TrInk: Ink Generation with Transformer Network

Anonymous ACL submission

Abstract

In this paper, we propose TrInk, a Transformer-based model for ink generation, enabling parallel training and better capturing global dependencies. To better facilitate the alignment between the input text and generated stroke points, we introduce scaled positional embeddings and a Gaussian memory mask in the cross-attention module. Additionally, we design both subjective and objective evaluation pipelines to comprehensively assess the legibility and style consistency of the generated handwriting. Experiments demonstrate that our Transformer-based model achieves a 35.56% reduction in character error rate (CER) and an 29.66% reduction in word error rate (WER) on the IAM-OnDB dataset compared to previous methods. We provide an demo page with handwriting samples from TrInk and baseline models at: `https://akahello-a11y.github.io/trink-demo/`

1 Introduction

Handwriting synthesis is the task of automatically generating realistic handwritten text from digital inputs. Automatic handwritten text generation can support a wide range of applications, including digital note-taking, educational tools, and generating training data to improve optical character recognition (OCR) systems (Li et al., 2023; Fujitake, 2024; Yeleussinov et al., 2023). However, due to the complex temporal dynamics and variability inherent in human handwriting, generating high-quality handwritten samples still faces challenges.

Deep learning-based handwritten text generation approaches can be roughly divided into image-based offline and stroke-based online methods. Offline handwriting synthesis focuses on producing a static image of handwritten text (Chang et al., 2018; Alonso et al., 2019; Kang et al., 2020; Haines et al., 2016). In contrast, online handwriting synthesis (also called ink generation) aims to generate a time-ordered sequence of pen-tip coordinates along with pen-state indicators (e.g., pen-up and pen-down), thereby reconstructing the full dynamic trajectory of the writing process. Compared with offline approaches, online handwriting synthesis (ink generation) outputs lightweight stroke vectors that can be rendered at any resolution, making them easy to transmit and display consistently across diverse devices. In this work, we focus on ink generation to generate handwriting samples that are stylistically consistent and highly legible.

Recent research on ink generation has predominantly relied on sequential models (Graves, 2013; Aksan et al., 2018; Chang et al., 2022). Graves (2013) leverages an LSTM-based network to predict future stroke points from the current pen position based on the given text. Aksan et al. (2018) introduces a conditional variational RNN that improves the model's capacity to capture handwritten digits. Building on Graves (2013), Chang et al. (2022) introduces style equalization method, equipped with a style encoder to explicitly model the style information.

While these approaches have demonstrated promising results, they remain fundamentally constrained by the sequential nature of recurrent architectures, which limits their ability to model long-range dependencies and hinders parallel training. Furthermore, alignment between the input text and generated strokes often requires careful design, such as attention windowing. Motivated by the success of Transformer (Vaswani et al., 2017) in various generative tasks (Li et al., 2019; Chen et al., 2020; Ding et al., 2021; Chang et al., 2023; Ma et al., 2024), we propose TrInk (Transformer for Ink Generation), a fully attention-based model tailored for ink generation. The encoder ingests the target text sequence, through multi-head self-attention, yields a contextual content representation for every character. The decoder receives the character representations together with the previous generated stroke points, and applies multi-head self- and cross-attention to compute decoder hidden states. These decoder hidden states are fed into a mixture-density network, which outputs a Gaussian mixture distribution from which the next pen offset and pen state are sampled. To improve alignment between the text and the stroke sequence, we apply a Gaussian memory mask to the cross-attention matrix, constraining the decoder's focus to progress strictly left-to-right along the encoded text as strokes are generated. We apply a learnable scale to sinusoidal positional embeddings to handle differences between text and ink points. Our main contributions are summarized as follows:

1. To the best of our knowledge, TrInk is the first to employ a Transformer encoder–decoder architecture for ink generation.

2. TrInk introduces a Gaussian memory mask to ensure the generated ink points follows the natural writing order, and a scale factor for the position embeddings to model the differing charateristics of the text and the ink points.

3. Our experimental results show that our proposed TrInk yields substantially higher legibility, particularly on long text, than the previous.

