Residual Decoder Adapter: ID-Preserving Tokenizer Adaption for Autoregressive Text Rendering

Supplementary Material

Contents

1. Implementation Details

1.1. Model Architecture

Hint Codebook We instantiate a paired codebook that mirrors the size and index space of the original tokenizer codebook. The embedding vectors are learned from scratch, but their indices remain aligned with the base codebook.

Projector Design We use a lightweight 1×1 Conv2d to map hint-codebook embeddings ($d_{\text{hint}} = 16$) to the residual decoder input space:

$$\text{Projector}_q : \text{Conv2d}(16, 256, 1) \quad (1)$$

Table 1. **Residual Decoder architecture.** The decoder follows a simple expand–process–upsample design.

Stage	Layer	Input → Output	Resolution
Input	-	[B, 256, h, w]	1/16
Expansion	Conv-in	256 → 1024	1/16
Processing	ResBlock	1024 → 1024	1/16
	AttnBlock	1024 → 1024	1/16
	ResBlock	1024 → 1024	1/16
Upsampling	Block 0	1024 → 512	1/16 → 1/8
	Block 1	512 → 512	1/8 → 1/4
	Block 2	512 → 512	1/4 → 1/2
	Block 3	512 → 256	1/2 → 1/1
	Block 4	256 → 128	1/1 → 1/1
Output	Conv-out	128 → 3	1/1

Figure 1. **Visualization of Sobel mask and frequency mask.**

Residual Decoder Architecture The architecture largely follows decoder of LlamaGenVQ [1] design with key modifications for high-resolution text detail capture. The full specification is shown in Tab. 1.

1.2. Training Configuration

All loss weights are set to 1.0. We use a high-pass mask M_q with $q = 0.8$ during training. The gradient-sensitive loss $\mathcal{L}_{\text{sobel}}$ emphasizes edge structures, while $\mathcal{L}_{\text{freq}}$ preserves high-frequency components in Fourier space. Visualization of these masks is provided in Fig. 1.

2. Computational Cost Analysis

2.1. Parameters and Latency

Tab. 2 quantifies the computational overhead introduced by RDA on different AR models and tokenizers. All measurements are conducted in inference mode on a single V100 GPU.

RDA introduces negligible overhead ($< 2\%$ latency),

Table 2. **Computational cost of RDA.** We report the overhead on both the tokenizer side and the end-to-end AR model.

Module	Params	Latency
Tokenizer		
LLamaGenVQ	72 M	34.50 ms
+ RDA	237 M	80.11 ms
ChameleonVQ	69 M	34.48 ms
+ RDA	234 M	80.90 ms
AR model		
Janus Pro 1B	2.09 G	11.20 s
+ RDA	2.15 G (+7.89%)	11.36 s (+1.43%)
Janus Pro 7B	7.42 G	14.63 s
+ RDA	7.58 G (+2.22%)	14.80 s (+1.16%)
Tar 7B	9.40G	72.93 s
+ RDA	9.56 G (+1.76%)	73.25 s (+0.42%)
Lumina-mgpt	7.04 G	212.16 s
+ RDA	7.20 G (+2.34%)	212.43 s (+0.13%)

Figure 2. **Comparison between ground-truth images and reconstructions from the base image tokenizer**. The reconstructed results blur fine-grained text strokes and distort glyph edges, indicating that textual details are significantly degraded during quantization and decoding.

while achieving substantial improvements in text rendering quality.

3. Understanding the Bottleneck

3.1. Tokenizer Reconstruction Limit

We verify that the base tokenizer exhibits inherent reconstruction limitations even when provided perfect ground-truth input. Fig. 2 shows that the quantization-decoding process inherently loses fine-grained details, blurring text strokes and distorting glyph edges. This confirms that the tokenizer's reconstruction capability is the primary bottleneck.

Figure 3. **Failure Case of Tokenizer.**

3.2. Dual Bottleneck in Text Rendering

General AR models face two bottlenecks:

1. **Token prediction:** Weak text token prediction from the AR model.
2. **Reconstruction:** Limited reconstruction fidelity from the tokenizer.

Text-specific fine-tuning addresses the first bottleneck, making the tokenizer decoder the dominant limitation. RDA directly targets this by enhancing reconstruction without modifying the AR model, enabling large improvements on text-tuned models.

