LongCat-Video Technical Report

Meituan LongCat Team

ABSTRACT

Video generation is a critical pathway toward world models, with efficient long video inference as a key capability. Toward this end, we introduce LongCat-Video, a foundational video generation model with 13.6B parameters, delivering strong performance across multiple video generation tasks. It particularly excels in efficient and high-quality long video generation, representing our first step toward world models. Key features include: **Unified architecture for multiple tasks**: Built on the Diffusion Transformer (DiT) framework, LongCat-Video supports *Text-to-Video*, *Image-to-Video*, and *Video-Continuation* tasks with a single model; **Long video generation**: Pretraining on *Video-Continuation* tasks enables LongCat-Video to maintain high quality and temporal coherence in the generation of minutes-long videos; **Efficient inference**: LongCat-Video generates $720p$, $30fps$ videos within minutes by employing a coarse-to-fine generation strategy along both the temporal and spatial axes. Block Sparse Attention further enhances efficiency, particularly at high resolutions; **Strong performance with multi-reward RLHF**: Multi-reward RLHF training enables LongCat-Video to achieve performance on par with the latest closed-source and leading open-source models. Code and model weights are publicly available to accelerate progress in the field.

GitHub: https://github.com/meituan-longcat/LongCat-Video

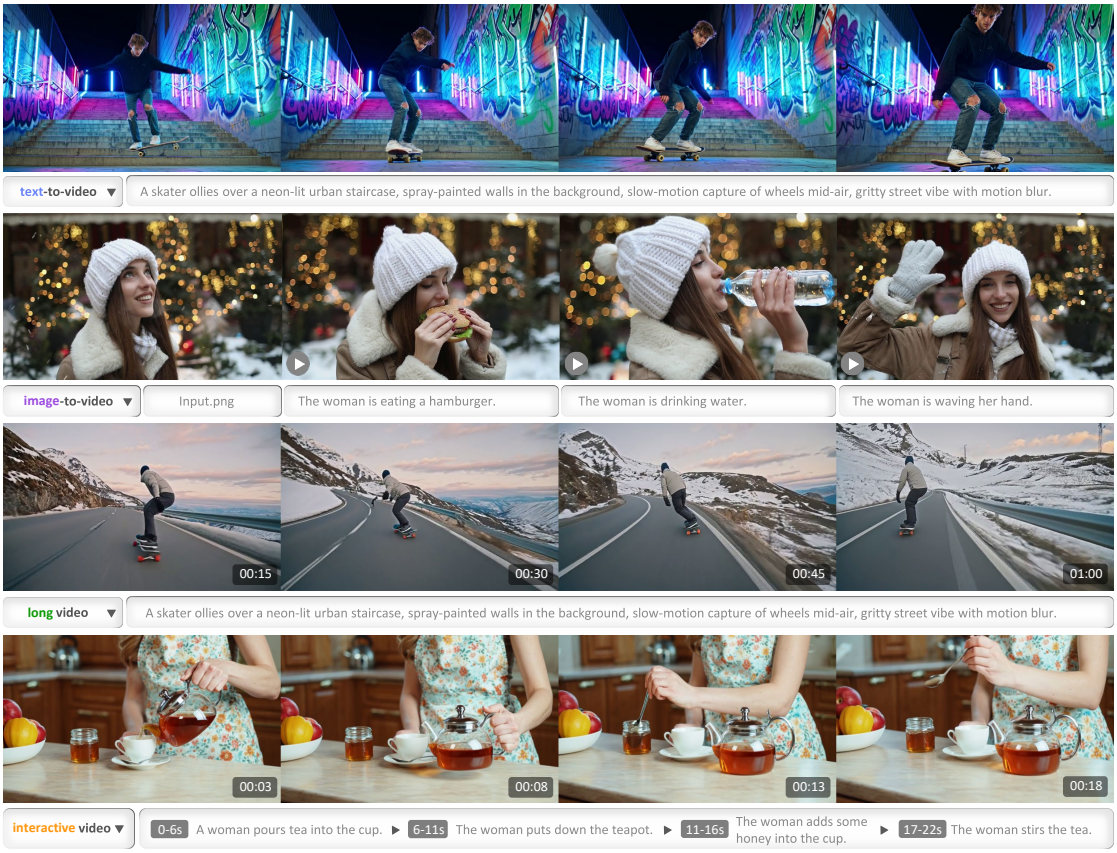

Figure 1: Examples on *Text-to-Video*, *Image-to-Video* and *Video-Continuation* tasks. *Video-Continuation* supports long video generation as well as interactive generation with multiple instructions. We unify these tasks with a single model.

Contents

1 Introduction

World models, which aim to understand, simulate, and predict complex real-world environments, constitute an important foundation for applying artificial intelligence in real-world scenarios. Video generation models serve as a critical pathway toward world models by compressing geometric, semantic, physical, and other forms of knowledge through video generation tasks, thereby enabling effective simulation and prediction of the physical world. Notably, efficient long video generation is particularly essential.

Over the past years, diffusion modeling and video generation have achieved remarkable breakthroughs. The quality of generated videos, instruction-following capabilities, and motion realism have all seen substantial improvements. Commercial products—such as Veo [Google, 2024], Sora [OpenAI, 2024], Seedance [Gao et al., 2025], Kling [Kuaishou, 2024], Hailuo [MiniMax, 2024], PixVerse [PixVerse, 2024] and others—and open-source solutions—such as Wanx [Wan et al., 2025], HunyuanVideo [Kong et al., 2024], Step-Video [Ma et al., 2025a], CogVideoX [Yang et al., 2024] and others—have demonstrated outstanding performance across various dimensions. These works are increasingly being integrated into content production pipelines, with widespread applications ranging from user-generated video content creation to film production, and from entertainment content creation to advertising creativity. Video generation [NVIDIA] is also establishing a robust foundation for world model applications such as autonomous driving and embodied AI, with the ongoing improvements in physical simulation and long video generation. These developments are further accelerating the deployment and evolution of intelligent systems in complex real-world scenarios.

In this report, we introduce LongCat-Video, a foundational video generation model with 13.6B parameters that delivers strong performance across general video generation tasks, particularly excelling in efficient, high-quality long video generation. LongCat-Video serves as a robust general-purpose model and marks our first step toward world models. Key features include:

- **Unified architecture for multiple tasks** Different use cases demand distinct video generation functionalities. For example, *Text-to-Video* is widely adopted for creative content production, while *Image-to-Video* is preferred when precise content control is required. LongCat-Video unifies *Text-to-Video*, *Image-to-Video*, and *Video-Continuation* tasks within a single video generation framework, distinguishing them by the number of conditioning frames—zero for *Text-to-Video*, one for *Image-to-Video*, and multiple for *Video-Continuation* generation. Through a multi-task training strategy, LongCat-Video natively supports all these tasks and delivers strong performance across them.
- **Long video generation** Long-video generation is critical for applications such as digital humans, embodied AI, and other complex tasks that require extended temporal coherence, which is also a key capability for world model applications. However, this remains a challenging problem due to generation error accumulated over time. While various methods [Chen et al., 2025] exist to finetune existing video foundation models for improved long-video generation, LongCat-Video is natively pretrained on *Video-Continuation* tasks, enabling it to produce minutes-long videos without color drifting or quality degradation.
- **Efficient inference** The computational cost of video generation increases substantially with higher video resolutions and frame rates, as attention complexity grows quadratically with the number of tokens. Inspired by Seedance [Gao et al., 2025], Hailuo [MiniMax, 2024] and related works, LongCat-Video adopts a coarse-to-fine strategy: videos are first generated at $480p, 15fps$, and subsequently refined to $720p, 30fps$. For high-resolution generation, we train an expert LoRA module to effectively leverage the base model's knowledge. Furthermore, we implement a block sparse attention mechanism, reducing attention computations to less than 10% of those required by standard dense attention. This design significantly enhances efficiency in the high-resolution refinement stage.
- **Strong performance with multi-reward RLHF** In post-training, we employ Group Relative Policy Optimization (GRPO) [Guo et al., 2025] method to further enhance model performance using multiple rewards. Comprehensive evaluations on both internal and public benchmarks, using human and model-based annotations, demonstrate that LongCat-Video achieves performance comparable to leading open-source video generation models as well as the latest commercial solutions. We are releasing the code, model weights, and key modules, including block sparse attention, to the community. We believe this work will help advance the development of video generation technology in both academic and industrial domains.

2 Data

Training a high-quality video generation model requires a large-scale, diverse, and high-quality dataset. To meet these requirements, we have developed a comprehensive data curation pipeline, as illustrated in Figure 2, which consists of two main stages: 1) **Data Preprocessing Stage**: This stage includes the acquisition of various data sources, deduplication, video transition segmentation, and black border cropping, ensuring the diversity and integrity of the

collected videos; 2) **Data Annotation Stage**: In this stage, video clips are annotated with multiple metrics and attributes to enrich the dataset and facilitate downstream tasks. We introduce the data curation pipeline in Section 2.1 and present the distribution of the curated training data in Section 2.2.

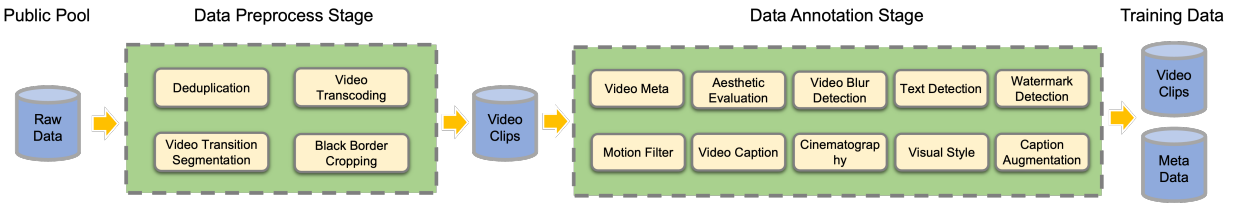

Figure 2: Overview of data curation pipeline. The data preprocessing stage extracts well-segmented video clips from raw source videos in the data pool. In the data annotation stage, each video clip is annotated with a variety of attributes, forming a comprehensive metadata database. This metadata database enables the convenient and flexible assembly of training datasets to support various training stages and objectives.

2.1 Data Curation Pipeline

2.1.1 Data Preprocessing Stage

We collect raw video data from a variety of sources. To eliminate redundant content, we perform deduplication using source video IDs and MD5 hashes. PySceneDetect [Castellano] and an in-house trained TransNetV2 [Souček and Lokoč, 2020] are employed to segment source videos into training-friendly clips while maintaining content consistency within each fragment—an essential factor for effective video generation model training. Additionally, black border cropping is applied using FFMPEG [FFmpeg Developers, 2014] during the video transition segmentation process to further improve data quality. Finally, all processed video clips are compressed and packaged, facilitating subsequent data cleaning and efficient data loading during training.

2.1.2 Data Annotation Stage

To meet the video filtering requirements at different training stages, we annotate video clips with a range of metrics and store them as a comprehensive metadata library. These metrics include basic video metadata (such as duration, resolution, frame rate, and bitrate), aesthetic score, blur score, text coverage, watermark detection, etc. Additionally, motion information is evaluated using extracted video optical flow to assess video dynamics, enabling us to filter out clips with minimal motion features. This metadata library facilitates flexible and targeted dataset construction for various training objectives.

The consistency between captions and video content is crucial for ensuring that the video generation model can accurately follow instructions. As illustrated in Figure 3, we decompose the video information and utilize multiple models to annotate various aspects of the video content.

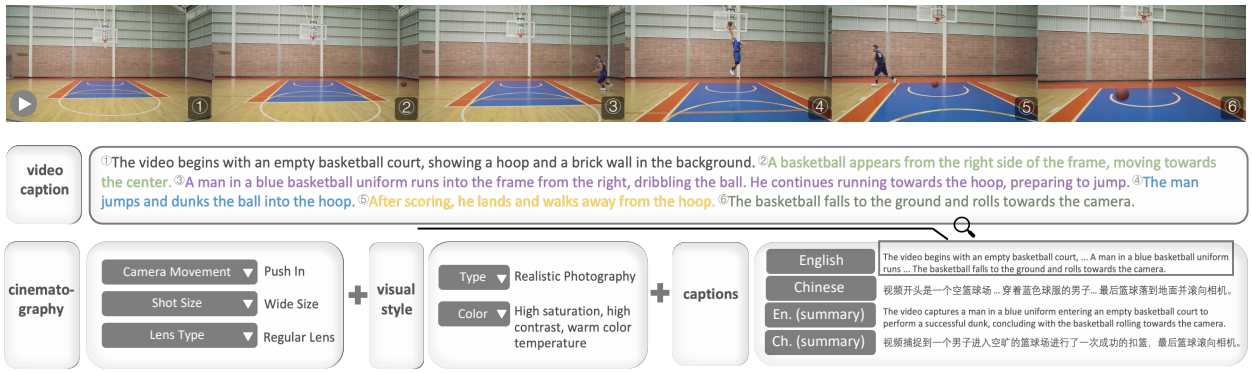

Figure 3: Overview of the video captioning workflow. The main content of each video is captured by a basic captioning model, and complemented by additional models that extract attributes such as cinematography and visual style. These elements are integrated to produce varied and informative captions, enhancing the quality and diversity of training data.

Basic video caption Videos contain complex information, including both appearance features and the temporal dynamics of actions and events. Many multimodal models are good at describing static images, but struggle to accurately capture actions and understand temporal relationships. We fine-tune the LLaVA-Video model [Zhang et al., 2024] using in-house constructed synthetic video-text pairs, improving its ability to describe both visual and temporal aspects. We also found that the amount and quality of temporal action annotations in the dataset are key to enhancing temporal understanding. To further improve this, we collected more videos with rich temporal events and used annotated data from Tarsier2 [Yuan et al., 2025a] for fine-tuning. This significantly boosts the model's ability to describe and understand temporal dynamics in videos.

Cinematography and visual style Cinematography in video includes elements such as camera movements, shot sizes, and lens types. To enable automatic recognition of camera movements, we annotated a dataset with categories including pan, tilt, zoom, and shark, and trained a dedicated classifier. The annotation of shot sizes and lens types requires image-level semantic understanding; for this purpose, we employ the Qwen2.5VL model [Bai et al., 2025], which excels at image analysis and accurately identifies these attributes. Visual style covers a broad range of characteristics, including general visual types such as realism, 2D anime, and 3D cartoon, as well as finer-grained attributes like color tones. For visual style annotation, we likewise utilize Qwen2.5VL, leveraging its strong image understanding capabilities to capture and interpret these diverse visual features.

