Imuru: Image-Only Style Transfer for Autoregressive Handwritten Text Generation

Haojie Cai*
New York University
New York, USA
hc3961@nyu.edu

Thomas Chen†
New York University
New York, USA
jc10883@nyu.edu

Scott Fan†
New York University
New York, USA
wf2060@nyu.edu

Bob (Shengtao) Yao†
New York University
New York, USA
sy3535@nyu.edu

Abstract

Generating high-quality styled text images that accurately reproduce both content and writing style remains challenging for handwritten text generation (HTG). While autoregressive HTG models show promising zero-shot style generalization, they suffer from limitations. Recent autoregressive approaches often relies on the textual transcription of the reference style image as an additional input to properly condition on the target style. In practice, this requirement limits usability and can cause the generation to fail when such annotations are unavailable or inaccurate. To address this limitation, we propose ***Imuru****, a novel autoregressive HTG framework built on Emuru that eliminates the need for reference-style transcriptions. In this work, we design a new training and inference pipeline that enables the model to condition on style directly from the reference style image. Experiments on multiple HTG benchmarks show that Imuru generalizes well to unseen handwriting and typewritten styles while maintaining strong content fidelity, and significantly improves the practicality of autoregressive HTG by requiring only a style image and target text at inference. The source codes and models are available at* https://github.com/Ruian7P/imuru.

1. Introduction

When we observe someone's handwriting, we can often recognize and reproduce their style without reading every word they wrote. The ability of extracting writing style from visual observation alone remains challenging in current handwritten text generation systems. A barrier appears because models require not only visual examples but also accurate transcriptions of those examples. This dependency becomes problematic when working with historical documents, degraded handwriting, or unfamiliar scripts. Such motivation drives us to build a system that learns handwriting styles purely from visual observation.

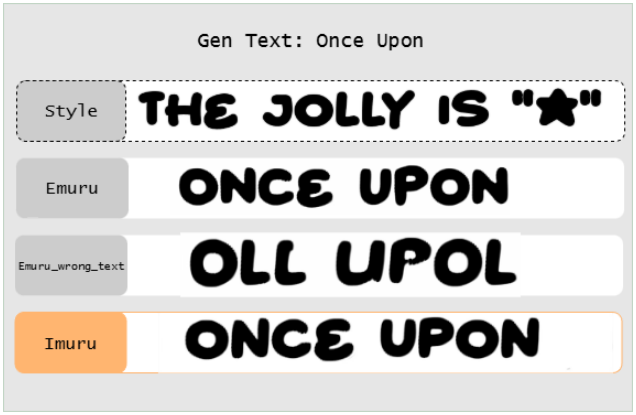

Figure 1. Comparison of text generation with incorrect style text transcription. Given a style image showing "THE JOLLY IS "U" and the target text "Once Upon", Emuru relies on the (incorrectly provided) style text "THE TIMES" and produces garbled output ("OLL UPOL"), while Imuru ignores the wrong text annotation and correctly generates "ONCE UPON" by learning style directly from the visual reference.

Handwritten Text Generation (HTG) [4–6, 11, 12, 14, 17, 18, 20, 21] aims to achieve this goal by generating images that contain user-specified text rendered in a handwriting style resembling reference samples. This task has received significant attention in computer vision due to its practical value in data augmentation for handwriting recognition and its impact on historical document analysis, where identifying and reproducing handwriting styles can be supportive for these studies. HTG can be formulated either online or offline. Online HTG models pen trajectories as stroke sequences, which often requires specialized hardware like digital pens. On the other hand, offline HTG operates directly on static images, making it more practical for reproducing styles from existing manuscripts. Thus, offline HTG constitutes the focus of this work.

*First author.
†Co-second authors; contributed equally.

Despite recent progress, generating high-quality styled handwritten text images offline remains challenging. Except generating the image with the target text, a successful HTG system must generalize to previously unseen writing styles, support variable-length generation with flexible spatial layouts, and avoid blurring background such as paper texture, noise, or scanning defects. Recent approaches have made important steps towards these goals, especially with the emergence of autoregressive frameworks. Emuru [14] has introduced the first autoregressive framework for offline HTG. The approach combines a VAE with an autoregressive encoder-decoder Transformer to generate arbitrary-length text images. By training exclusively on a large-scale synthetic dataset of over 100,000 fonts, Emuru achieves strong zero-shot generalization to unseen handwriting and typewritten styles, while eliminating background noise. However, there is a key limitation in Emuru, where it requires accurate textual transcriptions of the reference style images as input at inference time. This requirement reduces Emuru's practicalilty in real-world settings, as the model relies on the reference transcription to properly condition on the target style. When such style text is unavailable or inaccurate, the quality and the performance of Emuru can degrade substantially, limiting the deployability of autoregressive HTG frameworks.

