Towards Training-Free Scene Text Editing

Supplementary Material

6. AttnBoost Mechanism and Overshoot Scheduler

6.1. Attention-Modulated Overshooting

The AttnBoost module integrates attention mechanisms to achieve adaptive control over overshooting intensity, specifically targeting text regions while preserving non-text areas.

As shown in Fig. 7, attention mapping and aggregation mentioned in Sec. 3.3 extract text-to-image attention patterns and consolidate them through dimensional reduction. The resulting attention maps are then refined via spatial pooling to concentrate relevant information, followed by normalization to ensure consistent value ranges and numerical stability.

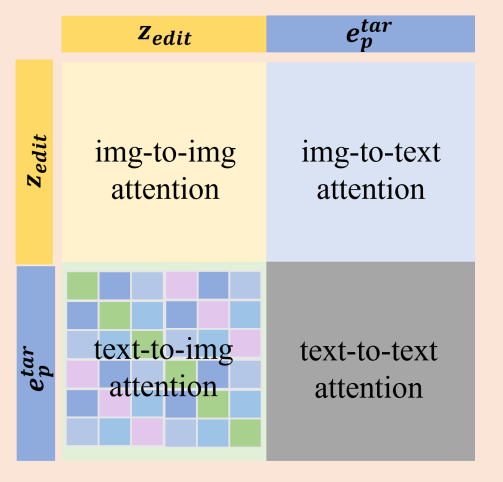

Figure 7. Computation graph of the full attention map between $\mathbf{z}_{\text{edit}}$ and $\mathbf{e}_p^{\text{tar}}$. The green-highlighted region in the lower-left corner illustrates the text-to-image attention components (A_{t2i}) extracted from each DoubleStream Transformer block, which are subsequently utilized for scheduler enhancement in the text rendering.

6.2. Implementation of Overshoot Scheduler

The Overshoot scheduler [16] implements a controlled trajectory deviation mechanism during the diffusion sampling process, leveraging attention-guided overshooting to enhance text rendering fidelity. The process, shown in Fig. 8, begins with a sample from the initial noise distribution, $\tilde{Z}_0 = X_0 \sim \pi_0$, and aims to compute the latent representation $\tilde{Z}_s$ at time $s = t + \varepsilon$ from the current state $\tilde{Z}_t$, where $\varepsilon > 0$ is the denoising step size.

In contrast to the standard Euler sampler, which updates as $\tilde{Z}_s = \tilde{Z}_t + \varepsilon v_\theta(\tilde{Z}_t, t)$, our overshooting sampler incorporates stochastic noise and attention modulation through a two-step procedure:

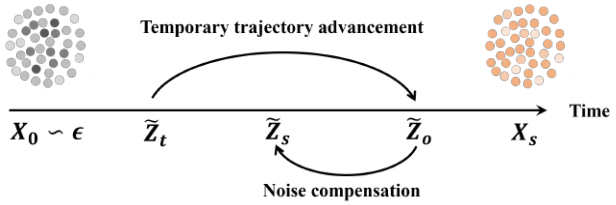

Figure 8. Schematic diagram of the Overshoot scheduler. Given the latent representation $\tilde{Z}_t$ at time t, the scheduler first advances the learned ODE trajectory beyond the target step to obtain $\hat{Z}_o$, then applies calibrated noise to return to the corrected state $\tilde{Z}_s$. The injected noise is precisely controlled to ensure $\tilde{Z}_s$ conforms to the marginal distribution of X_s.

1. **Temporary trajectory advancement:** The sampler first advances from the current timestep t to an overshoot point $o = s + \varepsilon c\hat{A}$, where $c \in \mathbb{R}^+$ is the overshoot intensity parameter and $\hat{A}$ denotes the normalized attention map derived from cross-modal interactions. The advanced latent representation is computed as:

$$\begin{aligned}\hat{Z}_o &= \tilde{Z}_t + v_\theta(\tilde{Z}_t, t) \odot (o - t) \\ &= \tilde{Z}_t + \varepsilon(1 + c\hat{A}) \odot v_\theta(\tilde{Z}_t, t),\end{aligned} \tag{16}$$

   Here, $v_\theta(\tilde{Z}_t, t)$ represents the velocity field parameterized by a neural network, and $\odot$ denotes element-wise multiplication with the attention map $\hat{A}$.