Caption augmentation To improve the model's robustness in handling diverse textual inputs, we enrich video captions through a variety of augmentation techniques. These include translating captions between Chinese and English to support both languages, as well as generating concise summaries to diversify caption styles. As illustrated in Figure 3, we further enhance caption diversity by randomly selecting elements from cinematography and visual style categories and integrating them with the augmented captions. This strategy ensures that each video clip is paired with multiple styles of textual descriptions, significantly increasing dataset diversity and enhancing the adaptability of the video generation model.

2.2 Data Distribution

Figure 4: We apply text embedding to video captions and perform clustering analysis. An LLM summarizes each cluster and assigns tags, enabling unsupervised categorization of the dataset.

As shown in Figure 4, we categorize video clips into several content types by performing cluster analysis on text embedding vectors derived from their captions. (e.g., personal interactions, artistic performances, natural landscapes, etc.). We then assess the data volume and distribution density for each category to evaluate the overall uniformity of the dataset. Based on this analysis, we implement targeted data supplementation or rebalancing strategies as needed. This systematic approach allows for dynamic and precise allocation of data subsets tailored to the specific requirements and objectives of different training phases, thereby optimizing the model training workflow.

3 Method

3.1 Model Architecture

Network Architecture We employ a standard DiT [Peebles and Xie, 2023] architecture with single-stream transformer blocks. Each block consists of a 3D self-attention layer, a cross-attention layer for text conditioning, and a Feed-Forward Network (FFN) with SwiGLU [Shazeer, 2020]. For modulation, we utilize AdaLN-Zero [Peebles and Xie, 2023], where each block incorporating a dedicated modulation MLP. To enhance training stability, RMSNorm [Zhang and Sennrich, 2019] is applied as QKNorm [Henry et al., 2020] within both the self-attention and cross-attention modules. Additionally, 3D RoPE [Su et al., 2024] is adopted for positional encoding of visual tokens. Detailed model specifications are summarized in Table 1.

Table 1: Model specifications of LongCat-Video.

Num. of Layers	Model Hidden Size	FFN Hidden Size	Num. of Attn. Heads	AdaLN Embedding Size
48	4096	16384	32	512

VAE and Text embedder For latent compression, we employ WAN2.1 VAE [Wan et al., 2025] to convert video pixels into latent tokens, achieving a compression ratio of $4 \times 8 \times 8$ along the temporal, height, and width dimensions. In addition, a patchify operation within the DiT model further compresses the latents with an additional $1 \times 2 \times 2$ ratio. As a result, the overall compression ratio from pixels to latents reaches $4 \times 16 \times 16$. For text encoding, we utilize umT5 [Chung et al., 2023], a multilingual text encoder that supports both English and Chinese captions.

3.2 Unified Model for Multiple Tasks

LongCat-Video is a unified video generation framework that supports *Text-to-Video*, *Image-to-Video*, and *Video-Continuation* tasks. We define all these tasks as video continuation, where the model predicts future frames conditioned on a given set of preceding condition frames. The primary difference between all these tasks is the number of condition frames provided, resulting in a hybrid input format for our network.

Unified Input Representation As illustrated in Figure 5, the network input consists of two sequences: the condition sequence $X_{\text{cond}} \in \mathbb{R}^{B \times N_{\text{cond}} \times H \times W \times C}$, which is the noise-free condition frames, and the noisy sequence $X_{\text{noisy}} \in \mathbb{R}^{B \times N_{\text{noisy}} \times H \times W \times C}$, which is the noisy frames to be denoised. Here, N_{cond} and N_{noisy} denote the lengths of the condition and noisy frames. B is the batch size, H and W are the spatial dimensions, and C is the number of channels. These two sequences are concatenated along the temporal axis to form the overall model input $X \in \mathbb{R}^{B \times (N_{\text{cond}}+N_{\text{noisy}}) \times H \times W \times C}$, expressed as $X = [X_{\text{cond}}, X_{\text{noisy}}]$ where $[\cdot]$ denotes the concatenation operation.

Similarly, the timesteps t are partitioned as $t = [t_{\text{cond}}, t_{\text{noisy}}]$, where t_{cond} corresponds to the timesteps of the condition frames and t_{noisy} to those of the noisy frames. This configuration of input sequences and timesteps enables the model to identify different task types based on input patterns. By explicitly structuring both the data and the associated timesteps, the model can effectively distinguish between various generation modes, thereby enhancing its flexibility and performance across a range of generative tasks. For the condition frames, we set t_{cond} to 0 to inject clear, lossless information, while t_{noisy} is sampled within the range $[0, 1]$. During loss computation, the contribution from the condition frames is omitted. The condition sequence remains fixed throughout both training and inference.

Block Attention with KVCache To accommodate the previously described input representation, we have designed a specialized attention mechanism within the unified model architecture, formulated as follows:

$$X_{\text{cond}} = \text{Attention}(Q_{\text{cond}}, K_{\text{cond}}, V_{\text{cond}}), \tag{1}$$

$$X_{\text{noisy}} = \text{Attention}(Q_{\text{noisy}}, [K_{\text{cond}}, K_{\text{noisy}}], [V_{\text{cond}}, V_{\text{noisy}}]), \tag{2}$$

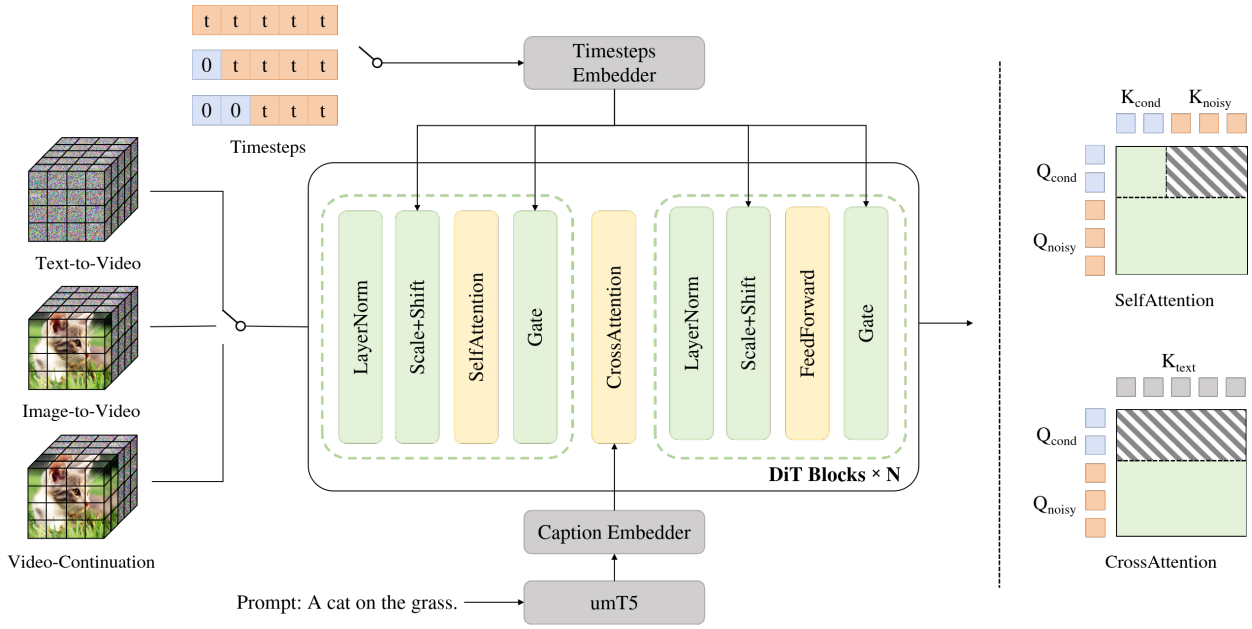

Figure 5: Left: Unified transformer for multiple generation tasks. Our model simultaneously supports *Text-to-Video*, *Image-to-Video* (with a single conditioning frame), and *Video-Continuation* (with multiple conditioning frames) tasks. The timestep configuration is consistent with the input, and the condition part are fixed to zero. Right: Block Causal Attention. In self-attention, the updates of the condition tokens are independent of the noisy tokens. In cross-attention, condition tokens do not participate in cross-attention computation.

where Q_{cond}, K_{cond}, and V_{cond} denote the query, key, and value of the condition tokens, and Q_{noisy}, K_{noisy}, and V_{noisy} correspond to those of the noisy tokens. This design ensures that the condition tokens are not influenced by the noisy tokens. Additionally, X_{cond} does not participate in the cross-attention computation. The computation related to condition tokens depends solely on the input video condition frames, allowing us to cache the KV features of the condition tokens and reuse them across all sampling steps, while ensuring consistency between training and inference. This strategy further enhances the efficiency of long video generation.

3.3 Multi-Reward GRPO Training

3.3.1 GRPO for Flow Matching Modeling

Although GRPO has achieved notable success in large language models [Guo et al., 2025] and image generation [Liu et al., 2025a, Xue et al., 2025, Li et al., 2025, He et al., 2025], its application to video generation is particularly challenging due to slow convergence and complex reward optimization. To overcome these issues, we introduce a series of techniques that significantly enhance both convergence speed and generation quality (Fig. 6) of GRPO for video generation tasks. The theoretical framework is outlined in Appendix A.1.1, and the complete GRPO training procedure is summarized in Algorithm 1.

GRPO as stochastic noise search We observe that GRPO for Flow Matching [Lipman et al., 2022] effectively simulates the gradients $\frac{dR}{dv_\theta}$ using stochastic noise search. In our reweighted version of the policy loss (See Appendix A.1.2 for details.), the gradient of the policy loss with respect to the model parameter θ is as follows:

$$\nabla_\theta \mathcal{L}_{\text{policy, reweighted}}(\theta) = -\frac{3}{2}\hat{A}_t^i \cdot \epsilon \cdot \nabla_\theta v_\theta \tag{3}$$

It is worth noting that Eq.(3) reveals that in flow matching models, GRPO fundamentally uses the relative advantage $\hat{A}_t^i$ and the noise term ϵ in the stochastic differential equation (SDE) sampling [Song et al., 2020] to approximate $\frac{dR}{dv_\theta}$, the gradient of the reward with respect to the velocity field, following the chain rule decomposition:

Algorithm 1 LongCat-Video's GRPO Training for Flow Matching Models

Require: Prompt distribution $\mathcal{C}$, group size G, total timesteps T, reward models $\{R_k\}_{k=1}^{n}$, weights $\{w_k\}_{k=1}^{n}$
Ensure: Optimized policy parameters θ

Initialize policy parameters θ, reference policy π_{ref}
repeat
  Sample batch of prompts $\{c_j\}_{j=1}^{B} \sim \mathcal{C}$
  for each prompt c_j **in parallel do**
    // Fix the initial noise and SDE timestep (Sec. 3.3.1)
    Sample initial noise $\boldsymbol{x}_T \sim \mathcal{N}(0, I)$
    Sample critical timestep $t' \sim \mathcal{U}(0, T'-1)$
    for $i = 1$ **to** G **do**
      Generate trajectory $\{\boldsymbol{x}_t^i\}_{t=0}^{T}$:
      for $t = T$ **to** 0 **do**
        if $t = t'$ **then**
          $\boldsymbol{x}_{t-1}^i \leftarrow \boldsymbol{x}_t^i + \text{drift}_\theta(\boldsymbol{x}_t^i, t, c_j)\Delta t + \sigma_t\sqrt{\Delta t}\epsilon$ // SDE step with truncated noise schedule (Sec. 3.3.1)
        else
          $\boldsymbol{x}_{t-1}^i \leftarrow \boldsymbol{x}_t^i + \text{drift}_\theta(\boldsymbol{x}_t^i, t, c_j)\Delta t$ // ODE step
        end if
      end for
      Compute rewards $\{R_k(\boldsymbol{x}_0^i, c_j)\}_{k=1}^{n}$
    end for
    for $k = 1$ **to** n **do**
      Compute $\mu_k \leftarrow \text{mean}(\{R_k(\boldsymbol{x}_0^i, c_j)\}_{i=1}^{G})$
      Compute $\sigma_k^j \leftarrow \text{std}(\{R_k(\boldsymbol{x}_0^i, c_j)\}_{i=1}^{G})$
      Collect $\{\sigma_k^j\}_{j=1}^{B}$ from all processes
      Compute $\sigma_{max,k} \leftarrow \max(\{\sigma_k^j\}_{j=1}^{B})$ // max group std (Sec. 3.3.1)
      for $i = 1$ **to** G **do**
        $\hat{A}_{k,t'}^i \leftarrow \frac{R_k(\boldsymbol{x}_0^i, c_j) - \mu_k}{\sigma_{max,k}}$
      end for
    end for
    for $i = 1$ **to** G **do**
      // Weighted relative advantage for multi-reward (Sec. 3.3.2)
      $\hat{A}_{\text{total}}^i \leftarrow \sum_{k=1}^{n} w_k \hat{A}_{k,t'}^i$
      // Reweighting of the Policy and KL Loss (Sec. 3.3.1)
      $\lambda_{\text{policy}} \leftarrow \sqrt{\frac{\frac{t'}{T}}{\Delta\frac{t'}{T}(1-\frac{t'}{T})}}$
      $\lambda_{\text{KL}} \leftarrow \frac{t'}{\Delta\frac{t'}{T}(1-\frac{t'}{T})}$
      $\mathcal{L}_{\text{policy}}^i \leftarrow \lambda_{\text{policy}} \cdot r_{t'}^i(\theta) \cdot \hat{A}_{\text{total}}^i$
      $\mathcal{L}_{\text{KL}}^i \leftarrow \beta\lambda_{\text{KL}} \cdot D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$
      $\mathcal{L}^i \leftarrow \mathcal{L}_{\text{policy}}^i - \mathcal{L}_{\text{KL}}^i$
    end for
  end for
  $\mathcal{L}_{\text{total}} \leftarrow \frac{1}{B \cdot G}\sum_{j=1}^{B}\sum_{i=1}^{G}\mathcal{L}^i$
  $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}_{\text{total}}$
until convergence

