FastVAR: Linear Visual Autoregressive Modeling via Cached Token Pruning

Supplementary Material

A. Results on ImageNet with VAR

In the main paper, we focus on the performance of our FastVAR on high-resolution image generation tasks. As stated in Section 3.2, applying token pruning on early small scale steps can lead to performance degradation due to the interference of the structure construction. Given that pruning on small token maps can not bring significant speedups, we thus do not design over-complex algorithms to further accelerate small-scale steps. However, for the sake of experimental completeness, we give the results of our FastVAR on the 256×256 class conditional generation on ImageNet in Tab. A.1. It can be seen that our FastVAR can achieve very competitive performance against existing methods, while maintaining a high speedup ratio. Since the VAE in the existing VAR methods adopts a high compression rate, *e.g.*, 16× downsample, the token map resolution at the largest scale in the VAR model [51] is only 16×16 for the 256×256 image generation. As a result, token pruning on this small-scale generation is much less robust compared to the larger resolution, *e.g.*, 1024×1024 resolution. We leave it for future work to design more generalized strategies to further include token maps at small scales.

Table A.1. Quantitative comparison on 256×256 generation on ImageNet.

Methods	#param	runtime	memory	IS↑	FID↓	Precision↑	Recall↑
VAR(d=24) [51]	1.0B	1.2s	14GB	313.7	2.29	82.50	57.45
VAR(d=30) [51]	2.0B	2.2s	22GB	306.6	2.05	81.76	58.20
CoDe(N=8) [10]	2.3B	1.7s	15GB	300.4	2.26	81.31	58.63
CoDe(N=9) [10]	2.3B	2.2s	16GB	297.2	2.16	81.07	59.03
FastVAR(d=24)	1.0B	1.1s	13GB	287.4	2.64	79.76	58.22
FastVAR(d=30)	2.0B	1.9s	15GB	288.7	2.30	80.72	58.64

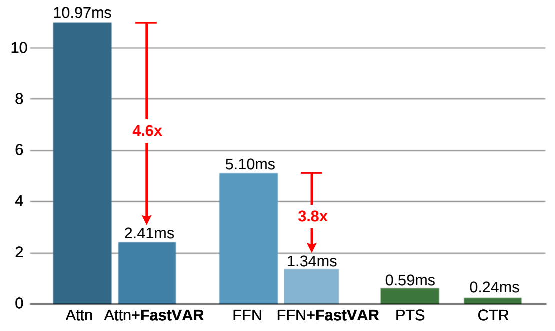

Figure A.1. Efficiency Profiling of different modules in one Transformer layer. Note that the runtime of the Attn and FFN baseline is evaluated using FlashAttention.

B. Further Efficiency Profiling

As demonstrated in the experiments, our FastVAR can achieve significant speedup without performance degradation, *e.g.*, 1.5× speedup for the HART backbone. However, this speedup ratio still shares some similar latencies as the unpruned benchmark, such as the forward pass at small scale steps. Here, we give a more fine-grained speedup comparison by directly comparing the attention and FFN under the condition of with and without the proposed FastVAR. As illustrated in Fig. A.1, FastVAR (ratio=75%) can bring even a 4.6× speedup for the attention and a 3.8× speedup for the FFN. This result demonstrates a promising speedup upper bound of our FastVAR.

Furthermore, compared to the runtime of the standard benchmark, our FastVAR adds additional token importance calculation, as well as token number restoration, which may introduce additional time. Here, we give experimental results to validate the efficiency of our FastVAR. As shown in Fig. A.1, the additional computational cost accompanying our FastVAR is almost negligible. For example, the proposed PTS occupies only 0.59 ms, while the CTR occupies 0.24 ms. Thus the total additional latency from our FastVAR, *i.e.*, 0.63 ms, occupies only 5% of the original attention module, which is significantly lower than the speedup brought from FastVAR.