2 Method

The Framework of TrInk comprises two main components: an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$. Encoder $\mathcal{E}$ aims to convert the input text into a sequence of vectors, where each vector represents the meaning of a token in its surrounding context. Decoder $\mathcal{D}$ aims to take the encoder's content vectors along with the previous stroke points and, at each time step, predict Gaussian distributions for the next pen coordinate and the stroke-end probability.

2.1 Encoder

From Fig. 1, we transform each character of the input "his operation was" (including the blank spaces) into a one-hot vector, yielding $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_T]$, and $\boldsymbol{h}_i \in \mathbb{R}^{|V|}$, where $|V|$ denotes the vocabulary size and T denotes the number of tokens in the text. After the linear projection and positional encoding, we obtain the Transformer encoder input $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T], \quad \boldsymbol{x}_i \in \mathbb{R}^d$, where d is the hidden-state dimension of the Transformer encoder. The high-dimensional text representation $\boldsymbol{C}$ generated by the encoder is then fed into the Transformer decoder to serve as the memory for cross-attention.

2.2 Scaled Positional Encoding

To account for the sequential order of both text tokens and stroke points in ink generation, we inject absolute position information using sinusoidal positional embeddings, as defined below:

$$\begin{aligned} PE(pos, 2i) &= \sin\Big(\frac{pos}{10000^{\frac{2i}{d}}}\Big), \\ PE(pos, 2i+1) &= \cos\Big(\frac{pos}{10000^{\frac{2i}{d}}}\Big), \end{aligned} \tag{1}$$

where pos denotes the position index and d is the model's hidden dimension. Because the encoder's domain is text and the decoder's domain is stroke points, using fixed positional embeddings alone cannot properly capture the differing scales and characteristics of these two inputs. We therefore employ these sinusoidal positional embeddings with trainable weights so that the embeddings can adaptively fit the output scales of both the encoder's and the decoder's linear layers, following (Li et al., 2019), as shown in Eq. 2

$$\boldsymbol{x}_i = \boldsymbol{f}_{\boldsymbol{\mathcal{E}}}(\boldsymbol{h}_i) + \alpha PE(i) \tag{2}$$

where α is the trainable weight. A similar formulation with a separate scaling parameter is applied in the decoder.

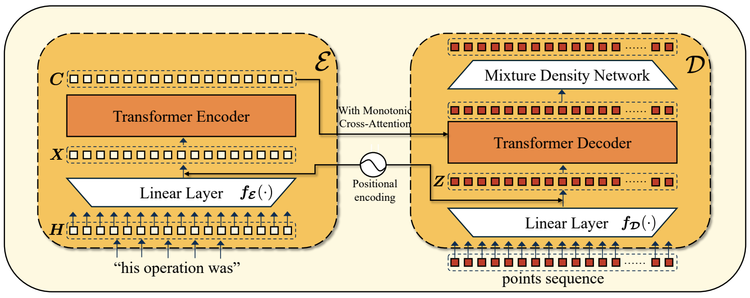
2.3 Decoder with Monotonic Cross-Attention

Each stroke point is represented as a 3-dimensional vector $[\Delta x, \Delta y, s]$, where Δx and Δy are the offsets along the x- and y-axes, and $s \in \{0, 1\}$ denotes the pen state (0 = pen-down, 1 = pen-up). After the linear projection and positional encoding, we obtain $\boldsymbol{Z} = [\,\boldsymbol{z}_1, \boldsymbol{z}_2 \ldots, \boldsymbol{z}_L\,]$, where $\boldsymbol{z}_i \in \mathbb{R}^d$ and L is the number of stroke points.