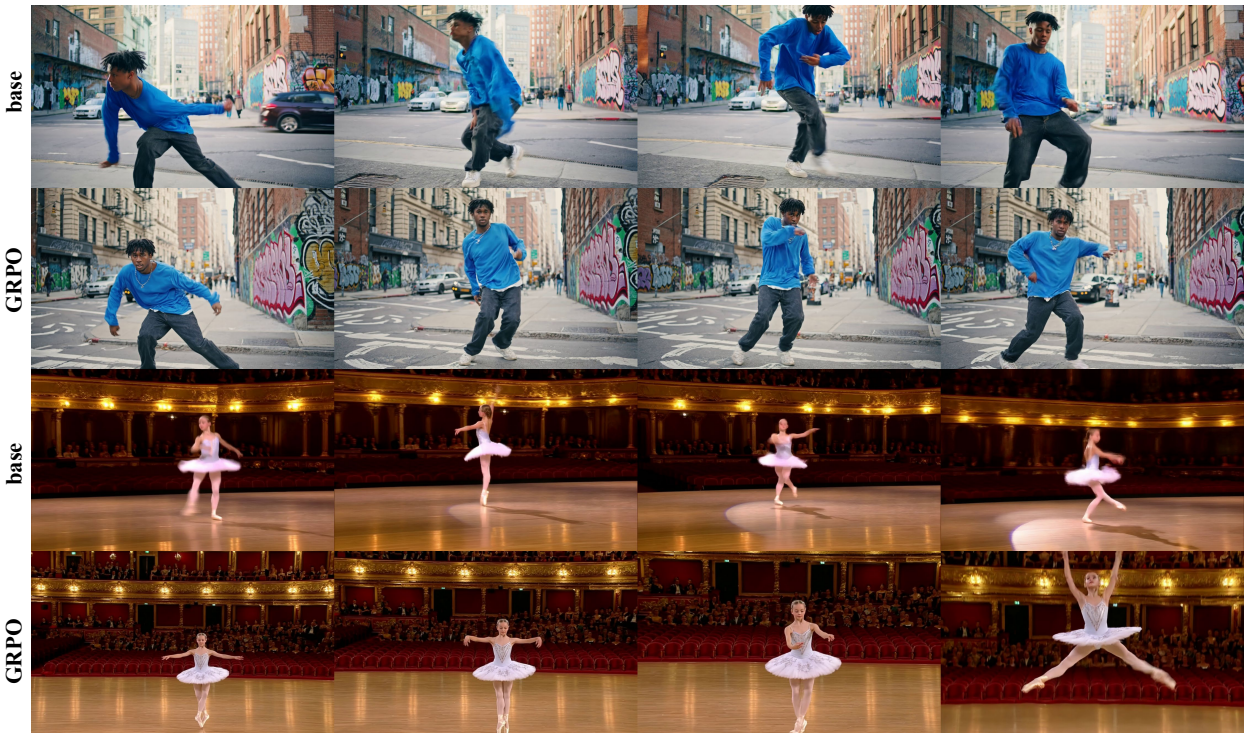

Figure 6: Our GRPO method significantly improves the video generation quality.

$$\frac{dR}{d\theta} = \frac{dR}{dv_\theta} \cdot \frac{dv_\theta}{d\theta} \tag{4}$$

where the GRPO framework provides the specific form:

$$\frac{dR}{dv_\theta} \approx -\frac{3}{2}\hat{A}_t^i \cdot \epsilon \tag{5}$$

Based on this finding, we design the following strategies.

Fix the stochastic timestep in SDE sampling Previous GRPO methods for Flow Matching sample trajectories using SDE sampling at all timesteps. This approach introduces temporal credit assignment ambiguity, as the reward is not accurately attributed to the specific timesteps that contributed to the final outcome. Instead, the reward is uniformly distributed across all timesteps, including those that may not have made a positive contribution. To address this ambiguity, we introduce a modified sampling scheme that isolates reward variation. Similar to concurrent works [He et al., 2025, Zhou et al., 2025], for each prompt c, samples share the same initial noise latent, and a single critical timestep t is randomly selected from the first T' timesteps ($T' < T$). SDE sampling with noise injection is applied only at t, while all other timesteps use deterministic ordinary differential equation (ODE) sampling. This approach enables precise credit assignment and leads to more stable, interpretable policy optimization. See Appendix A.1.3 for details.

Truncated noise schedule To enhance the diversity of SDE sampling, we adopt an amplified noise schedule with coefficient $a = 1$. However, this aggressive schedule can cause instability at high noise levels, as the diffusion coefficient $\sigma_t\sqrt{\Delta t}$ becomes excessively large when t approaches 1. We introduce a threshold-based clipping mechanism for the diffusion term. Specifically, the diffusion coefficient is clipped when it exceeds a predefined threshold τ:

$$\sigma_t\sqrt{\Delta t} \to \min\left(\sigma_t\sqrt{\Delta t}, \tau\right).$$

When clipping occurs, we set σ_t in the drift term to $\tau/\sqrt{\Delta t}$ for consistency. In our experiments, τ is set to 0.45.

Policy and KL Loss reweighting The gradient of the policy loss with respect to θ is as follows (See Appendix A.1.2 for details):

$$\nabla_\theta \mathcal{L}_{\text{policy}}(\theta) = -\frac{3}{2}\hat{A}_t^i \cdot \sqrt{\frac{\Delta t(1-t)}{t}} \cdot \epsilon \cdot \nabla_\theta v_\theta \tag{6}$$

We observe that the gradient magnitude is scaled by the factor $\kappa(t, \Delta t) = \sqrt{\frac{\Delta t(1-t)}{t}}$, which introduces two key optimization challenges: (1) **Vanishing gradient:** as $t \to 1$, $\kappa(t, \Delta t)$ approaches zero, causing the gradient magnitude to vanish in high noise stages; (2) **Small timestep:** video generation models typically use large shifts in timestep scheduling for both training and inference, resulting in small Δt values that further suppress the gradient magnitude.

To address these issues, we introduce a reweighting coefficient defined as:

$$\lambda_{\text{policy}}(t, \Delta t) = \kappa(t, \Delta t)^{-1} = \sqrt{\frac{t}{\Delta t(1-t)}}, \quad \mathcal{L}_{\text{policy, reweighted}}(\theta) = \lambda_{\text{policy}}(t, \Delta t) \cdot \mathcal{L}_{\text{policy}}(\theta) \tag{7}$$

Similarly, we also introduce a KL reweighting coefficient (See Appendix A.1.2 for details):

$$\lambda_{\text{KL}}(t, \Delta t) = k_{\text{KL}}(t, \Delta t)^{-1} = \frac{t}{\Delta t(1-t)}, \quad \mathcal{L}_{\text{KL, reweighted}}(\theta) = \lambda_{\text{KL}}(t, \Delta t) \cdot D_{\text{KL}}(\theta) \tag{8}$$

The reweighting coefficient effectively normalizes the gradient magnitude, eliminating the problematic temporal and step-size dependencies. This ensures stable and efficient optimization throughout the GRPO training (Figure 7a).

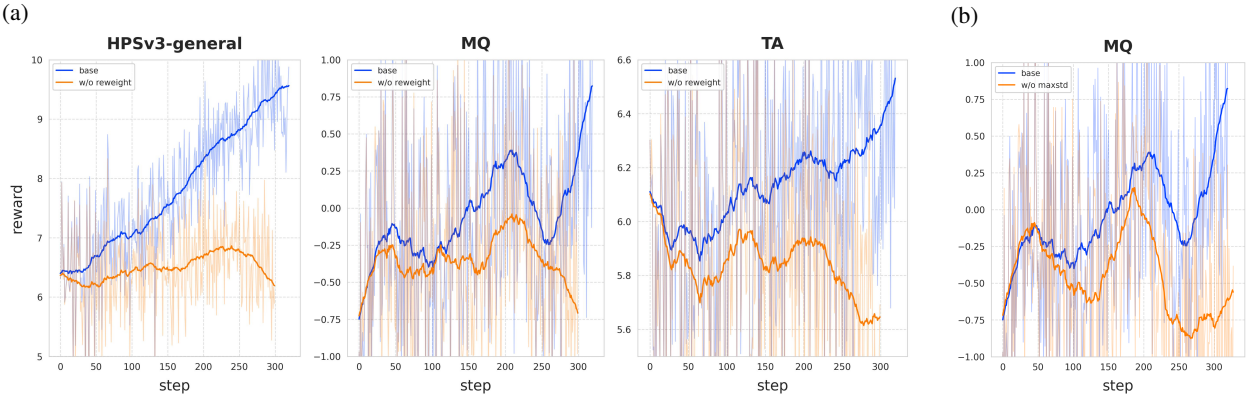

Figure 7: Ablation experiments on: (a) Policy and KL loss reweighting; (b) Max group standard deviation.

Max group standard deviation In the standard GRPO formulation, each prompt corresponds to a group of samples, and the relative advantage is computed using the group-specific standard deviation. However, reward dispersion varies across groups, and those with smaller standard deviations may yield unreliable advantage estimates due to inherent reward model inaccuracies.

To improve training stability, we address this by replacing the group-specific standard deviation with the maximum standard deviation observed across all groups. This adjustment reduces the gradient weight for samples from groups with potentially unreliable advantage estimates, while preserving the signal from groups with more reliable reward distributions. The modified advantage calculation becomes:

$$\hat{A}_{k,t}^i = \frac{R_k\left(\boldsymbol{x}_0^i, c_j\right) - \mu_k}{\sigma_{\max}} \tag{9}$$

where μ_k is the group mean for reward k, $\sigma_{\max} = \max_j \sigma_k^j$ is the maximum standard deviation across all groups for reward k. This modification ensures that samples from groups with small standard deviations receive appropriately scaled gradient updates and the training process becomes more robust to reward model inaccuracies (Figure 7b).

3.3.2 Reward Models and Multi-Reward Training

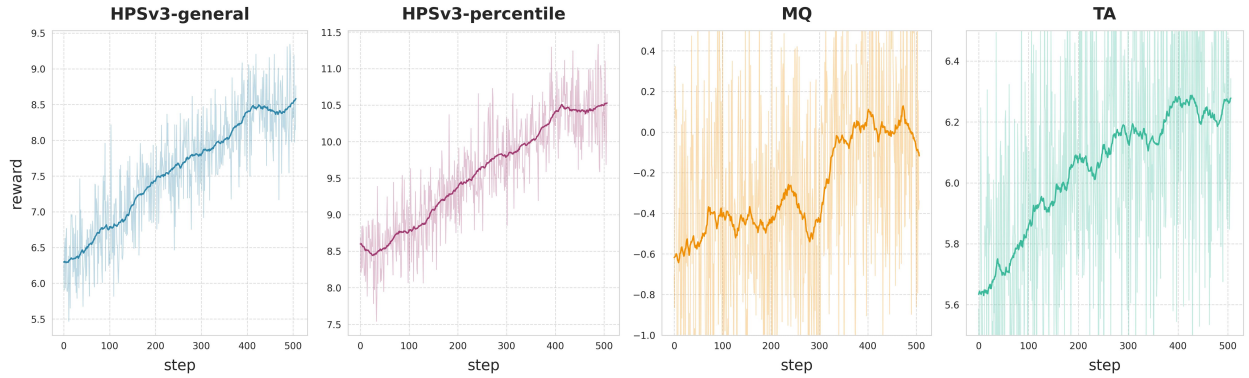

Figure 8: GRPO reward curves from the multi-reward training of LongCat-Video.

Figure 9: Reward hacking with single reward. Our multi-reward training approach prevents reward hacking for any single reward by establishing a balance among multiple rewards. For instance, the motion reward counteracts the static tendency induced by HPSv3 hacking while still leveraging HPSv3 to enhance visual quality.

Reward Models We utilize three specialized reward models to optimize visual quality (VQ), motion quality (MQ), and text-video alignment (TA) during training.

- **Visual Quality Assessment:** For VQ evaluation, we use HPSv3 [Ma et al., 2025b] as our base model, which inherently assesses both visual quality and text-video alignment. We combine two types of HPSv3-based rewards: ***HPSv3-general***, which is the mean score of all frames measured with the general prompt *"A high-quality image"* and focuses exclusively on visual quality; and ***HPSv3-percentile***, which is measured using the video caption to evaluate text-video alignment and uses the scores of the top 30% of all frames to mitigate the impact of low rewards resulting from content inconsistency caused by temporal changes.
- **Motion Quality Assessment:** For MQ evaluation, we employ a VideoAlign [Liu et al., 2025b]-based model fine-tuned on internal annotated datasets. To mitigate the model's preference for specific color, we use grayscale videos for both training and inference, which ensures the assessment focuses on motion characteristics rather than color attributes. Additionally, as illustrated in the validation loss curves during training (Figure 20), models trained with grayscale videos show a delayed increase in validation loss compared to those trained with RGB videos, indicating improved generalization and reduced overfitting in MQ reward model training.
- **Text-Video Alignment Assessment:** For TA evaluation, we also employ a VideoAlign-based model fine-tuned on internally annotated data. Unlike MQ evaluation, we retain the original color input processing to preserve the model's ability to assess semantic correspondence between text prompts and video content.

Multi-Reward Training For multi-reward GRPO training, the effective relative advantage in the policy loss for multi-reward optimization is exactly the weighted sum of the individual relative advantages (Refer to Appendix A.1.4 for details). Therefore, the corresponding policy loss becomes:

$$\mathcal{L}_{\text{policy, multi}}(\theta) = r_t^i(\theta) \cdot \left(\sum_{k=1}^{n} w_k \cdot \hat{A}_{k,t}^i \right) \tag{10}$$

where each relative advantage $\hat{A}_{k,t}^i$ is computed independently for reward R_k using group normalization.

In practice, the combination of multiple reward signals provides comprehensive guidance for the policy optimization process, ensuring balanced improvements in all aspects of video generation quality as shown in Figure 8. More importantly, the mutual constraints imposed by multiple rewards create a natural regularization effect that prevents over-optimization on any single metric and reduces the likelihood of reward hacking.

3.4 Efficient Video Generation

Inference efficiency remains a challenge for video generation, particularly for generating high-resolution, high-frame-rate videos. Therefore, we have introduced several optimizations to enhance inference efficiency. We distill the base model to reduce the necessary sampling steps. Additionally, we deploy coarse-to-fine (C2F) generation (Section 3.4.1) and block sparse attention (BSA) (Section 3.4.2) to further reduce the time cost in high-resolution video generation. As shown in Table 2, combining these strategies increases inference efficiency by more than $10\times$, allowing $720p, 30fps$ video generation within minutes. Additionally, we found that the coarse-to-fine generation strategy not only reduces inference cost but also improves generation quality, particularly enhancing visual details, as illustrated in Figure 10.

Table 2: Speed comparison under different inference settings.

Variant	LCM	C2F	BSA	Sampling Steps	Latency	Speedup
$480p \times 93$ frames	✗	✗	✗	50	341.5s	-
$480p \times 93$ frames	✓	✗	✗	16	61.3s	-
$720p \times 93$ frames	✗	✗	✗	50	1429.5s	1.0×
$720p \times 93$ frames	✓	✗	✗	16	244.6s	5.8×
$480p \times 93$ frames $\rightarrow 720p \times 93$ frames	✓	✓	✗	16/5	135.3s	10.6×
$480p \times 93$ frames $\rightarrow 720p \times 189$ frames	✓	✓	✗	16/5	302.9s	4.7×
$480p \times 93$ frames $\rightarrow 720p \times 93$ frames	✓	✓	✓	16/5	**116.5s**	**12.3**×
$480p \times 93$ frames $\rightarrow 720p \times 189$ frames	✓	✓	✓	16/5	**142.0s**	**10.1**×

$*$ The tests were conducted on a single H800 GPU with FlashAttention3 [Shah et al., 2024].