To address this limitation, we propose **Imuru** (Style-**I**mage-Only E**muru**), a novel autoregressive HTG framework based on Emuru with a redesigned training and inference pipeline that requires only style image for conditioning. Imuru learns to extract and use style representations directly from the reference style image, eliminating the need for style text at test time. Specifically, we restructure the training procedure so that the model is explicitly optimized to perform conditional generation given only a style image and a target text prompt, and we use a special token to support autoregressive decoding for target outputs. Extensive experiments on standard handwritten and typewritten benchmarks demonstrate that Imuru maintains the strong zero-shot generalization ability of Emuru while significantly improving practicality by removing the need for style text during inference.

Our contributions are summarized as follow:

- We propose Imuru, a offline autoregressive HTG model with a redesigned training and inference pipeline.
- We remove the dependency on reference-style text during inference, a key bottleneck that limits the practicality of prior autoregressive HTG models.
- We evaluate Imuru on multiple benchmarks and demonstrate robust performance and strong generalization to unseen styles.

2. Related Works

2.1. Handwritten Text Generation

Handwritten text generation involves generating realistic text images that mimic specific writing styles. Meanwhile, we want to maintain readability and flexibility in text content. GAN-based and diffusion-based methods are the top two approaches.

2.1.1. GAN-based Approaches

GAN-based [5, 6] approaches achieved promising results in styled handwriting generation by learning to disentangle writing style from content. However, we may encounter mode collapse during training. Additionally, GAN-based approaches have difficult generating arbitrary-length text. Moreover, they are struggling to produce writing styles outside their training distribution.

2.1.2. Diffusion-based Approaches

Diffusion models [4] produce outputs with higher quality through iterative denoising processes. Despite improvements in generation quality, diffusion-based methods suffer from computational inefficiency. This happens mainly because of their multi-step sampling procedures.

2.1.3. Autoregressive Image Generation

Autoregressive models [15] have achieved remarkable success in natural language generation. They can be adapted to images by treating them as sequences and generating through iterative next-token prediction.

Emuru [14] introduced the first autoregressive framework for zero-shot styled HTG, combining a VAE with an autoregressive Transformer to generate variable-length text images. By training exclusively on a large-scale synthetic dataset of over 100,000 fonts, Emuru achieves strong zero-shot generalization to unseen styles, eliminating the need for dataset-specific training while outperforming GAN and diffusion-based competitors on standard benchmarks.

2.2. Image Tokenization Approaches

Traditional autoregressive image generation relies on discrete tokenization. VQ-VAE [19] and VQ-VAE-2 [16] introduced vector quantization to convert images into discrete token sequences, but face optimization challenges including codebook collapse and gradient flow difficulties. Recent work has shown that continuous-token approaches overcome these limitations. GIVT [18] and LlamaGen [17] demonstrated that continuous autoregressive models using β-VAE with continuous latents can match diffusion model quality while avoiding quantization problems. However, these methods assume fixed output dimensions, unlike handwriting generation which requires variable-length outputs and autonomous stopping mechanisms.