Given the stroke embedding sequence $\boldsymbol{Z}$, the Transformer decoder first applies masked self-attention to enforce autoregressive dependencies among stroke points. It then performs cross-attention with the text representations $\boldsymbol{C}$ to align each generated stroke with the corresponding text content. To ensure that each decoding step attends to the most relevant region of the input text, we introduce a Gaussian-shaped cross-attention mask. For each decoder time step $t \in [1, L]$, we define its corresponding attention center μ_t on the text sequence $\boldsymbol{C}$ as:

$$\mu_t = \min\left(\frac{t}{r},\, T-1\right). \tag{3}$$

r denotes the average number of stroke points per character, estimated from the training data. Gaussian function centered at μ_t is used for each decoder



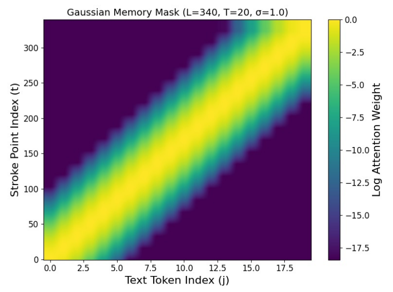


Figure 1: The proposed TrInk framework (left) and Gaussian memory mask (right).

step t, ensuring higher attention weights for text positions near the center and lower weights for distant ones. For each decoder step t and encoder position $j \in [1, T]$, the attention weight is defined as:

$$\boldsymbol{A}_{t,j} = \exp\left(-\frac{(j-\mu_t)^2}{2\sigma^2}\right), \tag{4}$$

where σ is a controllable parameter that determines the sharpness of the Gaussian distribution. We apply the logarithm to the $\boldsymbol{A}_{t,j}$ to obtain the cross-attention mask $\boldsymbol{M}_{t,j} = \log(\boldsymbol{A}_{t,j})$, allowing it to be added directly to the attention logits before the softmax, similar to standard attention masking in Transformers. This log-space formulation helps maintain numerical stability by avoiding extremely small values in the Gaussian tails. Gaussian cross-attention mask ensures that attention shifts monotonically from left to right across the input text. At each decoding step, encoder positions j near the center μ_t receive higher attention scores, while positions farther from μ_t are gradually suppressed, as illustrated in the right side of Fig. 1.

After the Transformer decoder, we adopt a mixture density network (Bishop, 1994) to model the output distribution like (Graves, 2013). Each decoder output token is mapped to a $(6K+2)$-dimensional vector, where K is the number of Gaussian mixture components. The output includes the parameters of K bivariate Gaussian distributions—mixture weights, means, standard deviations, and correlation coefficients—along with two additional scalars: an end-of-stroke probability and a sequence-level stop probability. During inference, we sample the pen-point coordinate at each time step from the predicted mixture of Gaussians, as in Graves (2013).

We adopt the same training objective as Graves (2013), minimizing the negative log-likelihood of the ground-truth trajectory. The loss function comprises three components: a mixture density loss for predicting stroke offsets, a Bernoulli loss for the end-of-stroke indicator, and a Bernoulli loss for determining sequence termination.

3 Experiments

3.1 Datasets

The original training dataset was collected from over 5,000 writers and was initially used for online handwriting recognition tasks. However, we observed that some ink samples were overly cursive that benefit the training of robust recognition systems but are not suitable for generating realistic handwriting. To tackle this issue, we leveraged an OCR engine to filter the dataset, selecting a curated subset of 600,000 high-quality ink samples, optimally prepared for handwriting generation. For evaluation, we use the IAM-OnDB (Liwicki and Bunke, 2005) test set, which is the most popular dataset for handwritten text recognition. We divide the test set into three subsets: full, short, and long. The long subset comprises samples exceeding 40 characters, while the short subset includes those with fewer than 10 characters.

3.2 Evaluation Pipeline

Inspired by text-to-speech evaluation protocols, we divide our evaluation into subjective and objective assessments. For the subjective evaluation, human raters score the generated handwriting samples based on legibility and stylistic consistency, each on a 1–5 scale, with higher scores indicating better quality. For the objective evaluation, we utilize a state-of-the-art OCR model (Li et al., 2023) to recognize the generated samples, comparing the outputs to the ground-truth text to compute the Character Error Rate (CER) and Word Error Rate (WER) as quantitative measures of legibility.