3.4.1 Coarse-to-Fine Generation

Training and inference on high-resolution, high-FPS videos incur substantial computational costs due to long token sequences. To address this, we propose a coarse-to-fine generation paradigm (Figure 11): first, the model generates a $480p, 15fps$ video; second, this video is upscaled to $720p, 30fps$ using trilinear interpolation and refined by a refinement expert. This approach greatly improves efficiency and enhances image quality and high-frequency details. The refinement expert is trained with LoRA fine-tuning on the base model. Since the refinement task is similar to the base model's generation task but follows a different denoising path, LoRA enables efficient adaptation while reusing the base model's capabilities. Besides, LoRA fine-tuning is decoupled from other training stages, converges faster, and significantly reduces memory usage.

Refinement using Flow Matching The training objective of refinement expert is to learn the transformation between the distribution of upsampled $480p, 15fps$ videos and the distribution of $720p, 30fps$ videos. We also utilize flow matching to model the mapping between these two distributions. The input to the network for the refinement stage training, denoted as $x_{t'}$, can be represented as follows:

$$x_{t'} = x_0 + (x_{thresh} - x_0) \cdot \frac{t'}{t_{thresh}}, t' \in [0, t_{thresh}], \tag{11}$$

$$x_{thresh} = (1 - t_{thresh}) \cdot x_{up} + t_{thresh} \cdot \epsilon, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{12}$$

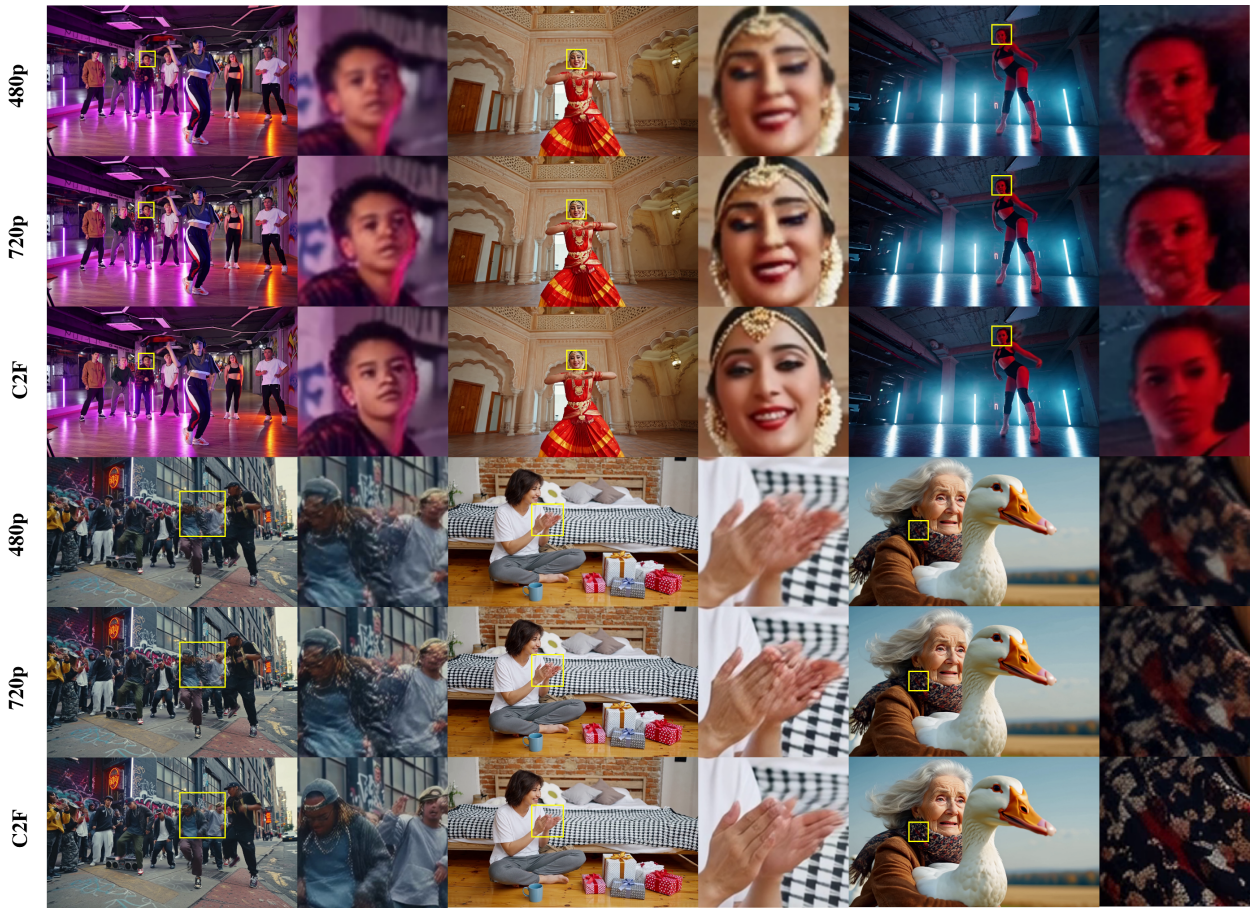

Figure 10: Comparison of native $480p$, native $720p$, and *coarse-to-fine* $720p$ generation. The coarse-to-fine strategy produces texture details and quality that surpass those of the native $720p$ generation and can also correct local distortions.

$$x_{up} = Encode(Upsample(Decode(x_{lr}))), \tag{13}$$

where x_{lr} is the output of the first stage, which is a latent representation of a low-resolution, low-frame-rate video, x_{up} represents the video latent obtained by applying the upsampling operation, denoted as $Upsample$, to x_{lr} in the RGB space, $Encode$ and $Decode$ respectively represent the encoding and decoding processes of the VAE.

To preserve the layout and structural information of low-resolution result, we apply a moderate level of noise, t_{thresh}, to x_{up}. The result after adding noise is x_{thresh}, which serves as the starting point for the refinement stage flow matching path, with the endpoint being x_0, the $720p, 30fps$ video latent. We sample noise intensity t' within the range from 0 to t_{thresh} for training. It should be noted that to ensure the numerical range of the ground truth in the refinement stage aligns with the base model, we need to apply numerical scaling to velocity $x_0 - x_{thresh}$. Finally, the ground truth $v_{t'}$ can be expressed as:

$$v_{t'} = \frac{x_0 - x_{thresh}}{t_{thresh}}. \tag{14}$$

This design is well-suited to the LoRA training mode, enabling significant reuse of the model's existing knowledge. It is evident that when t_{thresh} is equal to 1, the refinement stage training degenerates into a standard flow matching training process between the standard Gaussian distribution and the high-resolution video distribution. In practice, we set t_{thresh} to 0.5, and the refinement stage requires only 5 sampling steps, significantly improving efficiency. We further combine block sparse attention with the coarse-to-fine generation process, which accelerates sampling even further. Compared to the native generation process of $720p, 15fps$ videos, despite the token sequence length doubling, we achieve a **10.1**$\times$ acceleration in $720p, 30fps$ generation.

Refinement with Condition Frames In addition to the *Text-to-Video* task, we also support the refinement for the *Image-to-Video* and *Video-Continuation* tasks. In the conditional coarse-to-fine generation, we first use low-resolution

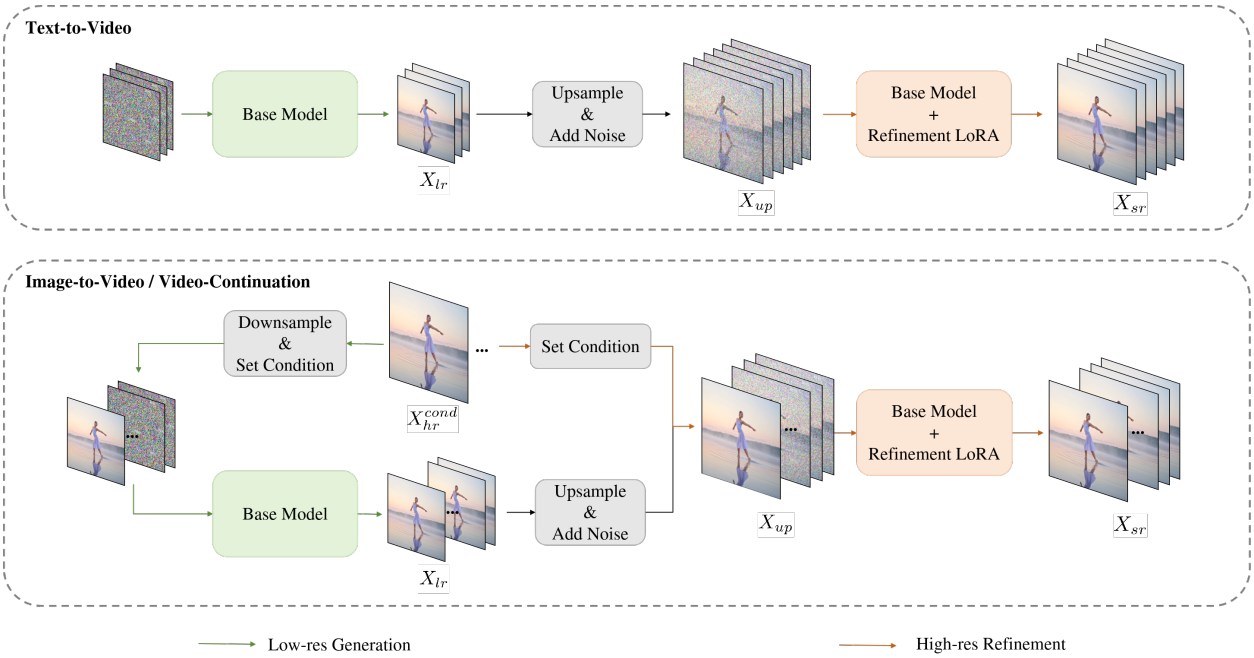

Figure 11: The coarse-to-fine generation processes for *Text-to-Video*, *Image-to-Video*, and *Video-Continuation* tasks. The green arrows indicate the low-resolution generation phase, while the orange arrows represent the refinement phase. Compared to *Text-to-Video*, *Image-to-Video* and *Video-Continuation* include additional configuration for the condition.

condition frames to generate a low-resolution video. This process can be represented as follows:

$$X_{lr} = BaseModel([Encode(X_{lr}^{cond}), \epsilon]), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{15}$$

$$X_{lr}^{cond} = Downsample(X_{hr}^{cond}), \tag{16}$$

where X_{hr}^{cond} represents the high-resolution condition RGB frames, X_{lr}^{cond} is the low-resolution condition RGB frames obtained using the spatial-temporal downsampling operation $Downsample$, and X_{lr} represents the non-condition part of the low-resolution video generated in the first stage. The generation process of the refinement stage can be represented as follows:

$$X_{up} = [X_{hr}^{cond}, Upsample(X_{lr})], \tag{17}$$

$$X_{sr} = Refinement(AddNoise(Encode(X_{up}))). \tag{18}$$

At the beginning of the refinement stage, we concatenate the high-resolution version of the condition RGB frames with trilinear upsampled X_{lr}, this concatenation is denoted as X_{up}. Then, we add noise at level t_{thresh} to VAE-encoded X_{up}. At this point, we have constructed the input for the refinement expert. The high-resolution video obtained after multiple steps of denoising is represented as X_{sr}. Through this design, we simultaneously support multiple tasks in refinement training, providing the coarse-to-fine generation with more application scenarios.

3.4.2 Block Sparse Attention

The computational speed of both training and inference for high-resolution video generation poses a major bottleneck for practical applications, primarily due to the quadratic complexity growth of self-attention with increasing token count. Trainable sparse attention mechanisms have demonstrated their effectiveness in large language models [Yuan et al., 2025b, Lu et al., 2025], and concurrent research has also validated their efficacy in video generation tasks [Zhang et al., 2025]. Given the high redundancy inherent in video latent representations, we developed a trainable sparse attention operator that significantly accelerates both training and inference. By retaining less than 10% of the original computational load, we can achieve near-lossless generation quality. Please refer to Appendix A.2 for details. Here we highlight some key points:

- Our 3D block sparse attention is open-sourced together with the base model, including both forward and backward implementations.This makes it convenient for the community to use as a modular component in their own projects.

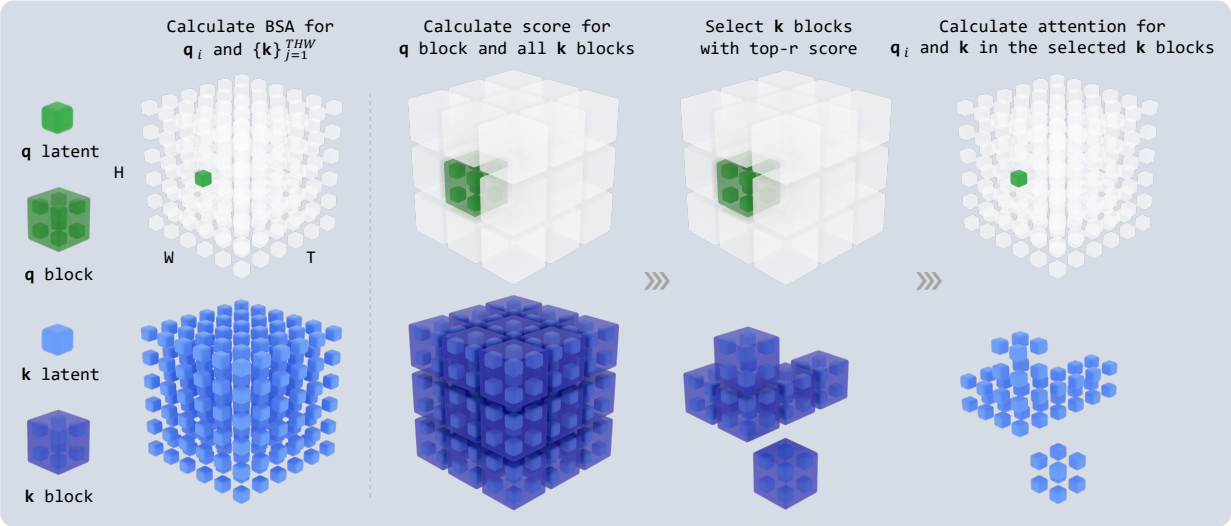

Figure 12: Illustration of 3D block sparse attention for query q_i and keys $\{k_j\}_{j=1}^{THW}$. **(a)** Partition q_i and all k_j into non-overlapping 3D blocks of size $t \times h \times w$. The block containing q_i is identified, and a similarity score is computed between this query block and each key block using their average values. **(b)** Select the top-r key blocks with the highest similarity scores. **(c)** Compute the standard attention between q_i and all keys within the selected r key blocks.

