

# VORTA: Efficient Video Diffusion via Routing Sparse Attention

## Appendix

We organize the appendix as follows:

### Methodological and implementation details:

- Appendix A.1: Methodological details
- Appendix A.2: Implementation details

### Additional experimental results and findings:

- Appendix B.1: Hyperparameters analysis
- Appendix B.2: Runtime analysis
- Appendix B.3: Attention pattern analysis
- Appendix B.4: Qualitative comparison
- Appendix B.5: VBench dimensional scores

### Limitations and border impact:

- Appendix C.1: Failure cases
- Appendix C.2: Border impact

## A Methodology and implementation details

### A.1 Methodological details

**Sliding tile attention.** Section 3.2 introduced the sliding tile attention [43], the sparse attention pattern we used to model local interactions. To keep the main text concise, we explained only the 1D case. Here, we present the 3D version of the sliding tile attention mask used in our implementation, as detailed in Algorithm 2.

**Bucketed coreset selection (BCS).** Section 3.2 also introduced the bucketed coreset selection (BCS) method, designed to reduce the computational overhead of modeling long-range interactions. In the naive setting, computing pairwise similarities across an input sequence of length  $L$  incurs a quadratic complexity of  $\mathcal{O}(L^2)$ . Bolya and Hoffman [2], Sun et al. [30] select a subset of tokens (e.g., 25%) as anchors and computing similarities between these anchors and the remaining tokens. This reduces the number of comparisons but still results in  $\mathcal{O}((3L/4) \cdot (L/4)) = \mathcal{O}(L^2)$  complexity.

In contrast, BCS achieves linear complexity  $\mathcal{O}(L)$  by employing a bucketing strategy. Each bucket, of size  $(t, h, w)$ , computes similarities between a central token and the remaining  $(thw - 1)$  tokens, yielding a per-bucket cost of  $\mathcal{O}(thw)$ . With  $L/(thw)$  such buckets, the total cost becomes  $\mathcal{O}((L/(thw)) \cdot thw) = \mathcal{O}(L)$ . No inter-bucket comparisons are performed, and the empirical results in Section 4.2 show that this approach is effective enough in selecting a representative subset of tokens for long-range interactions. Compared to global pairwise methods [2, 30], which require  $\mathcal{O}(L^2)$  operations, BCS offers a substantially more efficient  $\mathcal{O}(L)$  alternative.

### A.2 Implementation details

**Implementation details for our VORTA.** Besides the implementation introduced in Section 4.1, we also provide the implementation details for our VORTA model.

For router optimization, we train on the Mixkit dataset [15] for 100 steps using a learning rate of  $10^{-2}$  and a batch size of 4. Training completes in approximately one day using two H100 GPUs.

The sliding attention branch employs a window size of  $(18, 27, 24)$ , while the coreset attention branch uses a bucket size of  $(2, 3, 2)$  with a coreset ratio of  $r_{\text{core}} = 0.5$ .

---

**Algorithm 2** 3D sliding tiled attention mask

---

**Input:** video size  $\mathbf{v} = (F, H, W)$ , tile size  $\mathbf{t} = (t_F, t_H, t_W)$ , window size  $\mathbf{w} = (w_F, w_H, w_W)$ .  
1: *// Total sequence length for self-attention*  
2:  $L \leftarrow F \times H \times W$   
3:  $\mathbf{M} \leftarrow \mathbf{0}_{L \times L}$   
4: *// The attention mask between the  $i$ -th query and the  $j$ -th key*  
5: **for**  $i \leftarrow 1$  to  $L$  **do**  
6:   **for**  $j \leftarrow 1$  to  $L$  **do**  
7:     *// Get tile id from token id*  
8:      $q_F, q_H, q_W \leftarrow \text{get\_tile\_id\_3d}(i, \mathbf{v}, \mathbf{t})$   
9:      $k_F, k_H, k_W \leftarrow \text{get\_tile\_id\_3d}(j, \mathbf{v}, \mathbf{t})$   
10:    *// All queries within the same window share the same tile id as the window center tile id*  
11:     $q_F, q_H, q_W \leftarrow \text{get\_window\_center\_id}(q_F, q_H, q_W, \mathbf{w})$   
12:    *// true if key tile is within the local window of query tile*  
13:     $m \leftarrow \text{bool}(\text{abs}(q_F - k_F) \leq w_F/2)$   
14:     $m \leftarrow m \text{ and } \text{bool}(\text{abs}(q_H - k_H) \leq w_H/2)$   
15:     $m \leftarrow m \text{ and } \text{bool}(\text{abs}(q_W - k_W) \leq w_W/2)$   
16:    *// Save the mask*  
17:     $\mathbf{M}[i, j] \leftarrow m$   
18:   **end for**  
19: **end for**  
**Output:** sliding tile attention mask  $\mathbf{M}$ .