2. **Noise compensation and trajectory correction:** The oversampled latent $\hat{Z}_o$ is then corrected back to the target time s by introducing stochastic noise:

$$\tilde{Z}_s = a\hat{Z}_o + b\xi, \quad \xi \sim \mathcal{N}(0, I). \tag{17}$$

   The correction coefficients a and b are defined as:

$$a = \frac{s}{o}, \tag{18}$$

$$b = \sqrt{(1 - s)^2 - \frac{s^2(1 - o)^2}{o^2}}, \tag{19}$$

   This step ensures stability by compensating for the overshooting effect while preserving textual details.

The overall scheduler output for the next timestep is thus given by:

$$z_{t-1} = \mathcal{S}(z_t, \hat{A}, t), \tag{20}$$

where $\mathcal{S}$ encapsulates the overshooting and correction steps. This approach enables targeted improvements in text rendering quality within attention-masked regions without full

model fine-tuning, relying on well-aligned attention maps for optimal performance. The integration of attention modulation allows for adaptive control over overshooting intensity, focusing on text-relevant areas while minimizing artifacts in non-text regions.

7. More Analysis of Experiments

In this section, we present additional experiments to comprehensively analyze and validate our method.

7.1. Comparison with SOTA

Datasets. ScenePair collects 1,280 image pairs with text labels from ICDAR 2013 [18], HierText [24], and MLT 2017 [25], where each pair consists of two cropped text images with similar text length, style, and background, along with the original full-size images. We conduct quantitative analysis on the ScenePair dataset, where we pad the cropped images with similar background colors to a resolution of 384×256 to ensure consistent input size across all models, and all metrics are computed based on this preprocessing. We perform qualitative analysis using challenging full-size scene images selected from ICDAR 2013 [18], HierText [24], and MLT 2017 [25] datasets.

Quantitative Analysis. As shown in Table 6, our method achieves state-of-the-art performance across most image-quality metrics on the ScenePair dataset. TextFlow significantly outperforms competing methods in structural preservation with an SSIM of 89.03 and a PSNR of 22.47, while also demonstrating superior distortion reduction with an MSE of 0.91 and an FID of 13.53. Although TextCtrl attains the highest text accuracy with a character accuracy of 84.67% and an NED of 0.936, our method maintains competitive textual performance with 79.98% accuracy and 0.914 NED while delivering better overall visual quality and generalization capability. These quantitative results confirm TextFlow's balanced approach to preserving scene structure while achieving accurate text rendering.

Table 7 presents an extensive quantitative comparison of different methods on the TamperScene and AnyText-Bench datasets. Among training-based approaches, methods such as Longcat-edit and Flux-text demonstrate strong performance across various metrics. In the training-free category, particularly TextFlow-Longcat, our proposed TextFlow variants achieve the highest human evaluation (HE) scores and competitive accuracy metrics, outperforming existing training-free methods and narrowing the gap with training-based approaches. These results validate the effectiveness of our phase-aware guidance strategy in preserving style and ensuring textual accuracy without requiring task-specific training.

Qualitative Analysis. For comprehensive comparison on full-size images, as shown in Fig. 11, we select representative methods including: AnyText [40] from UNet-based approaches; Flux-Text [23] and textFlux [50] from DiT-based STE specific methods; general editing models Flux-Kontext [22] and Qwen-image [47]; and the training-free approach FlowEdit [20]. The results demonstrate that our method achieves superior performance in both style consistency and text accuracy. Notably, our approach successfully renders challenging out-of-vocabulary words like "HELLOW" in the first example, demonstrating its robust semantic understanding and effective visual-textual alignment. Furthermore, the integrated attention mechanism enables precise spatial localization of target regions without requiring explicit masks, achieving accurate local editing through attention-guided refinement in a fully mask-free paradigm.

