Handwritten Text Generation with Character-Specific Encoding for Style Imitation

Jan Zdenek(✉) and Hideki Nakayama

The University of Tokyo, Tokyo, Japan
jan@nlab.ci.i.u-tokyo.ac.jp, nakayama@ci.i.u-tokyo.ac.jp

Abstract. In this paper, we propose a novel method for handwritten text generation that uses a style encoder based on a vision transformer network that encodes handwriting style from reference images and allows the generator to imitate it. The encoder learns to disentangle style information from the content by learning to recognize who wrote the text, and the self-attention mechanism in the encoder allows us to produce character-specific encodings by using characters in the target sequence as queries. Our method can also generate handwritten text images in random styles by sampling random latent vectors instead of encoding style vectors from reference images.

We demonstrate through experiments that our proposed method outperforms existing methods for handwritten text generation in terms of the quality of generated images and their fidelity with respect to the distribution of real images. Furthermore, it achieves significantly better performance at imitating handwriting styles defined by reference images. Our model generalizes well to unseen data and can generate handwritten images of words and character sequences as well as imitate handwriting styles not included in the training data.

Keywords: Handwritten text generation · Handwriting imitation · Handwritten text recognition

1 Introduction

Significant progress has been achieved in image generation in recent years, particularly thanks to the emergence of new approaches such as generative adversarial networks (GANs) [12], variational auto-encoders (VAEs) [26], and more recently also diffusion models [17]. Generative models can now produce very accurate and detailed images that are difficult to discern from real ones [24,32,39]. The original GAN architecture could only generate images from randomly sampled latent vectors, which did not provide a way to control what was generated. However, further research proposed various methods to manipulate the generation process by conditioning on class labels [30], text embeddings [40]+, segmentation maps [36], reference images [32], etc. This extended the possibilities of image generation beyond a simple generation of random objects.

G. A. Fink et al. (Eds.): ICDAR 2023, LNCS 14188, pp. 313–329, 2023.
https://doi.org/10.1007/978-3-031-41679-8_18

One of the domains that have adopted recent methods for image generation is handwritten text. Multiple fields and applications can benefit from the ability to generate images of handwritten text automatically. In handwritten text recognition (HTR), being able to generate a large number of diverse handwriting samples for training can improve the accuracy and robustness of recognition models. Handwritten text generation can also be used to create handwritten text assets for games and virtual reality applications, and it can also help in product design when handwritten text is required or desirable.

In recent years, several works have proposed methods for generation of handwritten text images. Some of them can only generate handwritten text with random styles of writing [2,9,38], while some can imitate existing styles from reference images [4,10,22]. The earlier proposed approaches suffer from poor visual quality due to limitations such as being able to generate only fixed-size images [2]. However, recent methods can generate images that are difficult for humans to distinguish from real ones [11,38]. JokerGAN [38] achieves outstanding performance, but it can only generate handwritten text in random styles. In this work, we leverage the performance of JokerGAN by using it as a base for our model and modify it to enable style imitation by generating handwritten text with guidance by reference images. To encode style features from reference images, we train a style encoder together with the rest of the model by making it learn to recognize who wrote the text, inspired by [10]. As vision transformers (ViT) [8] have shown to excel at various vision-based tasks [7,34,39], we employ an encoder based on a ViT network. We further significantly improve the performance of our model by using the target character sequence as a query for self-attention in the transformer, which allows us to generate specific encodings for different characters.

As our proposed model extends the original JokerGAN to support imitation of style from reference images, we call it JokerGAN++.

In summary, the main contributions of our work are as follows:

- We propose a new method to imitate handwriting styles by using a ViT-based encoder that uses target character sequences as queries to produce character-specific style encodings. The experiments show that it can imitate handwriting styles more accurately than existing methods and also generate more authentic images with respect to the distribution of real images.
- We conduct experiments on data augmentation for HTR with generative models and show that HTR models trained on data augmented by images generated by our model outperform models trained on data augmented by images generated by existing methods.
- We demonstrate that our method for generation of handwritten text can also be used to erase handwritten text from images.

2 Related Work

Generation of realistic images of handwritten text is a challenging task. Conventional methods in the past required a lot of manual manipulation of the source images, which involved clipping of individual characters. Those were then combined by various rendering techniques to produce new images of handwritten text [15,35].

The past decade has witnessed the success of deep neural networks, leading to their utilization in various domains and applications, including generation of handwritten text. The first attempts at applying deep neural networks to handwritten text generation focused on online handwritten trajectories. Recurrent neural networks were used to learn and generate the temporal data [13], and further improvement was achieved by adding a discriminator network and employing adversarial training [20]. Manipulating the style of generated images was accomplished by disentangling the style and content of handwritten text [1]. Deep neural networks require a lot of data for successful training, but collecting a large amount of online handwritten data is a demanding task that involves special equipment to record handwriting trajectories. However, collecting offline handwritten data is much easier as it only requires obtaining images of handwritten text. It has been demonstrated that generative adversarial networks (GAN) [12] and variational auto-encoders [26] are capable of generating images of handwritten digits, and it is possible to control which digits are generated with conditional GANs [30]. Recent progress in raster image generation showed the potential of generative models to create very realistic and detailed images [6,23,31], and it helped drive the research on offline handwritten text generation.

