generate synthetic samples tailored to each target dataset. This process involves conditioning the generation on a subset of the original dataset's samples. Specifically, VATr++ and DiffPen use crops from 15 and 5 randomly selected line images, respectively, while Emuru operates with a single line reference. The textual content for the synthetic data is chosen to align with each dataset: excerpts from Giacomo Leopardi's prose for the Leopardi dataset, passages from George Washington's diaries for Washington, and a medieval Latin Bible for Saint Gall. This selection ensures linguistic consistency between the synthetic and real data.

4.3. Evaluation Protocol

To analyze the impact of pretraining and fine-tuning in scenarios where only a small portion of the target dataset is annotated, we fine-tune models on progressively smaller subsets of the training data. Specifically, we consider fractions of 100%, 50%, 25%, 10%, 5%, 2.5%, and 1.25% of the available training lines. For comparison, we also train models from scratch using the same subsets. Moreover, we assess direct transfer by applying pretrained models to the target datasets without fine-tuning.

To compare the considered HTG models, we first consider their generation performance, expressed in terms of multiple commonly applied scores. Specifically, these are: Fréchet Inception Distance (**FID**) [26], Kernel Inception Distance (**KID**) [4], **HWD** [42], the binarized version of FID and KID (dubbed **BFID** and **BKID**, respectively), obtained by computing the scores on binarized images, and the Absolute Difference in the CER (**ΔCER**) of the off-the-shelf TrOCR-Base [33] model on the reference and generated images. Moreover, since the main goal of this work is to compare HTG approaches in terms of their effectiveness in providing synthetic data for HTR, we consider the recognition performance of the considered DefCRNN trained on such data. We report the performance in terms of CER, which is standard for text recognition.

4.4. Results

Generation Performance. First, we evaluate the considered HTG models in terms of their generation capabilities. Recall that none of them have been trained on the target datasets, making this a zero-shot evaluation of their ability to produce handwriting samples that align with the target styles. The quantitative performance comparison is reported in Table 1. From the FID, KID, and HWD scores, it is evident that Emuru consistently outperforms the other models across nearly all metrics, leveraging its zero-shot capabilities to generate more style-faithful samples. This observation is confirmed by the qualitative examples in Figure 2. Additionally, the scatter plots in Figure 3, which report the distribution of the generated datasets in terms of TrOCR CER and HWD relative to the respected target dataset, show that the images generated by Emuru exhibit the lowest readability according to the TrOCR model. However, from the ΔCER values in Table 1, we can argue that