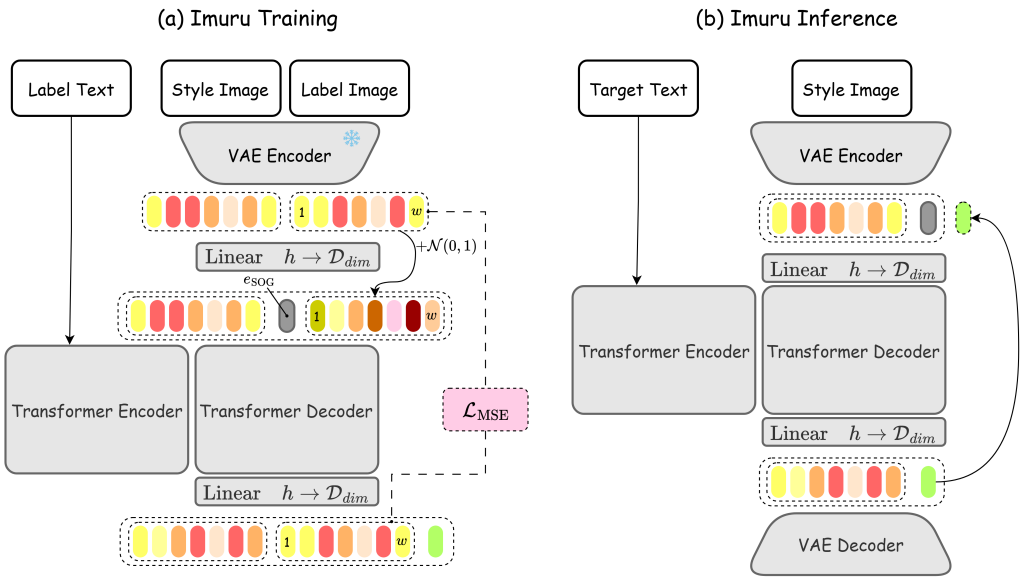

Figure 2. The training and inference pipeline of our Imuru framework. (a) Imuru Training: a trained VAE encodes style and label images into latents, and the Transformer learns to predict label latents from style latents and label text. (b) Imuru Inference: given a style image and target text, Imuru autoregressively generates latent features that the VAE decoder turns into the final styled text image.

3. Methodology

Our framework consists of two main components: a Variational Autoencoder (VAE) that encodes and reconstructs images in a latent space, and an autoregressive Encoder-Decoder Transformer that predicts the latent sequence of a target text line based on a reference style image. We first train the VAE first to obtain a reliable latent representation space. We then introduce Imuru, a Transformer-based model that encodes the style image and the target text and autoregressively generates the corresponding latent sequence in the VAE space. Fig. 2 provides an overview of the training and inference pipeline of Imuru.

3.1. VAE

We reuse the continuous β-VAE, provided in [14], frozen, to project the style image I_s into a compressed latent space and then convert the embeddings generated by the autoregressive Transformer into the output image I_g. Specifically, The VAE Encoder takes the images with dimension $I^{3\times W\times H}$ as input and maps them into latent space representations. Given the VAE's number of channels, c, and its downscaling factor f, the resulting latent vector will have shape $c \times h \times w$, where $w = W/f$, and $h = H/f$. In most of the handwritten datasets, lines of text often have fixed height but arbitray length. Taking this into account, we model the image as a w-long sequence of $h \times c$ vectors. Intuitively, the array of w vectors encodes the entire line of text with each w vector encoding a vertical slice. The w vectors are then fed as input to the VAE decoder and it is tasked to reconstruct the grayscale text image I_T without the background, therefore enforcing the latent space to represent the text style rather capture the background content. The model was trained with a combination of reconstruction loss, KL divergence, Cross-Entriopy from the pretrained Writer Identification network, and a CTC loss from a pretrained HTR(Handwritten Text Recoginition) model.

3.2. Imuru Autoregressive Transformer

Imuru is a generative autoregressive Transformer consisting of a Transformer Encoder $\mathcal{E}$ and a Transformer Decoder $\mathcal{D}$. To accelerate training, we build Imuru on top of a pretrained encoder-decoder Transformer models, such as T5 [15]. During training, $\mathcal{E}$ encodes the style text, while $\mathcal{D}$ predicts VAE latents conditioned jointly on the style image latents and the encoded style text. Specifically, we first freeze the pretrained VAE, and feed it with both the style image I_{style} and the label image I_{label}. This yields the style image latents $v_{\text{style}} = [v_1^{\text{style}}, \ldots, v_m^{\text{style}}]$ and the label image latents $v_{\text{label}} = [v_1^{\text{label}}, \ldots, v_n^{\text{label}}]$, where m and n are the corresponding sequence lengths and all vectors lie in the VAE latent space with dimension of d. During the training process, v_{label} serves not only as the supervision ground truth for loss computation, but also as the reference sequence for the Transformer Decoder $\mathcal{D}$. Thus, to improve robustness, we add Gaussian noise to the label sequence during training, obtaining noisy label sequence $\tilde{v}_{label}$. Then, a linear projection maps the style latents v_{style} and the noisy label latents $\tilde{v}_{label}$ into the Trans-

former hidden space of dimension $\mathcal{D}_{\text{dim}}$. Let $e_{\text{style}} = [e_1^{\text{style}}, \ldots, e_m^{\text{style}}]$ and $e_{\text{label}} = [v_1^{\text{label}}, \ldots, v_n^{\text{label}}]$ denote the projected style and label sequences in the Transformer hidden space. We construct the Transformer Decoder input sequence $x_{\mathcal{D}}$ by prepending the projected style sequence e_{style}, followed by a special learned start-of-generation embedding e_{SOG}, and finally the projected label representation e_{label}. Formally, the input sequence to $\mathcal{D}$ is

$$x_{\mathcal{D}} = [\underbrace{e_1^{\text{style}}, \ldots, e_m^{\text{style}}}_{e_{\text{style}}}, e_{\text{SOS}}, \underbrace{e_1^{\text{label}}, \ldots, e_n^{\text{label}}}_{e_{\text{label}}}]. \quad (1)$$