- We implemented ring block sparse attention to support context parallelism (See A.2.2 for details), which supports efficient training of large-scale models.
- Users can implement other sparse attention patterns based on our implementation, such as cumulative distribution function (CDF) based or block-wise 2D+1D, by customizing the block selection mask (See A.2.3 for details).
- In our experiments, the top-k[1] block sparse attention pattern achieved lossless sparse attention adaptation after training, eliminating the need for specially designed patterns; for simplicity, LongCat-Video adopted the top-k approach.

4 Training

As illustrated in Figure 13, the overall training procedure comprises three main components. The process begins with base model training, which includes progressive pre-training and supervised fine-tuning (SFT) to produce a base video generation model. This is followed by Reinforcement Learning from Human Feedback (RLHF) training, where Group Relative Policy Optimization (GRPO) is employed to enhance model performance by aligning outputs with human preferences. The final component is acceleration training, which involves model distillation and the development of a refinement expert LoRA module for coarse-to-fine generation. For both RLHF and acceleration training, we utilize the LoRA mechanism to facilitate the stacking of various enhancements and to ensure flexibility for future extensions.

4.1 Base Model Training

Flow Matching We employ the flow matching framework to model the diffusion process. During training, given a noise-free video latent x_0, a random noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and a timestep $t \in [0, 1]$, the network predicts the velocity $v_t = \frac{dx_t}{dt}$ of x_t moving towards x_0 at time t. x_t can be represented as the linear interpolation as

$$x_t = (1 - t) \cdot x_0 + t \cdot \epsilon. \tag{19}$$

The ground truth velocity is

$$v_t = x_0 - \epsilon. \tag{20}$$

The network output can be denoted as $v_{pred}(x_t, c, t; \theta)$, where c represents the task conditions (text prompt, conditional image/video latents), and θ represents the model parameters. The model parameters θ are optimized by minimizing the mean squared error (MSE) between model prediction v_{pred} and the ground truth velocity v_t, denoted as a loss function

$$\mathcal{L} = \mathbb{E}_{\epsilon, x_0, c, t} \left\| v_{pred}(x_t, c, t; \theta) - v_t \right\|^2. \tag{21}$$

[1]Note: To avoid confusion between top-k and the abbreviation 'k' for 'key', we refer to it as top-r in other parts of the report.

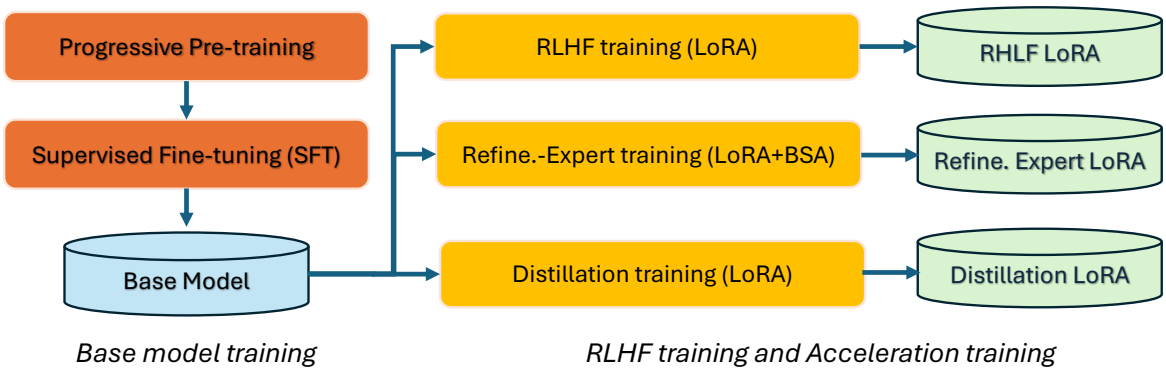

Figure 13: Overview of training process.

During training, we sample timestep t from a uniform distribution, and apply a logit-normal-like loss weighting scheme. We found that this strategy is more stable than sampling timesteps directly from the logit-normal distribution. Additionally, we adaptively adjust the timestep shift based on the volume of noise tokens [Esser et al., 2024], such that higher noise levels are preferred for videos with higher resolution and longer length.

Progressive Pre-training During pretraining, we employ a progressive training strategy to improve efficiency, as outlined in Table 3. The training process consists of multiple stages, beginning with model pre-training on low-resolution images to facilitate efficient learning of semantic and visual representations. After the image training stage reaches convergence, the process transitions to a dedicated video training phase, where the model captures fundamental motion dynamics. Following this, the training proceeds through several multi-task stages, during which *Text-to-Image (T2I)*, *Text-to-Video (T2V)*, *Image-to-Video (I2V)*, and *Video-Continuation (VC)* tasks are jointly optimized. For *Video-Continuation (VC)* task, we also perturb conditional frames with per-frame independent noise levels [Chen et al., 2024] to enhance robustness to color drift. These stages progress from low-resolution to high-resolution settings. At each stage, training samples are assigned to specific size buckets according to the closest aspect ratio, thereby maximizing computational efficiency. The AdamW [Loshchilov and Hutter, 2017] optimizer is used with a constant learning rate within each stage, and the learning rate is gradually reduced as training progresses to subsequent stages.

Table 3: Outline of the progressive training stages.

Training tasks	Size bucket	Learning rate	Iterations
T2I	$256p$	1e-4	285k
T2I + T2V	$256p \times 93$ frames	1e-4	140k
T2I + T2V + I2V + VC	$256p \times 93$ frames	5e-5	164k
T2I + T2V + I2V + VC	$480p \times 93$ frames	5e-5	36k
T2I + T2V + I2V + VC	$480p + 720p \times 93$ frames	2e-5	53k

Supervised Fine-Tuning (SFT) After pretraining, we conduct a supervised fine-tuning (SFT) stage using a carefully curated, high-quality dataset. The data is filtered based on multiple metrics, including aesthetic score, video quality, and motion quality, among others. To ensure balanced category representation, samples are selected inversely proportional to their density in the caption embedding space. In addition to the general high-quality dataset, we incorporate specialized datasets to further enhance the model's instruction-following capabilities, particularly for camera motion and visual style.

Table 4: Specifications of supervised fine-tuning (SFT) stage.

Training Tasks	Size Bucket	Learning rate	Iterations
T2I + T2V + I2V + VC	$480p + 720p \times 93$ frames	1e-5	7.5k

4.2 RLHF Training

After training the base model, we further improve its performance through a post-training stage that incorporates multiple video quality-related rewards using the GRPO method as described in Section 3.3. The key training specifications are listed in Table 5. For the complete experimental setup, please refer to Appendix A.1.5. We employ only *Text-to-Video* tasks in the GRPO training, and find that the improvements of instruction-following, visual quality and motion quality generalize well to *Image-to-Video* and *Video-Continuation* tasks. Proposing task-specific rewards for each task (e.g. quality degradation penalty of long-video generation for *Video-Continuation*) remains a future work.

Table 5: Specifications of RLHF training stage.

Training tasks	Size bucket	Group size	Prompts per step	Sampling steps	SDE steps range	Learning rate	Iterations
T2V	$480p + 720p \times 93$ frames	4	64	16	[0, 6]	1e-4	0.5k

4.3 Acceleration Training

As described in Section 3.4, we distill the model and train a refinement expert module to enable efficient inference.

Distillation training We have adopted Classifier-Free Guidance (CFG) distillation and consistency model (CM) distillation [Ren et al., 2024, Wang et al., 2024] to enhance model inference speed. In the CFG distillation step, we distill a general negative prompt using CFG-Zero [Fan et al., 2025] with a default guidance strength of 4.0. The combination of CFG distillation and CM distillation enables inference with 16 steps with quality comparable to inference results with more than 50 steps. We use a LoRA training strategy to allow flexible stacking of various model enhancement and further extensions.

Table 6: Specifications of distillation training.

Stage	Training Tasks	Size Bucket	Learning rate	Iterations
CFG distillation	T2I + T2V + I2V + VC	$480p + 720p \times 93$ frames	5e-5	2k
CM distillation	T2I + T2V + I2V + VC	$480p + 720p \times 93$ frames	5e-5	3k

Refinement expert training During the refinement LoRA training process, we initially use full attention for training. Once the loss converges and stabilizes, we activate BSA to continue training. We set the sparsity of BSA to 93.75% and the initial noise intensity for the refinement stage to 0.5. In terms of training data, we use Gray-Level Co-occurrence Matrix(GLCM) [Haralick et al., 2007] filter to keep only data with rich texture details for training. We apply a series of degradation operations to the training data to enhance the model's ability to refine details and improve robustness. Note that we train the refinement expert on data with mixed frame rates, enabling it to support both spatial-only refinement and spatial-temporal refinement.

Table 7: Specifications of refinement expert training.

Training Stage	Sparsity	t_{thresh}	Size bucket	Learning rate	Iterations
Full Attention	-	0.5	$720p \times 93$ *or* 189 frames	5e-5	500
Sparse Attention	93.75%	0.5	$720p \times 93$ *or* 189 frames	5e-5	500

4.4 Training Infrastructure

Our distributed training infrastructure incorporates mechanisms such as **DeepSpeed-Zero2** [Rasley et al., 2020], **Context Parallelism**, **Ring Attention**, and **Activation Checkpointing**, enabling efficient training of video generation models at the 13B-parameter scale. To support mixed-resolution training, we adopt a bucket-based strategy that groups data with similar resolutions into the same bucket for batch processing. Furthermore, we employ a cache mechanism to eliminate computation bubbles arising from VAE operations across different ranks, thereby improving computational efficiency and resource utilization. These methods collectively enable the training process to achieve *Model Flops Utilization* (MFU) rates ranging from 33% to 38%.

5 Evaluation

This section presents a comprehensive evaluation of LongCat-Video's performance across multiple dimensions of video generation quality. We establish rigorous assessment protocols through both internal benchmarks and public evaluation frameworks, providing a holistic view of the model's capabilities in *Text-to-Video* and *Image-to-Video* generation tasks. The subsequent subsections present representative examples of LongCat-Video outputs across various video generation tasks.

5.1 Internal Benchmarks

We introduce an internal benchmarking suite to assess model performance across two core tasks: *Text-to-Video* and *Image-to-Video*. The benchmark encompasses a total of 1,628 samples, categorized into 1,228 *Text-to-Video* cases (evaluated via 500 human and 728 automatic assessments) and 400 *Image-to-Video* cases. For *Text-to-Video*, evaluation is conducted based on the following four key dimensions:

- **Text-Alignment** evaluates whether the video comprehensively encompasses the information conveyed in the text and accurately interprets the relevant semantic expressions. It includes precise understanding of descriptions related to objects, people, scenes, styles, and other key elements.
- **Visual quality** is assessed from two perspectives: plausibility and realism. Plausibility focuses on the visual presentation of the video, examining whether it adheres to objective physical principles and identifying any issues such as distortion or unnatural appearances. Realism evaluates whether the scenes and subjects depicted in the video possess a sense of authenticity, aiming to avoid the presence of unrealistic elements.
- **Motion quality** assesses the normalcy of motion within the video. It examines whether motion trajectories are coherent and actions are smooth, in accordance with physical laws. For human motion, object motion, and camera motion, the evaluation determines whether each type of movement reflects realistic behavior, avoiding issues such as prolonged stillness or excessive jitter.
- **Overall quality** represents a comprehensive quality score for the generated video based on the aforementioned sub-dimensions.

For *Image-to-Video*, we further incorporate an "Image-Alignment" dimension in addition to the above four dimensions for evaluation:

- **Image-Alignment** evaluates the extent to which the generated video faithfully preserves key attributes and relationships of both the subject and background from the reference image, while maintaining the overall style of the original reference.

Evaluation Protocol The evaluation of video result in this report comprises both human and automatic model-based assessments. For human evaluation, following prior practice [Gao et al., 2025], we employ two complementary methodologies: absolute Mean Opinion Score (MOS) ratings and relative Good-Same-Bad (GSB) assessments. The former utilizes a 5-point scale for pointwise evaluation to quantitatively measure perceptual quality across various dimensions. Detailed descriptors were established for each scoring tier to ensure metric interpretability. The final score for each model is calculated as a weighted (2:1) average of human evaluation and automatic evaluation. The latter adopts a pairwise comparative approach, which provides more discriminative model performance rankings.

Quality Control To ensure annotation quality, a comprehensive and rigorous pre-annotation training process was implemented for all annotators. Each video was independently annotated by three annotators. In cases where significant discrepancies were identified between any two annotations, two additional annotators were introduced to reassess the video. The final score for each video was derived by averaging the ratings provided by all involved annotators. This consensus-based approach enhances the reliability and objectivity of the annotation outcomes.

For automatic evaluation, we have specifically trained a vision-language judge model based on high-quality human-annotated data, capable of quantitatively evaluating text alignment, visual quality, and motion quality. Internal evaluations demonstrate that this judge model achieves correlations consistently exceeding 0.92 with human assessments across all dimensions.

Data Taxonomy for Text-to-Video Evaluation Our text-to-video evaluation benchmark comprises two distinct subsets: 500 prompts designed for human evaluation and 728 for automatic evaluation. The human evaluation subset is characterized by its exceptional semantic diversity, spanning 48 distinct categories. This design ensures a balanced

assessment, preventing the overrepresentation of any single capability, with the most frequent category constituting only 39.2% of the prompts. Critically, the benchmark features a long tail of specialized tasks: 58.3% of categories appear with a frequency of 5% or less. These range from foundational abilities such as *Entity Generation* and *Action* to complex functions like *Physical Simulation* and *Inductive Reasoning*. Furthermore, the prompts exhibit significant structural diversity. Their lengths follow a pronounced bimodal distribution: 34.8% are concise ($\leq$20 words) and 34.6% are highly detailed ($\geq$51 words), with an overall range of 4 to 121 words. To ensure comprehensive coverage for the automatic evaluation subset, we curate prompts from high-quality public datasets, including T2VCompbench [Sun et al., 2025] and MovieGen [Polyak et al., 2024], and supplement them with in-house prompts to cover a wide array of video generation scenarios.

Text-to-Video Evaluation Leveraging our internal benchmark, we first conducted a comprehensive comparative evaluation of LongCat-Video against several leading video generation models in text-to-video setting. Specifically, we compare with two advanced proprietary models Veo3 [Google, 2024] and PixVerse-V5 [PixVerse, 2024], as well as the current SOTA open-source model Wan 2.2-T2V-A14B [Wan et al., 2025].

