Open-Set Personalized Handwriting Generation with DiffusionPen

Atul Pandey
Dept. of Computer Science and Engineering
University at Buffalo, SUNY
Professor - David Doermann
atulpriy@buffalo.edu

***Abstract*—The goal of this project was simple to state and harder to deliver: given a few handwriting samples from a person and some arbitrary text, produce that text in their handwriting. I started from DiffusionPen [1], a few-shot latent-diffusion model for handwritten text generation, and spent the semester turning its public release into something that actually works in the open-set setting, where the writer is not in the training data and no writer ID is available at inference. Getting there meant fixing a text-conditioning path that was silently dead, correcting the classifier-free-guidance branch, chasing down a short-word hallucination ("the" kept coming out as "thee"), moving training onto multiple GPUs, and adding the unglamorous glue — crop cleanup, ink-cropping, contrast normalization, word stitching — that a real document needs. I also extended the pipeline to the GNHK in-the-wild color dataset and ran a side-by-side against Emuru [3], a recent autoregressive model. The two fail in different places, and that observation is what motivates *EmuruPen*: a hybrid that keeps DiffusionPen's denoiser but swaps its closed-set style encoder for Emuru's handwriting-domain VAE through one small trained projection. The report walks through both architectures, the fixes, and qualitative results on words, full sentences, and a complete letter generated for an unseen writer, and closes with where this goes next, including font-based synthetic training to cover the digits, punctuation, and rare glyphs that IAM barely contains.**

***Index Terms*—Handwritten text generation, personalized handwriting synthesis, latent diffusion, few-shot style transfer, open-set generation, autoregressive generation, cross-attention masking, distributed training.**

I. Introduction

Two people writing the same word rarely produce anything that looks alike. Slant, spacing, stroke width, pressure, the way letters join, where the baseline sits — all of it varies, and all of it is what makes handwriting recognizably someone's. That is also what makes personalized handwritten text generation (HTG) awkward as a conditional image problem: the model has to keep the content you asked for while borrowing a style it has only seen a handful of times. The concrete target for this project was:

> Given some input text and four to six handwriting samples from a person, generate new words, sentences, and short documents in that person's style — with no writer ID and no per-writer retraining.

This work was carried out as a graduate thesis project at the University at Buffalo, Department of Computer Science and Engineering.

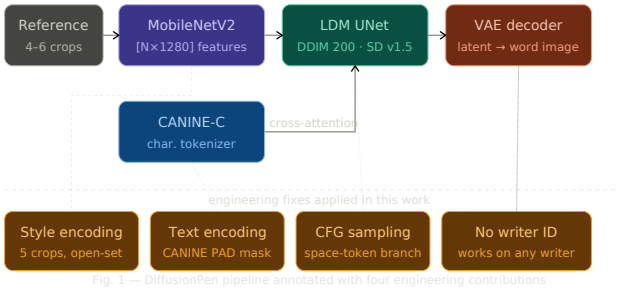


Fig. 1. The DiffusionPen pipeline I started from, with the four fixes from this project marked on it. Reference crops go through a MobileNetV2 style encoder; the target string is tokenized character-by-character by CANINE-C; style is added to the diffusion timestep embedding while text conditions every cross-attention block in the UNet; generation happens in the latent space of a frozen Stable Diffusion v1.5 VAE. The bottom row points at where each fix lives.

I built on DiffusionPen [1], an ECCV 2024 few-shot diffusion model that pairs a MobileNetV2 style encoder, a CANINE-C character text encoder, a latent-diffusion UNet, and a Stable-Diffusion VAE decoder (Fig. 1). The field has moved around a lot — RNN stroke models [5] early on, then GANs [6], [7], then transformers [8], and lately diffusion [1], [9], [10] and an autoregressive comeback in Emuru [3]. DiffusionPen is a good place to stand because the design is clean and the quality is there, but the released code is closed-set: inference wants a training-set writer ID and a JSON lookup of that writer's crops, which is useless if the person you care about was never in IAM.