4. Failure Case Analysis

We categorize failures into two types based on their source:

4.1. Tokenizer-Level Failures

When characters are extremely small or visually ambiguous, the tokenizer may assign incorrect visual tokens, leading to unrecoverable errors. Fig. 3 illustrates such cases.

4.2. AR Model-Level Failures

When the AR model generates malformed glyph structures during generation, RDA cannot correct them since it operates on the decoded output. Fig. 4 shows examples where the AR model produces structurally incorrect characters.

5. Training Analysis

5.1. Training Stability and Convergence

Fig. 5 presents training curves for RDA. The optimization is stable throughout training. Critically, without residual perceptual loss $\mathcal{L}_{\text{perc}}^{\text{res}}$, the residual branch fails to converge and produces only blurry gray regions. This underscores the importance of perceptual supervision for learning meaningful high-frequency details.

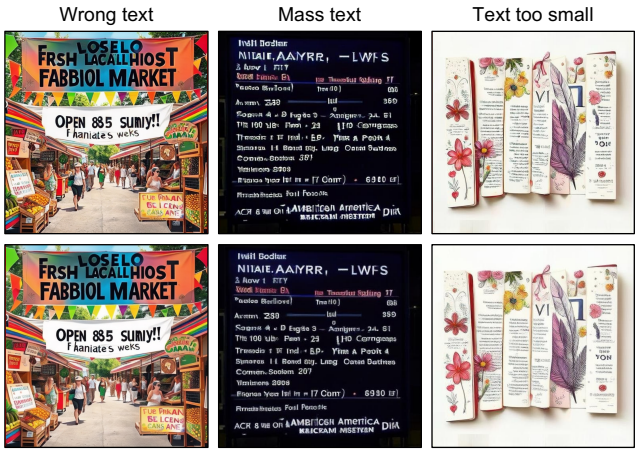

Figure 4. **Failure Case of AR model.**

Figure 5. **Visualization of loss curve.**

6. Design Justifications

6.1. Why Shared-ID Preserves Compatibility

The Shared-ID mechanism ensures that the token ID distribution remains identical to the base tokenizer.

Since AR models learn a distribution over token IDs (not codebook embeddings), they can directly benefit from improved reconstruction without retraining.

6.2. Why TAR Can Use RDA (LlamaGen-VQ)

TAR adapts LlamaGen-VQ by modifying the tokenization and embedding pipeline before decoding. In contrast, RDA operates after the decoder and refines pixel-level outputs without altering token IDs. As illustrated in Fig. 6, this separation allows RDA trained on LlamaGen-VQ to be directly applied to TAR without additional training.

7. Extended Experiments

7.1. Tokenizer Results on Additional Datasets

We also conduct evaluations on StyledTextVisionBlend and TextScenesHQ of TextAtlasEval [3] to assess robustness.

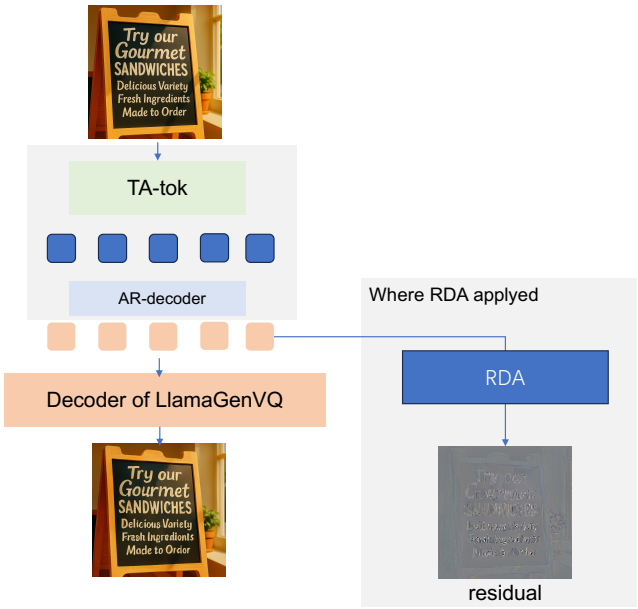

Figure 6. **Where RDA is applied in TAR.** TAR modifies the tokenization and embedding pipeline before decoding, while RDA attaches after the decoder and refines pixel-level outputs without altering token IDs.