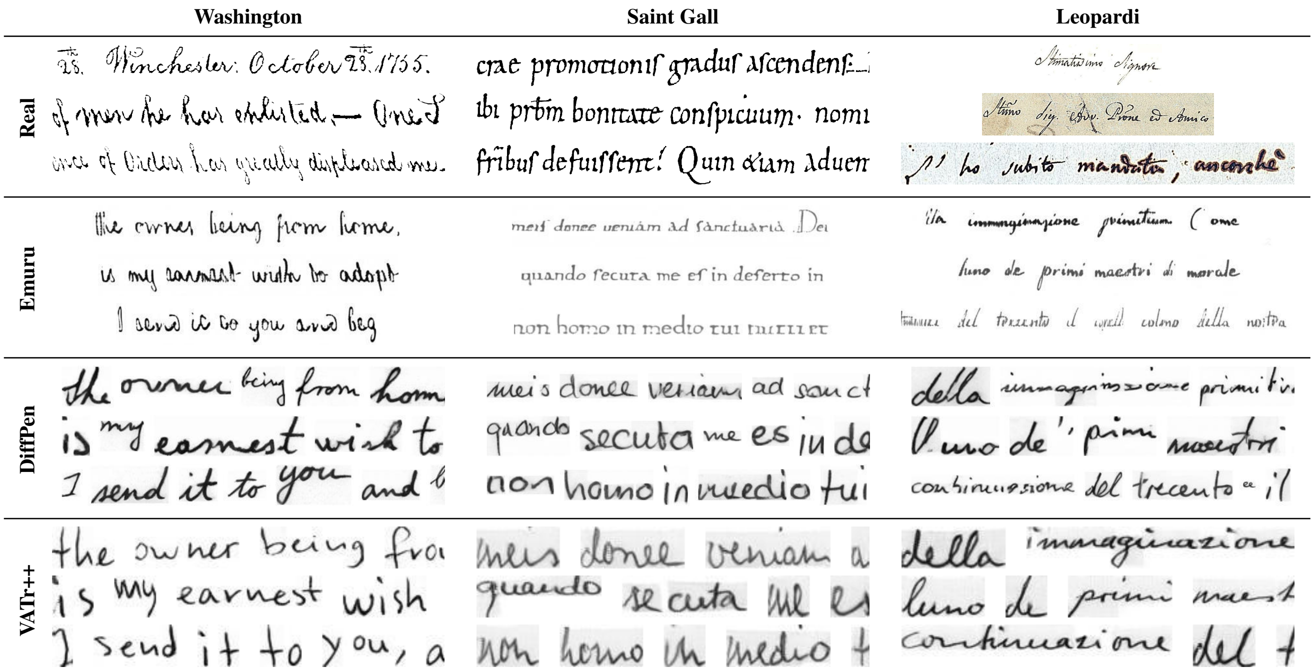

Figure 2. Qualitative comparison of the considered HTG models when generating samples from the considered target datasets. Note that none of the considered models has been trained on the target datasets.

Dataset	Model	HWD↓	FID↓	BFID↓	KID↓	BKID↓	ΔCER ↓
Leopardi	Emuru	**2.05**	**208.4**	**57.7**	0.246	**0.051**	**0.4**
	DiffPen	3.15	244.9	127.4	0.257	0.110	1.6
	VATr++	3.02	217.7	90.8	**0.230**	0.084	21.8
Washington	Emuru	**1.63**	**108.3**	**24.6**	**0.104**	**0.016**	**9.7**
	DiffPen	2.56	171.6	91.9	0.163	0.085	9.8
	VATr++	3.24	217.1	105.4	0.228	0.092	20.4
Saint Gall	Emuru	**1.53**	**243.0**	**32.2**	0.320	**0.014**	**7.0**
	DiffPen	3.44	250.0	40.9	0.310	0.024	11.1
	VATr++	3.82	251.6	73.0	**0.306**	0.060	10.2

Table 1. Generation scores computed on the three target datasets generated with the considered HTG models.

this reduced readability is a consequence of Emuru's ability to faithfully replicate the target handwriting style.

Direct Transfer Recognition Performance. The recognition performance achievable by the DefCRNN model when pretrained on the generated data and then directly applied to the real, target datasets are reported in Tables 2 to 4 (in the first column relative to the CER) and depicted in Figure 4. As can be observed, Emuru's generated samples allow for achieving the best performance in this zero-shot HTR scenario, consistently outperforming the other models. A possible explanation for this advantage can be found in the scatter plots in Figure 3, where the generated samples from Emuru exhibit greater variability, forming more dispersed clusters. This diversity may contribute to the model's robustness when directly applied to unseen handwriting styles. Furthermore, it is worth noting that the handwriting in the Washington and Saint Gall datasets is quite regular (see Figure 2). Since Emuru is trained on a large corpus of synthetic typewritten and calligraphic fonts, it can effectively approximate the structured styles characteristic of these datasets. This alignment is reflected in the CER scores achieved in the direct transfer setting: 18.1 for Saint Gall and 15.3 for Washington.

Fine-tuning Recognition Performance. From the results in Tables 2 to 4 and Fig. 4, it can also be observed the effect of fine-tuning DefCRNN on a varying number of real data from the target dataset after being pretrained on the samples synthesized by the considered HTG models. It can be observed that Emuru is the most effective in providing pretraining data for the HTR model when only a very limited amount of real data is available for fine-tuning. In particular, when fewer than 130 real images are used (*e.g.*, 10% of Leopardi, which corresponds to 130 images, or 25% of Saint Gall, which includes 125 images), the style similarity between the pretraining and target dataset plays a crucial role. In other words, the closer the synthetic samples are to the target handwriting style, the greater the benefit for HTR performance in low-data regimes. However, as the number of real training samples increases beyond this threshold, the influence of style similarity of the pretraining dataset diminishes and its diversity becomes the dominant factor in improving HTR performance. This explains why, when more than 130 real images are available, DefCRNN pretrained on the more stylistically varied DiffPen-generated datasets has better performance than when pretrained on Emuru-generated data. In other words, when more fine-tuning data are available DiffPen's style variability provides