Table A.2. More evaluation results on HPSv2.1 [58] and ImageReward [62] benchmarks.

benchmark	HART	ToMe	Ours	Infinity	ToMe	Ours
Latency↓	950ms	800ms	630ms	2600ms	1130ms	950ms
Speedup↑	1.0×	1.2×	1.5×	1.0×	2.3×	2.7×
HPSv2.1↑	28.75	27.04	27.85	30.36	29.85	30.03
ImageReward↑	0.5658	0.4988	0.5370	0.9245	0.8992	0.9129

C. Comparison on More Benchmarks

In the main paper, we compare our FastVAR with different methods on the Geneval [20] and MJHQ30K [29] datasets. In order to provide a systematic evaluation, we further compare different methods on more benchmarks including HPSv2.1 [58], and ImageReward [62]. The experimental results are given in Tab. A.2. It can be seen that our FastVAR maintains consistently favorable performance than other competitive token pruning baseline, while allowing for significant speedup than the original backbones. For instance, FastVAR achieves 0.81 higher HPSv2.1 score than ToMe [3] while being more efficient. The above results on more evaluation benchmarks further demonstrate the effectiveness of our FastVAR.

Table A.3. Ablation experiments of applying extreme pruning ratios to other VAR backbone HART [49]. We set $N = 2$ in all setups, *i.e.*, only the last two scale steps are pruned with FastVAR.

ratios	Speedup↑	Latency↓	Throughput↑	FID↓	CLIP↑	GenEval↑
no_pruning	1.0×	0.95s	1.05	30.61	28.47	0.51
{50%,75%}	1.5×	0.63s	1.59	28.19	28.34	0.51
{50%,100%}	1.9×	0.51s	1.96	48.54	28.46	0.48

Table A.4. Ablation experiments of different caching scale steps.

cached steps	GenEval				MJHQ30K	
	two_object↑	position↑	color_attr↑	Overall↑	FID↓	CLIP↑
no_pruning	0.62	0.13	0.18	0.51	30.61	28.47
K-N-3	0.53	0.11	0.15	0.47	40.61	27.36
K-N-2	0.60	0.13	0.19	0.50	32.39	27.94
K-N-1	0.57	0.13	0.20	0.49	29.83	28.25
K-N	0.57	0.16	0.24	0.51	28.19	28.34

D. Ablation on Caching Step

In the proposed Cached Token Restoration (CTR), we use the token map from the last scale step of the Structure Construction Stage $\mathcal{S}$, *i.e.*, the $(N - K)$-th step, as the cache, which will be used to restore the original token numbers during token pruning. Here, we conduct ablation experiments to justify the rationality by setting different scale steps as the caching step. The results are shown in Tab. A.4. It can be seen that setting the last element in $\mathcal{S}$ as the caching step achieves consistently the best results on all evaluation metrics. Notably, this setup has almost no performance degradation compared to the unpruned baseline models. In addition, we observe a steady performance drop when the caching step gradually moves small. This is because we use the cached token map to approximate the pruned tokens, so the gap between the last element in $\mathcal{S}$ and the steps in $\mathcal{T}$ is the smallest. Therefore, using the step that is closer to the pruned scale steps as the caching step can achieve better performance.

E. Discussion on Extreme Pruning Ratios

In the main paper, we mentioned that different backbones exhibit different levels of tolerance for the pruning ratio. For example, we used an even 100% ratio for the last two scale steps of the Infinite model [22]. However, we point out that this extreme pruning ratio does not apply to HART model [49]. Specifically, we apply the $N = 2$ and {50%, 100%} FastVAR to the HART model. The experimental results are shown in Tab. A.3. It can be seen that the extreme pruning ratio produces serious performance degradation for HART. We argue that this is due to the difference in the pre-trained model size. Specifically, the size of Infinite 2B is significantly larger than the 700M of HART. The stronger capabilities of the larger model allow for modeling more challenging textures in the earlier scale steps. As a contrast, the smaller model relies on test-time scaling [47] to use longer scale steps to produce complex details, and thus suffers from severe degradation when extreme pruning is applied on the last few steps.