There are two approaches to offline handwritten text generation: 1) generation of handwritten text in random styles given by randomly sampled latent vectors, and 2) generation of handwritten text in a specific style defined by a reference image. Most existing methods support generation either only in random styles [2,9,38] or only in specific styles given by reference images [4,21,22]. Recently, methods that support both types of generation have also emerged [10,11].

Application of GANs to offline handwritten text generation was first proposed by Alonso et al. [2] and their model consisted of BigGAN [6] for image generation, an LSTM [18] to encode the target word into a fixed-length vector used as a conditional input for the generator, and a text recognition network to ensure that the generator produces legible images of the target word. Due to its design, the model was restricted to generating text images of a fixed size regardless of the length of the generated word, which causes distortions. ScrabbleGAN [9] resolved this problem by replacing the LSTM-based encoder with a bank of base filters for each letter in the alphabet, which allows generation of images in variable sizes. The generator in ScrabbleGAN uses k base filters corresponding to k letters in the target word. The size of the model grows almost linearly with respect to the size of the character set, which makes it unfeasible for languages with large character sets, such as Chinese or Japanese. To alleviate this issue, JokerGAN [38] replaces the bank of base filters with a single base filter for all characters and uses multi-class conditional batch normalization to generate different characters depending on the conditional input, which is obtained by embedding individual characters in the target word. JokerGAN also introduced a new type of conditional input that makes the generator aware of the position of all characters in the target word with respect to the baseline and mean line, which reduces distortions in generated images.

Kang et al. [22] proposed a method that generates handwritten words from style features extracted from reference images in a few-shot setting and textual features of a predefined text length. In a later work [21], they extended their previous method to support generation of long character sequences and text lines. The recent surge of transformers in computer vision tasks inspired [4] to use transformers to generate styled handwritten text. A transformer is used to model the target word and style features extracted from a reference image by a CNN, and the output is passed to a CNN that upsamples and generates a text image in the desired resolution. The method works in a few-shot setting and requires multiple words as a reference to extract and reproduce the style accurately. HiGAN [10] extended ScrabbleGAN to support generation in specific styles by adding a CNN to encode style from reference images, and a later work [11] improved the performance by modifying the network architecture and using contextual loss in training.

Besides methods that generate handwritten text from latent or encoding vectors, there are methods that synthesize handwritten text in a given style from skeleton images of handwriting [14] and from machine-printed text [28] using an image-to-image translation approach.

Table 1. Comparison of functionality of existing methods and our proposed method. *Latent* means that the method can generate images from random latent vectors. *Reference* means that generation process can be guided by a reference image.

Method	Latent	Reference	Few-shot	One-shot
Alonso et al. [2]	✓			
ScrabbleGAN [9]	✓			
GANWriting [22]		✓	✓	
HWT [4]		✓	✓	
JokerGAN [38]	✓			
HiGAN [10]	✓	✓		✓
HiGAN+ [11]	✓	✓		✓
Ours	✓	✓		✓

Table 1 compares the functionality of our method and existing methods for handwritten text generation. While most methods can generate text either only in random styles using random latent vectors or only in specific styles, guided by reference images, our method supports both types and can generate handwritten text in both random styles and styles defined by a reference image. Our method also requires only a single handwritten word as a reference to imitate the style, unlike some of the other methods that work in a few-shot setting and require multiple words as a reference.

3 Proposed Method

Our proposed method, JokerGAN++, is based on JokerGAN [38] model for handwritten text generation. JokerGAN boasts a high quality of generated handwritten text images, but it cannot imitate specific handwriting styles. Therefore, we modify the original architecture and add a style encoder network to control the style of generated handwriting by reference images. We also revise the discriminator to exploit semantic information about text in images to learn whether the image is real or not. In Sect. 3.1, we describe individual parts of our proposed model, and in Sects. 3.2 and 3.3, we introduce our key contributions to the architecture in detail.

3.1 Model Architecture

Generator. Our generator $\mathcal{G}$ is based on [38]. It supports generation of character sequences of arbitrary length by concatenating k identical base filters that are passed into $\mathcal{G}$, corresponding to the length k of the target character sequence. Generation of different characters is achieved by multi-class conditional batch normalization (MCCBN) that is conditioned on the target sequence of characters. MCCBN is also used to generate handwriting in different styles by concatenating the character sequence embeddings with style codes. In [38], style codes are obtained randomly by sampling from a normal distribution. In our work, we use a style encoder to imitate existing handwriting styles given by reference images; however, random styles can also be generated by using randomly sampled style codes. We also empirically find that using identical encoding vectors in each block of the generator yields better results in our task than using hierarchical input [6] for conditional batch normalization, and injecting noise for additional diversity also hurts the performance.