The thing that makes open-set work is almost a loophole. When you hand the UNet a style tensor derived from images, the writer-class path is bypassed — so you can pass a dummy ID and let the style come entirely from the user's crops. Everything else in this report builds on that. The contributions:

- An **open-set custom-writer pipeline** that generates words, sentences, and short documents from four to six reference crops, no writer ID, no retraining.
- A **text-conditioning fix** that took the model from noise to legible cursive by making the CANINE encoder actually participate in the forward pass.
- A **classifier-free-guidance correction**: the unconditional branch now uses a space-token embedding instead of

`context=None`, so cross-attention stays alive in both passes.

- A **cross-attention mask fix plus a short fine-tune** that propagates the CANINE PAD mask and kills the "the"/"thee" bug.
- A **multi-GPU DDP training setup** with mixed precision and checkpoint handling that survives the prefix mess in the released weights.
- **Document-level post-processing**, a **GNHK color extension**, and an **Emuru comparison** that leads into the proposed *EmuruPen* hybrid.

Sections are in the usual order: related work next, then both architectures (Sec. III), the fixes (Sec. IV), the GNHK extension (Sec. V), qualitative results (Sec. VI), the Emuru comparison and EmuruPen (Sec. VII), and then lessons, limitations, ethics, and conclusions.

II. Background and Related Work

A. How Handwriting Generation Got Here

The earliest neural approaches modeled pen trajectories directly with RNNs [5]. They work, but they need online data with timestamps, which most handwriting collections do not have. Once people moved to offline images, GANs took over for a while. GANwriting [6] showed few-shot writer conditioning was feasible; SmartPatch [14] leaned on patch-level supervision to sharpen things up; VATr [7] dropped the GAN in favor of a transformer driven by a learned archetype book. Diffusion then narrowed the realism gap fast. WordStylist [9] treated HTG as latent diffusion conditioned on a learned writer-ID embedding — clean, but closed-set by design. DiffusionPen [1] swapped that embedding for a style-image encoder trained with a mixed triplet/classification objective, which is the model I use here. One-DM [10] pushed toward one-reference conditioning. Emuru [3] went a different direction entirely and made the whole thing autoregressive, decoding a word image as a sequence of VAE patches.

B. Conditioning, Briefly

Every one of these systems has to be told two things: what to write and whose hand to write it in. Content is usually handled at the character level — WordStylist uses a learned embedding table, DiffusionPen and One-DM reuse the pretrained CANINE-C [2] encoder. Style runs the gamut from one-hot writer IDs (closed-set), through learned writer embeddings, to image-conditioned encoders that can in principle generalize. The last category is the only one that makes open-set inference possible, and it is the one this project lives in.

C. Datasets

IAM [11] is the standard benchmark — word crops of English from many writers, with a fixed train/test split. GNHK [12] is newer and messier: English handwriting photographed in the wild, in color, on noisy backgrounds. I trained primarily on IAM and used GNHK to show the color extension and to stress the open-set path, since no GNHK writer overlaps IAM.

TABLE I
DiffusionPen pipeline components.

Component	Description
Style encoder	MobileNetV2, multi-crop → style vector
Text encoder	CANINE-C, character-level, padded
Latent UNet	SD v1.5 backbone, modified cross-attn
VAE	Frozen SD v1.5 VAE, latent → image
Scheduler	DDIM sampling at inference

III. System Architecture

A. DiffusionPen

Fig. 1 and Table I cover the pieces. Style side: up to five reference crops pass through a MobileNetV2 backbone (trained separately with a triplet loss plus writer classification), the per-image features get mean-pooled, projected through `style_lin`, and added to the diffusion timestep embedding. If you have fewer than five crops, the pipeline just replicates what it has. Text side: CANINE-C tokenizes the target string character-wise, pads it to a fixed length, and the embeddings become the `context` for every cross-attention block in the UNet. Generation runs in the SD v1.5 VAE latent space, guided by classifier-free guidance [13] that trades off how faithfully the text comes out against how strongly the style shows. The detail that matters most for this whole project: once a style extractor is supplied at inference, the UNet ignores the writer class label. That is the entire reason open-set works without touching the architecture.