F. Limitation and Future Work

Our FastVAR can effectively alleviate the quadratically increasing complexity with scales, benefiting from the proposed cached token pruning. Nonetheless, our work can be further improved in the future in the following aspects. First, the proposed FastVAR focuses mainly on the acceleration of the large-scale step which occupies the main inference time. Therefore, our method can be further improved in accelerating small-resolution image generation tasks by designing more generalized pruning strategies to include pruning small-scale token maps as well. Second, we have revealed the scale-wise sensitivity of pre-trained VAR models, *i.e.*, large-scale steps are more robust to small-scale steps for pruning, which inspires us to adopt a progressive pruning ratio schedule. Therefore, utilizing more fine-grained pruning prior, *e.g.*, layer-wise or even developing adaptive pruning ratios, is promising to achieve higher speedup ratios. Third, we show that the current FastVAR can be combined with Flash Attention to achieve combined speedup. As other potential work on accelerating VAR models emerges, such as network quantization or fewer decoding steps, our FastVAR could potentially integrate with these approaches to achieve further acceleration.

G. Algorithm of FastVAR

In Algo. 1, we give the Pytorch-like pseudocode of the proposed FastVAR. Thanks to the simplicity and generality, our proposed FastVAR can be seamlessly integrated into various VAR models using a few code lines.

H. More Visual Results

In this section, we provide more visual results, which are organized as follows:

- In Fig. A.2, we give more visualization results about the intermediate outputs of the pre-trained VAR model at different scale steps.
- In Fig. A.3, we give more qualitative results of Infinite [22] on the MJHQ30K dataset.
- Fig. A.4 gives more qualitative results of the HART [49] model on the MJHQ30K [29] dataset.
- In Fig. A.5, Fig. A.6, and Fig. A.7, we give more generation results on the zero-shot higher-resolution image synthesis tasks.

Algorithm 1: The pseudo-code of FastVAR algorithm, Pytorch-like

```
def pivotal_token_selection(x, topk):
   # calculate the direct-through component
   pool_x = rearrange(x, 'b (h w) c -> b c h w')
   pool_x = adaptive_avg_pool2d(x, (1, 1))
   pool_x = rearrange(pool_x, 'b c 1 1 -> b 1 c')
   score = sum((x - pool_x)**2, dim=-1)
   # select the topK high frequency tokens
   pivotal_idx = argsort(score, dim=1, descending=True)[:, :topk, :]
   return gather(x, dim=1, index=pivotal_idx)
def cached_token_restoration(x, cache):
   # up-sample cache features to the size of x
   restored_x = interpolate(cache)
   restored_x = rearrange(restored_x, 'b c h w -> b h w c')
   # fuse the cached and the current tokens
   restored_x.scatter_(dim=1, index=pivotal_idx, src=x)
   return restored_x
```