3.3 Main Results

Table 1 presents the objective evaluation results on the IAM-OnDB test set. Our pipeline employs a

Method	**IAM-OnDB**				
	Full		**Long Texts**		**Short Texts**
	CER, % ↓	**WER, % ↓**	**CER, % ↓**	**WER, % ↓**	**CER, % ↓**
AlexRNN (Graves, 2013)	9.0	53.6	15.6	48.6	27.8
AlexRNN (Top-k)	8.8	42.6	10.0	40.1	18.3
Style Equalization (Chang et al., 2022)	8.7	47.4	11.6	46.0	24.4
Style Equalization (Top-k)	6.5	40.0	8.7	39.7	18.2
TrInk	8.5	43.2	9.3	43.3	21.9
TrInk (Top-k)	**5.8**	**37.7**	**6.8**	**36.3**	**17.6**

Table 1: Comparison of different methods on three test sets (k = 5).

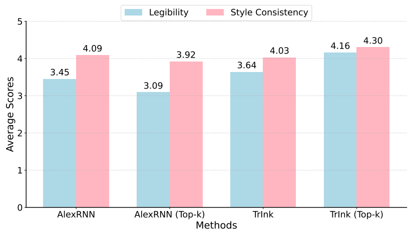


Figure 2: Subjective Evaluation of Handwriting Quality Across Methods.

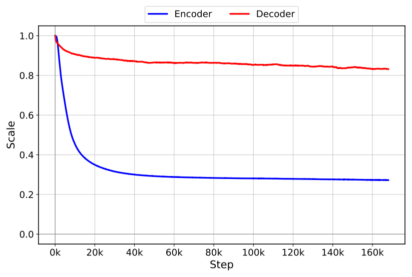


Figure 3: Trainable Positional Encoding Weights (Encoder and Decoder) Over Training.

Table 2: Effectiveness of Gaussian Memory Mask in Cross-Attention Alignment.

Method	**Gaussian memory mask**	**CER, % ↓**
TrInk	✗	70.1
	✓	8.5

Top-k sampling strategy where k candidate handwriting samples are first generated, then ranked by TrOCR according to their CER scores against the ground-truth text, with the optimal sample (minimum CER) selected as final output. As shown in Table 1, TrInk consistently outperforms all baselines, including both the standard AlexRNN and its variant with style equalization, across all evaluation settings. TrInk with the Top-k strategy achieves the best performance, with a 35.56% reduction in CER and a 29.66% reduction in WER on the full test set compared to AlexRNN. For long-text generation, TrInk shows even greater improvements, with a 56.41% reduction in CER and a 25.31% reduction in WER compared to AlexRNN. These reductions further highlight the effectiveness of TrInk.

Figure. 2 presents the results of our subjective experiments. The final scores for each method were the average ratings for two metrics: style consistency and legibility. From Figure. 2, we can observe that TrInk outperforms AlexRNN in both metrics, further validating the effectiveness of TrInk.

3.4 Ablation Study

Figure. 3 illustrates the changes of the two trainable weights in the positional encoding for the encoder and decoder during training. Notably, these weights converge to different values, indicating a significant discrepancy, showing that adopting fixed positional embeddings may fail to capture the differing scales and characteristics of the two input modalities. We also investigated the effect of the Gaussian memory mask on the IAM-OnDB full test set, as shown in Table 2. Removing the Gaussian memory mask leads to a significant drop in the legibility of the generated samples. This is mainly because the model fails to learn proper alignment between the text and stroke points without the guidance of the mask.

4 Conclusion

This paper presents TrInk, the first ink-generation model built on a Transformer encoder–decoder architecture. To achieve precise alignment between input text and generated stroke sequences, we introduce a scaled positional encoding with learnable weights and a Gaussian memory mask. We also devise both subjective and objective evaluation protocols for ink generation. Experimental results demonstrate that TrInk markedly outperforms traditional LSTM-based approaches, producing handwriting samples with superior style consistency and legibility.

5 Limitations

Despite the promising results, TrInk has two limitations. First, training our Transformer-based architecture requires considerable computational resources. The increased model capacity and parallel attention mechanisms lead to higher memory consumption and longer convergence time compared to lightweight RNN-based alternatives.

Second, our current experiments are conducted solely on English handwriting datasets. As handwriting conventions vary significantly across scripts and languages (e.g., cursive structures in Arabic, character-based layouts in Chinese), it remains unclear how well our model generalizes to multilingual settings. Developing a unified ink generation framework capable of generating stylistically consistent samples across multiple languages would be an important direction for future work.