SynData4CV
#49

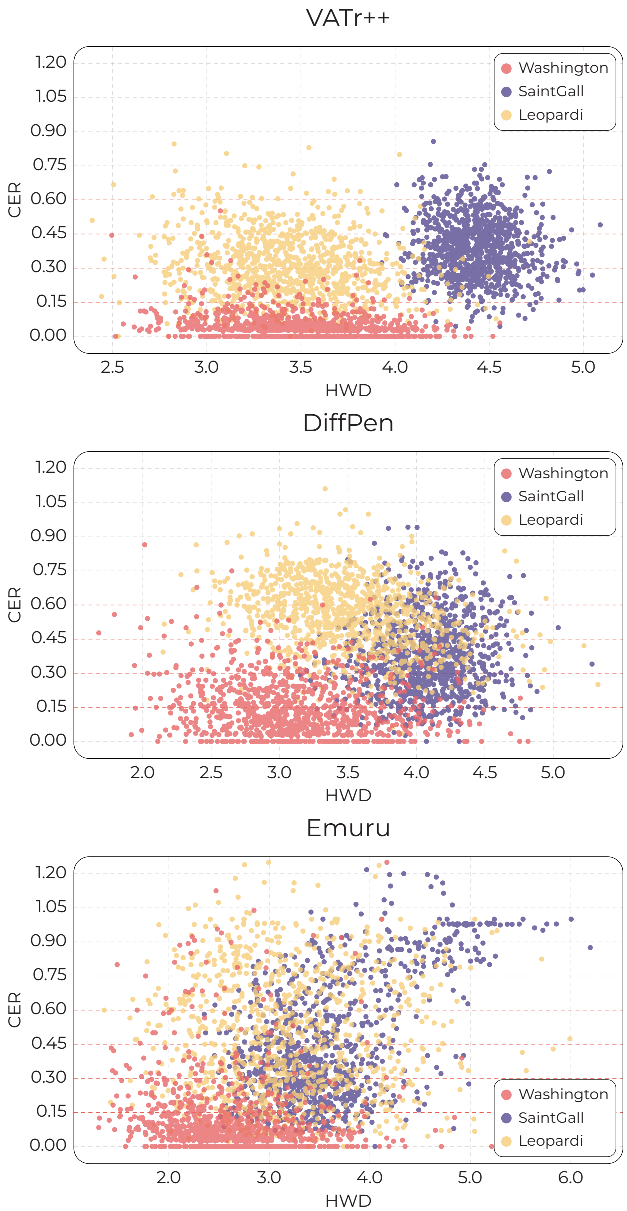

Figure 3. Distribution of 1000 random samples from each synthetic dataset generated by the HTG models, in terms of CER and HWD *w.r.t.* the real samples. The horizontal lines indicate the CER thresholds used for filtering. We omit the separators for HWD-based filtering for clarity, since they depend on the percentiles.

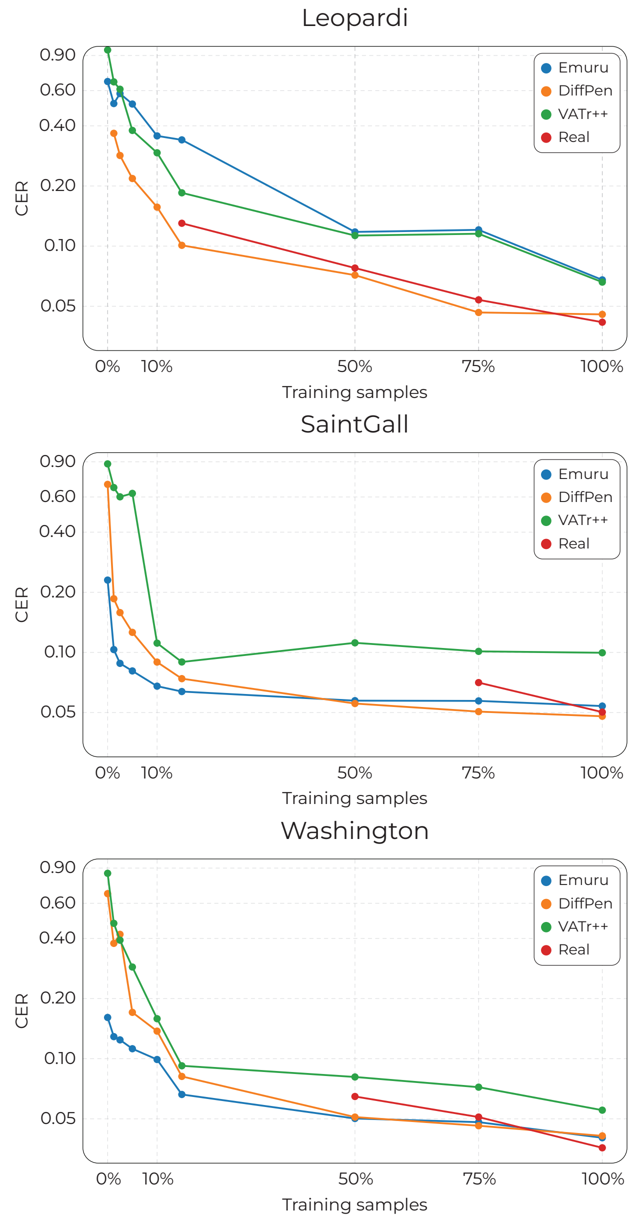

Figure 4. CER scores obtained by DefCRNN when fine-tuned with different portions of the three target datasets, after having been pretrained on all the synthetic data generated with the considered HTG models. We report the CER obtained when training only on real data for comparison.

a stronger generalization capability to the HTR model.