Discriminator. The discriminator $\mathcal{D}$ learns to predict whether an image is real or generated by $\mathcal{G}$. The discriminator used in [9,38] learns to solve this binary classification problem from real and generated image samples without any explicit information about the character sequences in the images. We add a new component into the discriminator that implicitly learns to recognize individual characters in the image from output feature maps of discriminator layers and uses this semantic information to modulate the feature maps. More details follow in Sect. 3.3.

Style Encoder. We use a style encoder network $\mathcal{E}$ to produce encodings of handwriting styles that can be used as conditional input for $\mathcal{G}$. Detailed description of $\mathcal{E}$ can be found in Sect. 3.2.

Text Recognizer. The objective of text recognizer $\mathcal{R}$ is to promote generation of legible text that matches the target character sequence. Following [9,38], we use a simple network that predicts local patterns without global context to focus on legibility of individual characters. The text recognizer is trained only on real labeled images, and text recognition loss calculated on generated images is used to provide guidance and optimize $\mathcal{G}$.

Figure 1 shows a diagram of the whole model. To simplify the diagram and reduce visual clutter, we do not include all loss functions.

The training process is similar to [10,38], so we simplify the explanation. We alternate between two optimization passes. In the first pass, we optimize $\mathcal{D}$ by adversarial loss, $\mathcal{R}$ by CTC loss for text recognition, and the writer identification module by cross-entropy loss. In the second pass, we optimize $\mathcal{G}$ and the style encoding module in $\mathcal{E}$ by 1) adversarial loss, 2) CTC text recognition loss on generated samples to ensure that $\mathcal{G}$ generates legible images, 3) cross-entropy loss for writer identification using the writer identification module on generated samples, 4) L1 reconstruction loss of style codes calculated between random latent vectors $\mathbf{z}$ and style encoding vectors obtained by $\mathcal{E}$ from images generated by $\mathcal{G}$ and conditioned on $\mathbf{z}$, and 5) KL-Divergence loss to regularize the style latent space so that it matches normal distribution.

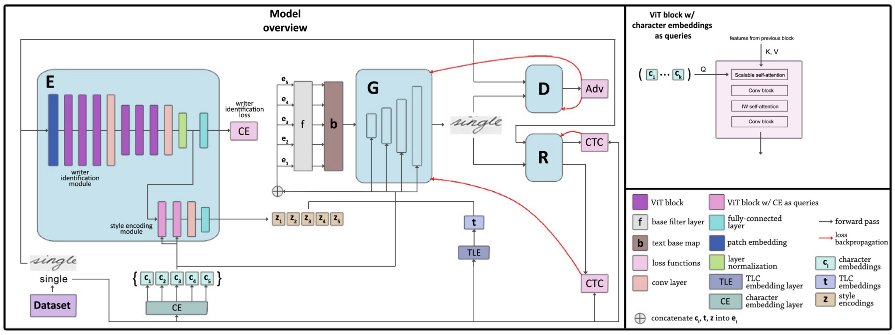

Fig. 1. Overview of the proposed model and the ViT block with character embeddings as queries.

3.2 ViT-Based Style Encoder

To generate images of handwriting that imitate an existing style, we use an encoder network $\mathcal{E}$ to extract the style information from reference images. Inspired by [10], we use $\mathcal{E}$ that is jointly trained by learning to identify the writer of a handwriting sample and learning to encode style encodings for the generator. Instead of using a simple convolutional network as [10] does, our $\mathcal{E}$ is based on ViT. There are two reasons why we employ a ViT-based network for the style encoder. First, ViTs have shown to be exceptionally powerful at modeling local and global information in images, which makes them a great choice to encode handwriting style from an image because a style is defined by both global features (e.g., slant) and local features (e.g., shape of characters, stroke width). Second, ViT allows us to incorporate information about the target character sequence in the encoding mechanism, so we are able to produce style encodings that are not only dependent on the handwriting style, but also on the individual characters appearing in the sequence. As a result, we can create character-specific style encodings, which distinguishes our method from [10] that extracts identical style encoding for the whole character sequence without considering the differences needed to capture to correctly encode the style for different characters.

The identifier module of $\mathcal{E}$ takes handwriting images as input and is composed of a convolutional patch embedding layer and transformer layers from [37] and a single fully-connected layer. It is trained to predict writers of handwriting images, which makes it learn and model differences in handwriting styles.

The encoding module of $\mathcal{E}$ takes features extracted by the transformer layers of the identifier module and disentangles the handwriting styles to produce style encoding vectors. Similarly to the identifier module, it consists of transformer blocks and fully-connected layers to yield fixed-sized length encoding for each character in the target sequence. Each transformer block in the encoding module consists of multi-head scalable self-attention and interactive windowed self-attention introduced in [37] along with fully-connected layers. ScalableViT [37] is a variant of ViT that shows excellent performance across many vision tasks and we empirically found it to yield the best results among several popular state-of-the-art ViT architectures. The feature inputs from the identifier module are averaged across the spatial dimensions before being passed to the encoding module to reduce differences in features based on the characters in the text from the reference image as we want to produce encodings specific to the characters in the target sequence.