The MOS evaluation results are illustrated in Figure 14. Our analysis reveals that LongCat-Video demonstrates a highly competitive and well-balanced performance. A standout achievement is its excellence in Visual Quality, where it achieves a score that is nearly on par with the top performer, Wan 2.2, and significantly surpasses PixVerse-V5, which shows a clear deficit in this area. In terms of Overall Quality, LongCat-Video establishes itself as a top-tier model, achieving a score superior to both PixVerse-V5 and Wan 2.2-T2V-A14B. While Veo3 leads in this category, its advantage is built upon superior text-alignment and motion scores. In contrast, our model provides a more consistent, high-quality experience. For Text-Alignment, LongCat-Video delivers robust results, proving its strong capability in semantic understanding, though Veo3 sets a particularly high benchmark.

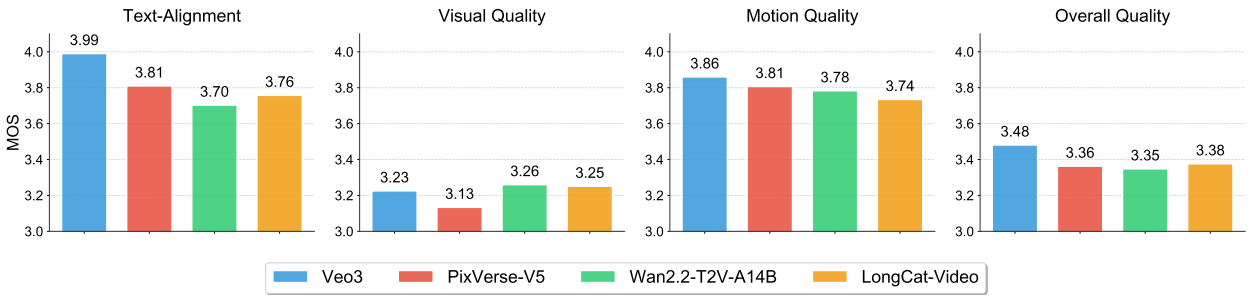

Figure 14: Text-to-Video MOS evaluation results on our internal benchmark.

The GSB evaluation results are shown in Figure 15. The user preference study indicates that LongCat-Video's performance, while trailing the state-of-the-art closed-source model Veo3, is highly competitive and on par with other leading proprietary models like PixVerse-V5. In the direct comparison, LongCat-Video and PixVerse-V5 are nearly tied in overall quality (242 vs. 246), with our model demonstrating a distinct advantage in visual quality. More importantly, when benchmarked against the current state-of-the-art open-source model, Wan2.2-T2V-A14B, our model shows a clear superiority. LongCat-Video was preferred by users in overall quality, driven by significant leads in both text-alignment and motion quality.

Data Taxonomy for Image-to-Video Evaluation Our benchmark for *Image-to-Video* evaluation is built upon a curated set of 100 first-frame reference images, designed to exhibit comprehensive diversity across multiple dimensions. These dimensions include **style** (e.g., photorealism, ink wash, 2D/3D animation, oil painting, sketch), **content** (e.g., human subjects, animals, plants, food, vehicles, indoor/outdoor environments), and **quality** (high vs. standard). Each image is further defined by metadata such as aspect ratios (1:1, 16:9, 9:16) and resolutions ($720p$, $1080p$, $2K$). To rigorously evaluate model sensitivity and dependency, each reference image is paired with a set of four distinct prompt types: (1) *detailed prompts* that specify fine-grained attributes; (2) *concise prompts* with minimal instructions; (3) *contradictory prompts* designed to conflict with the visual reference; and (4) *empty prompts* to assess unconditional generation based on the image. This quadripartite prompt structure enables a robust assessment of the model's cross-modal alignment and generative capabilities.

Image-to-Video Evaluation We then compare LongCat-Video against several leading video generation models in image-to-video generation setting. Concretely, we compare with two advanced proprietary models Seedance 1.0 [Gao et al., 2025] and Hailuo-2, as well as the current SOTA open-source model Wan 2.2-I2V-A14B [Wan et al., 2025].

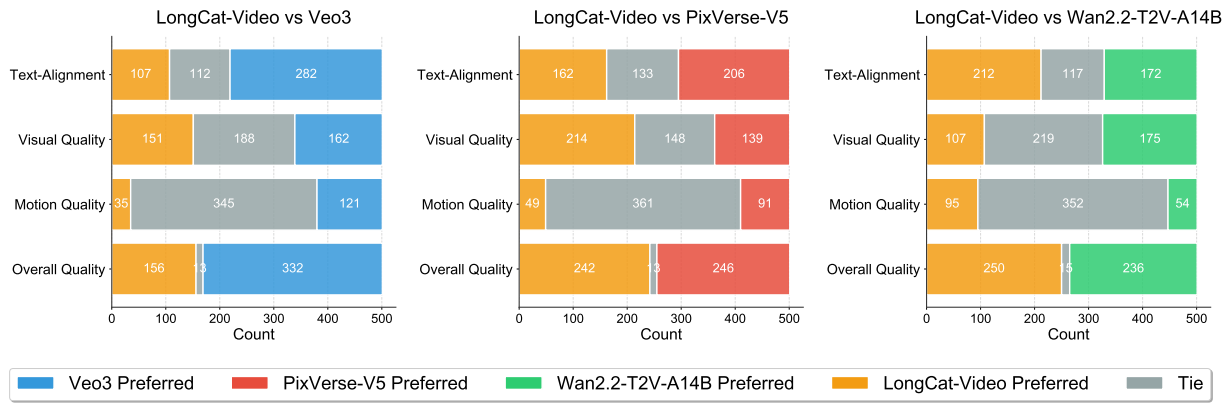

Figure 15: Text-to-Video GSB evaluation results on our internal benchmark.

The MOS evaluation results are illustrated in Figure 16. As shown in the figure, LongCat-Video achieves the highest score in Visual Quality (3.27), indicating its strength in generating aesthetically pleasing frames. However, it scores lower on Image-Alignment (4.04) and Motion Quality (3.59) compared to the other models. Hailuo-02 and Wan2.2-I2V-A14B perform best in Image-Alignment (4.18), while Hailuo-02 leads in Motion Quality (3.80). In the Overall Quality evaluation, LongCat-Video (3.17) is rated as competitive, though it trails the other models, with Seedance 1.0 achieving the highest overall score of 3.35. This suggests that while our model excels in visual fidelity, there is room for improvement in maintaining temporal consistency and alignment with the source image.

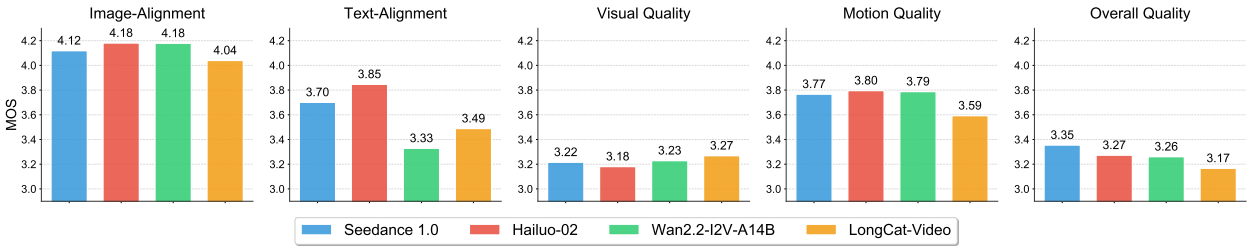

Figure 16: Image-to-Video MOS evaluation results on our internal benchmark.

5.2 Public Benchmarks

As a supplement to internal benchmarks, we also evaluated LongCat-Video on the widely used public benchmark VBench [Huang et al., 2024, Zheng et al., 2025]. Specifically, we conducted assessments on the latest version of VBench 2.0. The evaluation results are shown Table 8. On VBench 2.0, Long-Cat Video also demonstrated strong performance, with a total score second only to Veo3 [Google, 2024] and Vidu Q1 [Shengshu, 2024]. It is noteworthy that LongCat-Video led all other methods in the *Commonsense* dimension, indicating that our approach excels in aspects such as motion rationality and physical laws. This aligns with Long-Cat Video's outstanding long video generation capabilities and represents a key advantage in moving towards world model development.

Table 8: Text-to-Video evaluation results on VBench 2.0 benchmark.

Model name	Accessibility	Evaluation Date	Creativity↑	Commonsense↑	Controllability↑	Human Fidelity↑	Physics↑	Total Score↑
HunyuanVideo [Kong et al., 2024]	Open Source	2025-03	41.84%	63.44%	28.60%	82.41%	60.20%	55.30%
Wan2.1 [Wan et al., 2025]	Open Source	2025-03	55.25%	63.98%	37.32%	81.60%	62.84%	60.20%
Sora-480p [OpenAI, 2024]	Proprietary	2025-03	60.57%	64.32%	22.09%	87.72%	57.18%	58.38%
Kling1.6 [Kuaishou, 2024]	Proprietary	2025-03	48.58%	65.45%	33.05%	83.56%	64.35%	59.00%
Vidu Q1 [Shengshu, 2024]	Proprietary	2025-04	56.54%	65.98%	38.13%	81.24%	71.63%	62.70%
Seedance 1.0 Pro [Gao et al., 2025]	Proprietary	2025-06	53.04%	64.31%	39.84%	77.06%	64.81%	59.81%
Veo3 [Google, 2024]	Proprietary	2025-09	60.85%	69.48%	47.04%	86.88%	69.35%	66.72%
LongCat-Video	Open Source	2025-10	54.73%	70.94%	44.79%	80.20%	59.92%	62.11%

5.3 Text-to-Video Examples

Figure 17: Results on *Text-to-Video* generation.

5.4 Image-to-Video Examples

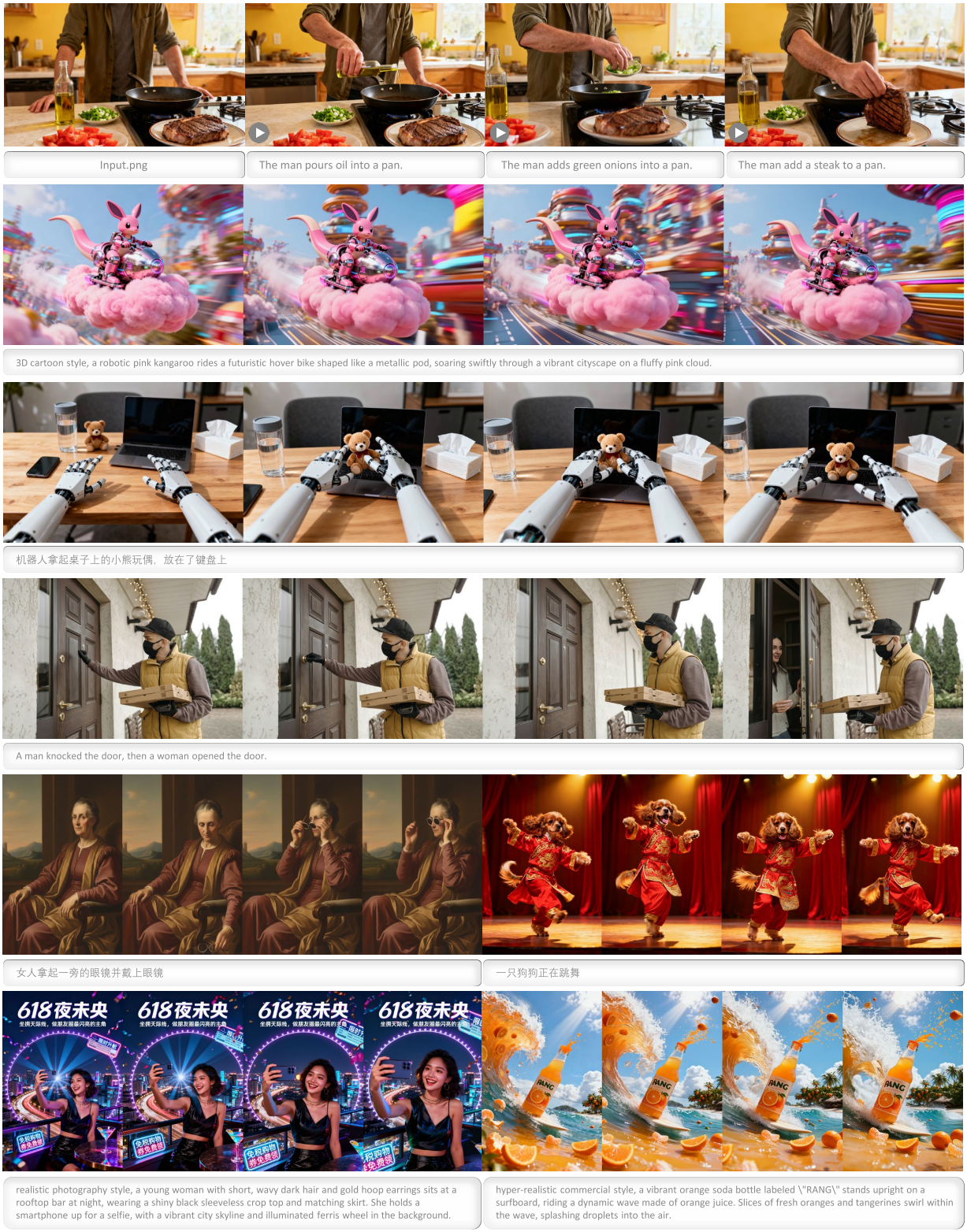

Figure 18: Results on *Image-to-Video*. As shown in the top row, given the same initial image, LongCat-Video accurately responds to instructions for various actions.