---

854 Regarding video configurations during both training and inference, HunyuanVideo [13] utilizes  
855 videos with 117 frames at a resolution of  $720 \times 1280$  (720p), while Wan 2.1 [32] uses 77-frame  
856 videos at the same resolution. For rendering, HunyuanVideo outputs videos at 24 frames per second  
857 (FPS), whereas Wan 2.1 generates videos at 15 FPS, as specified in their respective repositories.

858 **Implementation details for baseline methods.** To evaluate efficiency, the PAB [47] is tested on  
859 720p videos from HunyuanVideo [13], aligning with the setup used in other methods. However,  
860 processing at this resolution with PAB requires over 80 GB of GPU memory. To address this  
861 limitation, we apply sequential CPU offloading [31]. For a fair comparison, we exclude the latency  
862 caused by model loading and offloading from the reported results. Despite this, the end-to-end  
863 runtime with CPU offloading exceeds 2000 seconds per video, which is substantially slower than the  
864 original pretrained model and offers no practical efficiency advantage. The original STA kernel  
865 supports video generation only at a fixed resolution of  $768 \times 1280$ , which exceeds the 720p  
866 resolution used by the pretrained model. To ensure consistency in evaluation, we reimplemented the  
867 kernel using FlexAttention to support 720p video generation.

868 **Evaluation metrics.** To evaluate how the generated outputs differ from those of pretrained models,  
869 we use LPIPS [45] as a reference metric. However, since the pretrained models do not represent the  
870 ground truth, divergence from them does not necessarily indicate degraded performance. Multiple  
871 generated outputs may be equally valid, provided they align with the input prompts. LPIPS is only  
872 used as a comparative reference. The primary evaluation of video quality and prompt alignment  
873 should be based on VBench [9] scores.

874 **The improved performance of acceleration methods.** Interestingly, despite using less  
875 computation, VORTA slightly outperforms the original pretrained models in terms of VBench score.  
876 Similar findings have been reported in other acceleration methods [14, 36]. A possible explanation  
877 lies in the redundancies present in overparameterized models, which can introduce marginal negative  
878 effects. By pruning these redundancies, these acceleration methods may contribute to slight  
879 performance gains, although they are not intended for this purpose.