To produce character-specific style encodings, we use characters in the target sequence as queries for self-attention in the encoding module. Therefore, unlike regular self-attention where query $Q()$, key $K()$, and value $V()$ all have the same input, the $Q()$ input here is an embedded character c in the target character sequence while $K()$ and $V()$ inputs are features $\mathbf{X}$ from the previous transformer block. The self-attention in our module is thus calculated as

$$Attn\,(\mathbf{X}, c) = softmax(\frac{Q(c)K(\mathbf{X})}{\sqrt{d}})V(\mathbf{X}). \tag{1}$$

We produce a fixed-size style encoding vector for each character in the target sequence. Style encodings of all characters are then employed as conditional input for MCCBN in the generator.

3.3 Character Modulation

The original discriminator in [9,38] does not use any semantic information about characters in the image to predict whether an image is real or generated. We strive to improve the performance of the discriminator by modeling the semantic information. Similarly to techniques such as [19], we add a branch with softmax activation after each block in the discriminator. The feature maps $\mathbf{X}$ where $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ are passed to a convolutional layer f with the kernel size of 1×1 and K output channels where K corresponds to the number of characters in the character set, such as alphanumeric characters. This aggregates channel-wise features and reduces the number of channels to match the number of characters. In order to utilize the information aggregated in the reduced feature space, we need a gating mechanism that promotes emphasis of a single channel as we ideally want each channel to correspond to one character. To achieve this, we employ a softmax activation function across the channel dimension. The process can be denoted as

$$\hat{\mathbf{X}} = softmax\left(f\left(\mathbf{X}, \mathbf{W}\right)\right), \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{C \times K}$. The extracted features $\hat{\mathbf{X}}$ are aggregated with the original features $\mathbf{X}$ by concatenating them across the channel dimension to produce $\tilde{\mathbf{X}} \in \mathbb{R}^{H \times W \times (C+K)}$, and finally the concatenated features are passed to another 1×1 convolutional layer to reduce the number of channel dimensions back to C. Figure 2 illustrates the whole process.

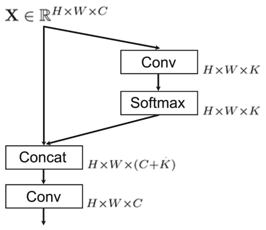

Fig. 2. Diagram of the character modulation component used in the discriminator.

4 Experiments

4.1 Implementation Details

Our model is based on JokerGAN [38], whose core consists of BigGAN [6] layers that are modified for generation of images with a fixed height and variable width. Besides our stated modifications, the architecture of $\mathcal{G}$, $\mathcal{D}$ and $\mathcal{R}$ is identical as in [38]. The identification module of $\mathcal{E}$ consists of a patch embedding layer (filter size 8, stride 4) and two ViT stacks, each comprised of 3 blocks of ViT layers followed by a convolutional layer. The encoding module consists of two blocks of ViT [37] with character embeddings as queries for self-attention, followed by a convolutional layer for downsampling and a fully-connected layer to obtain fixed-size style encoding vectors. The architecture of ViT layers is from [37]. We use the Adam optimizer [25] with a learning rate of 0.0002 for training. Our model is implemented in PyTorch.

4.2 Datasets

We use the following two datasets in our experiments.

- **IAM**. The IAM dataset [29] contains approximately 80k grayscale images of handwritten words in English, divided into training, test and validation sets. The training set consists of about 40k and test set of about 10k images. The words are written by 657 different people and all words written by one person only appear in one of the sets to achieve mutual exclusivity of authors in training, validation, and test sets. The data was created and preprocessed for training of HTR models.
- **GNHK**. The GNHK dataset [27] is composed of images of unconstrained handwritten text in the wild captured by mobile phone cameras. The training set we use in our experiments contains about 28k images of individual handwritten words cropped from the original text images. Due to the nature of the data, the GNHK dataset consists of images with more variety in style and more noise, which makes it more challenging than the IAM dataset.

4.3 Handwritten Text Image Generation

We evaluate the quality of handwritten text image generation and handwriting style imitation using several metrics to measure different performance aspects.

- **Visual Quality**. Our primary metrics of visual quality are Frechet Inception Distance (FID) [16] and Kernel Inception Distance (KID) [5], which are widely used to evaluate GANs. FID and KID compare the distributions of generated images and real samples. We also use the structural similarity index (SSIM) that measures structural similarity between real and generated images.
- **Style Imitation**. We use the writer identification error rate (WIER) [11] to evaluate how well a model can imitate styles. A writer identification network is trained on the test set of images and used to predict writers for images that are generated with test set images as reference. Misclassified samples are deemed as failures of the generative model to accurately imitate the handwriting style. We measure WIER when generating the identical word as in the reference image (WIER-I) as well as when generating a random word (WIER-R). SSIM also indicates how similar the styles in generated and reference images are.
- **Diversity and Fidelity**. To measure how diverse and accurate the generated images are with respect to the distribution of real images, we use GAN-train and GAN-test evaluation [33]. Originally, GAN-train is measured by training an image classifier on generated data and testing on real data and approximates image generation recall, and GAN-test is measured by training on real data and testing on generated data and approximates image generation precision. GAN-tt is an average of GAN-train and GAN-test to consider their trade-off and combine them into one score. In our case, we use a HTR model in place of an image classifier and word recognition accuracy as the underlying metric to calculate GAN-train and GAN-test scores.
- **Readability**. GAN-test also indicates the readability of generated samples as it measures if a HTR model trained on real data can read them.