5.5 Long-Video Generation Examples

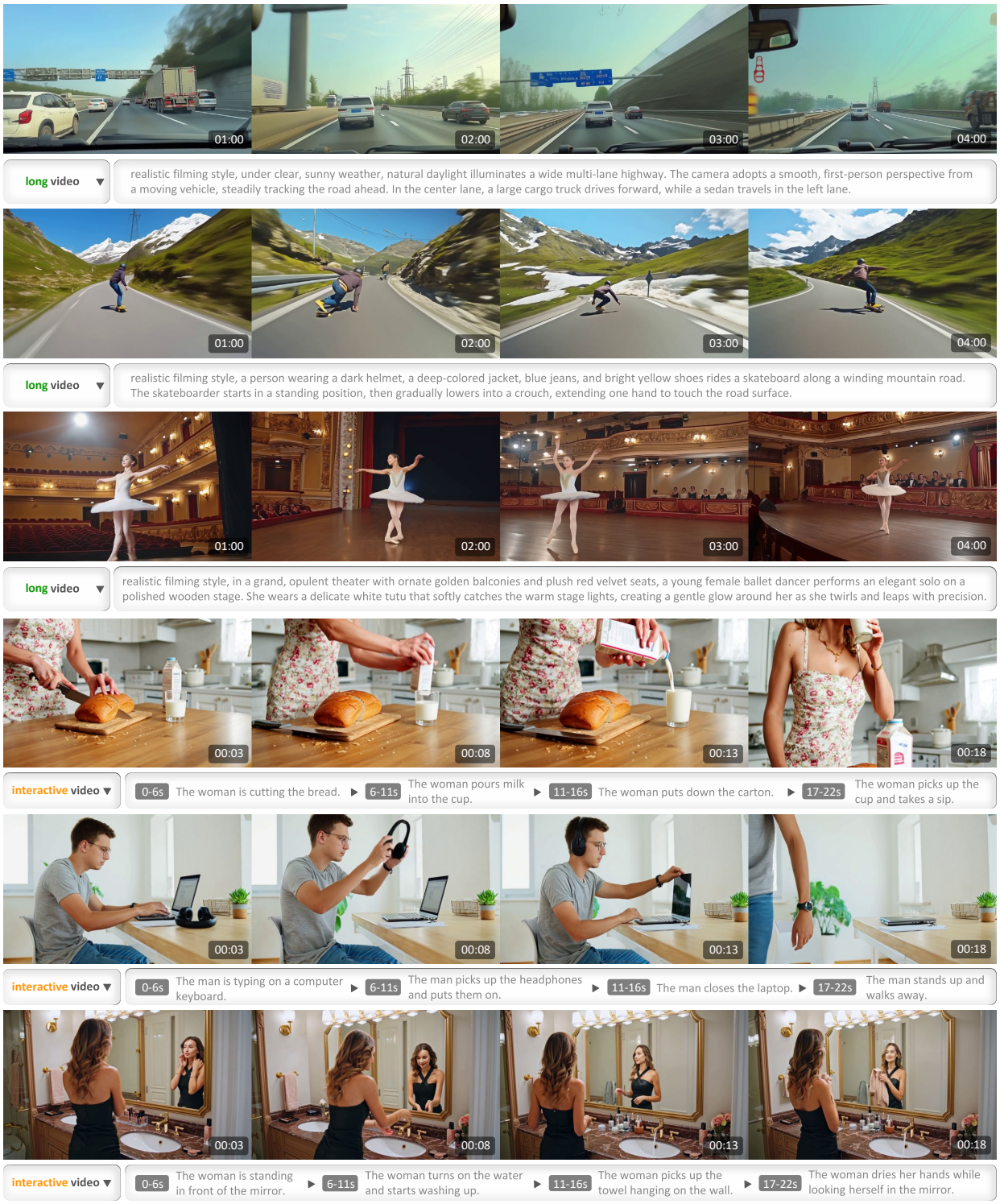

Figure 19: Results on *Video-Continuation*. LongCat-Video supports minutes-long video generation without quality degradation, as well as interactive video generation with changing instructions for each clip.

6 Conclusion and Future Work

We introduce LongCat-Video, a 13B-parameter foundational video generation model that unifies *Text-to-Video*, *Image-to-Video*, and *Video-Continuation* tasks within a single framework. LongCat-Video demonstrates strong performance across all supported tasks, particularly excelling in long video generation, which is enabled by pretraining on the *Video-Continuation* task. As a robust general-purpose video generation model, LongCat-Video is applicable to a wide range of video content creation scenarios. Moreover, it marks our first step toward developing world models. Efficient long video generation addresses the rendering problem of world models, enabling models to express their world knowledge through generated video content. Future directions include better modeling of physical knowledge, multi-modal memory integration in video generation, and the incorporation of knowledge from LLM and MLLM.

7 Contributors and Acknowledgments

Contributors are listed in alphabetical order by their last names. Names marked with an asterisk (*) indicate people who have left our team.

Contributors

Xunliang Cai	Qilong Huang	Zhuoliang Kang	Hongyu Li	Shijun Liang
Liya Ma	Siyu Ren	Xiaoming Wei	Rixu Xie	Tong Zhang

Acknowledgments

Xuezhi Cao	Hui Chen	Fengjiao Chen	Tianye Dai	Feng Gao
Ying Guo*	Xiaoyu Li	Shengxi Li	Hao Lu	Xiaofeng Mei*
Zhuqi Mi	Xin Pan	Liang Shi	Yuchen Tang	Chao Wang
Ziwen Wang	Wei Yi	Yong Zhang	Zizhe Zhao	

References

Google. Veo. https://deepmind.google/models/veo/, 2024.

OpenAI. Sora. https://openai.com/sora/, 2024.

Yu Gao, Haoyuan Guo, Tuyen Hoang, Weilin Huang, Lu Jiang, Fangyuan Kong, Huixia Li, Jiashi Li, Liang Li, Xiaojie Li, et al. Seedance 1.0: Exploring the boundaries of video generation models. *arXiv preprint arXiv:2506.09113*, 2025.

Kuaishou. Kling. https://klingai.com, 2024.

MiniMax. Hailuo. https://hailuoai.video/, 2024.

PixVerse. Pixverse. https://app.pixverse.ai, 2024.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Guoqing Ma, Haoyang Huang, Kun Yan, Liangyu Chen, Nan Duan, Shengming Yin, Changyi Wan, Ranchen Ming, Xiaoniu Song, Xing Chen, et al. Step-video-t2v technical report: The practice, challenges, and future of video foundation model. *arXiv preprint arXiv:2502.10248*, 2025a.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

NVIDIA. Cosmos. URL https://github.com/nvidia-cosmos.

Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, Junchen Zhu, Mingyuan Fan, Hao Zhang, Sheng Chen, Zheng Chen, Chengcheng Ma, Weiming Xiong, Wei Wang, Nuo Pang, Kang Kang, Zhiheng Xu, Yuzhe Jin, Yupeng Liang, Yubing Song, Peng Zhao, Boyuan Xu, Di Qiu, Debang Li, Zhengcong Fei, Yang Li, and Yahui Zhou. Skyreels-v2: Infinite-length film generative model, 2025. URL https://arxiv.org/abs/2504.13074.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Brandon Castellano. PySceneDetect. URL https://github.com/Breakthrough/PySceneDetect.

Tomáš Souček and Jakub Lokoč. Transnet v2: An effective deep network architecture for fast shot transition detection, 2020. URL https://arxiv.org/abs/2008.04838.

FFmpeg Developers. Ffmpeg. https://ffmpeg.org, 2014.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*, 2024.

Liping Yuan, Jiawei Wang, Haomiao Sun, Yuchen Zhang, and Yuan Lin. Tarsier2: Advancing large vision-language models from detailed video description to comprehensive video understanding. *arXiv preprint arXiv:2501.07888*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers. *arXiv preprint arXiv:2010.04245*, 2020.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. *arXiv preprint arXiv:2304.09151*, 2023.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025a.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Junzhe Li, Yutao Cui, Tao Huang, Yinping Ma, Chun Fan, Miles Yang, and Zhao Zhong. Mixgrpo: Unlocking flow-based grpo efficiency with mixed ode-sde. *arXiv preprint arXiv:2507.21802*, 2025.

Xiaoxuan He, Siming Fu, Yuke Zhao, Wanli Li, Jian Yang, Dacheng Yin, Fengyun Rao, and Bo Zhang. Tempflow-grpo: When timing matters for grpo in flow models, 2025. URL https://arxiv.org/abs/2508.04324.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yujie Zhou, Pengyang Ling, Jiazi Bu, Yibin Wang, Yuhang Zang, Jiaqi Wang, Li Niu, and Guangtao Zhai. G^2rpo: Granular grpo for precise reward in flow models, 2025. URL https://arxiv.org/abs/2510.01982.

Yuhang Ma, Yunhao Shui, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. Hpsv3: Towards wide-spectrum human preference score. *arXiv preprint arXiv:2508.03789*, 2025b.

Jie Liu, Gongye Liu, Jiajun Liang, Ziyang Yuan, Xiaokun Liu, Mingwu Zheng, Xiele Wu, Qiulin Wang, Wenyu Qin, Menghan Xia, et al. Improving video generation with human feedback. *arXiv preprint arXiv:2501.13918*, 2025b.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024. URL https://arxiv.org/abs/2407.08608.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025b.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. Vsa: Faster video diffusion with trainable sparse attention. *arXiv preprint arXiv:2505.13389*, 2025.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion, 2024. URL https://arxiv.org/abs/2407.01392.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Yuxi Ren, Xin Xia, Yanzuo Lu, Jiacheng Zhang, Jie Wu, Pan Xie, Xing Wang, and Xuefeng Xiao. Hyper-sd: Trajectory segmented consistency model for efficient image synthesis. *Advances in Neural Information Processing Systems*, 37: 117340–117362, 2024.

Fu-Yun Wang, Zhaoyang Huang, Alexander Bergman, Dazhong Shen, Peng Gao, Michael Lingelbach, Keqiang Sun, Weikang Bian, Guanglu Song, Yu Liu, et al. Phased consistency models. *Advances in neural information processing systems*, 37:83951–84009, 2024.

Weichen Fan, Amber Yijia Zheng, Raymond A Yeh, and Ziwei Liu. Cfg-zero*: Improved classifier-free guidance for flow matching models. *arXiv preprint arXiv:2503.18886*, 2025.

Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 2007.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506, 2020.

Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. T2v-compbench: A comprehensive benchmark for compositional text-to-video generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8406–8416, 2025.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024.

Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Lulu Gu, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, et al. Vbench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025.

Shengshu. Vidu. https://vidu.cn, 2024.

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

A Appendix

A.1 Appendix-A

A.1.1 GRPO Preliminaries

The GRPO method optimizes the generative flow model by maximizing the following objective function:

$$\mathcal{J}_{\mathrm{GRPO}}(\theta)=\mathbb{E}_{c\sim\mathcal{C},\{x^i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|c)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T}\sum_{t=0}^{T-1}\left(\mathcal{L}_{\text{policy}}\left(\theta\right)-\beta D_{\mathrm{KL}}\left(\pi_\theta\|\pi_{\text{ref}}\right)\right)\right], \tag{22}$$

Below we elaborate on each component of this objective.

Sampling Process. A group of G samples $\left\{\boldsymbol{x}^i\right\}_{i=1}^{G}$ is drawn from the current policy $\pi_{\theta_{\text{old}}}$ conditioned on the prompt c. Each sample is generated by discretizing the reverse-time stochastic differential equation (SDE):

$$x_{t+\Delta t}=x_t+\left[v_\theta\left(x_t,t,c\right)+\frac{\sigma_t^2}{2t}\left(x_t+(1-t)v_\theta\left(x_t,t,c\right)\right)\right]\Delta t+\sigma_t\sqrt{\Delta t}\epsilon, \tag{23}$$

with $\epsilon\sim\mathcal{N}(0,\mathbf{I})$ and noise schedule $\sigma_t=a\sqrt{t/(1-t)}$. This process yields complete trajectories $\left\{\left(x_T^i,x_{T-1}^i,\cdots,x_0^i\right)\right\}_{i=1}^{G}$ for policy optimization.

Policy Loss. The policy loss $\mathcal{L}_{\text{policy}}\left(\theta\right)=r_t^i(\theta)\hat{A}_t^i$ consists of two elements:

1) Importance ratio: $r_t^i(\theta)=\frac{p_\theta\left(x_{t-1}^i|x_t^i,c\right)}{p_{\theta_{\text{old}}}\left(x_{t-1}^i|x_t^i,c\right)}$ quantifies the probability change for transition $x_t^i\to x_{t-1}^i$ between policy updates, where the transition probability follows:

$$p_\theta\left(x_{t-1}\mid x_t,c\right)=\mathcal{N}\left(x_{t-1};\mu_\theta\left(x_t,t,c\right),\sigma_t^2\Delta t\mathbf{I}\right). \tag{24}$$

2) Group-relative advantage: $\hat{A}_t^i=\frac{R\left(x_0^i,c\right)-\text{mean}\left(\left\{R\left(x_0^j,c\right)\right\}_{j=1}^{G}\right)}{\text{std}\left(\left\{R\left(x_0^j,c\right)\right\}_{j=1}^{G}\right)}$ provides normalized advantage estimates by comparing individual rewards against group statistics.

KL Regularization. The KL divergence term $D_{\mathrm{KL}}\left(\pi_\theta\|\pi_{\text{ref}}\right)$ ensures training stability by constraining policy deviation from the reference policy. For the flow matching formulation, this term can be expressed as:

$$D_{\mathrm{KL}}\left(\pi_\theta\|\pi_{\text{ref}}\right)=\frac{\Delta t}{2}\left(\frac{\sigma_t(1-t)}{2t}+\frac{1}{\sigma_t}\right)^2\left\|v_\theta\left(x_t,t,c\right)-v_{\text{ref}}\left(x_t,t,c\right)\right\|^2, \tag{25}$$

with β controlling the regularization strength.