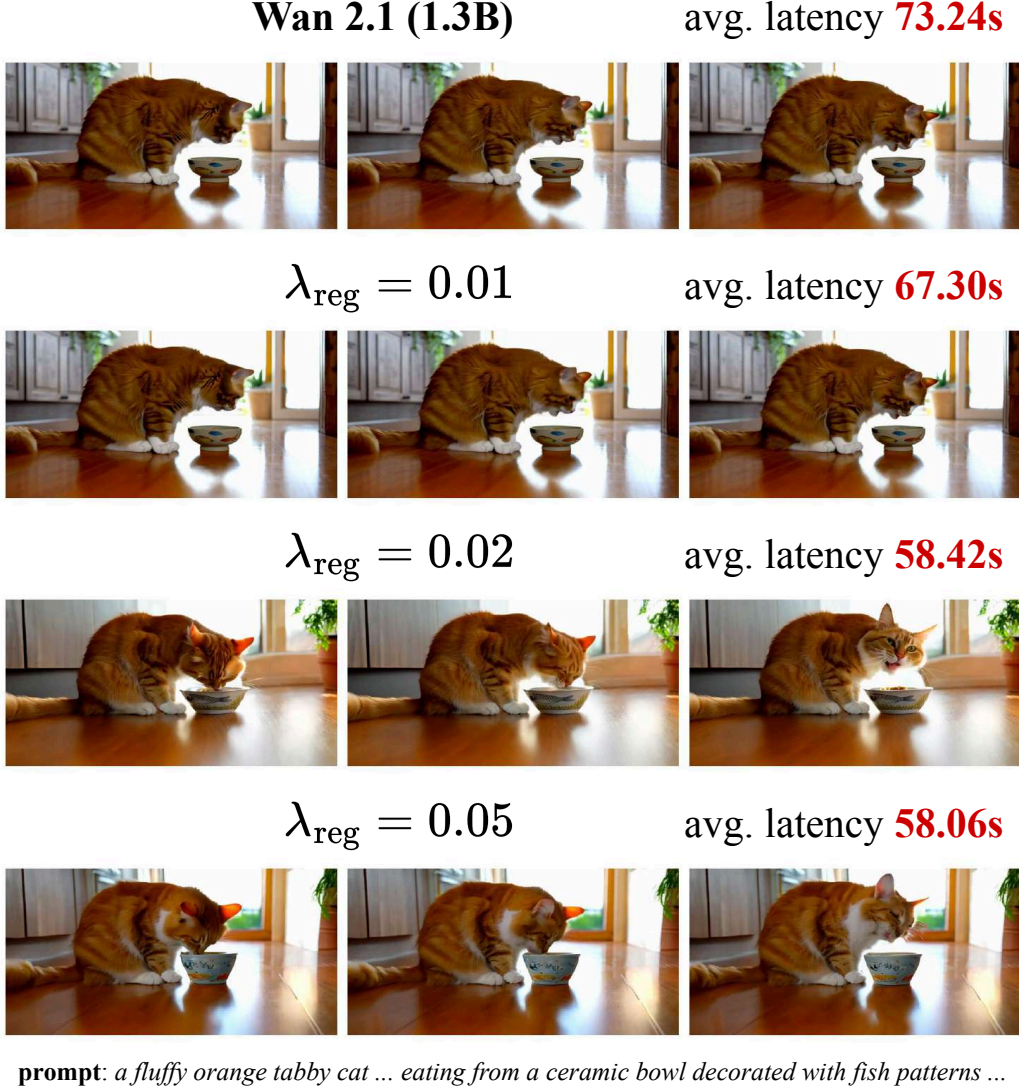


Figure 9: Qualitative evaluation of varying the regularization weight  $\lambda_{\text{reg}}$ .

## 880 B Additional experimental results and findings

### 881 B.1 Hyperparameters analysis

882 Figure 9 shows the effect of varying the regularization weight  $\lambda_{\text{reg}}$ . When  $\lambda_{\text{reg}}$  is small ( $\lambda_{\text{reg}} = 0.01$ ),  
 883 the speedup is limited, and in most scenarios, the router tends to select full attention. In contrast, with  
 884 a large  $\lambda_{\text{reg}} = 0.05$ , the video exhibits noticeable distortion (e.g., the cat’s head in the final frame). A  
 885 moderate value of  $\lambda_{\text{reg}} = 0.02$  achieves a good trade-off between acceleration and output quality.

886 Figure 10 provides further analysis of alternative pooling strategies in the coreset attention branch.  
 887 Using average pooling with a  $r_{\text{core}} = 50\%$  coreset ratio significantly underperforms compared to  
 888 BCS pooling, primarily because it lacks a selection mechanism. As the coreset ratio  $r_{\text{core}}$  increases,  
 889 the VBench score improves; however, a ratio of  $r_{\text{core}} = 50\%$  is sufficient to achieve strong  
 890 performance. Higher ratios yield marginal performance gains while introducing additional  
 891 computational overhead during generation.

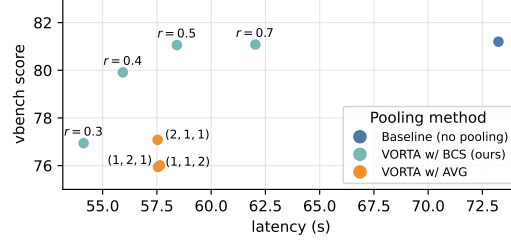


Figure 10: Effect of varying the coreset size in BCS and the kernel size in average pooling.

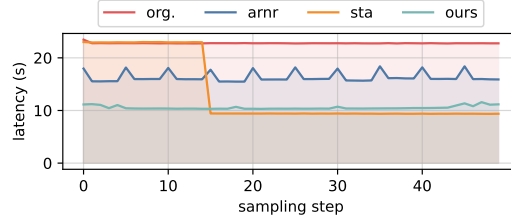


Figure 11: Per-step sampling latency on HunyuanVideo [13]. ARnR [30] yields smaller speedups overall. STA [43] shows no speedup in early timesteps.