Table 2. Comparison of performance of handwritten text generation methods on the IAM dataset in terms of metrics for image generation quality based on the distance between real and generated image distributions (FID, KID), structural similarity of images (SSIM), style imitation accuracy measured by WIER, and diversity and quality of image generation measured by training and testing with HTR models (GAN-test, GAN-train, GAN-tt). Evaluation on real data is provided for reference.

Method	FID↓	KID↓	SSIM↑	WIER_I↓	WIER_R↓	GAN-test↑	GAN-train↑	GAN-tt↑
ScrabbleGAN [9]	19.98	1.359	–	–	–	93.11	27.67	60.39
HWT [4]	18.76	1.214	0.182	0.829	0.855	64.06	22.15	43.11
JokerGAN [38]	4.63	0.186	–	–	–	80.86	54.42	67.64
HiGAN-L [10]	17.69	1.107	–	–	–	**96.15**	31.93	64.38
HiGAN-R [10]	12.43	0.669	0.196	0.628	0.674	96.61	36.03	66.62
HiGAN+ [11]	5.94	0.368	0.332	0.526	0.575	95.65	30.72	63.19
Ours (latent)	3.00	0.098	–	–	–	94.27	55.67	**74.97**
Ours (reference)	**2.14**	**0.078**	**0.429**	**0.327**	**0.499**	81.11	**61.90**	71.51
Real data	0.02	0.002	–	0.043	–	83.49	–	–

As shown in Table 2, our method outperforms existing methods in virtually all metrics. GAN-test is the only metric in which it slightly falls behind. This can be attributed to the fact that our model generates more diverse samples that might be harder for a HTR model to read correctly. The diversity is measured by GAN-train in which our model surpasses other methods with a high GAN-test score by a large margin. In addition, the GAN-test score of our method when using reference images for generation is similar to word recognition accuracy of a HTR model trained and tested on real data, which also suggests that the distribution of images generated with our method is closer to the distribution of real data. Our method also achieves the best WIER scores, indicating that it can imitate styles from reference images better than other methods. We state results of our method when generating both in random styles from randomly sampled latent vectors (latent) and styles from reference images (reference).

SSIM and WIER are only applicable for methods that imitate style from reference images; therefore, we do not use them for evaluation of methods that generate images in random styles. We also include real data results for reference where WIER represents the error rate of a writer identifier trained and tested on real data, and the value in the GAN-test column represents word recognition accuracy of a HTR model trained and tested on real data.

Table 3 shows results of experiments on GNHK. The trends are similar to those in Table 2, further attesting our method outperforms the competition. GNHK dataset is more complex than IAM, so the performance of all models is worse compared to IAM. In particular, the diversity is limited and HTR models trained only on generated images perform poorly as shown by low GAN-train. Due to the nature of the dataset, we use top-5 error rate for WIER for GNHK.

Table 3. Comparison of performance of handwritten text generation methods on the GNHK dataset. The same metrics as in Table 2 are used.

Method	FID↓	KID↓	SSIM↑	WIER$_I$ (top-5)↓	GAN-test↑	GAN-train↑	GAN-tt↑
ScrabbleGAN [9]	20.92	1.603	–	–	92.24	8.98	50.61
JokerGAN [38]	10.35	0.515	–	–	76.97	10.85	43.91
HiGAN-L [10]	14.27	0.840	–	–	**93.89**	8.12	51.00
HiGAN-R [10]	8.12	0.366	**0.327**	0.695	92.09	9.58	50.84
HiGAN+ [11]	8.04	0.459	0.235	0.293	82.76	6.10	44.43
Ours (latent)	8.69	0.395	–	–	89.88	**12.30**	**51.09**
Ours (reference)	**5.99**	**0.194**	0.269	**0.129**	82.39	12.15	47.27
Real data	0.03	0.005	–	0.081	63.32	–	–

4.4 Ablation Study

We perform ablation studies to validate the effectiveness of individual proposed components and modifications. We use the IAM dataset and evaluate the performance in two settings, 1) handwritten text generation guided by a reference image, and 2) handwritten text generation from randomly sampled latent vectors. Baseline refers to [38] with the style encoder from [10].