Effect of Filtering. Finally, we consider the effect of filtering with the two proposed CER-based and HWD-based strategies (as observed from Tables 2 to 4). Notably, no clear trend emerges between HTR performance and filtering based on handwriting style similarity (HWD). This suggests that strictly enforcing style consistency between the synthetic and real datasets does not necessarily lead to better recognition performance. Conversely, filtering based on CER appears to have a more direct impact. The best-performing configurations are typically those where the filtering threshold is set to $\text{CER}_{0.30}$ or when no filtering is applied at all. This suggests that the amount of training samples is a more important factor than their quality. A weak filter removes the worst examples while keeping enough variety in the data, while a strict filter may remove too many samples and hurt performance. Moreover, a too-strict CER-based filtering could remove too many samples, preventing the HTR model from converging (as in the case of pretraining on DiffPen-generated data for Leopardi with a filter with

Data	Filter	#samples	CER 0%	1.25%	2.5%	5%	10%	25%	50%	75%	100%
Real	-	1.3K	-	-	-	-	-	13.0	7.8	5.4	4.2
Emuru	$HWD_{25\%}$	21.5K	62.7	48.6	47.4	26.9	20.4	12.9	8.4	8.0	4.1
Emuru	$HWD_{50\%}$	43.0K	**49.0**	34.9	29.7	23.8	15.2	**10.0**	7.7	5.5	4.6
Emuru	$HWD_{75\%}$	64.5K	49.3	45.5	52.0	53.5	30.9	19.7	10.9	11.5	6.2
Emuru	$CER_{0.15}$	7.4K	67.7	34.3	28.9	24.6	19.6	16.1	7.9	6.3	5.2
Emuru	$CER_{0.30}$	23.3K	62.4	**21.7**	**22.9**	**16.5**	**14.3**	10.6	7.7	5.9	4.0
Emuru	$CER_{0.45}$	39.8K	65.5	24.9	23.9	21.3	14.7	11.5	7.8	6.0	5.5
Emuru	$CER_{0.60}$	53.3K	66.2	29.6	29.9	18.8	15.2	11.5	7.9	6.2	5.1
Emuru	-	87.8K	66.7	51.7	58.0	51.4	35.6	34.0	11.8	12.1	6.8
DiffPen	$HWD_{25\%}$	22.0K	80.5	54.4	42.2	30.1	23.5	15.8	10.0	**4.5**	**3.9**
DiffPen	$HWD_{50\%}$	43.9K	76.0	41.9	31.5	22.1	17.3	10.7	7.5	5.2	4.5
DiffPen	$HWD_{75\%}$	65.9K	83.4	56.9	49.8	33.8	24.8	14.9	9.8	8.3	6.3
DiffPen	$CER_{0.15}$	0.2K	-	-	-	-	98.9	12.4	7.4	5.6	4.5
DiffPen	$CER_{0.30}$	3.5K	-	44.4	36.7	27.4	24.7	15.7	8.4	5.4	4.5
DiffPen	$CER_{0.45}$	19.7K	74.8	46.7	42.0	25.8	23.2	14.4	9.0	4.7	4.0
DiffPen	$CER_{0.60}$	51.8K	80.2	53.3	43.0	26.7	22.9	14.4	9.0	7.3	5.6
DiffPen	-	87.8K	-	36.7	28.4	21.8	15.7	10.1	7.2	4.7	4.6
VATr++	$HWD_{25\%}$	22.0K	96.9	59.4	49.0	30.7	24.1	15.9	10.2	5.1	4.6
VATr++	$HWD_{50\%}$	43.9K	94.7	63.6	59.6	33.8	24.9	16.1	9.5	9.6	4.3
VATr++	$HWD_{75\%}$	65.9K	89.2	65.0	69.0	38.3	27.3	18.4	11.9	11.2	4.2
VATr++	$CER_{0.15}$	9.9K	94.6	49.2	40.1	31.2	25.8	16.9	**6.8**	5.0	4.4
VATr++	$CER_{0.30}$	43.8K	94.3	64.5	57.7	40.4	26.7	17.0	11.5	10.3	4.1
VATr++	$CER_{0.45}$	74.2K	92.0	66.3	62.9	37.8	28.0	19.0	11.6	11.2	4.4
VATr++	$CER_{0.60}$	85.4K	95.7	67.0	60.9	37.4	30.2	17.6	11.2	11.6	6.5
VATr++	-	87.8K	95.9	66.4	61.0	37.9	29.3	18.5	11.3	11.5	6.6

Table 2. CER scores (multiplied by 100) obtained by pretraining the DefCRNN on the generated data and then fine-tuned on different portions of the Leopardi dataset. Bold indicates the best overall score for each fine-tuning setting, underline the best score within each setting (HTG model and filtering strategy).