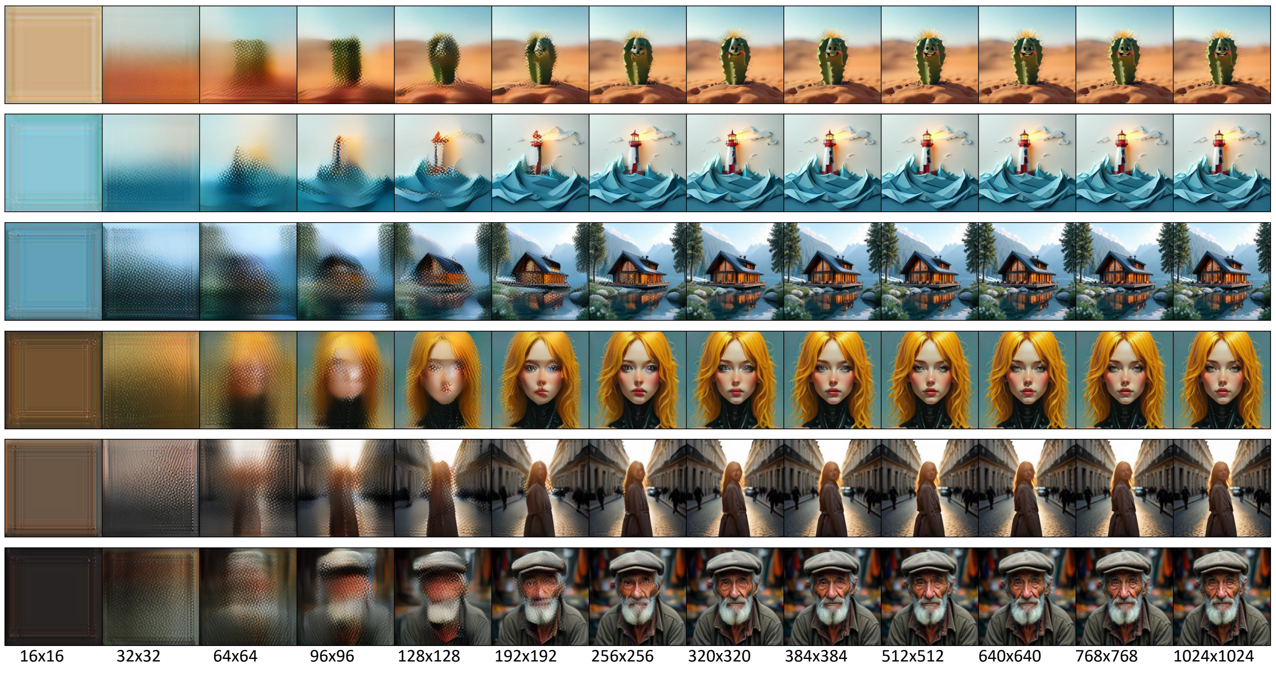

Figure A.2. More visualization results of the intermediate outputs at different scale steps of pre-trained VAR model [22].