As can be seen in Table 4 and 5, all of our new components and modifications improve the performance of generation either from reference images or random latent vectors. WIER improves particularly thanks to our newly proposed ViT-based style encoder, which can encode the handwriting styles better than a conventional CNN-based encoder. The performance further improves when we input the target character sequence into the encoder and encode specific style vectors for individual characters in the sequence that we want to generate. The ViT-based style encoder also enhances the overall quality of generated images and their fidelity to the distribution of the original real data as measured by FID and KID, and also similarity to reference images in terms of the structure as measured by SSIM. Since the style encoder is not used for generation with random styles, replacing a CNN-based style encoder with our ViT-based one does not significantly affect performance when generating from random latent vectors instead of reference images as can be seen in Table 5. Using strict style conditioning without introducing additional randomness by appending a random latent vector to the style encoding and using identical style conditions for all blocks in the generator instead of hierarchical input improves the performance of generation from latent vectors. Finally, our newly proposed character modulation in the discriminator improves performance of generation in both settings.

Table 4. Ablation study of individual changes to the baseline model evaluated on the IAM dataset. Style of generated images is defined by reference images.

Method	FID↓	KID↓	SSIM↑	WIER_I↓	GAN-test↑	GAN-train↑
baseline	3.698	0.171	0.266	0.498	69.82	50.22
+ strict style conditioning	3.634	0.154	0.263	0.503	64.99	47.88
+ non-hierarchical conditioning	3.719	0.178	0.271	0.499	75.82	48.53
+ character modulation in $\mathcal{D}$	3.252	0.116	0.284	0.459	85.76	51.35
+ ViT-based style encoder	2.560	0.075	0.327	0.407	88.58	55.86
+ character specific style encoding	2.136	0.078	0.429	0.327	81.11	61.90

Table 5. Ablation study of individual changes to the baseline model evaluated on the IAM dataset. Style of generated images is random, defined by random latent vectors.

Method	FID↓	KID↓	GAN-test↑	GAN-train↑
baseline	7.330	0.575	82.74	48.19
+ strict style conditioning	4.003	0.202	80.92	48.46
+ non-hierarchical conditioning	3.261	0.122	82.05	54.19
+ character modulation in $\mathcal{D}$	3.047	0.104	87.83	50.74
+ ViT-based style encoder	3.042	0.076	95.08	50.57
+ character specific style encoding	3.002	0.098	94.27	55.67

4.5 Data Augmentation for HTR

Creating annotation for training of machine learning models is a demanding and expensive process, so in real life, we may encounter situations where we only have unlabeled or partially labeled data. In this experiment, we follow [38] and simulate the situation that we have partially labeled data by using only 5k images with text annotations from the IAM dataset and the rest without text annotations. We train generative models for handwritten text on both unlabeled and labeled data since only the text recognizer requires text annotation for training, but the rest of the model can be optimized on unlabeled data.

For data augmentation evaluation, we use a HTR model [3] trained only on 5k labeled images from IAM as a baseline (IAM-5k). Each of our trained generative models is then used to augment the training dataset by generating additional 100k images for training of HTR models. Word error rate (WER) and normalized edit distance (NED) are used as metrics for evaluation. Table 6 illustrates that using additional training data generated by handwritten text generation models improves the HTR performance, and in particular, our proposed model achieves the biggest performance boost out of all tested models. We also include the results of a HTR model trained on the complete IAM dataset of 40k labeled images for reference.

Table 6. Comparison of different models when used to generate additional data for training of a handwritten text recognition model.

Data	WER↓	NED↓
IAM-40k	16.52	4.95
IAM-5k	34.17	11.90
IAM-5k + ScrabbleGAN 100k [9]	30.65	10.00
IAM-5k + HWT 100k [4]	33.58	11.59
IAM-5k + JokerGAN 100k [38]	28.50	9.26
IAM-5k + HiGAN-L 100k [10]	30.42	10.01
IAM-5k + HiGAN-R 100k [10]	30.96	10.44
IAM-5k + HiGAN+ 100k [11]	27.76	8.94
IAM-5k + Ours (latent) 100k	27.17	8.73
IAM-5k + Ours (reference) 100k	**25.00**	**8.08**

4.6 Qualitative Evaluation

Figure 3 shows results of handwritten text generation and style imitation with different methods. The models are trained on the training set of the IAM dataset and the reference images used in Fig. 3 are from the test set of IAM. As can be seen, our model can accurately imitate the styles in the reference images, which shows that it generalizes well and it can imitate styles that it did not see during training. Since JokerGAN and ScrabbleGAN cannot imitate styles, the text in Fig. 3 is generated in random styles for these two methods.

Figure 4 shows images generated by models trained on the GNHK dataset. Style imitation is less accurate than in the case of IAM because the GNHK dataset contains a larger variety of images of unrestricted handwritten text in RGB colorspace. However, images produced by our model still exhibit a significant style similarity to reference images.

4.7 Text Erasing

We demonstrate that while our proposed method is primarily intended for handwritten text generation, it can be also used to erase text from documents, as shown in Fig. 5. When we include whitespace in the character set that the model learns to generate, our model can erase text from a reference image by generating whitespace characters with style guided by the reference image. Note that it can erase text while preserving the original background and text lines.