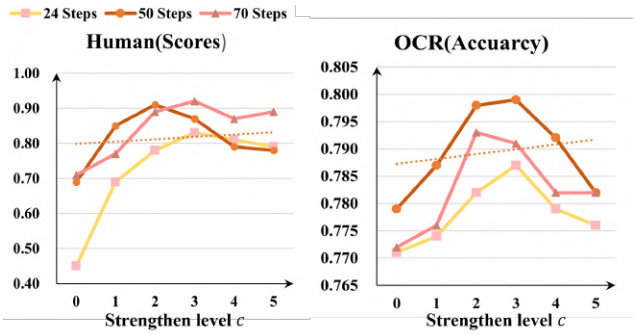

Figure 9. Ablation study on parameter c. The results indicate that configurations with $c = 2$ or $c = 3$ at 50 steps achieve optimal performance in terms of both generation accuracy and human evaluation scores.

7.2. Strength of V_Δ in FMS

Table 8 presents an ablation study on the strength parameter V_Δ using the ScenePair dataset. As the strength increases from 0.2 to 1.0, most metrics improve, with SSIM, PSNR, and ACC reaching their highest values at strength 1.0 or 2.0, while FID achieves the lowest (best) at strength 5.0. Notably, a strength of 1.0 yields a balanced performance with high SSIM (89.03), PSNR (22.47), ACC (79.98%), and competitive FID (13.53). Beyond 1.0, although image similarity metrics (SSIM, PSNR) continue to improve slightly, textual accuracy (ACC, NED) begins to decline, indicating a trade-off between style preservation and text fidelity. These results suggest that a moderate strength around 1.0 optimally balances the two objectives.

7.3. Strength of AttnBoost

Additionally, we perform an ablation study on the intensity parameter c of the AttnBoost mechanism, as illustrated in Fig. 9. The evaluation is conducted on the ScenePair dataset to measure text accuracy, supplemented by a human assessment phase that comprehensively evaluates both accuracy and aesthetic quality. The manual scoring uses a

Table 6. Performance of more methods on the ScenePair dataset.

Method	ScenePair					
	SSIM ($\times 10^{-2}$) ↑	PSNR ↑	MSE ($\times 10^{-2}$) ↓	FID ↓	ACC (%) ↑	NED ↑
TextCtrl [54]	37.56	14.99	4.47	43.78	**84.67**	**0.936**
Flux-Text [23]	86.45	17.95	1.93	54.84	70.94	0.877
TextFlow (Ours)	**89.03**	**22.47**	**0.91**	**13.53**	79.98	0.914

Table 7. Performance of different methods on TamperScene and AnyText-Bench. HE represents Human Evaluation.

Type	Method	ScenePair	TamperScene-2k			AnyText-Bench-en		
		HE	ACC(%) ↑	NED ↑	HE	Sen.acc(%)	NED	HE
Training-Based	TextFlux [50]	6.9	19.70	0.42	5.4	8.13	0.21	6.2
	Flux-Text [23]	7.1	18.60	0.42	5.9	**38.89**	**0.65**	8.1
	Qwen-image [47]	8.0	10.75	0.37	7.7	3,74	0.15	7.5
	Flux-Kontext [22]	7.3	17.79	0.45	7.2	19.53	0.39	7.9
	Longcat-Edit [38]	8.2	0.65	0.29	4.9	5.66	0.26	8.1
Training-Free	FlowEdit [20]	7.5	5.56	0.25	6.9	1.60	0.10	5.3
	TextFlow-Kontext	8.3	18.75	**0.45**	8.0	26.07	0.45	8.0
	TextFLow-Longcat	**8.5**	**20.95**	0.44	**8.7**	38.75	0.61	**8.5**

Table 8. Ablation of V_Δ strength on ScenePair.

Stren.	SSIM ↑	PSNR ↑	MSE ↓	FID ↓	ACC(%) ↑	NED ↑
0.2	84.81	18.19	2.67	23.16	75.20	0.895
0.5	85.14	19.83	2.06	19.21	76.92	0.908
0.7	85.92	20.86	1.46	17.25	79.90	**0.928**
1.0	89.03	22.47	0.91	**13.53**	**79.98**	0.914
2.0	**89.30**	**23.47**	**0.90**	14.20	77.62	0.894
5.0	88.47	21.02	0.90	**12.83**	76.88	0.872

Table 9. Ablation of Overshoot scheduler on ScenePair.