The results are summarized in Tab. 3.

7.2. AR Model Results on TextAtlasEval

We also evaluate general AR models on the TextAtlasEval. The results are reported in Tab. 4.

Across different AR backbones and resolutions, RDA provides consistent improvements on TextAtlasEval benchmarks.

8. More Visualizations

8.1. Tokenizer Reconstruction Results

We evaluate the reconstruction quality of different image tokenizers, including LlamaGen-VQ and Chameleon-VQ [2], across multiple datasets. For clarity, Fig. 7 presents results based on LlamaGenVQ, while Fig. 8 shows results from Chameleon-VQ. Across both tokenizers, applying RDA leads to visibly sharper text strokes and improved structural consistency, demonstrating that our method generalizes to different tokenizer architectures.

8.2. General AR Generation Results

Janus Pro Fig. 9 shows generation results from Janus Pro.
TAR-1B Fig. 10 shows generation results from TAR-1B.
TAR-7B Fig. 11 shows generation results from TAR-7B.

Table 3. **Comparison of text image reconstruction performance across image tokenizers.** StyledTextSynth is evaluated at 512 and 1024.

Model	Data	AR Free	StyledTextVisionBlend Acc.	F1.	SSIM	LPIPS	TextScenesHQ Acc.	F1.	SSIM	LPIPS
Low Resolution										
LlamagenVQ	50M	✗	65.58	71.34	82.80	7.61	11.11	16.35	50.36	46.27
w/ RDA	5M	✔	78.94	82.75	85.91	6.46	19.73	26.45	51.26	45.92
ChameleonVQ	-	✗	58.75	62.65	81.59	7.94	10.12	13.73	49.71	46.34
w/ RDA	5M	✔	72.16	76.39	83.70	7.20	14.59	19.80	50.20	46.02
High Resolution										
LlamagenVQ	50M	✗	92.04	92.45	92.62	5.67	33.47	40.98	52.50	42.55
w/ RDA	5M	✔	93.39	92.42	93.32	5.60	42.15	47.72	53.29	42.55
ChameleonVQ	-	✗	91.70	91.49	92.25	5.10	30.49	36.31	51.79	42.11
w/ RDA	5M	✔	92.19	91.78	92.21	4.62	36.47	42.29	52.12	42.14

Table 4. **Comparison of general AR models before and after applying RDA.** Each cell shows the result *w*/*wo* applying RDA.

Model	Size	Res	TextVisionBlend Acc. ↑	F1. ↑	CER ↓	StyledTextSynth Acc. ↑	F1. ↑	CER ↓	TextScenesHQ Acc. ↑	F1. ↑	CER ↓
Janus Pro	1B	384	0.68/**0.74**	1.22/**1.30**	0.94/**0.93**	0.47/**0.64**	0.91/**1.21**	0.97/0.97	0.39/**1.02**	0.70/**1.70**	0.94/**0.90**
	7B	384	0.37/**0.56**	0.69/**1.04**	0.97/**0.96**	0.46/**0.89**	0.60/**1.14**	0.98/**0.97**	0.68/**1.02**	1.19/**1.70**	0.92/**0.90**
TAR	1.5B	512	0.87/**1.32**	1.55/**2.26**	0.96/**0.95**	1.17/**2.02**	2.15/**3.55**	0.96/**0.94**	2.31/**3.59**	3.59/**5.04**	0.89/**0.87**
	1.5B	1024	2.81/**4.05**	4.00/**4.57**	0.92/0.92	3.48/**5.10**	5.52/**7.10**	0.89/**0.86**	2.31/**3.59**	3.59/**5.04**	0.89/**0.85**
	7B	512	4.90/**7.29**	7.34/**10.11**	0.87/**0.85**	3.62/**6.39**	6.21/**9.99**	0.92/**0.88**	9.84/**13.63**	13.31/**16.61**	0.77/**0.74**
	7B	1024	7.56/**7.98**	10.33/**10.74**	0.84/0.84	7.70/**8.32**	11.70/**12.40**	0.86/**0.85**	13.94/**14.47**	16.59/**16.83**	0.75/0.75

8.3. Text-Specialized AR Generation Results

Janus Pro (Fine-tuned) Fig. 12 shows generation results from fine-tuned Janus Pro.