B. EmuruPen, the Proposed Hybrid

The Emuru comparison in Section VII is what made me sketch this. The short version is that the two models break in different places, so a hybrid is worth trying. *EmuruPen* (Fig. 2) keeps DiffusionPen's UNet, CANINE encoder, and VAE decoder untouched and replaces only the closed-set MobileNetV2 style encoder with Emuru's handwriting-domain VAE encoder followed by a small trained projection. If z_E is the Emuru style latent from the crops, the projection learns $s_D = Wz_E + b$ so that s_D lands in the style space DiffusionPen already expects. Only that projection trains; the Emuru VAE, CANINE, the VAE decoder, and the DiffusionPen UNet all stay frozen. That keeps the training cost in the hours-not-days range and removes any catastrophic-forgetting risk. The bet is not that the hybrid fixes everything — it is that MobileNetV2 was trained to tell writers apart, which is not the same thing as encoding stroke style, whereas a handwriting VAE latent should carry slant, spacing, and stroke texture more directly.

IV. Engineering Contributions

A. The Text-Conditioning Path Was Dead

The first sign something was wrong: training ran fine, loss went down, and the samples were gray noise that did not respond to the text at all. It took a while to find because nothing errored. The text-encoder weights loaded under `module.*` keys (left over from a `DataParallel` save)

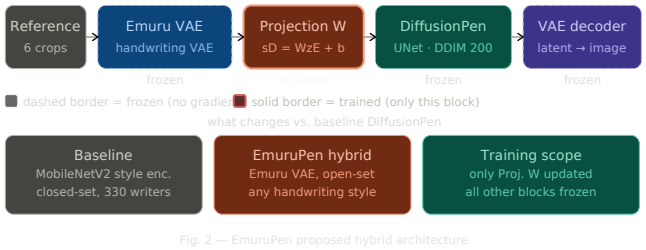


Fig. 2. Proposed EmuruPen hybrid. Reference crops are encoded by Emuru's frozen handwriting-domain VAE, mapped by a small trained projection W into DiffusionPen's style space, and fed to the frozen DiffusionPen UNet and VAE decoder. The projection is the only thing that trains.

while the runtime model exposed bare keys, so CANINE was never actually updated and contributed no gradient. Fixing the key mapping and confirming CANINE was in the forward pass is what turned the model into an HTG system at all. Everything downstream is meaningless without it, so this was the fix that mattered most even though it is the least interesting to describe.

B. The CFG Branch Was Toggling, Not Contrasting

The original sampler passed `context=None` for the unconditional branch. That does not give you blank-text conditioning — it skips cross-attention entirely in that branch while the conditional branch still uses it, so guidance was contrasting "cross-attention on" against "cross-attention off" rather than "no text" against "target text." Switching the unconditional branch to a space-token embedding keeps the path identical across both passes, so guidance amplifies the difference you actually want.

C. Moving Training onto Multiple GPUs

The single-GPU `DataParallel` loop was too slow to iterate on. I rebuilt it around `torch.distributed` with one process per GPU and `DistributedDataParallel`, AMP mixed precision on the UNet forward, and a checkpoint loader that adds or strips the `module.` prefix depending on what it is loading — necessary because the released checkpoint was saved under `DataParallel` and nothing else would resume cleanly. On the cluster, a couple of GPUs sat in a persistent error state, so those got excluded with `CUDA_VISIBLE_DEVICES` and the real runs went on the healthy node. None of this is novel, but it is the difference between a training schedule that finishes and one that does not. Table II lists the setup.