Data	Filter	#samples	CER 0%	1.25%	2.5%	5%	10%	25%	50%	75%	100%
Real	-	0.5K	-	-	-	-	-	-	-	7.1	5.0
Emuru	$HWD_{25\%}$	17.3K	37.1	12.2	10.9	9.4	**6.7**	6.7	6.9	6.3	6.0
Emuru	$HWD_{50\%}$	34.6K	29.0	13.7	11.6	10.8	7.5	6.8	7.0	6.9	6.6
Emuru	$HWD_{75\%}$	51.9K	26.6	29.0	28.4	11.0	7.2	7.5	6.7	5.8	5.5
Emuru	$CER_{0.15}$	5.7K	20.0	11.9	10.0	9.0	7.6	6.9	6.5	6.4	6.2
Emuru	$CER_{0.30}$	22.8K	21.6	12.6	11.5	9.6	7.8	6.9	6.3	6.1	5.9
Emuru	$CER_{0.45}$	37.6K	21.5	13.2	10.9	9.7	7.5	6.6	6.6	5.9	5.7
Emuru	$CER_{0.60}$	46.3K	**18.1**	26.2	25.4	10.1	7.2	7.2	6.7	6.4	6.4
Emuru	-	70.5K	23.0	**10.3**	**8.8**	**8.1**	6.8	**6.4**	5.7	5.7	5.4
DiffPen	$HWD_{25\%}$	17.6K	59.1	27.8	21.2	16.4	9.6	7.4	7.5	6.0	6.4
DiffPen	$HWD_{50\%}$	35.2K	62.4	27.7	21.2	15.6	10.0	9.0	7.8	6.2	5.8
DiffPen	$HWD_{75\%}$	52.9K	64.5	35.6	27.0	20.0	8.0	7.7	7.9	6.4	6.5
DiffPen	$CER_{0.15}$	4.9K	54.7	56.2	56.2	21.6	8.6	8.5	10.5	8.9	5.3
DiffPen	$CER_{0.30}$	25.3K	59.2	31.4	24.2	17.4	8.0	7.7	7.7	7.0	6.3
DiffPen	$CER_{0.45}$	49.3K	65.3	32.8	24.0	18.7	9.7	8.4	8.4	7.8	8.0
DiffPen	$CER_{0.60}$	63.5K	64.2	37.9	29.5	21.1	8.2	9.9	8.2	7.2	6.9
DiffPen	-	70.5K	69.4	18.6	15.8	12.6	8.9	7.4	**5.5**	**5.1**	**4.8**
VATr++	$HWD_{25\%}$	17.6K	87.7	36.8	26.0	19.8	9.4	9.1	7.6	7.0	6.0
VATr++	$HWD_{50\%}$	35.2K	94.2	37.6	26.2	20.0	10.3	9.5	9.3	8.1	9.1
VATr++	$HWD_{75\%}$	52.9K	88.7	56.6	43.1	29.6	9.6	10.4	9.3	8.2	7.6
VATr++	$CER_{0.15}$	1.9K	-	-	-	-	-	9.9	9.5	7.4	8.0
VATr++	$CER_{0.30}$	18.0K	93.4	31.8	22.5	19.2	9.7	8.2	9.0	7.3	6.0
VATr++	$CER_{0.45}$	47.7K	92.1	64.6	60.8	45.0	9.4	9.3	8.8	7.5	6.6
VATr++	$CER_{0.60}$	65.6K	91.9	71.2	75.2	50.4	9.4	9.2	9.6	9.0	8.3
VATr++	-	70.5K	87.9	66.9	60.1	62.5	11.1	9.0	11.2	10.1	10.0

Table 3. CER scores (multiplied by 100) obtained by pretraining the DefCRNN on the generated data and then fine-tuned on different portions of the Saint Gall dataset. Bold indicates the best overall score for each fine-tuning setting, underline the best score within each setting (HTG model and filtering strategy).

$CER_{0.15}$). For these reasons, not filtering can yield better results, as it maximizes variability in the pretraining data, improving the HTR model's generalizability.