References

Emre Aksan, Fabrizio Pece, and Otmar Hilliges. 2018. Deepwriting: Making digital ink editable via deep generative modeling. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–14.

Eloi Alonso, Bastien Moysset, and Ronaldo Messina. 2019. Adversarial generation of handwritten text images conditioned on sequences. In *2019 international conference on document analysis and recognition (ICDAR)*, pages 481–486. IEEE.

Christopher M Bishop. 1994. Mixture density networks.

Bo Chang, Qiong Zhang, Shenyi Pan, and Lili Meng. 2018. Generating handwritten chinese characters using cyclegan. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 199–207. IEEE.

Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, and 1 others. 2023. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*.

Jen-Hao Rick Chang, Ashish Shrivastava, Hema Koppula, Xiaoshuai Zhang, and Oncel Tuzel. 2022. Style equalization: Unsupervised learning of controllable generative sequence models. In *International Conference on Machine Learning*, pages 2917–2937. PMLR.

Mingjian Chen, Xu Tan, Yi Ren, Jin Xu, Hao Sun, Sheng Zhao, Tao Qin, and Tie-Yan Liu. 2020. Multispeech: Multi-speaker text to speech with transformer. *arXiv preprint arXiv:2006.04664*.

Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and 1 others. 2021. Cogview: Mastering text-to-image generation via transformers. *Advances in neural information processing systems*, 34:19822–19835.

Masato Fujitake. 2024. Dtrocr: Decoder-only transformer for optical character recognition. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 8025–8035.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Tom SF Haines, Oisin Mac Aodha, and Gabriel J Brostow. 2016. My text in your handwriting. *ACM Transactions on Graphics (TOG)*, 35(3):1–18.

Lei Kang, Pau Riba, Yaxing Wang, Marçal Rusinol, Alicia Fornés, and Mauricio Villegas. 2020. Ganwriting: content-conditioned generation of styled handwritten word images. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 273–289. Springer.

Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2023. Trocr: Transformer-based optical character recognition with pre-trained models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 13094–13102.

Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. 2019. Neural speech synthesis with transformer network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6706–6713.

Marcus Liwicki and Horst Bunke. 2005. Iam-ondb-an on-line english sentence database acquired from handwritten text on a whiteboard. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 956–961. IEEE.

Xin Ma, Yaohui Wang, Gengyun Jia, Xinyuan Chen, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. 2024. Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Arman Yeleussinov, Yedilkhan Amirgaliyev, and Lyailya Cherikbayeva. 2023. Improving ocr accuracy for kazakh handwriting recognition using gan models. *Applied Sciences*, 13(9):5677.

A Appendix

A.1 Training Configuration

Our model is trained on 8 NVIDIA V100 GPUs with a per-GPU batch size of 64. We adopt the Adam optimizer with a learning rate of 0.0001. Both the encoder and decoder are implemented as 3-layer Transformers, each with 4 attention heads and a hidden dimension of 512. In the Gaussian memory mask, we set the scaling factor $r = 17$ in Eq. 3. The mixture density network (MDN) outputs a 20-component Gaussian mixture ($K = 20$) to parameterize the pen trajectory distribution at each decoding step.

A.2 Evaluation Pipelines

Subjective Evaluation: We conducted a subjective evaluation with 20 human evaluators to score samples generated by four methods: AlexRNN and TrInk (both with and without the Top-k strategy). For the experiment, 96 text inputs were used to generate samples, and each output was rated on two criteria—style consistency and legibility using a 5-point Likert scale (1: lowest, 5: highest). Higher scores indicate better performance.

Objective Evaluation: We first convert the generated handwriting samples into standardized textline images to simulate realistic OCR application scenarios. These images are then fed into the state-of-the-art TrOCR model (Li et al., 2023) for text recognition. The outputs from TrOCR are systematically compared with the ground-truth text to compute Character Error Rate (CER) and Word Error Rate (WER), which quantify character-level inaccuracies and word-level mismatches, respectively. Notably, WER is excluded for short-text evaluations due to its instability when applied to limited word counts, as minor errors disproportionately skew the metric. Lower CER values indicate higher legibility, providing an automated and reproducible measure of text quality. During the evaluation of style equalization, we dynamically sample style inputs from the training set for the style encoder, ensuring that each synthesized handwriting sample corresponds to a unique, randomly selected style reference from the training dataset.