## B.2 Runtime analysis

Section 4.2 analyzed the average runtime (over diffusion steps) of each component, as shown in Figure 7. Figure 11 further presents the runtime breakdown per diffusion step. Overall, ARnR exhibits relatively high runtime, primarily due to limited acceleration in attention operations. Additionally, its periodic global similarity computation, with  $\mathcal{O}(L^2)$  complexity, introduces a bottleneck. This is evident from latency spikes every five steps. In contrast, STA shows significantly higher runtime during the initial 15 steps due to the lack of early-stage acceleration, leading to reduced overall efficiency. Our VORTA achieves near-constant runtime across all steps, demonstrating stable and efficient performance.

## B.3 Attention pattern analysis

Figures 12 and 13 show the router gate values for Hunyuan [13] and Wan 2.1 (14B) [32], respectively, as supplementary results for Section 3.1.

## B.4 Qualitative comparison

In addition to the qualitative results in Figures 1 and 6 and the quantitative results in Table 1, we present further visualizations for Wan 2.1 (14B) in Figure 14. VORTA inherits the strong performance of the pretrained ViTs while offering a significant speedup.

## B.5 VBench dimensional scores

VBench evaluates the generated videos across 16 dimensions. Due to space constraints, we report only three aggregated scores in Table 1, with the complete set of scores provided in Table 3 for completeness.

## C Limitations and border impact

### C.1 Failure cases

VORTA does not modify the pretrained parameters of VDiTs, and therefore its performance inherently depends on the quality of the underlying pretrained model. As illustrated in Figure 15,

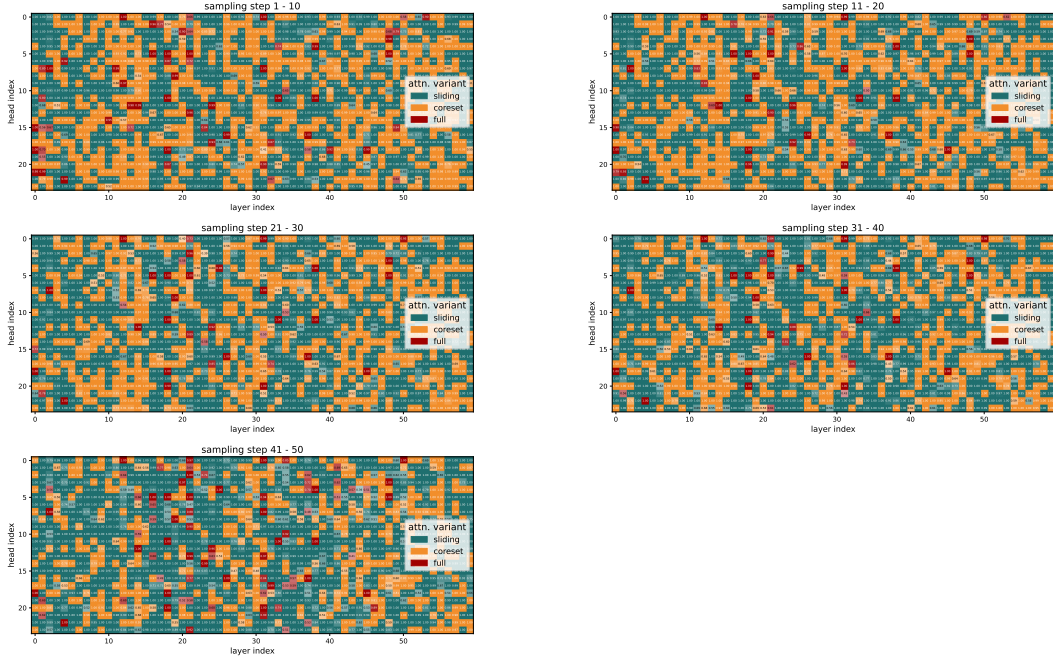


Figure 12: Optimized gate values for HunyuanVideo [13]. The 50 sampling steps are grouped into 5 equal intervals, and the averaged gate value within each interval is reported. Each color corresponds to a distinct attention branch, with color intensity indicating confidence.

Table 3: Quantitative results for VBench dimensions [9] for HunyuanVideo [13] and Wan (14B) [32].