D. The "Thee" Bug

This one cost me the most time. Long enough into training, normal words came out clean — `"hello"`, `"University"`, `"Buffalo"` all fine — but short words were consistently broken, and the cleanest tell was that `"the"` rendered as `"thee"` (or some other four-character smear) in every writer style I tried.

The cause is in how the text gets padded. CANINE-C works at the character level, and the pipeline pads every input to a fixed token length, so `"the"` is three real embeddings followed by a long tail of PAD. CANINE hands back an attention mask that says which positions are PAD, but the released UNet has `mask = None` hardcoded inside its CrossAttention, so the softmax spreads weight over every position including the padding. For a three-letter word the PAD tail outvotes the real characters and the model drifts toward the longer-word shapes it saw more often in training.

The obvious idea — just pass the mask at inference — does not work. The UNet was trained with no mask, so masking only at test time is a train/test mismatch and the output visibly degrades. The fix has to happen in training. I fine-tuned the trained EMA checkpoint with the mask propagated end-to-end (I call this checkpoint `finetuned_mask`), leaving everything else alone. After that, `"the"` is three characters and short words behave across reference writers. Fig. 3 is the before/after; Table III is the summary.

TABLE II
TRAINING AND INFRASTRUCTURE CONFIGURATION.

Item	Working setup
Primary model	DiffusionPen latent diffusion
Datasets	IAM; GNHK color extension
Hardware	A5000 GPU cluster via SLURM
Parallelism	PyTorch DDP (NCCL)
Precision	AMP mixed precision
Optimizer	AdamW
Sampling	DDIM, CFG-guided
Checkpoint	EMA weights preferred for inference

Fig. 3. The "Thee" bug. Before the context-mask fine-tune, "the" grows a fourth character because cross-attention is also attending to PAD. After a short fine-tune with the CANINE mask propagated through cross-attention, it attends only to real tokens and "the" is three characters again.

E. Open-Set Custom-Writer Inference

The released scripts insist on a real IAM writer ID and pull that writer's crops from a JSON file, which caps you at the training writers. The replacement, `sample_custom_writer.py`, drops that restriction. A `crop_style_images.py` utility takes arbitrary handwriting images, segments them into words by ink bounding box, whitens the background, runs a light median filter, and saves clean crops. Those crops go through the trained style encoder, get mean-pooled (replicated if there are fewer than five), and projected through `style_lin`. A dummy writer ID is passed but ignored, since the style extractor is active, and CFG uses the space-token branch from earlier. For multi-word input each word is generated on its own, ink-cropped to drop the fixed-width padding, contrast-normalized, and composited into lines.

TABLE III
THE "THEE" BUG: CAUSE AND MITIGATION.

Aspect	Details
Symptom	"the" → "thee"; short words drift
Trigger	CANINE PAD-token leakage
Root cause	`mask=None` in CrossAttention
Inference-only fix	Fails (train/test mismatch)
Working fix	Short fine-tune with mask propagated
Inference flag	`--use_context_mask`
Check	"the" renders as three characters

The ink-crop step sounds trivial but is not optional — the UNet always emits a fixed-size frame, so without it every word occupies the same width and you get a rigid grid instead of text that flows. One practical note that took some trial and error: crop quality feeds straight into stroke width and slant, so tight, aspect-preserving crops with clean backgrounds make a visible difference.

V. GNHK COLOR EXTENSION

DiffusionPen's paper only shows GNHK [12] qualitatively, so I wired up full color training. That meant a custom `utils/GNHK_dataset.py` that reads the GNHK manifest and yields image/text/writer tuples without converting to grayscale, a style encoder trained on the GNHK writer distribution, color-aware augmentation that leaves ink hue alone while normalizing background brightness, and separate color inference scripts. Because no GNHK writer appears in IAM training, this path also doubles as a second open-set stress test.