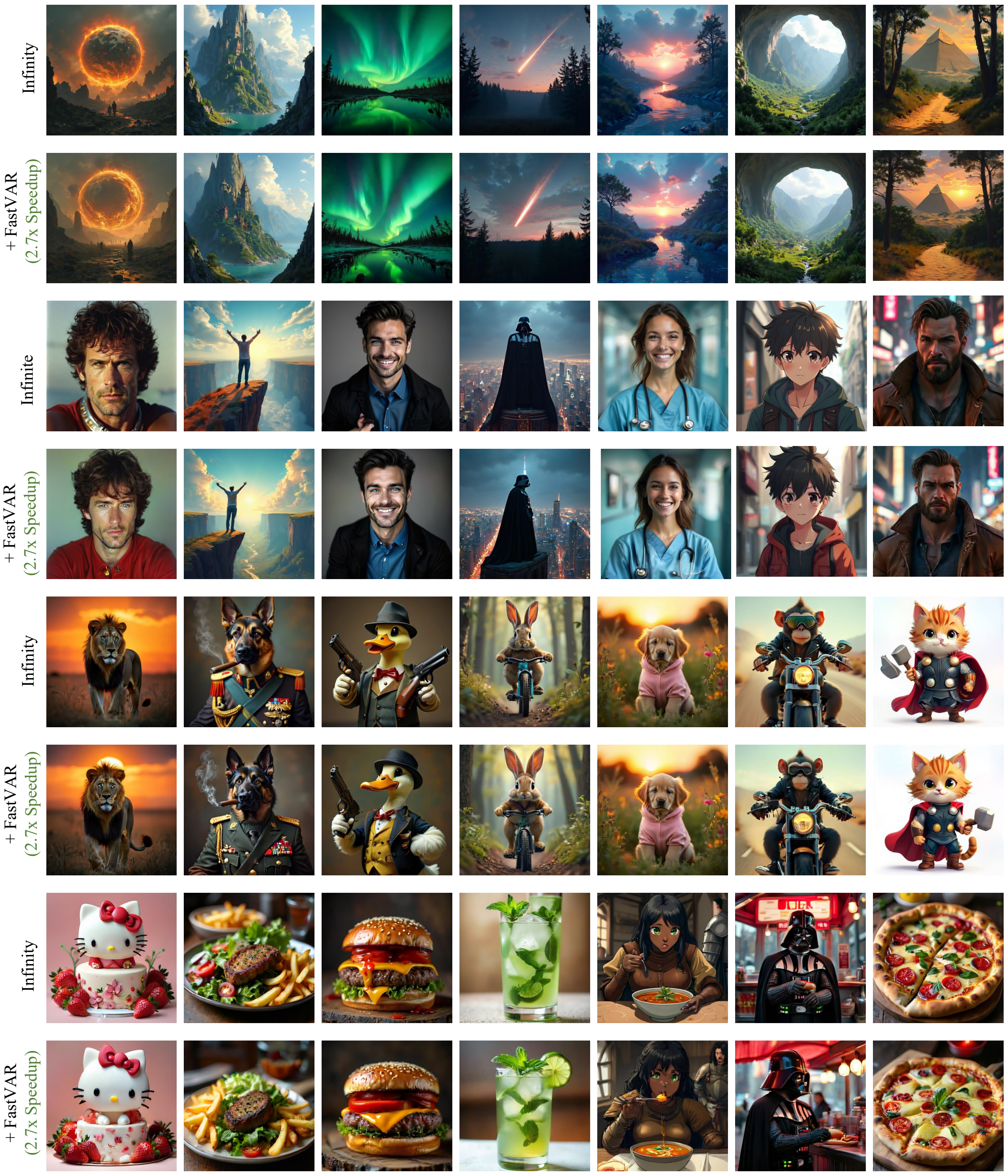

Figure A.3. More qualitative comparison with the Infinite model on the MJHQ30K dataset.

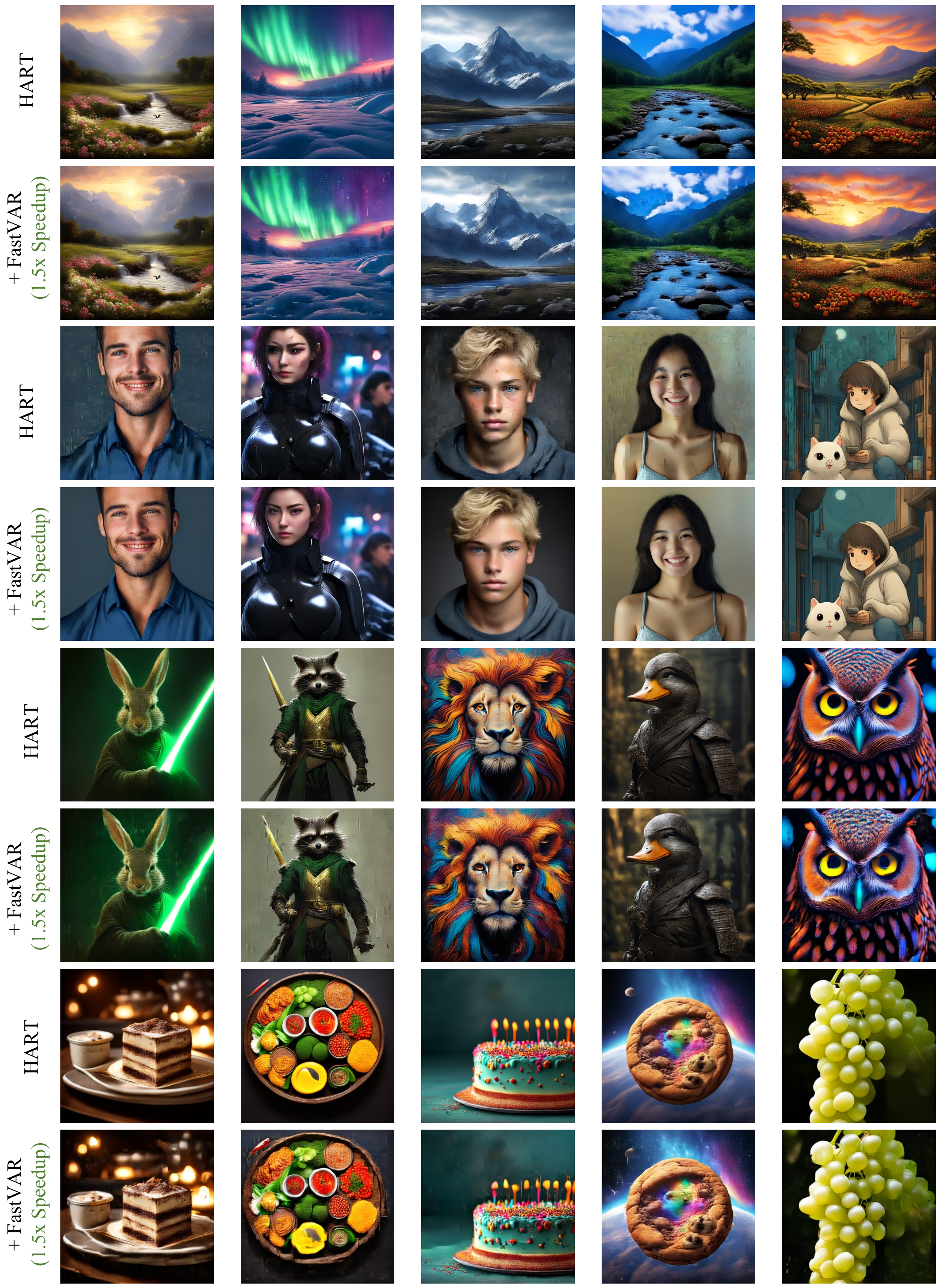

Figure A.4. More qualitative comparison with the HART backbone on the MJHQ30K dataset.