Method	Aesthetic Quality $\uparrow$	Appearance Style $\uparrow$	Background Consistency $\uparrow$	Color $\uparrow$	Dynamic Degree $\uparrow$	Human Action $\uparrow$	Imaging Quality $\uparrow$	Motion Smoothness $\uparrow$
HunyuanVideo	62.05	77.88	93.80	91.35	38.89	96.00	63.33	96.98
+ ARnR	62.58	80.36	93.99	87.54	39.77	93.73	63.21	96.12
+ STA	63.87	79.48	94.47	86.39	39.58	97.00	61.92	96.79
+ PAB	63.53	75.85	94.82	87.92	36.11	98.00	63.40	97.34
+ VORTA	62.33	80.63	94.62	92.13	37.50	97.00	63.16	95.84
+ VORTA & PAB	63.16	80.43	95.22	90.92	36.31	97.50	62.05	96.18
+ PCD	62.69	76.21	94.44	85.94	31.94	94.00	63.08	96.46
+ VORTA & PCD	61.43	76.92	94.87	89.33	35.42	98.00	62.42	95.11
Wan 2.1 (14B)	63.33	82.12	94.55	83.26	37.50	99.00	63.99	91.60
+ VORTA	63.79	83.32	95.28	87.04	39.58	100.00	63.07	92.17

Method	Mutiple Objects $\uparrow$	Objects Class $\uparrow$	Overall Consistency $\uparrow$	Scene $\uparrow$	Spatial Relationship $\uparrow$	Subject Consistency $\uparrow$	Temporal Flickering $\uparrow$	Temporal Style $\uparrow$
HunyuanVideo	52.21	84.41	74.30	65.59	78.06	90.30	98.57	69.61
+ ARnR	52.09	87.49	69.88	69.21	80.89	90.55	99.28	70.70
+ STA	56.55	87.82	75.01	64.08	80.60	88.12	98.40	69.59
+ PAB	56.86	84.97	74.97	63.91	78.34	90.89	98.61	70.50
+ VORTA	58.00	89.56	74.62	61.87	78.30	92.43	98.47	69.44
+ VORTA & PAB	59.17	89.90	75.82	62.99	77.43	92.27	97.96	70.29
+ PCD	54.04	81.88	75.07	66.03	74.48	90.64	97.37	70.52
+ VORTA & PCD	51.07	85.05	74.93	67.09	73.70	91.34	97.50	70.69
Wan 2.1 (14B)	74.62	86.47	75.49	64.70	79.16	91.21	97.69	71.54
+ VORTA	75.76	88.45	75.09	63.29	80.28	90.76	97.78	70.70

when the pretrained VDiTs fail to generate high-quality videos, resulting in distortions or non-physical outputs, VORTA exhibits similar deficiencies. In some cases, it even produces outputs of lower quality than the original VDiTs. This degradation is attributed to computations on erroneous generations, which amplify distortions in the resulting videos.



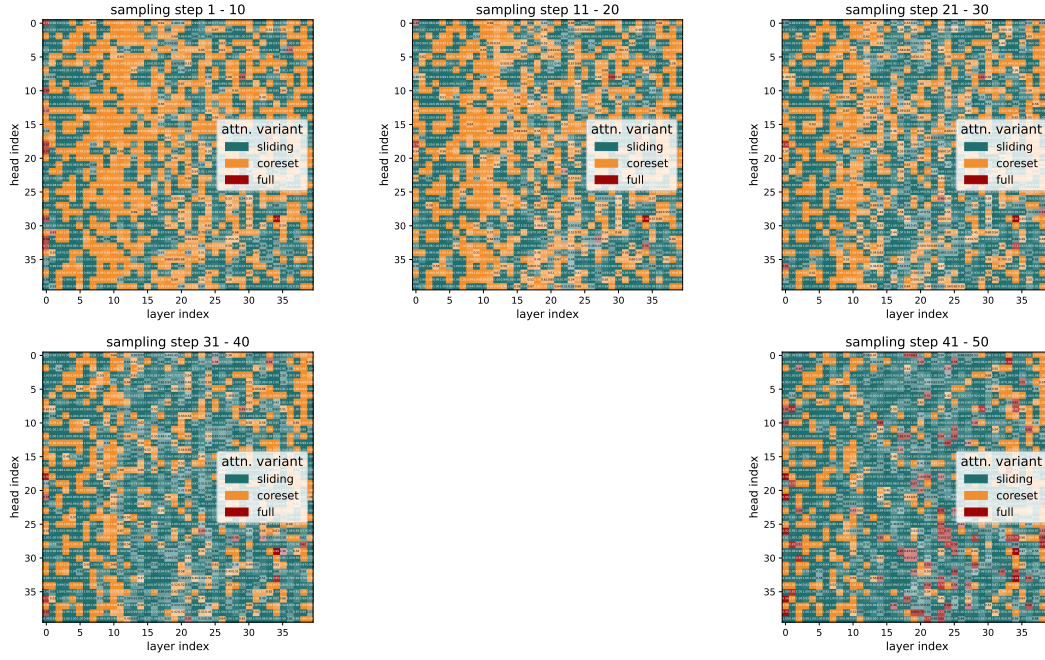


Figure 13: Optimized gate values for Wan 2.1 (14B) [32].