VI. QUALITATIVE RESULTS

A. Setup and the Reference Style

Unless I say otherwise, results use the `finetuned_mask` EMA checkpoint with CFG-guided DDIM sampling and a fixed seed, through the open-set path from Section IV-E. Every open-set result below borrows the single writer style in Fig. 4 — the four-to-six reference crops all come from that one sample — so each generated word, sentence, and letter should be read against that target.

B. One Style, Many Words — and Many Styles, One Word

First a sanity check that the model spans the IAM writers: the same word `"hello"` across eight writer IDs (Fig. 5). Quality tracks how much training data each writer had. IDs around 50, 100, and 338 come out decent and legible; the extreme IDs degrade because there is not much data there. I used writer 50 as a clean default for IAM-conditioned demos.

C. A Full Sentence

For an end-to-end stress test I generated a long sentence — *"The University at Buffalo quickly maximizes diverse knowledge zones with jazzy vibrant research pursuits"* — in the reference style of Fig. 4. The output (Fig. 6) holds a consistent cursive slant the whole way across. The short word

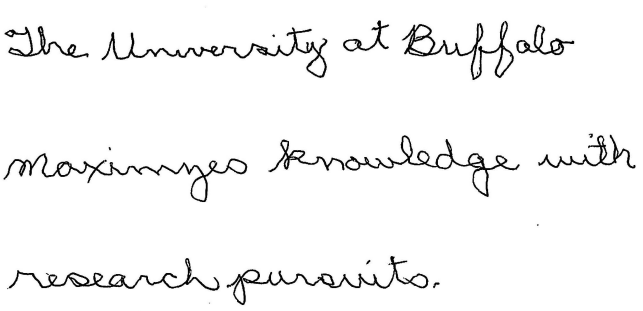


Fig. 4. The reference style. Every open-set result in this section imitates this writer; the style crops are taken from this single sample, and the writer does not appear in IAM training.

Fig. 5. Same word "hello" across eight IAM writer IDs, identical text conditioning. Writer IDs around 50, 100, and 338 look decent; the extremes show artefacts where the IAM distribution is thin.

"The" is correct now, and the longer words stay readable. A few visually awkward tokens still come out stylized, which I would call style variation rather than a spelling failure.

D. Word-Level, Next to Emuru

Fig. 7 lines up isolated words across the broken baseline, the fixed DiffusionPen, and Emuru, under two shared style references. Fixed DiffusionPen is the bolder, more expressive one and gets short words right after the mask fine-tune; Emuru is lighter and cleaner on a white background but can go soft on long words.

E. A Full Sentence in the Reference Hand

Fig. 8 is a representative sentence-level result in the reference style: the same line generated through the open-set pipeline. It is clean and the slant is consistent end to end, which is the behavior I wanted out of the mask fine-tune at sentence scale. Run the same line through Emuru and you get something tidier on the background but flatter in character — the same trade-off the word grid shows, just at sentence length.

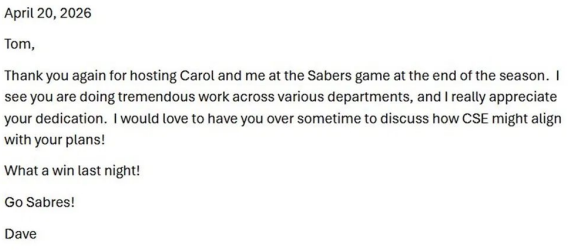
F. The Letter

The realistic test was converting a short typed letter (Fig. 9) — date line, salutation, two short paragraphs, sign-off, signature — into handwriting using only the reference crops from Fig. 4. DiffusionPen (Fig. 10) keeps the layout, holds a consistent cursive weight, wraps correctly, and even gets a natural pen-lift on the signature; a few words drift in scale and some rare proper nouns come out partly garbled. Emuru (Fig. 11) gives a cleaner, lighter version on a native white background but with less of the writer's character. Both used the identical six crops from the one sample in Fig. 4.