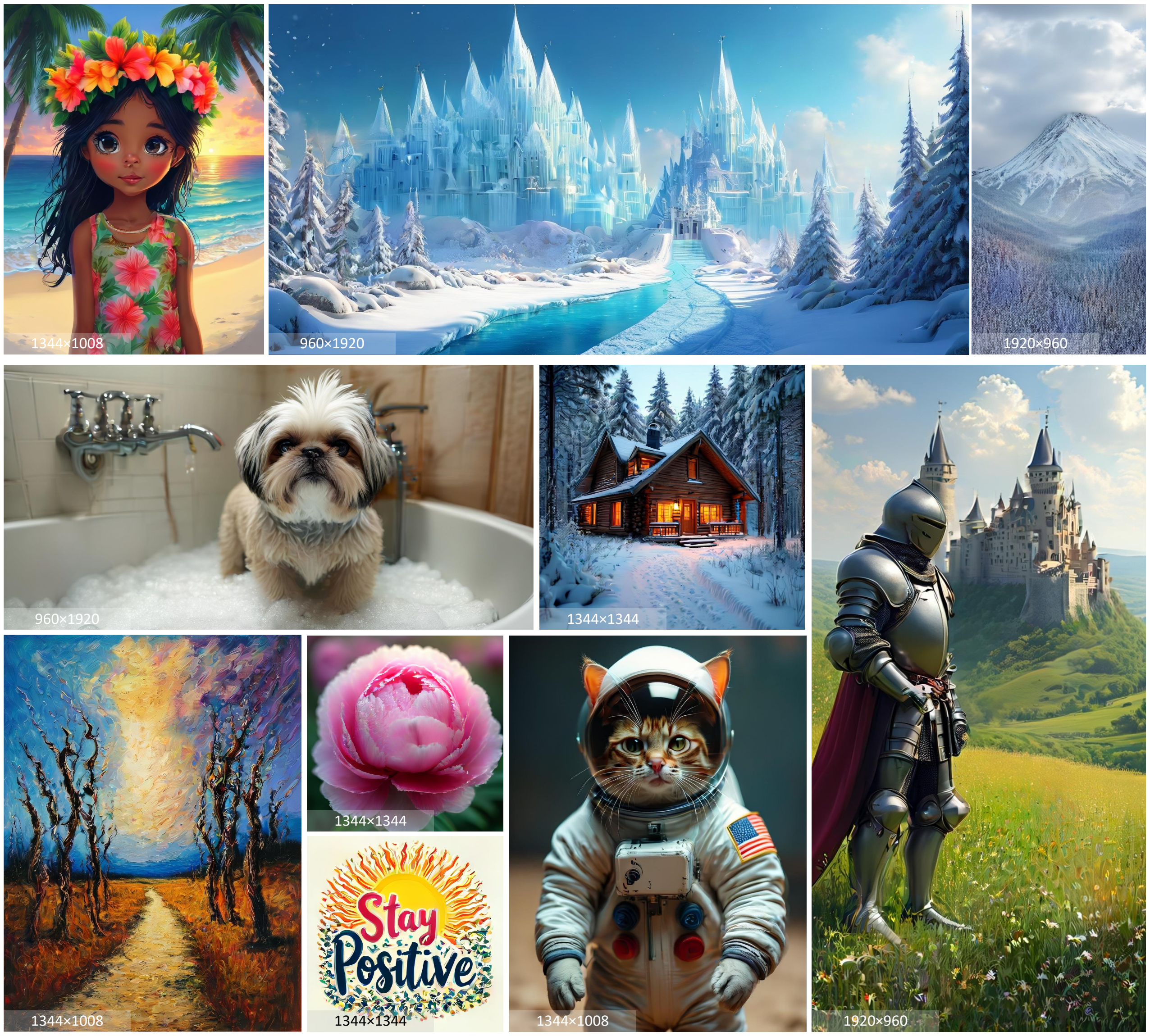

Figure A.5. Moreover generation results of the high-resolution image synthesis with Infinite+FastVAR. The images are scaled for better presentation.