Lumina-mGPT (512px) Fig. 13 shows generation results fine-tuned Lumina-mGPT at 512 resolution.

Lumina-mGPT (1024px) Fig. 14 shows generation results fine-tuned Lumina-mGPT at 1024 resolution.

9. Additional Information

9.1. Recaption Prompt

We use the following prompt to generate recaptions via Qwen-2.5-VL:

2

Recaption Prompt Carefully describe the image by precisely combining visual elements with all visible text. The final caption must integrate the visual scene and quoted text into a coherent, factual narrative of around 100 words. Extract every piece of visible text from the image—no omissions—and enclose each text string in double quotes (""). Specify the approximate position of each text (e.g., top-left, center, bottom-right). Avoid adding any imaginative, inferred, or generic descriptions not grounded in the image.

References

[1] 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] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models, 2025. 3

[3] Alex Jinpeng Wang, Dongxing Mao, Jiawei Zhang, Weiming Han, Zhuobai Dong, Linjie Li, Yiqi Lin, Zhengyuan Yang, Libo Qin, Fuwei Zhang, Lijuan Wang, and Min Li. Textatlas5m: A large-scale dataset for dense text image generation, 2025. 3

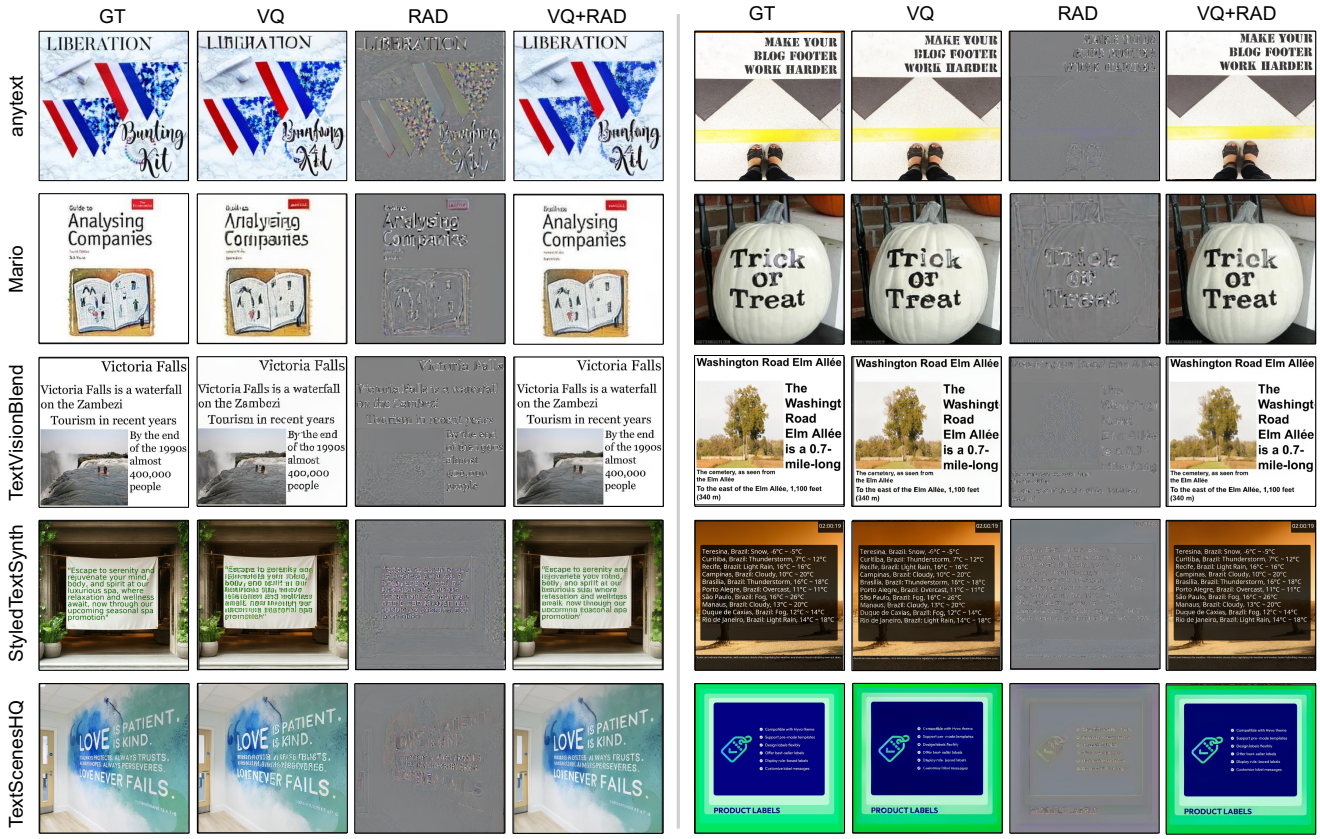

Figure 7. **Qualitative Results of LlamaGenVQ Applying RDA.** Left: low-resolution setting. Right: high-resolution setting.

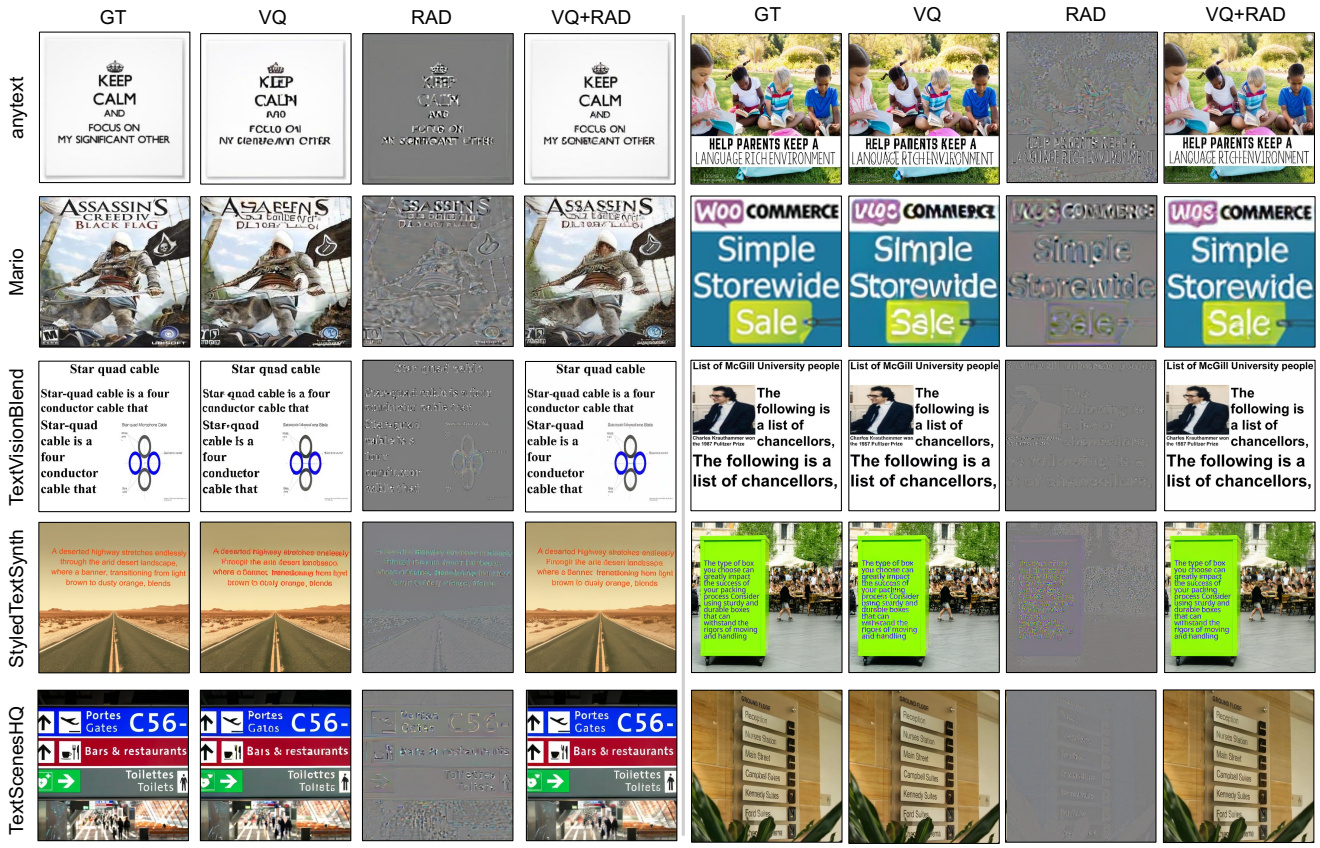

Figure 8. **Qualitative Results of ChameleonVQ Applying RDA.** Left: low-resolution setting. Right: high-resolution setting.