Whitespace characters are not included in the original training data. As a solution, we randomly add a whitespace character to the beginning or the end of a word by padding the image with the left or right edge pixels. The size of the padding corresponds to the set approximate width of one character.

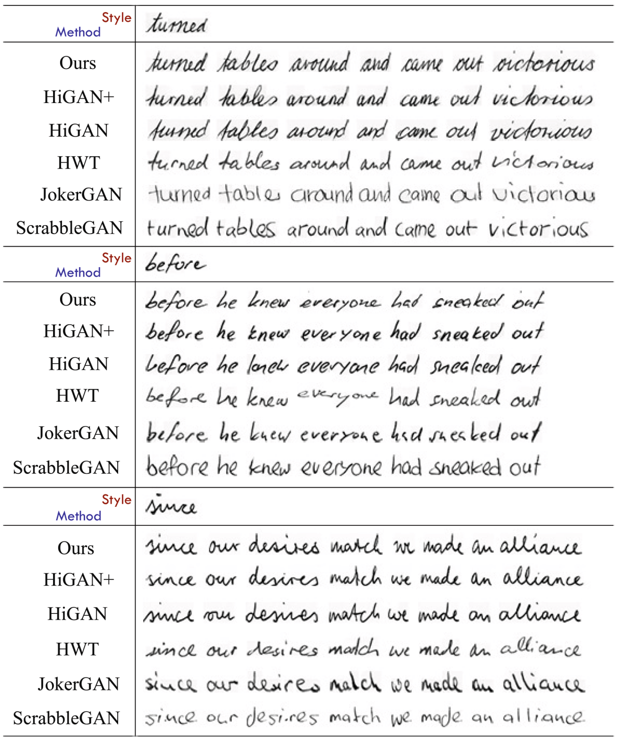
Style
Method
turned
Ours
HiGAN+
HiGAN
HWT
JokerGAN
ScrabbleGAN
turned tables around and came out victorious
before
before he knew everyone had sneaked out
since
since our desires match we made an alliance

Fig. 3. Results of handwriting style imitation with different methods. The reference style images are from the test set of the IAM dataset and the models did not see those handwriting styles during training.

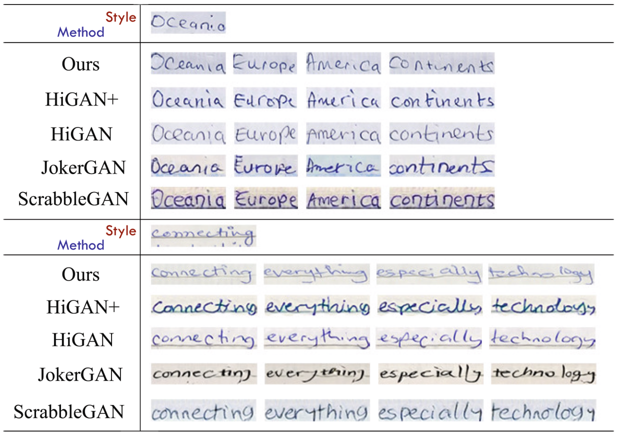
Style
Method
Oceania
Ours
HiGAN+
HiGAN
JokerGAN
ScrabbleGAN
Oceania Europe America continents
connecting
connecting everything especially technology

Fig. 4. Results of handwriting style imitation with different methods. The reference style images are from the test set of the GNHK dataset.

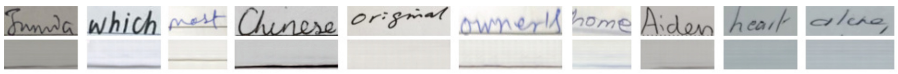

Fig. 5. Results of text erasing with our proposed method.

Table 7. Comparison of existing models and our proposed model in terms of size in megabytes. Only the modules necessary to store to perform generation, that is generator (Gen) and encoder (Enc), are considered.

Method	Size (MB)		
	Gen	Enc	Total
ScrabbleGAN [9]	81.8	N/A	81.8
JokerGAN [38]	11.0	N/A	11.0
GANWriting [22]	95.6	76.5	172.1
HWT [4]	80.7	50.6	131.3
HiGAN [10]	38.6	20.5	59.1
HiGAN+ [11]	15.0	6.7	21.7
Ours	11.6	12.6	24.2

4.8 Model Size

Table 7 shows a comparison of the size of our model and existing models for handwritten text generation. We only consider the size of the modules that are needed at inference time, which is the generator and encoder. In the case of models that only generate handwritten text in random styles, there is no encoder. We denote the size of the models in megabytes. Our model not only achieves better performance in terms of generation quality, but as can be seen, it is also one of the most lightweight models.