In parallel, the label text T_{label} in the label image I_{label} is tokenized and passed through the Transformer Encoder $\mathcal{E}$, producing a sequence of text embeddings $s = [s_1, \ldots, s_k]$ which are used as keys and values for cross-attention in the decoder. Once we obtain decoder input embeddings $x_{\mathcal{D}}$ and the encoder outputs s, we feed them into the Transformer Decoder $\mathcal{D}$. Causal masked self-attention is applied over $x_{\mathcal{D}}$, while unmasked cross-attention is computed between $x_{\mathcal{D}}$ and s. The decoder then outputs a sequence of hidden states h with sequence length of $m + n + 1$. Finally, we linearly project the output sequence h back into the $d-$dimensional VAE latent space to obtain the predicted output sequence $\hat{v}_{\text{pred}} = [\hat{v}_1^{\text{pred}}, \ldots, \hat{v}_{m+n+1}^{\text{pred}}]$. This output sequence is used for the training loss computation.

Imuru is trained following a teacher-forcing strategy, where start-of-generation token e_{SOG} is trained to predict the first token in the label sequence v_{label}, and each latent e_i^{label} in the label sequence is trained to predict the subsequent token v_{i+1}^{label}. This introduces a one-step shift between input and target latents. Imuru is optimized with a mean square error loss between the prediction and the ground truth of the label image latents. Specifically, the loss function is defined as

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^{n} ||\hat{v}_{m+i}^{\text{pred}} - v_i^{\text{label}}||_2^2, \quad (2)$$

where $|| \cdot ||_2$ denotes the ℓ_2 norm on the VAE dimension. This loss encourages the decoder to reconstruct the entire label image latent sequence given the style image latents and the encoded style text. The Gaussian noise on $\tilde{v}_{label}$ serves as a regularizer, making the model more robust to small perturbations in the latent space and reducing overfitting.

3.3. Inference

At inference time, Imuru is provided with a reference style image I_{style} and a textual prompt T_{target} describing the desired output. We first pass I_{style} through the VAE encoder to obtain the style latent sequence v_{style}, which is then mapped into the Transformer hidden space to form $e_{\text{style}} = [e_1^{\text{style}}, \ldots, e_m^{\text{style}}]$. At the same time, the prompt T_{target} is tokenized and fed into the Transformer Encoder $\mathcal{E}$, yielding encoder outputs s. Since the label image is not available at test time, the label latent sequence is generated autoregressively in our model. We initiate the decoder input with the style prefix and the start-of-generation token. The decoder input embeddings is then $x_{\mathcal{D}}^{(0)} = [e_1^{\text{style}}, \ldots, e_m^{\text{style}}, e_{\text{SOG}}]$. We forward $x_{\mathcal{D}}^{(0)}$ and the encoder outputs s into $\mathcal{D}$. The hidden state at the last position of the decoder output is then projected back to the VAE latent space, producing the first predicted latent $\hat{v}_1^{\text{pred}}$. For each subsequent step $t \geq 1$, we project the already generated latents $\hat{v}_1^{\text{pred}}, \ldots, \hat{v}_t^{\text{pred}}$ into the Transformer hidden space, obtaining $\hat{e}_1^{\text{pred}}, \ldots, \hat{e}_t^{\text{pred}}$, and concatenate them with $x_{\mathcal{D}}^{(0)}$. The decoder input is updated as

$$x_{\mathcal{D}}^{(t)} = [e_1^{\text{style}}, \ldots, e_m^{\text{style}}, e_{\text{SOG}}, \hat{e}_1^{\text{label}}, \ldots, \hat{e}_t^{\text{label}}], \quad (3)$$

and passed again through $\mathcal{D}$ to obtain the next latent $\hat{v}_{t+1}^{\text{pred}}$. Repeating this procedure yields an autoregressively generated sequence.