Fig. 6. Full-sentence generation with `finetuned_mask`, style from the reference in Fig. 4. It includes the previously broken short word "The" (now fine) next to longer words, with a consistent slant across the line.

Fig. 7. Word-level comparison. Columns: DiffusionPen baseline (no mask), then fixed DiffusionPen and Emuru under `ref00`, then the same pair under `ref02`. The mask fine-tune is most obvious on "the", "at", and "of".

Fig. 8. Sentence-level open-set generation in the reference style of Fig. 4. Consistent slant and spacing across the whole line; the Emuru contrast is discussed in the text rather than shown, since the failure modes are easier to read at word scale.



April 20, 2026

Tom,

Thank you again for hosting Carol and me at the Sabers game at the end of the season. I see you are doing tremendous work across various departments, and I really appreciate your dedication. I would love to have you over sometime to discuss how CSE might align with your plans!

What a win last night!

Go Sabres!

Dave

Fig. 9. The source typed letter for the document case study. Both models render this same content in the reference writer's hand.

G. Color and Category Coverage

Separate from the grayscale open-set pipeline, the GNHK color extension renders colored ink across content/style categories — in- and out-of-vocabulary words, seen and unseen styles, and special strings like digits and punctuation. Fig. 12 is a color showcase of those categories. This is an extra capability of the extended pipeline, not part of the single-reference open-set results above, and it regenerates on demand for arbitrary words and colors.

VII. Comparison with Emuru and the EmuruPen Hybrid

A. Why Emuru

I set Emuru [3] up as an external baseline in its own conda environment on the same cluster, sharing the style crops so both models see the exact same references. Emuru is a different animal — an autoregressive transformer that decodes a word image as a sequence of VAE patches, pretrained on synthetic font crops and fine-tuned for handwriting. I picked it because it ships public weights, a working inference snippet, and a CVPR 2025 paper, so a fair comparison did not require reproducing anyone's training.

B. They Break Differently

Across words, sentences, and the letter, neither model wins outright (Table IV). DiffusionPen is stronger on style fidelity and on long or rare words; Emuru is cleaner on background, more reliable on very short words, and more consistent in scale. Pushing Emuru on the structured letter exposed three categorical weaknesses: digits and numbers, because its VAE patches were pretrained on alphabetic data; punctuation, which is narrower than a patch and gets hallucinated into a nearby cursive stroke; and vertical placement, which wanders with descenders because there is no baseline head, so aligning to ruled lines is hard. DiffusionPen's fixed latent geometry gives it stable baselines and full Unicode through CANINE, but it pays for that with noisier backgrounds and the short-word problem I had to fine-tune away. Those two profiles are close to complementary, which is exactly the opening EmuruPen aims at: take style from Emuru's handwriting VAE, keep DiffusionPen's controllable, text-faithful denoiser.

VIII. Lessons from the Build

A few things from this generalize past this one model. The biggest: train/test consistency is not negotiable. Both of the worst bugs — masking only at inference, and toggling cross-attention via `context=None` in one CFG branch — degraded output silently, with no error anywhere, and the only reason I found them was comparing the distribution of the conditioning signals between training and inference. Second, EMA weights beat the raw weights at every checkpoint I sampled, for the price of one extra parameter copy and a slow moving average. Third, fixed-spatial-dimension latent diffusion is brittle to inference-time shape changes — feeding the UNet unseen latent widths produced garbled output even when the arithmetic checked out, so I always pad or crop to the trained shape and handle layout afterward. And finally, open-set inference can be cheap when the style conditioner is image-based: here the real work was the data pipeline, not the model, because there was already a code path that ignored the writer ID.