Data	Filter	#samples	CER 0%	1.25%	2.5%	5%	10%	25%	50%	75%	100%
Real	-	0.5K	-	-	-	-	-	-	6.5	5.1	3.6
Emuru	$HWD_{25\%}$	5.7K	18.7	17.6	15.3	13.7	10.8	7.5	5.7	5.5	5.1
Emuru	$HWD_{50\%}$	11.4K	18.8	16.7	14.4	13.5	10.1	6.8	6.2	5.5	5.5
Emuru	$HWD_{75\%}$	17.1K	15.9	13.5	13.5	13.3	**8.7**	5.8	9.2	8.3	7.0
Emuru	$CER_{0.15}$	16.7K	15.8	13.4	12.5	12.2	10.3	6.2	5.1	5.6	4.3
Emuru	$CER_{0.30}$	20.4K	**15.3**	15.2	**11.9**	**10.3**	9.5	6.3	**4.4**	**4.2**	**3.5**
Emuru	$CER_{0.45}$	21.5K	16.6	13.9	13.4	11.1	9.8	6.7	6.1	5.2	4.7
Emuru	$CER_{0.60}$	21.9K	17.1	14.5	13.2	12.1	9.7	**5.0**	6.3	5.7	4.1
Emuru	-	23.1K	16.1	**12.9**	12.4	11.2	9.9	6.6	5.0	4.8	4.0
DiffPen	$HWD_{25\%}$	5.8K	-	32.7	24.4	18.8	51.1	12.9	7.1	6.2	4.2
DiffPen	$HWD_{50\%}$	11.6K	72.9	31.8	23.3	20.5	13.9	8.3	7.5	6.2	3.9
DiffPen	$HWD_{75\%}$	17.3K	72.6	29.6	24.7	19.9	11.6	7.2	8.1	6.3	5.1
DiffPen	$CER_{0.15}$	13.8K	68.7	28.1	21.7	18.3	13.4	9.8	6.9	5.0	4.8
DiffPen	$CER_{0.30}$	20.3K	68.3	30.5	23.8	19.5	14.4	9.0	6.7	6.1	5.2
DiffPen	$CER_{0.45}$	22.4K	78.0	29.1	23.4	19.3	13.8	7.2	7.4	6.4	4.7
DiffPen	$CER_{0.60}$	22.9K	72.2	32.0	25.4	20.7	12.4	7.0	8.9	6.8	6.3
DiffPen	-	23.1K	67.0	37.8	41.9	17.1	13.7	8.2	5.1	4.6	4.1
VATr++	$HWD_{25\%}$	5.8K	-	41.9	31.3	23.2	51.8	9.9	8.4	6.2	3.9
VATr++	$HWD_{50\%}$	11.6K	86.8	46.1	29.7	22.5	16.0	9.0	6.5	5.3	4.1
VATr++	$HWD_{75\%}$	17.3K	81.7	45.4	35.9	27.2	14.8	10.2	8.4	6.3	5.4
VATr++	$CER_{0.15}$	22.0K	78.3	64.3	24.7	20.3	16.4	9.0	6.2	5.7	3.9
VATr++	$CER_{0.30}$	23.0K	83.0	62.4	28.2	17.8	13.7	10.3	6.3	5.0	5.0
VATr++	$CER_{0.45}$	23.1K	81.0	45.4	34.5	27.4	14.7	9.1	8.2	6.4	6.3
VATr++	$CER_{0.60}$	23.1K	81.9	44.5	34.5	28.9	14.7	10.4	8.2	6.2	5.4
VATr++	-	23.1K	84.8	47.6	39.2	28.8	15.9	9.2	8.1	7.2	5.5

Table 4. CER scores (multiplied by 100) obtained by pretraining the DefCRNN on the generated data and then fine-tuned on different portions of the Washington dataset. Bold indicates the best overall score for each fine-tuning setting, underline the best score within each setting (HTG model and filtering strategy).

5. Conclusion

We have explored low-resource HTR on historical manuscripts and we have proposed a pipeline leveraging synthetic data generated by state-of-the-art HTG models for pretraining an HTR model, which is then fine-tuned on a few real samples. Our findings give the following insights:

- **Style Fidelity.** Among the evaluated HTG models, Emuru consistently generates the most style-faithful handwriting samples, leading to superior zero-shot HTR performance, indicating the importance of style fidelity in zero-shot scenarios.
- **Style Variability.** When only a small number of real images (fewer than 130) are available for fine-tuning, pretraining on data that closely matches the target handwriting style leads to better recognition results. However, if more real samples are available, diversity in the pretraining set becomes more important than style similarity. In these cases, training on the more varied DiffPen-generated datasets leads to better generalization and improved HTR performance. This shows the effect of style diversity depending on the amount of fine-tuning data.
- **Filtering.** There is no clear benefit from filtering based on HWD. In contrast, filtering based on CER with a too strict threshold can lead to removing too many useful training samples or even hinder the HTR model convergence.

By shedding light on the existing HTG capabilities, this study aims to help the design of novel HTG models for boosting HTR in low-resources scenarios.