A.3 Visualization of Generated Samples

We present a collection of generated handwriting samples based on 13 text prompts of varying lengths. Each row illustrates outputs from four models: AlexRNN, AlexRNN (Top-k), TrInk, and TrInk (Top-k), displayed from left to right. As observed, TrInk consistently produces handwriting that is more legible and stylistically consistent than that of the RNN-based.

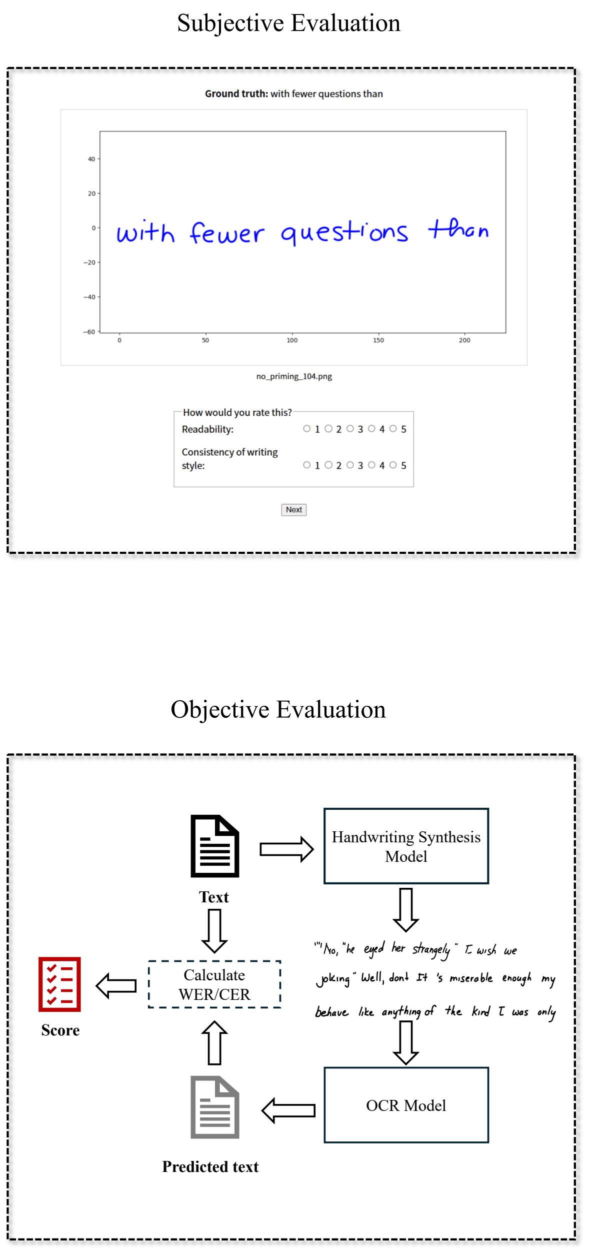

Subjective Evaluation
Ground truth: with fewer questions than
with fewer questions than
no_priming_104.png
How would you rate this?
Readability: ○1 ○2 ○3 ○4 ○5
Consistency of writing style: ○1 ○2 ○3 ○4 ○5
Next
Objective Evaluation
Text
Handwriting Synthesis Model
"No," he eyed her strangely." I wish we joking." Well, dont It 's miserable enough my behave like anything of the kind I was only
Calculate WER/CER
Score
OCR Model
Predicted text

Figure 4: Subjective and Objective Evaluation Pipelines.