IX. Limitations and Ethics

On the technical side: generating each word independently loses sentence-level scale, so words occasionally come out oversized. Rare proper nouns outside the training vocabulary degrade visibly. There is an `n/m` confusion that comes from how close those characters sit in the released CANINE space — fixing it properly means retraining CANINE with a

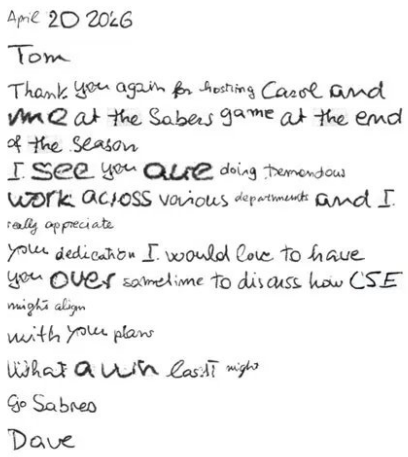


Fig. 10. DiffusionPen's open-set version in the reference style of Fig. 4: `finetuned_mask`, EMA weights, context-mask fix, ink-crop, contrast normalization. Strong style match and a believable signature; a few words vary in scale and rare proper nouns suffer.

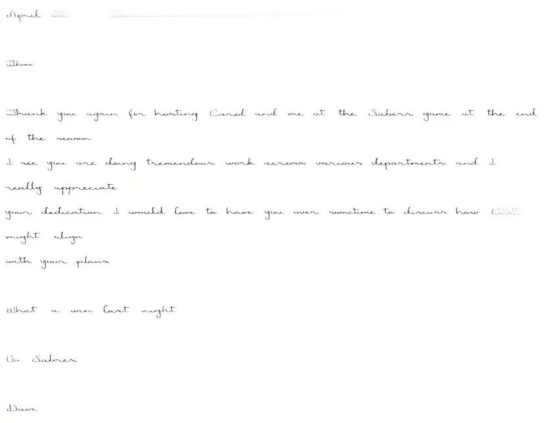


Fig. 11. Emuru's version of the same letter with the same six crops. Cleaner and lighter on a native white background, but the style identity is weaker.

Fig. 12. Color and category showcase from the GNHK extension across in/out-of-vocabulary words, seen/unseen styles, and special strings. Generated independently of the single-reference open-set results.

TABLE IV
DIFFUSIONPEN VS. EMURU: WHERE EACH ONE BREAKS.

Aspect	DiffusionPen	Emuru
Short words	Fixed via mask fine-tune	Clean (AR exact)
Long/rare words	Mostly correct	Can drift
Style fidelity	Bold, expressive	Light, less distinct
Background	Needs post-proc	Clean white, native
Open-set	Patched via pipeline	Native, zero-shot
Punctuation	Not rendered	Native, partial artifacts
Scale/baseline	Per-word variable	Consistent x-height
Controllability	CFG-controllable	Fixed autoregressive

contrastive character objective. Punctuation and digits are not rendered natively because DiffusionPen's training crops do not contain them, and DiffusionPen still leans on background post-processing. Emuru is the inverse: clean, but it drifts on long or rare words and has no baseline control.

On ethics: this technology is genuinely useful for data augmentation, accessibility, education, historical-document restoration, and creative tools, but the same capability enables impersonation and forgery. Any real deployment should label synthetic output, attach watermarking or provenance metadata, only use style references with consent, and refuse to produce signatures or legal and financial documents in a real person's hand without authorization.

X. FUTURE WORK

Three things are next. First, finish EmuruPen — train the projection layer from Section III with everything else frozen, so the change is one small matrix that trains in hours. Second, font-based synthetic training: render synthetic word crops from a handwriting-style font catalog (Lucida Handwriting, Segoe Script, Bradley Hand, Ink Free, Monotype Corsiva, the italic families) at several sizes with mild rotation and scale jitter, treat each font as a synthetic "writer" during style-encoder pretraining, then adapt back to real IAM/GNHK crops so the output does not end up looking like a typeface. The point is to finally cover the digits, punctuation, and uppercase

glyphs that IAM barely has, and to widen the range of stroke widths and slants the style encoder ever sees; fonts would be used under their applicable licenses. Third, a small regression head predicting baseline, ascender, and descender per word, which should fix the descender-dependent vertical wobble and make ruled-paper placement reliable. The longer-horizon *StyleFusion-Diffusion* idea — replacing MobileNetV2 with a hybrid CNN + state-space + local-transformer + wavelet encoder — is scoped but I have not started it.