Figure 9. **Qualitative Results of Janus Pro Applying RDA**

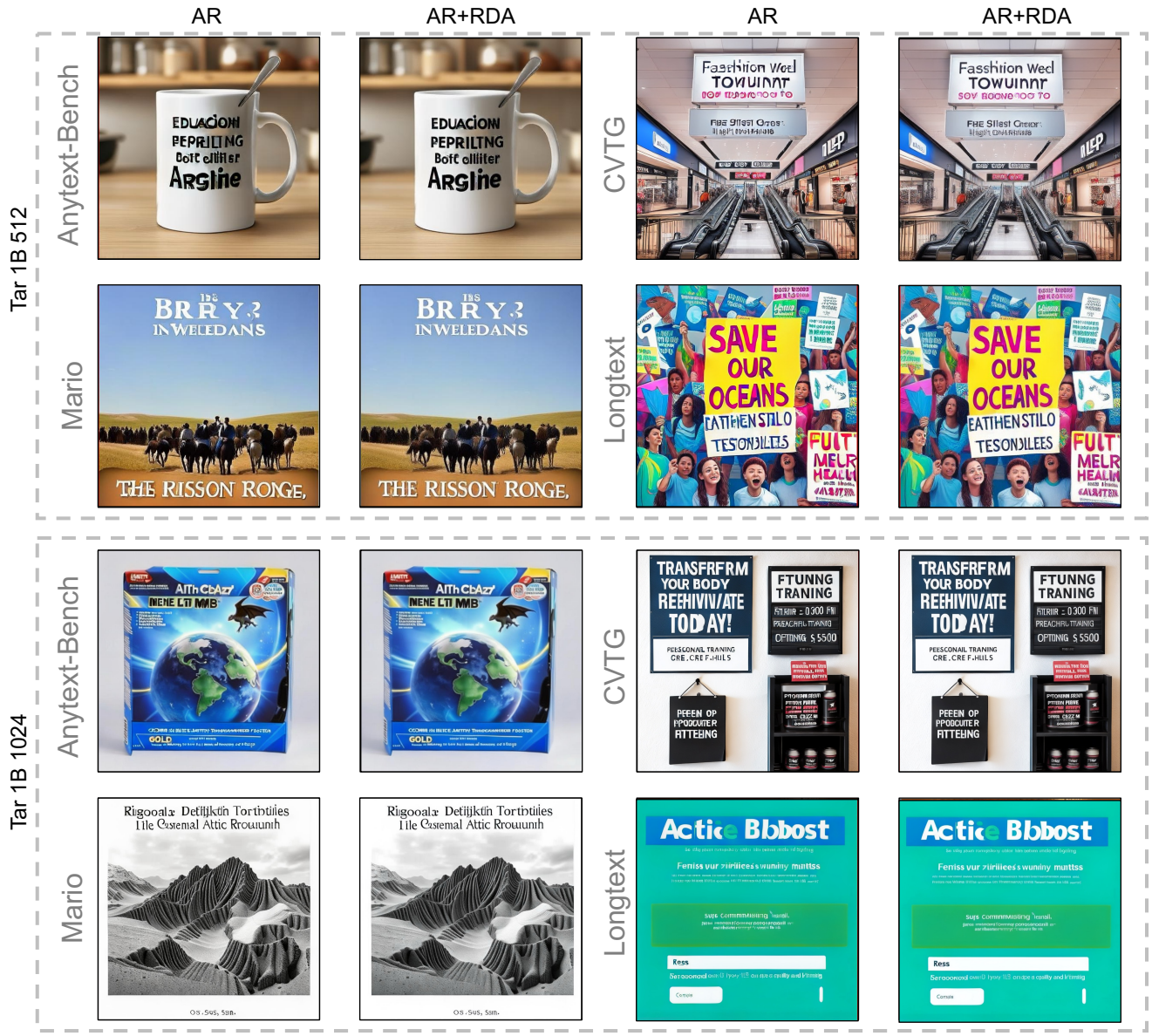

Figure 10. **Qualitative Results of Tar 1B Applying RDA**

Figure 11. **Qualitative Results of Tar 7B Applying RDA**

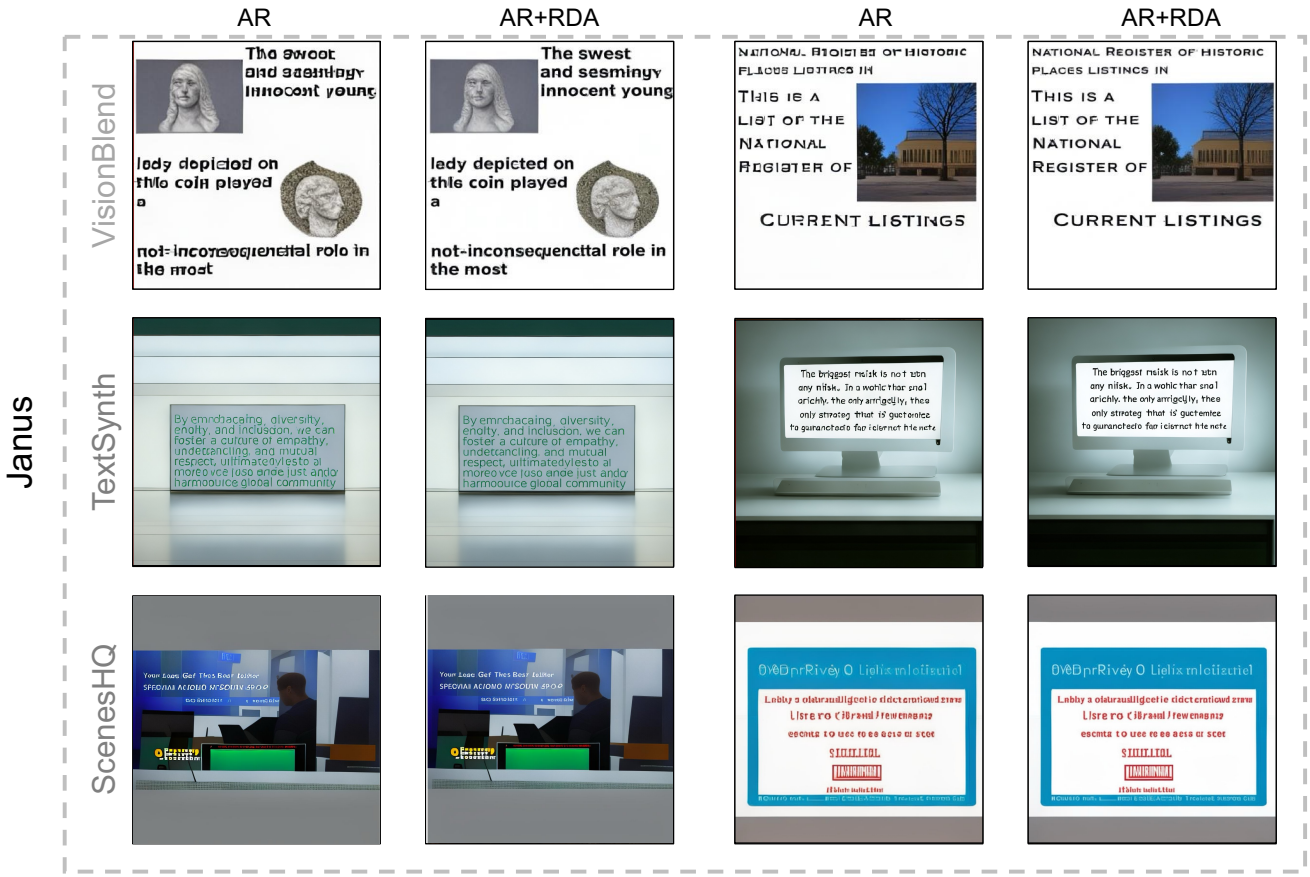

Figure 12. **Qualitative Results of Finetuned Janus Pro Applying RDA**

Figure 13. **Qualitative Results of Finetuned Luminamgpt 512 Applying RDA**

Figure 14. **Qualitative Results of Finetuned Luminamgpt 1024 Applying RDA**