As in Emuru [14], all traing images are padded with white space for batching, which induces a compact cluster of padding latents in the VAE space. We follow the logic of stopping generation in Emuru, where at inference time, we compute the cosine similarity between the most recent predictions and the padding token, and terminate generation once the model emits P consecutive latents whose similarity to the padding token exceeds a fixed threshold. After stopping, we feed the predicted sequence $\hat{v}_{\text{pred}}$ to the VAE decoder. This will result the final image rendering the target text T_{target} in the handwriting style captured from the style image I_{style}.

4. Experiments

4.1. Experimental Setup

Datasets. We evaluate on multiple benchmark datasets representing diverse handwriting styles and languages. Following standard HTG protocols, we use both word-level and line-level annotations from the IAM Handwriting Database (161 writers for testing) [10], the CVL Database (283 authors for testing with English and German manuscripts) [8], and the RIMES Database (French handwritten letters) [1]. We also evaluate on the Karaoke dataset [14] containing song lyrics in English, French, German, and Italian rendered with 100 publicly available fonts (both calligraphy and typewritten styles). Notably, while competing methods are trained on IAM, our model is trained exclusively on synthetic data: a large-scale pretraining dataset of 23M samples (font-square-pretrain-20M) followed by fine-tuning on 10M variable-length samples (font-square-paris-ft) [21], making all evaluation datasets effectively out-of-distribution. In our experiments, We use first 500 tar shards in font-square-

pretrain-20M for pretraining, and first 50 tar shards in font-square-pairs-ft for fine-tuning.

Metrics. Following HTG evaluation best practices, we employ multiple metrics to assess generation quality. We use Fréchet Inception Distance (FID) [7] and Kernel Inception Distance (KID) [3] to measure distributional similarity in feature space. To capture style fidelity while reducing background sensitivity, we compute Binarized FID (BFID) [14] on Otsu-thresholded images. For perceptual style assessment, we use Handwriting Distance (HWD) [13], a task-specific metric comparing styles through learned representations. To evaluate text readability, we measure the Absolute Character Error Rate Difference (△CER) [14] using TrOCR-Base [9], ensuring generated images maintain similar readability to references without stylistic collapse—△CER near zero is ideal. Lower values indicate better performance across all metrics. We report KID multiplied by 10^3.

Baselines. For our experimental comparison, we evalaute a range of state-of-the-art methods for HTG that span both adversarial-based and denoising diffusion-based frameworks. To ensure fairness and reproducibility, we restrict our comparisons to approaches for which official implementations and pretrained weights are publicly available. Our benchmark includes convolutional GAN-based models such as HiGAN+ [6] and TS-GAN [5], as well as Transformer-based GAN architectures including HWT [2], VATr [**?**], and VATr++ [20]. In addition, we incorporate recent diffusion-based HTG methods, namely, DiffPen [11] and One-DM [4].

Implementation Details. Our VAE is built upon the pretrained Emuru VAE [14], which consists of four encoder and four decoder layers with output channel size of $32, 64, 128, 256$, respectively, a downscaling factor $f = 8$ and a single output channel. We fine-tune the pretrained VAE on our training dataset to ensure the learned VAE latent space better aligns with our dataset distribution. Fine-tuning is performed for 2 epochs using a batch size of 64 and a learning rate of 5×10^{-5}, without using writer-id loss.

The autoregressive text-to-image generation backbone builds upon the T5 encoder-decoder architecture [15], agumented with two linear adapter layers. One adapter projects the VAE latent embeddings into the T5 decoder's internal dimensionality($D_{dim} = 1024$), while the second replaces the original T5 decoder output projection to map back the VAE latent space. Our Imuru-Large model builds upon T5-Large, and Imuru-Small is based on T5-Small. The text input to the encoder is tokenized using the Google ByT single-character tokenizer. During training, we employ the AdamW optimizer with a learning rate of 2×10^{-5}. In the pretraining stage, we train the model for 5 epochs using a batch size of 32 and apply noisy teacher forcing on label representations with probability 0.1. During this stage, all training images are padded or cropped to a fixed width of 1024 pixels. In the fine-tuning stage, we train the model for 1 epochs using a batch size of 2, without limiting the maximum width of each images. Gradient accumulation is used to simulate a larger effective batch size, and noisy teacher forcing is disabled during this stage. All of our models are trained and evaluated on a single NVIDIA RTX Pro 6000 Blackwell GPU.

Table 1. Zero-shot evaluation on the IAM **word-level** dataset. All competing methods are trained on IAM, while Imuru is trained exclusively on synthetic data and performs inference using image-only style conditioning. Lower is better (↓) for all metrics. KID is reported $\times 10^3$.