References

[1] José Carlos Aradillas, Juan José Murillo-Fuentes, and Pablo M Olmos. Boosting offline handwritten text recognition in historical documents with few labeled lines. *IEEE Access*, 2021. 1, 2

[2] 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*, 2019. 2

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

[4] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *ICLR*, 2018. 5

[5] Théodore Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In *NeurIPS*, 2016. 2

[6] Théodore Bluche and Ronaldo Messina. Gated convolutional recurrent neural networks for multilingual handwriting recognition. In *ICDAR*, 2017. 2

[7] Théodore Bluche, Jérôome Louradour, and Ronaldo Messina. Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention. In *ICDAR*, 2017. 2

[8] Silvia Cascianelli, Marcella Cornia, Lorenzo Baraldi, Maria Ludovica Piazzi, Rosiana Schiuma, and Rita Cucchiara. Learning to Read L'Infinito: Handwritten Text Recognition with Synthetic Training Data. In *CAIP*, 2021. 1, 2, 3, 5

[9] Silvia Cascianelli, Marcella Cornia, Lorenzo Baraldi, Rita Cucchiara, et al. Boosting Modern and Historical Handwritten Text Recognition with Deformable Convolutions. *IJDAR*, pages 1–15, 2022. 3

[10] Edgard Chammas, Chafic Mokbel, and Laurence Likforman-Sulem. Handwriting recognition of historical documents with few labeled data. 2018. 2

[11] Nicole Dalia Cilia, Claudio De Stefano, Francesco Fontanella, and Alessandra Scotto di Freca. A ranking-based feature selection approach for handwritten character recognition. *Pattern Recognit. Lett.*, pages 77–86, 2019. 2

[12] Tarin Clanuwat, Alex Lamb, and Asanobu Kitamoto. KuroNet: Pre-Modern Japanese Kuzushiji Character Recognition with Deep Learning. In *ICDAR*, 2019. 2

[13] Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. CANINE: Pre-training an Efficient Tokenization-Free Encoder for Language Representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022. 4

[14] Iulian Cojocaru, Silvia Cascianelli, Lorenzo Baraldi, Massimiliano Corsini, and Rita Cucchiara. Watch Your Strokes: Improving Handwritten Text Recognition with Deformable Convolutions. In *ICPR*, 2020. 3

[15] Denis Coquenet, Clément Chatelain, and Thierry Paquet. Recurrence-free unconstrained handwritten text recognition using gated fully convolutional network. In *ICFHR*, 2020. 2

[16] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *CVPR*, 2017. 3

[17] Haisong Ding, Bozhi Luan, Dongnan Gui, Kai Chen, and Qiang Huo. Improving handwritten OCR with training samples generated by glyph conditional denoising diffusion probabilistic model. In *ICDAR*, 2023. 2

[18] Andreas Fischer, Volkmar Frinken, Alicia Fornés, and Horst Bunke. Transcription alignment of Latin manuscripts using hidden Markov models. In *HIP*, 2011. 5

[19] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognit. Lett.*, pages 934–942, 2012. 5

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

[21] Ji Gan and Weiqiang Wang. HiGAN: Handwriting Imitation Conditioned on Arbitrary-Length Texts and Disentangled Styles. In *AAAI*, 2021. 2

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

[23] 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. 3

[24] Adeline Granet, Emmanuel Morin, Harold Mouchère, Solen Quiniou, and Christian Viard-Gaudin. Transfer learning for handwriting recognition on historical documents. In *ICPRAM*, 2018. 1, 2

[25] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *NeurIPS*, 2009. 2

[26] 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. In *NeurIPS*, 2017. 5

[27] José Carlos Aradillas Jaramillo, Juan José Murillo-Fuentes, and Pablo M Olmos. Boosting handwriting text recognition in small databases with transfer learning. In *ICFHR*, 2018. 2

[28] 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. 2

[29] 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. 2

[30] Lei Kang, Pau Riba, Marçal Rusiñol, Alicia Fornés, and Mauricio Villegas. Pay attention to what you read: non-recurrent handwritten text-line recognition. *Pattern Recognit.*, 129:108766, 2022. 1, 2

[31] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2013. 4

[32] Praveen Krishnan, Rama Kovvuri, Guan Pang, Boris Vassilev, and Tal Hassner. TextStyleBrush: Transfer of Text Aesthetics from a Single Example. *IEEE Trans. PAMI*, 2023. 2

[33] Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. TrOCR: Transformer-based optical character recognition with pre-trained models. *AAAI*, 2023. 1, 2, 4, 5

[34] Troy Luhman and Eric Luhman. Diffusion Models for Handwriting Generation. *arXiv preprint arXiv:2011.06704*, 2020. 2