XI. Conclusion

This project took DiffusionPen from a closed-set paper artifact to a working open-set handwriting generator. The progress that mattered came from unglamorous places: getting text conditioning to actually fire, fixing the CFG branch, propagating the cross-attention mask to kill the "thee" bug, moving training onto multiple GPUs, and adding the document-level glue. The system now produces readable cursive words, sentences, and a full letter for a writer it never trained on, from a handful of crops. Putting it next to Emuru made the complementary-strengths picture obvious, and that is what the scoped EmuruPen hybrid is built to exploit. The near-term work — finishing EmuruPen and adding font-based synthetic training for digits, punctuation, and rare glyphs — follows directly from what is here. All code changes and the GNHK extension stay open-source under the original project's MIT license.

Acknowledgment

Thanks to the DiffusionPen authors [1] for releasing weights and training code, to the Emuru authors [3] for a public release that was actually runnable, and to the University at Buffalo CCR for cluster access. This thesis was supervised within the Department of Computer Science and Engineering at the University at Buffalo.

References

[1] K. Nikolaidou, G. Retsinas, G. Sfikas, and M. Liwicki, "DiffusionPen: Towards controlling the style of handwritten text generation," in *Proc. ECCV*, 2024.

[2] J. H. Clark, D. Garrette, I. Turc, and J. Wieting, "CANINE: Pre-training an efficient tokenization-free encoder for language representation," *Trans. Assoc. Comput. Linguistics*, vol. 10, pp. 73–91, 2022.

[3] V. Pippi, F. Quattrini, S. Cascianelli, A. Tonioni, and R. Cucchiara, "Zero-Shot Styled Text Image Generation, but Make It Autoregressive," in *Proc. CVPR*, 2025.

[4] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. CVPR*, 2022, pp. 10684–10695.

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

[6] L. Kang, P. Riba, Y. Wang, M. Rusiñol, A. Fornés, and M. Villegas, "GANwriting: Content-conditioned generation of styled handwritten word images," in *Proc. ECCV*, 2020, pp. 273–289.

[7] V. Pippi, S. Cascianelli, and R. Cucchiara, "Handwritten text generation from visual archetypes," in *Proc. CVPR*, 2023, pp. 22458–22467.

[8] A. K. Bhunia *et al.*, "Handwriting transformers," in *Proc. ICCV*, 2021, pp. 1086–1094.

[9] K. Nikolaidou *et al.*, "WordStylist: Styled verbatim handwritten text generation with latent diffusion models," in *Proc. ICDAR*, 2023, pp. 384–401.

[10] G. Dai, Y. Zhang, Q. Ke, Q. Guo, and S. Huang, "One-Shot Diffusion Mimicker for Handwritten Text Generation," in *Proc. ECCV*, 2024.

[11] U.-V. Marti and H. Bunke, "The IAM-database: an English sentence database for offline handwriting recognition," *Int. J. Document Anal. Recognit.*, vol. 5, no. 1, pp. 39–46, 2002.

[12] A. W.-C. Lee, J. Chung, and M. Lee, "GNHK: A dataset for English handwriting in the wild," in *Proc. ICDAR*, 2021, pp. 399–412.

[13] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[14] A. Mattick, M. Mayr, M. Seuret, A. Maier, and V. Christlein, "SmartPatch: Improving handwritten word imitation with patch discriminators," in *Proc. ICDAR*, 2021, pp. 268–283.