Method	FID↓	BFID↓	KID↓	ΔCER↓	HWD↓
TS-GAN	129.57	86.45	141.08	0.28	4.22
HiGAN+	50.19	21.92	43.39	0.20	3.12
HWT	27.83	15.09	19.64	0.15	2.01
VATr	30.26	15.81	22.31	**0.00**	2.19
VATr++	31.91	17.15	23.05	0.07	2.54
One-DM	27.54	10.73	21.39	0.10	2.28
DiffPen	**15.54**	**6.06**	11.55	0.06	**1.78**
Imuru (Ours)	90.66	60.3	**0.078**	0.73	2.43

4.2. Results

Zero-Shot Inference on IAM. Most existing HTG approaches achieve strong performance by training on a single target dataset, most commonly the word-level IAM dataset, and by conditioning generation on both style images and their corresponding textual content. In Table 1, we compare such IAM-trained methods with **Imuru**, which is trained exclusively on large-scale synthetic data and performs inference using *only* a reference style image and a target text prompt. We refer to this evaluation protocol as *zero-shot inference*, as Imuru has never observed IAM samples during training. Despite relying on weaker conditioning information at inference time, Imuru achieves performance that is competitive with existing methods across multiple metrics, demonstrating strong generalization and effective style extraction directly from image-only cues.

Zero-Shot Inference on CVL and RIMES. Dataset-specific training strategies severely limit the generalization ability of many HTG models. As shown in Table 3, competing approaches trained on IAM exhibit a marked performance degradation when evaluated on CVL and RIMES line-level benchmarks. In contrast, Imuru maintains stable performance across both datasets, despite not having access to style text during inference. Interestingly, while some

Figure 3. Visual comparison of generated text images across six benchmark datasets (IAM Words, IAM Lines, CVL Lines, RIMES Lines, Karaoke Typewritten, Karaoke Calligraphy). Given a style image (top row), each method generates the target text shown in the reference (bottom row, green). Our Imuru model achieves superior text correctness while maintaining style consistency.

competing methods preserve text readability—as reflected by relatively stable △CER scores—they often fail to accurately capture the target handwriting style. These results indicate that directly modeling style in a latent image space enables Imuru to generalize more robustly across datasets and writing styles.

Zero-Shot Inference on the Karaoke Dataset. We further evaluate Imuru on the Karaoke benchmark, which contains calligraphic and typewritten text images rendered in multiple languages and does not include background artifacts. Unlike IAM, CVL, and RIMES, Karaoke emphasizes stylistic diversity rather than writer identity. As reported

Table 2. Zero-shot evaluation on the IAM **line-level** dataset. All competing methods are trained on IAM, while Imuru is trained exclusively on synthetic data and performs inference using image-only style conditioning. Lower is better (↓) for all metrics. KID is reported $\times 10^3$.

Method	FID↓	BFID↓	KID↓	ΔCER↓	HWD↓
TS-GAN	44.17	19.45	45.42	0.02	3.21
HiGAN+	74.41	34.18	77.27	**0.00**	3.25
HWT	44.72	30.26	43.49	0.33	2.97
VATr	35.32	27.97	33.61	0.02	2.37
VATr++	34.00	21.67	29.68	0.03	2.38
One-DM	43.89	21.54	44.48	0.13	2.83
DiffPen	**12.89**	**6.87**	9.73	0.03	2.13
Imuru (Ours)	17.21	8.67	**0.012**	0.2	**1.83**

Table 3. Zero-shot evaluation on the CVL and RIMES line-level datasets. All competing methods are trained on IAM, while Imuru is trained exclusively on synthetic data and performs inference using image-only style conditioning. Lower is better (↓) for all metrics. KID is reported $\times 10^3$.

Dataset	Method	FID↓	BFID↓	KID↓	ΔCER↓	HWD↓
CVLLines	TS-GAN	42.12	31.97	43.15	0.13	3.07
	HiGAN+	78.44	39.47	80.39	0.12	3.07
	HWT	31.22	16.73	26.14	0.38	2.59
	VATr	34.40	24.64	32.21	0.06	2.36
	VATr++	35.53	19.87	34.15	0.12	2.18
	One-DM	60.45	26.58	64.13	0.06	2.66
	DiffPen	40.40	17.50	38.21	**0.01**	2.99
	Imuru (Ours)	**19.14**	**14.03**	**0.016**	0.25	**1.65**
RIMESLines	TS-GAN	109.04	36.39	132.90	0.12	3.26
	HiGAN+	160.57	47.38	183.82	0.14	3.39
	HWT	118.21	35.26	128.66	0.45	3.36
	VATr	113.76	30.21	114.21	0.07	3.09
	VATr++	110.04	35.61	104.05	0.10	2.83
	One-DM	121.18	36.07	121.67	0.20	3.36
	DiffPen	89.79	**18.25**	94.78	**0.04**	2.58
	Imuru (Ours)	**48.11**	27.89	**0.03**	0.35	**2.04**

in Table 4, Imuru achieves consistently strong performance across both calligraphy-style and typewritten text images. Notably, this robustness is achieved without conditioning on any style-related textual information, highlighting Imuru's ability to infer stylistic characteristics directly from visual cues alone.