Method	Setting	ACC(%) ↑	NED ↑
TextFlow-Kontext	+ Overs. + AttnBoost	81.16	**0.93**
TextFlow-Kontext	+ Overs.	79.99	0.91
TextFlow-Kontext	-	78.72	0.92
TextFLux	+ Overs.	**81.24**	0.92
TextFLux	-	80.40	0.91

100-point system, with 50 points allocated to text accuracy and 50 points to aesthetics, and the results are reported in percentage form. Tests are performed under different inference steps with varying intensity values. Results indicate that the highest scores are achieved when $c = 2$ or $c = 3$, while larger values of c do not lead to significant performance improvements. Although inference with 70 steps yields marginally better results, considering the trade-off between inference efficiency and computational cost, we select 50 steps with $c = 2$ and 50 steps as the default configuration, offering the most balanced solution in practice.

To further validate the effectiveness of AttnBoost, we visualize its attention heatmaps in Fig. 10. The results clearly demonstrate its role in enhancing textual accuracy.

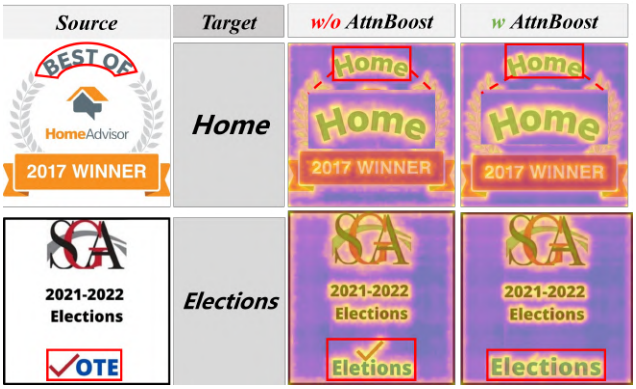

Figure 10. Visualized heatmaps of models with and without AttnBoost.

7.4. Ablation of Overshoot Scheduler

Table 10 ablates the overshoot scheduler on the ScenePair dataset. For TextFlow-kontext, introducing the overshoot scheduler alone boosts ACC from 78.72% to 79.99%, and further adding AttnBoost achieves the highest accuracy (81.16% ACC and 0.93 NED). Similarly, textFlux also benefits from the overshoot scheduler, with ACC increasing from 80.40% to 81.24%. These results confirm that both the overshoot scheduler and AttnBoost contribute to improved textual accuracy.

7.5. Time and GPU costs

Table 10 reports inference time, GPU memory consumption, and accuracy on the TamperScene dataset using an A6000 GPU. For Flux-kontext, integrating TextFlow increases inference time from 260.21s to 483.43s and memory

Table 10. Inference time and memory cost on A6000, accuracy on TamperScene.

Model	Setting	Time(s) ↓	Mme(GB) ↓	ACC(%)
F-Kontext	Base	**260.21**	**40.12**	17.79
F-Kontext	Base+TextFlow	483.43	41.57	**18.75**
Longcat-Edit	Base	**136.54**	**37.24**	0.65
Longcat-Edit	Base+TextFlow	252.62	38.67	**20.95**
FlowEdit	Base(FLUX.1dev)	100.62	32.20	5.56

usage from 40.12GB to 41.57GB, while improving accuracy from 17.79% to 18.75%. For Longcat-edit, TextFlow raises time from 136.54s to 252.62s and memory from 37.24GB to 38.67GB, but delivers a dramatic accuracy boost from 0.65% to 20.95%. FlowEdit (based on FLUX.1-dev) serves as a baseline with 100.62s, 32.20GB, and 5.56% accuracy. These results demonstrate that TextFlow achieves substantial gains in textual accuracy at the cost of moderate increases in computational resources, particularly for models that initially exhibit low text accuracy.

7.6. Visualize Results and Limitations

Visualize Results Fig. 12 demonstrates the editing performance of TextFlow across diverse challenging scenarios. Our method exhibits remarkable capability in handling special symbols, significant background luminance variations, fine-grained regions, artistic typography, and structurally complex layouts. Particularly noteworthy is its performance in the second row, where TextFlow successfully maintains style consistency and achieves accurate text rendering even when dealing with circular text arrangements, a particularly challenging case that requires sophisticated geometric adaptation.