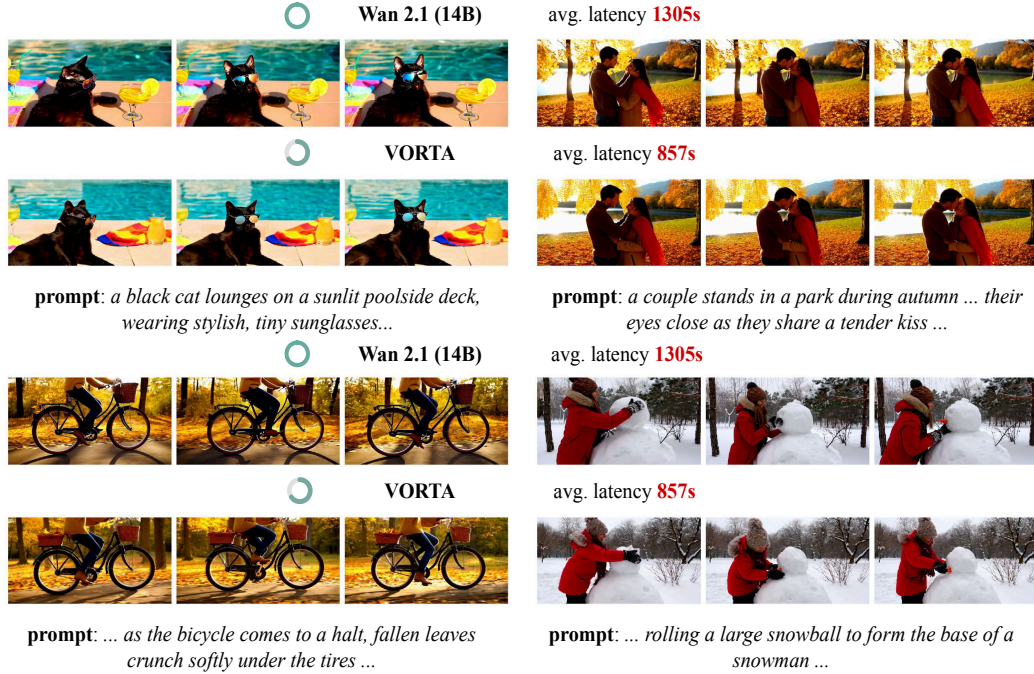


Figure 14: Qualitative comparison on Wan 2.1 (14B) [32].

## 920 C.2 Border impact

921 Generative models pose risks of producing biased, privacy-invasive, or harmful content. While our  
 922 method accelerates video generation, it may also propagate such risks. It is imperative that  
 923 researchers, developers, and platform providers actively assess and mitigate these potential harms to  
 924 promote responsible use.

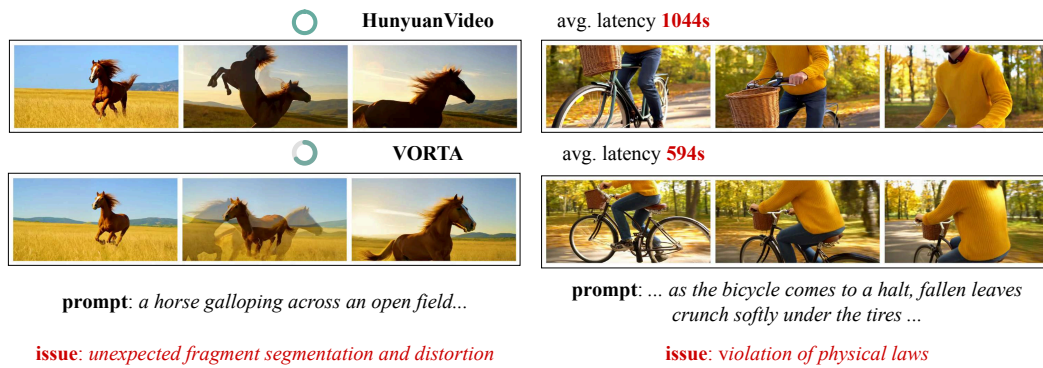


Figure 15: Failure cases. When the pretrained model exhibits distortions or non-physical phenomena, VORTA inherits these issues. We refer the reader to the supplementary video for a more illustrative example.