Comparison between Imuru and Emuru. To isolate the impact of our transcription-free design, we conduct a controlled comparison between Imuru and Emuru under identical training conditions. Both models use the T5-Small backbone and are trained on the same synthetic dataset (font-square-pretrain-20M) for 5 epochs with identical hyperparameters. The only difference lies in the conditioning signal: Emuru relies on both style images and their corresponding text transcriptions, while Imuru operates using style images alone.

Table 4. Zero-shot evaluation on the Karaoke dataset. All competing methods are trained on IAM, while Imuru is trained exclusively on synthetic data and performs inference using image-only style conditioning. Lower is better (↓) for all metrics. KID is reported $\times 10^3$.

Karaoke Calligraphy	FID↓	BFID↓	KID↓	ΔCER↓	HWD↓
TS-GAN	60.30	12.68	64.80	0.23	4.59
HiGAN+	125.75	69.41	136.75	0.08	4.90
HWT	62.69	43.03	59.35	0.32	4.50
VATr	72.22	47.66	67.70	0.05	3.89
VATr++	67.16	46.53	58.57	**0.01**	3.96
One-DM	59.73	38.30	56.55	0.04	4.31
DiffPen	34.19	25.78	28.91	0.16	4.18
Imuru (Ours)	**13.21**	**10.08**	**0.005**	0.15	**1.8**

Karaoke Typewritten	FID↓	BFID↓	KID↓	ΔCER↓	HWD↓
TS-GAN	141.41	75.78	157.33	0.32	4.70
HiGAN+	135.34	63.39	146.34	0.07	5.19
HWT	72.78	37.40	62.77	0.37	4.57
VATr	80.38	41.02	70.46	0.05	4.14
VATr++	76.03	41.69	63.17	**0.01**	4.15
One-DM	70.75	44.06	60.90	0.05	4.80
DiffPen	78.07	61.16	67.17	0.14	4.71
Imuru (Ours)	**10.88**	**3.56**	**0.006**	0.1	**1.07**

As shown in Table 5, Imuru achieves overall stronger performance than Emuru across most evaluation benchmarks. Despite using weaker conditioning, Imuru consistently improves visual quality and style fidelity, as reflected by lower FID, BFID, and HWD scores on RIMES and both Karaoke settings. In addition, Imuru maintains comparable or improved readability, with lower ΔCER and KID in most cases.

These results indicate that removing the dependency on style text transcriptions does not degrade generation quality. On the contrary, Imuru demonstrates slightly better overall performance and stronger robustness across datasets with diverse style characteristics, highlighting the practical advantages of transcription-free conditioning in real-world scenarios.

5. Conclusions

In this paper, we presented Imuru, an autoregressive model for offline styled text image generation based on Emuru. Our approach utilizes a text-image representation model (VAE) and an autoregressive Transformer, both trained and finetuned on large, diverse, and synthetic handwritten datasets rendered in various fonts and writing styles. Imuru attempts to improve upon the Emuru model by modifying the Transformer architecture, introducing synchronization tokens for transcription-free image generation, stop tokens to ensure well-defined sequence termination, and classifier-free guidance for improved text fidelity. During evaluation time, we considered multiple datasets, both handwritten and

Table 5. Comparison between Imuru and Emuru (T5-Small, 5 epochs). Imuru removes dependency on style text transcriptions. Lower is better (↓). KID $\times 10^3$.

Dataset / Metric	Emuru (Small)	Imuru (Small)
Karaoke Calligraphy		
FID ↓	50.58	17.19
BFID ↓	47.58	12.51
HWD ↓	3.25	2.38
KID ↓	0.028	0.01
ΔCER ↓	0.65	0.22
Karaoke Typewritten		
FID ↓	35.14	14.04
BFID ↓	14.40	4.52
HWD ↓	2.09	1.24
KID ↓	0.02	0.01
ΔCER ↓	0.57	0.17
RIMES Lines		
FID ↓	120.84	64.35
BFID ↓	95.36	41.68
HWD ↓	3.01	2.14
KID ↓	0.10	0.05
ΔCER ↓	1.01	0.62

typewritten, and measured various metrics for consistency.