Fig. 13 (a) and (b) demonstrate the scalability of TextFlow across challenging editing tasks, including variations in word length and simple stylistic changes, highlighting its robust generalization to complex scenarios without task-specific tuning. Meanwhile, Fig. 13 (c) illustrates its flexibility in responding to diverse user instructions, accurately performing edits according to different textual prompts, which underscores its adaptability for interactive applications.

Limitations Fig. 14 illustrates certain limitations of our proposed method. The approach exhibits challenges in accurate word spatial localization and inter-word gap recognition, as evidenced by the case where "ROYAL" → "PEN" incorrectly merges both words into "PEN". Additionally, the method demonstrates insufficient capability in handling images with perspective distortion, as shown in the "Just" → "God" example, where it erroneously modifies all textual elements while leaving residual background artifacts. For irregular character arrangements such as handwritten fonts, the editing process fails to achieve satisfactory results, as seen in the unsuccessful "Geek" → "Models" conversion. Furthermore, the method occasionally produces blurred rendering outputs, particularly evident in the "Thes" → "what" transformation, where character clarity is compromised.

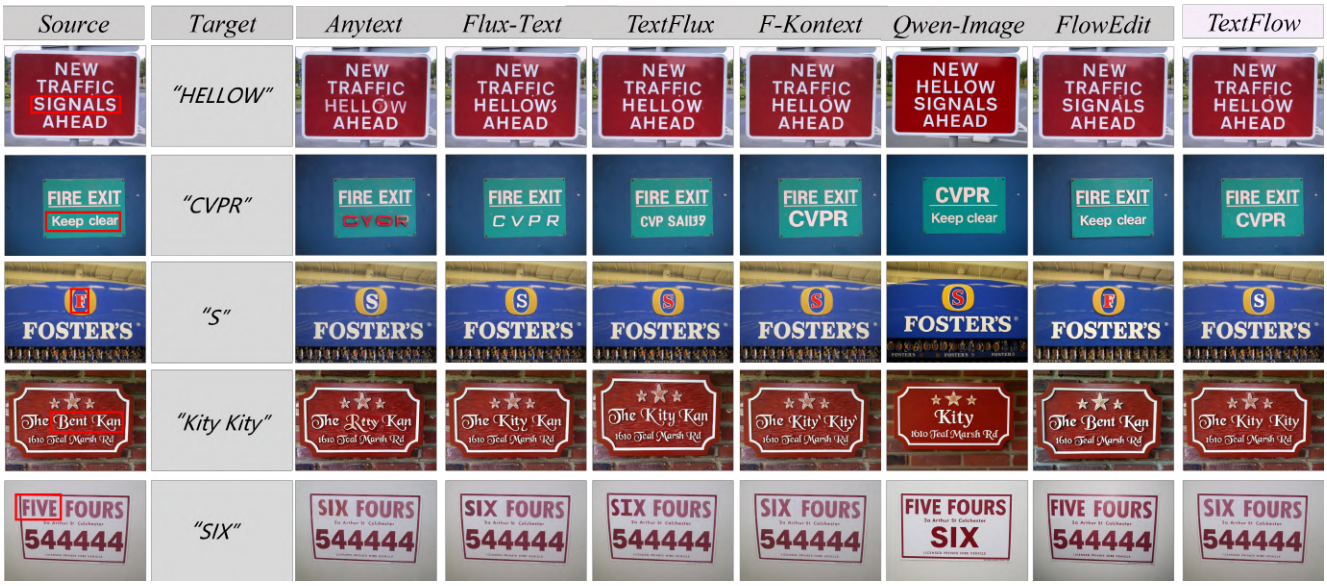

Figure 11. Comparative analysis of editing performance with additional models on full-size images.

Figure 12. Additional visual results demonstrating robust performance across challenging scenarios, including artistic typography, small-font text, complex layouts, and special symbols.

Figure 13. Extensive study on varying length, style change, and different prompts.

Figure 14. Limitations in editing accuracy and practical utility within complex scenarios.