[35] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*, 2014. 3

[36] Bastien Moysset, Christopher Kermorvant, and Christian Wolf. Full-page text recognition: Learning where to start and when to stop. In *ICDAR*, 2017. 2

[37] Konstantina Nikolaidou, George Retsinas, Vincent Christlein, Mathias Seuret, Giorgos Sfikas, Elisa Barney Smith, Hamam Mokayed, and Marcus Liwicki. WordStylist: Styled Verbatim Handwritten Text Generation with Latent Diffusion Models. In *ICDAR*, 2023. 2

[38] Konstantina Nikolaidou, George Retsinas, Giorgos Sfikas, and Marcus Liwicki. DiffusionPen: Towards Controlling the Style of Handwritten Text Generation. *ECCV*, 2024. 1, 3, 4

[39] Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. Dropout improves recurrent neural networks for handwriting recognition. In *ICFHR*, 2014. 2

[40] Vittorio Pippi, Silvia Cascianelli, and Rita Cucchiara. Handwritten Text Generation from Visual Archetypes. In *CVPR*, 2023. 1, 2, 4

[41] Vittorio Pippi, Silvia Cascianelli, Christopher Kermorvant, and Rita Cucchiara. How to choose pretrained handwriting recognition models for single writer fine-tuning. In *ICDAR*, 2023. 1, 2, 5

[42] Vittorio Pippi, Fabio Quattrini, Silvia Cascianelli, and Rita Cucchiara. HWD: A Novel Evaluation Score for Styled Handwritten Text Generation. In *BMVC*, 2023. 3, 4, 5

[43] Vittorio Pippi, Fabio Quattrini, Silvia Cascianelli, Alessio Tonioni, and Rita Cucchiara. Zero-Shot Styled Text Image Generation, but Make It Autoregressive. In *CVPR*, 2025. 1, 2, 3, 4

[44] Joan Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *ICDAR*, 2017. 2, 3

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 4

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 4

[47] Xi Shen and Ronaldo Messina. A method of synthesizing handwritten chinese images for data augmentation. In *ICFHR*, 2016. 2

[48] 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*, pages 2298–2304, 2016. 2, 3

[49] Mohamed Ali Souibgui, Ali Furkan Biten, Sounak Dey, Alicia Fornes, Yousri Kessentini, Lluis Gomez, Dimosthenis Karatzas, and Josep Llados. One-shot Compositional Data Generation for Low Resource Handwritten Text Recognition. In *WACV*, 2022. 2

[50] Yann Soullard, Wassim Swaileh, Pierrick Tranouez, Thierry Paquet, and Clement Chatelain. Improving text recognition using optical and language model writer adaptation. In *ICDAR*, 2019. 2

[51] Felipe Petroski Such, Dheeraj Peri, Frank Brockler, Hutkowski Paul, and Raymond Ptucha. Fully convolutional networks for handwriting recognition. In *ICFHR*, 2018. 2

[52] Bram Vanherle, Vittorio Pippi, Silvia Cascianelli, Nick Michiels, Frank Van Reeth, and Rita Cucchiara. VATr++: Choose Your Words Wisely for Handwritten Text Generation. *IEEE Trans. PAMI*, 2024. 1, 2, 3

[53] 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

[54] Paul Voigtlaender, Patrick Doetsch, and Hermann Ney. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *ICFHR*, 2016. 2

[55] Christoph Wick, Jochen Zöllner, and Tobias Grüning. Rescoring Sequence-to-Sequence Models for Text Line Recognition with CTC-Prefixes. *arXiv preprint arXiv:2110.05909*, 2021. 1

[56] Christoph Wick, Jochen Zöllner, and Tobias Grüning. Transformer for Handwritten Text Recognition Using Bidirectional Post-decoding. In *ICDAR*, 2021. 2

[57] Curtis Wigington, Seth Stewart, Brian Davis, Bill Barrett, Brian Price, and Scott Cohen. Data augmentation for recognition of handwritten words and lines using a CNN-LSTM network. In *ICDAR*, 2017. 2

[58] Curtis Wigington, Chris Tensmeyer, Brian Davis, William Barrett, Brian Price, and Scott Cohen. Start, Follow, Read: End-to-End Full-Page Handwriting Recognition. In *ECCV*, 2018. 2

[59] Mohamed Yousef and Tom E Bishop. OrigamiNet: Weakly-Supervised, Segmentation-Free, One-Step, Full Page Text Recognition by learning to unfold. In *CVPR*, 2020. 2

[60] Yuanzhi Zhu, Zhaohai Li, Tianwei Wang, Mengchao He, and Cong Yao. Conditional Text Image Generation with Diffusion Models. In *CVPR*, 2023. 2