Author Contributions

Haojie Cai is the first author corresponding to this paper. Thomas Chen, Scott Fan, and Bob Yao contributes equally to this project. Haojie Cai conceived the main idea and proposed the overall framework, trained the model, and assist other members for code writing. Thomas Chen contributed to VAE training, assisted with evaluation and wrote the VAE section, part of the Experiments section and Conclusion. Scott Fan selected benchmark examples and helped developed evaluation scripts for VATr++, DiffPen, Emuru, and Ref, and contributed to writing the Abstract, Introduction, Related Work, and part of the Experiments section. Shengtao (Bob) Yao built the evaluation and inference pipeline, implemented benchmarking metrics (FID/KID/HWD/CER), and conducted dataset preprocessing and evaluation experiments. All authors reviewed and approved the final manuscript.

Acknowledgments

This work is inspired by and extends the autoregressive framework introduced in Emuru [14] by Pippi et al. from the University of Modena and Reggio Emilia and Google.

References

[1] Emmanuel Augustin, Matthieu Carré, Emmanuèle Grosicki, J.-M. Brodin, Edouard Geoffrois, and Francoise Preteux. RIMES evaluation campaign for handwritten mail processing. In *International Workshop on Frontiers in Handwriting Recognition (IWFHR'06),*, pages 231–235, La Baule, France, 2006. 4

[2] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, and Mubarak Shah. Handwriting transformers, 2021. 5

[3] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans, 2021. 5

[4] Gang Dai, Yifan Zhang, Quhui Ke, Qiangya Guo, and Shuangping Huang. One-shot diffusion mimicker for handwritten text generation, 2024. 1, 2, 5

[5] Brian Davis, Chris Tensmeyer, Brian Price, Curtis Wigington, Bryan Morse, and Rajiv Jain. Text and style conditioned gan for generation of offline handwriting lines, 2020. 2, 5

[6] Ji Gan, Weiqiang Wang, Jiaxu Leng, and Xinbo Gao. Higan+: Handwriting imitation gan with disentangled representations. *ACM Trans. Graph.*, 42(1), 2022. 1, 2, 5

[7] 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, 2018. 5

[8] Florian Kleber, Stefan Fiel, Markus Diem, and Robert Sablatnig. Cvl-database: An off-line database for writer retrieval, writer identification and word spotting. In *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 560–564. IEEE, 2013. 4

[9] Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. Trocr: Transformer-based optical character recognition with pre-trained models, 2022. 5

[10] Urs-Viktor Marti and Horst Bunke. The iam-database: an english sentence database for offline handwriting recognition. *Int. J. Document Anal. Recognit.*, 5(1):39–46, 2002. 4

[11] Konstantina Nikolaidou, George Retsinas, Giorgos Sfikas, and Marcus Liwicki. Diffusionpen: Towards controlling the style of handwritten text generation. In *European Conference on Computer Vision (ECCV)*, 2024. 1, 5

[12] Vittorio Pippi, Silvia Cascianelli, and Rita Cucchiara. Handwritten text generation from visual archetypes, 2023. 1

[13] Vittorio Pippi, Fabio Quattrini, Silvia Cascianelli, and Rita Cucchiara. Hwd: A novel evaluation score for styled handwritten text generation, 2023. 5

[14] Vittorio Pippi, Fabio Quattrini, Silvia Cascianelli, Alessio Tonioni, and Rita Cucchiara. Zero-shot styled text image generation, but make it autoregressive. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7910–7919, 2025. 1, 2, 3, 4, 5, 8

[15] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a

unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. 2, 3, 5

[16] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2, 2019. 2

[17] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation, 2024. 1, 2

[18] Michael Tschannen, Cian Eastwood, and Fabian Mentzer. Givt: Generative infinite-vocabulary transformers, 2024. 1, 2

[19] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018. 2

[20] Bram Vanherle, Vittorio Pippi, Silvia Cascianelli, Nick Michiels, Frank Van Reeth, and Rita Cucchiara. Vatr++: Choose your words wisely for handwritten text generation, 2024. 1, 5

[21] Carmine Zaccagnino, Fabio Quattrini, Vittorio Pippi, Silvia Cascianelli, Alessio Tonioni, and Rita Cucchiara. Autoregressive styled text image generation, but make it reliable. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2026. 1, 4