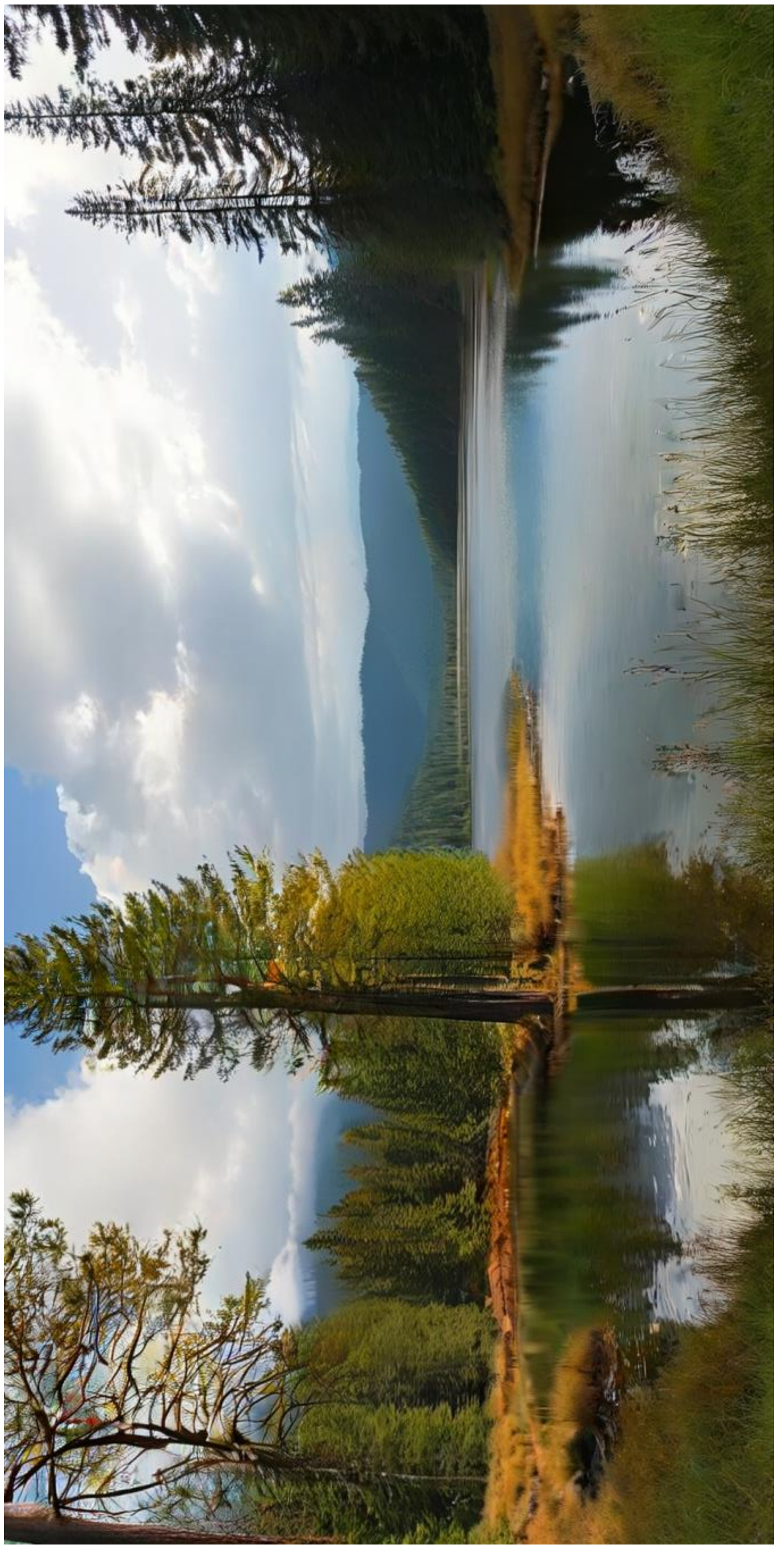

Figure A.6. Moreover generation results of the high-resolution image synthesis with Infinite+FastVAR.