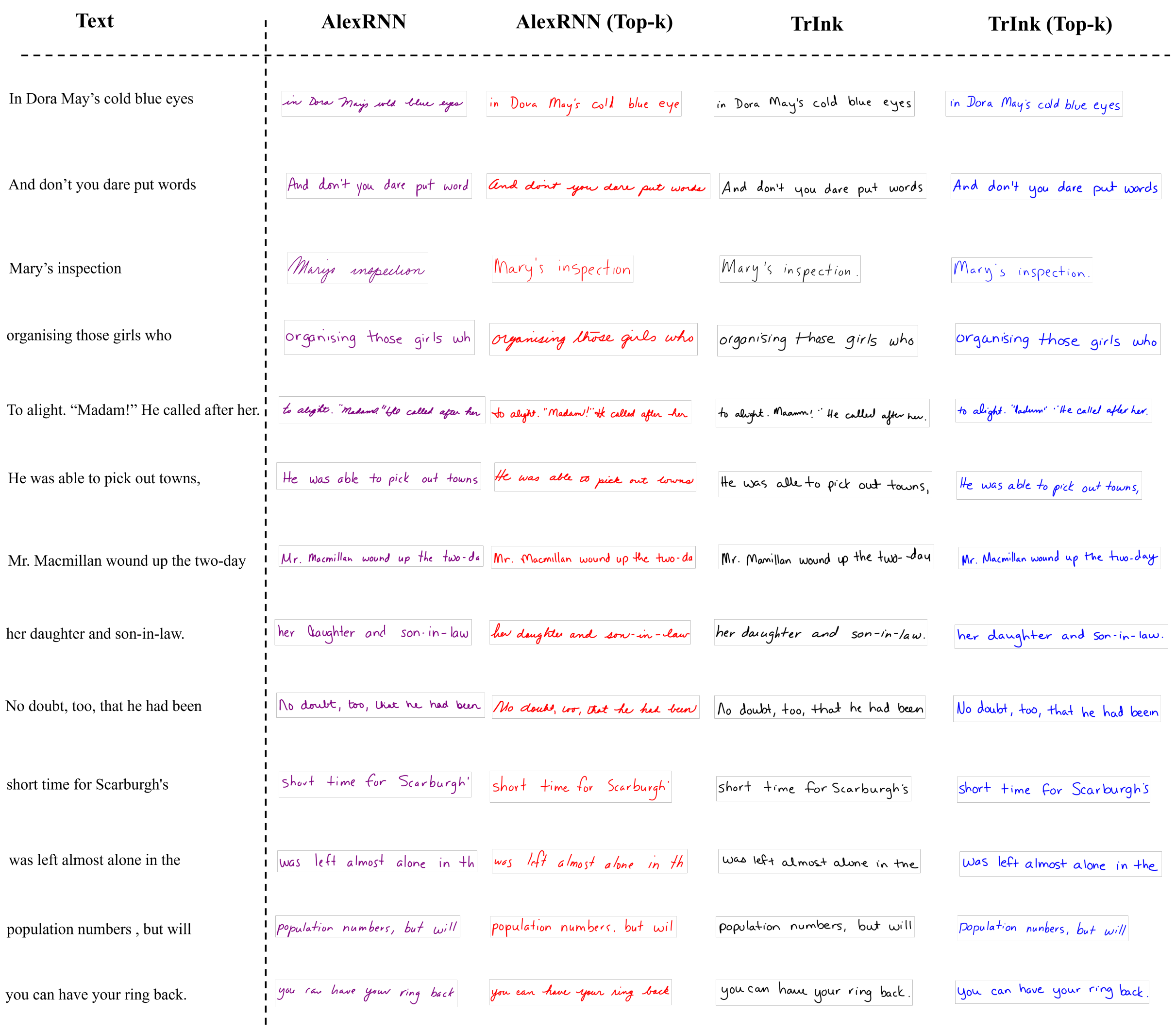

Text
AlexRNN
AlexRNN (Top-k)
TrInk
TrInk (Top-k)
In Dora May’s cold blue eyes
And don’t you dare put words
Mary’s inspection
organising those girls who
To alight. “Madam!” He called after her.
He was able to pick out towns,
Mr. Macmillan wound up the two-day
her daughter and son-in-law.
No doubt, too, that he had been
short time for Scarburgh's
was left almost alone in the
population numbers , but will
you can have your ring back.

Figure 5: Samples for AlexRNN and TrInk.