5 Conclusion

We have proposed a new method for generation of handwritten text images. Our method can not only generate handwriting in a random style, but it can also imitate a specific handwriting style passed to the model as a reference in the form of a raster image. Experiments show that our method outperforms existing methods in terms of the quality of generation and similarity to the style of handwriting in reference images. The performance has particularly improved thanks to our newly proposed ViT-based style encoder that takes the target character sequence that we want to generate as an additional input to produce character-specific style encodings. We also show that images generated by our model can be used for data augmentation for training of OCR models for handwritten text.

Acknowledgment. This work was supported by JSPS KAKENHI Grant Number JP22H00540.

References

1. Aksan, E., Pece, F., Hilliges, O.: DeepWriting: making digital ink editable via deep generative modeling. In: CHI (2018)
2. Alonso, E., Moysset, B., Messina, R.: Adversarial generation of handwritten text images conditioned on sequences. In: ICDAR (2019)
3. Baek, J., et al.: What is wrong with scene text recognition model comparisons? Dataset and model analysis. In: ICCV (2019)
4. Bhunia, A.K., Khan, S., Cholakkal, H., Anwer, R.M., Khan, F.S., Shah, M.: Handwriting transformers. In: ICCV (2021)
5. Bińkowski, M., Sutherland, D.J., Arbel, M., Gretton, A.: Demystifying MMD GANs. In: ICLR (2018)
6. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: ICLR (2018)
7. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12346, pp. 213–229. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_13
8. Dosovitskiy, A., et al.: An image is worth 16x16 words: transformers for image recognition at scale. In: ICLR (2021)
9. Fogel, S., Averbuch-Elor, H., Cohen, S., Mazor, S., Litman, R.: ScrabbleGAN: semi-supervised varying length handwritten text generation. In: CVPR (2020)
10. Gan, J., Wang, W.: HiGAN: handwriting imitation conditioned on arbitrary-length texts and disentangled styles. In: AAAI (2021)
11. Gan, J., Wang, W., Leng, J., Gao, X.: HiGAN+: handwriting imitation GAN with disentangled representations. ACM Trans. Graph. **42**(1), 1–17 (2022)
12. Goodfellow, I.J., et al.: Generative adversarial networks. In: NIPS (2014)
13. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
14. Guan, M., Ding, H., Chen, K., Huo, Q.: Improving handwritten OCR with augmented text line images synthesized from online handwriting samples by style-conditioned GAN. In: ICFHR (2020)
15. Haines, T.S.F., Mac Aodha, O., Brostow, G.J.: My text in your handwriting. ACM Trans. Graph. **35**(3), 1–18 (2016)
16. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: NIPS (2017)
17. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: NeurIPS (2020)
18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
19. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR (2018)
20. Ji, B., Chen, T.: Generative adversarial network for handwritten text. arXiv preprint arXiv:1907.11845 (2019)
21. Kang, L., Riba, P., Rusiñol, M., Fornés, A., Villegas, M.: Content and style aware generation of text-line images for handwriting recognition. TPAMI **44**(12), 8846–8860 (2022)
22. Kang, L., Riba, P., Wang, Y., Rusiñol, M., Fornés, A., Villegas, M.: GANwriting: content-conditioned generation of styled handwritten word images. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12368, pp. 273–289. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58592-1_17

23. Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., Aila, T.: Training generative adversarial networks with limited data. In: NeurIPS (2020)
24. Karras, T., et al.: Alias-free generative adversarial networks. In: NeurIPS (2021)
25. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
26. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
27. Lee, A.W.C., Chung, J., Lee, M.: GNHK: a dataset for English handwriting in the wild. In: ICDAR (2021)
28. Luo, C., Zhu, Y., Jin, L., Li, Z., Peng, D.: SLOGAN: handwriting style synthesis for arbitrary-length and out-of-vocabulary text. IEEE Trans. Neural Netw. Learn. Syst. (2022)
29. Marti, U.V., Bunke, H.: The IAM-database: an English sentence database for offline handwriting recognition. IJDAR **5**(1), 39–46 (2002)
30. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
31. Miyato, T., Koyama, M.: cGANs with projection discriminator. In: ICLR (2018)
32. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: CVPR (2019)
33. Shmelkov, K., Schmid, C., Alahari, K.: How good is my GAN? In: ECCV (2018)
34. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: transformer for semantic segmentation. In: ICCV (2021)
35. Wang, J., Wu, C., Xu, Y.Q., Shum, H.Y.: Combining shape and physical models for online cursive handwriting synthesis. IJDAR **7**(4), 219–227 (2005)
36. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional GANs. In: CVPR (2018)
37. Yang, R., et al.: ScalableViT: rethinking the context-oriented generalization of vision transformer. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision – ECCV 2022. LNCS, vol. 13684, pp. 480–496. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-20053-3_28
38. Zdenek, J., Nakayama, H.: JokerGAN: memory-efficient model for handwritten text generation with text line awareness. In: ACM Multimedia (2021)
39. Zhang, B., et al.: StyleSwin: transformer-based GAN for high-resolution image generation. In: CVPR (2022)
40. Zhang, H., et al.: StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks. In: ICCV (2017)