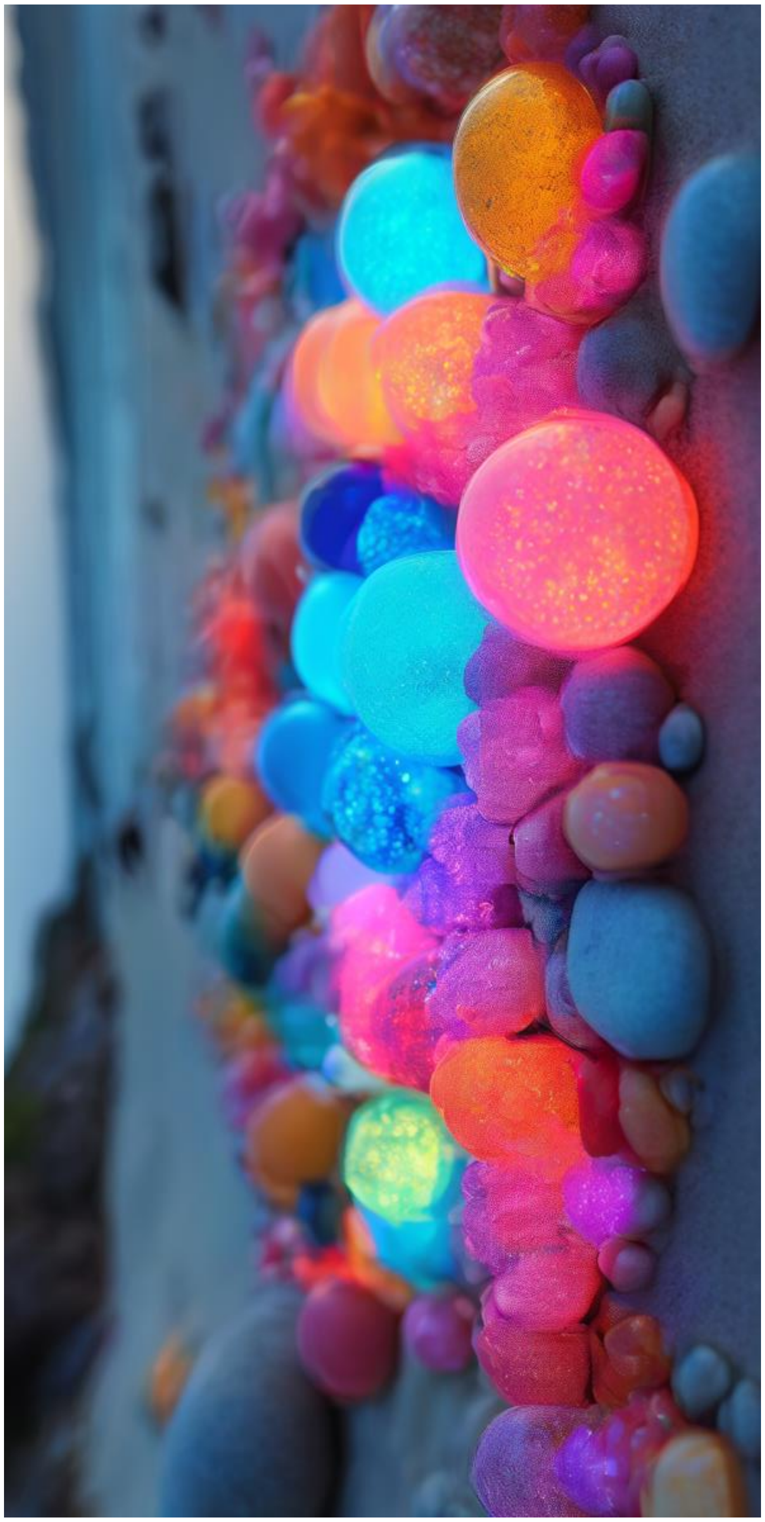

Figure A.7. Moreover generation results of the high-resolution image synthesis with Infinite+FastVAR.