A.1.2 The Gradient of the Policy and KL Loss

We derive the gradient of the policy loss $\mathcal{L}_{\text{policy}}(\theta)=r_t^i(\theta)\hat{A}_t^i$ with respect to the parameters θ, The gradient computation proceeds as follows:

$$\nabla_\theta\mathcal{L}_{\text{policy}}(\theta)=\hat{A}_t^i\nabla_\theta r_t^i(\theta).$$

$$\nabla_\theta r_t^i(\theta)=\frac{p_\theta\left(x_{t-1}^i\mid x_t^i,c\right)}{p_{\theta_{\text{old}}}\left(x_{t-1}^i\mid x_t^i,c\right)}\nabla_\theta\log p_\theta\left(x_{t-1}^i\mid x_t^i,c\right)=\nabla_\theta\log p_\theta\left(x_{t-1}^i\mid x_t^i,c\right).$$

Combining these results gives the policy gradient:

$$\nabla_\theta\mathcal{L}_{\text{policy}}(\theta)=\hat{A}_t^i r_t^i(\theta)\nabla_\theta\log p_\theta\left(x_{t-1}^i\mid x_t^i,c\right). \tag{26}$$

We now compute the score function $\nabla_\theta \log p_\theta\left(x_{t-1} \mid x_t, c\right)$. The conditional distribution is Gaussian:

$$p_\theta\left(x_{t-1} \mid x_t, c\right)=\mathcal{N}\left(x_{t-1} ; \mu_\theta\left(x_t, t, c\right), \sigma_t^2 \Delta t I\right).$$

$$\nabla_\theta \log p_\theta=\frac{1}{\sigma_t^2 \Delta t}\left(x_{t-1}-\mu_\theta\right) \cdot \nabla_\theta \mu_\theta.$$

From the SDE sampling process, we have the reparameterization:

$$x_{t-1}=\mu_\theta+\sigma_t \sqrt{\Delta t} \epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$

Substituting:

$$\nabla_\theta \log p_\theta=\frac{1}{\sigma_t^2 \Delta t}\left(\sigma_t \sqrt{\Delta t} \epsilon\right) \cdot \nabla_\theta \mu_\theta=\frac{1}{\sigma_t \sqrt{\Delta t}} \epsilon \cdot \nabla_\theta \mu_\theta.$$

$$\mu_\theta=x_t+\left[v_\theta\left(x_t, t, c\right)+\frac{\sigma_t^2}{2 t}\left(x_t+(1-t) v_\theta\left(x_t, t, c\right)\right)\right](-\Delta t) \tag{27}$$

Simplifying the drift term:

$$\begin{aligned}\text{drift} &= v_\theta+\frac{\sigma_t^2}{2 t} x_t+\frac{\sigma_t^2}{2 t}(1-t) v_\theta \\ &= v_\theta\left(1+\frac{\sigma_t^2(1-t)}{2 t}\right)+\frac{\sigma_t^2}{2 t} x_t\end{aligned} \tag{28}$$

Thus:

$$\mu_\theta=x_t-\Delta t \cdot \text{drift} \tag{29}$$

Taking the gradient with respect to θ (noting that x_t is constant):

$$\nabla_\theta \mu_\theta=-\Delta t \cdot \nabla_\theta \text{drift}=-\Delta t \cdot\left(1+\frac{\sigma_t^2(1-t)}{2 t}\right) \nabla_\theta v_\theta \tag{30}$$

Substituting into $\nabla_\theta \log p_\theta$:

$$\begin{aligned}\nabla_\theta \log p_\theta &= \frac{1}{\sigma_t \sqrt{\Delta t}} \epsilon \cdot\left[-\Delta t \cdot\left(1+\frac{\sigma_t^2(1-t)}{2 t}\right) \nabla_\theta v_\theta\right] \\ &= -\frac{\sqrt{\Delta t}}{\sigma_t}\left(1+\frac{\sigma_t^2(1-t)}{2 t}\right) \epsilon \cdot \nabla_\theta v_\theta\end{aligned} \tag{31}$$

Therefore, the gradient of the policy loss is:

$$\nabla_\theta \mathcal{L}_{\text{policy}}(\theta)=\hat{A}_t^i r_t^i(\theta) \cdot\left[-\frac{\sqrt{\Delta t}}{\sigma_t}\left(1+\frac{\sigma_t^2(1-t)}{2 t}\right) \epsilon \cdot \nabla_\theta v_\theta\right] \tag{32}$$

Now, we substitute $a=1$ and $\sigma_t=\sqrt{\frac{t}{1-t}}$ (so $\sigma_t^2=\frac{t}{1-t}$). Computing the coefficient term:

$$1+\frac{\sigma_t^2(1-t)}{2 t}=1+\frac{\frac{t}{1-t} \cdot(1-t)}{2 t}=1+\frac{1}{2}=\frac{3}{2} \tag{33}$$

And the scaling term:

$$\frac{\sqrt{\Delta t}}{\sigma_t} = \frac{\sqrt{\Delta t}}{\sqrt{\frac{t}{1-t}}} = \sqrt{\Delta t} \cdot \sqrt{\frac{1-t}{t}} = \sqrt{\frac{\Delta t(1-t)}{t}} \tag{34}$$

Substituting these simplifications, we obtain the final policy gradient expression:

$$\nabla_\theta \mathcal{L}_{\text{policy}}(\theta) = -\frac{3}{2}\hat{A}_t^i \sqrt{\frac{\Delta t(1-t)}{t}}\epsilon \cdot \nabla_\theta v_\theta \tag{35}$$

By introducing a reweighting coefficient defined as:

$$\lambda_{\text{policy}}(t, \Delta t) = \kappa(t, \Delta t)^{-1} = \sqrt{\frac{t}{\Delta t(1-t)}} \tag{36}$$

The reweighted policy loss becomes:

$$\mathcal{L}_{\text{policy, reweighted}}(\theta) = \lambda_{\text{policy}}(t, \Delta t) \cdot \mathcal{L}_{\text{policy}}(\theta) \tag{37}$$

This yields the modified gradient:

$$\nabla_\theta \mathcal{L}_{\text{policy, reweighted}}(\theta) = -\frac{3}{2}\hat{A}_t^i \cdot \epsilon \cdot \nabla_\theta v_\theta \tag{38}$$

Similarly, the gradient of the KL divergence term can be derived as:

$$\nabla_\theta D_{\text{KL}}(\theta) = \Delta t \cdot \frac{9}{4} \cdot \frac{1-t}{t} \cdot (v_\theta - v_{\text{ref}}) \cdot \nabla_\theta v_\theta \tag{39}$$

This expression reveals that the KL loss gradient suffers from the same scaling issues as the policy loss gradient. To address this, we also introduce a KL reweighting coefficient:

$$\lambda_{\text{KL}}(t, \Delta t) = k_{\text{KL}}(t, \Delta t)^{-1} = \frac{t}{\Delta t(1-t)} \tag{40}$$

The reweighted KL loss becomes:

$$\mathcal{L}_{\text{KL, reweighted}}(\theta) = \lambda_{\text{KL}}(t, \Delta t) \cdot D_{\text{KL}}(\theta) \tag{41}$$

yielding the simplified gradient:

$$\nabla_\theta \mathcal{L}_{\text{KL,reweighted}}(\theta) = \frac{9}{4} \cdot (v_\theta - v_{\text{ref}}) \cdot \nabla_\theta v_\theta \tag{42}$$

Based on the reweighting coefficients for the policy loss and KL loss, the revised GRPO objective function is as follows:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{c\sim\mathcal{C},\ t'\sim\mathcal{U}(0,T'-1),\\ \{\boldsymbol{x}^i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|c,t')}} \left[\frac{1}{G}\sum_{i=1}^{G}\left(\lambda_{\text{policy}}(\frac{t'}{T}, \Delta\frac{t'}{T}) \cdot \mathcal{L}_{\text{policy}}(\theta) - \beta\lambda_{\text{KL}}(\frac{t'}{T}, \Delta\frac{t'}{T}) \cdot D_{\text{KL}}\left(\pi_\theta \| \pi_{\text{ref}}\right)\right)\right] \tag{43}$$

A.1.3 Fix the stochastic timestep in SDE sampling

As described in Para. "Fix the stochastic timestep in SDE sampling" in Sec. 3.3.1, the objective function is accordingly simplified to focus only on the critical stochastic timestep:

$$\mathcal{J}_{\text{GRPO-Selective}}(\theta) = \mathbb{E}_{c\sim\mathcal{C},\ t'\sim\mathcal{U}(0,T'-1),\ \{x^i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot|c,t')}\left[\frac{1}{G}\sum_{i=1}^{G}\left(r^i_{t'}(\theta)\hat{A}^i - \beta D_{\text{KL}}\left(\pi_\theta\|\pi_{\text{ref}}\right)_{t'}\right)\right], \tag{44}$$

where $t' \sim \mathcal{U}(0, T'-1)$ indicates uniform sampling of the critical timestep from the first T' steps. We set $T' = 6$ in our experiments. (The total sampling steps for training is set to 16.)

A.1.4 Multi-reward GRPO Training

Eq.(38) reveals that in flow matching models, GRPO fundamentally uses the relative advantage $\hat{A}^i_t$ and the noise term ϵ to estimate the gradient of the reward with respect to the velocity field, following the chain rule decomposition:

$$\frac{dR}{d\theta} = \frac{dR}{dv_\theta}\cdot\frac{dv_\theta}{d\theta} \tag{45}$$

where the GRPO framework provides the specific form:

$$\frac{dR}{dv_\theta} \approx -\frac{3}{2}\hat{A}^i_t\cdot\epsilon \tag{46}$$

When optimizing for multiple reward functions $R_1, R_2, \ldots, R_n$ with corresponding weights $w_1, w_2, \ldots, w_n$, the total gradient is given by the weighted sum:

$$\nabla_\theta J_{\text{total}} = \sum_{k=1}^{n} w_k \cdot \frac{dR_k}{d\theta} \tag{47}$$

Applying the chain rule decomposition for each reward:

$$\nabla_\theta J_{\text{total}} = \sum_{k=1}^{n} w_k \cdot \left(\frac{dR_k}{dv_\theta}\cdot\frac{dv_\theta}{d\theta}\right) = \left(\sum_{k=1}^{n} w_k \cdot \frac{dR_k}{dv_\theta}\right)\cdot\frac{dv_\theta}{d\theta} \tag{48}$$

Substituting the GRPO expression for each reward gradient:

$$\nabla_\theta J_{\text{total}} = \left(\sum_{k=1}^{n} w_k \cdot \left(-\frac{3}{2}\hat{A}^i_{k,t}\cdot\epsilon\right)\right)\cdot\frac{dv_\theta}{d\theta} = -\frac{3}{2}\left(\sum_{k=1}^{n} w_k\cdot\hat{A}^i_{k,t}\right)\cdot\epsilon\cdot\nabla_\theta v_\theta \tag{49}$$

This demonstrates that the effective relative advantage in the policy loss for multi-reward optimization is exactly the weighted sum of the individual relative advantages. Therefore, the corresponding policy loss becomes:

$$\mathcal{L}_{\text{policy, multi}}(\theta) = r^i_t(\theta)\cdot\left(\sum_{k=1}^{n} w_k\cdot\hat{A}^i_{k,t}\right) \tag{50}$$

where each relative advantage $\hat{A}^i_{k,t}$ is computed independently for reward R_k using group normalization:

$$\hat{A}^i_{k,t} = \frac{R_k\left(\boldsymbol{x}^i_0, \boldsymbol{c}\right) - \text{mean}\left(\left\{R_k\left(\boldsymbol{x}^j_0, \boldsymbol{c}\right)\right\}_{j=1}^{G}\right)}{\sigma_{\max,k}} \tag{51}$$

Table 9: GRPO Experiment Settings

Parameter	Value	Parameter	Value
Group size	4	# Sampling steps	16
Prompts per update	64	Timeshift	12
SDE steps range	[0, 6]	CFG	4
Online training	True	Learning rate	1e-4
Policy loss weight	1	LoRA dim	128
KL loss weight	3e-4	LoRA alpha	64
HPSv3-general reward weight	1	LoRA layers	Linear layers in all Self-Attention,
HPSv3-percentile reward weight	1		Cross-Attention, FFN layers
MQ reward weight	1		
TA reward weight	1		

A.1.5 GRPO Experiment Settings

A.2 Appendix-B

A.2.1 Modeling of Block Sparse Attention

3D Block Rearrangement We consider a video sequence with shape $T \times H \times W$, stored in memory in the order T, H, W. This sequence is divided into $N_T \times N_H \times N_W$ 3D blocks, where $N_T = \lceil T/t \rceil$, $N_H = \lceil H/h \rceil$, and $N_W = \lceil W/w \rceil$, and each block has shape $t \times h \times w$. The blocks are arranged in memory in the order $[N_T, N_H, N_W]$ (block-wise order), and within each block, the elements are stored in the order $[t, h, w]$ (intra-block order). After this rearrangement, we obtain a reshaped sequence.

Block Selection Mask Construction Let X be the input tensor after rearrangement. We compute the query Q and key K matrices using learnable weights W_q and W_k:

$$Q = XW_q \in \mathbb{R}^{b \times n_h \times s_q \times d}, \quad K = XW_k \in \mathbb{R}^{b \times n_h \times s_k \times d},$$

where b is the batch size, n_h is the number of attention heads, s_q and s_k are the sequence lengths for queries and keys respectively (with $s_q = s_k = T \times H \times W$ in this case), and d is the feature dimension.

To reduce computational cost, we perform average pooling over each block. Let $n = t \times h \times w$ be the number of elements per block. The pooled query Q_{pool} and key K_{pool} are computed by averaging over the elements within each block:

$$Q_{\text{pool}}[:, :, b_q, :] = \frac{1}{n} \sum_{j=0}^{n-1} Q[:, :, (b_q - 1)n + j, :] \quad \text{for} \quad b_q = 1, \ldots, N_q,$$

$$K_{\text{pool}}[:, :, b_k, :] = \frac{1}{n} \sum_{j=0}^{n-1} K[:, :, (b_k - 1)n + j, :] \quad \text{for} \quad b_k = 1, \ldots, N_k,$$

where $N_q = s_q/n$ and $N_k = s_k/n$ are the number of query and key blocks respectively.

The pooled score matrix S_{pool} is then calculated as:

$$S_{\text{pool}} = \frac{Q_{\text{pool}} K_{\text{pool}}^\top}{\sqrt{d}} \in \mathbb{R}^{b \times n_h \times N_q \times N_k},$$

where $K_{\text{pool}}^\top$ denotes the transpose of the last two dimensions of K_{pool}.

For each query block $i \in [0, N_q - 1]$, we select the top r key blocks based on the highest scores in $S_{\text{pool}}[:, :, i, :]$. This allows us to construct a binary mask matrix $M \in \mathbb{R}^{b \times n_h \times s_q \times s_k}$ as follows:

$$M[:, :, in : (i+1)n, jn : (j+1)n] = \begin{cases} 1 & \text{if key block } j \text{ is in the top-}r \text{ neighbors of query block } i \\ 0 & \text{otherwise} \end{cases},$$

Attention with Block Selection Mask Finally, we compute the masked attention. The attention score matrix S is:

$$S = \frac{QK^\top}{\sqrt{d}} \in \mathbb{R}^{b\times n_h \times s_q \times s_k},$$

where $K^\top$ is the transpose of the last two dimensions of K. We then apply the mask:

$$S_{\text{masked}} = \begin{cases} S & \text{where } M = 1 \\ -\infty & \text{where } M = 0 \end{cases},$$

and the attention weights are obtained by applying softmax along the last dimension:

$$O = \text{softmax}(S_{\text{masked}}).$$

A.2.2 Modeling of Ring Block Sparse Attention for Context Parallelism

We extend the sparse attention computation with context parallelism. Given a tensor parallelism size of N_{cp}, each parallel rank maintains a local segment of $\frac{T\times H\times W}{N_{cp}}$ latents. Let $Q_i, K_i, V_i \in \mathbb{R}^{b\times n_h \times \frac{T\times H\times W}{N_{cp}} \times d}$ denote the query, key, and value tensors respectively for the i-th rank.

Local Block Selection Mask Construction To compute the block-sparse attention mask $M_i \in \mathbb{R}^{b\times n_h \times \frac{N_q}{N_{cp}} \times N_k}$ for rank i, each rank first computes its own local pooled keys:

$$K_{\text{pool}_j}[:,:,b_j,:] = \frac{1}{n}\sum_{m=0}^{n-1} K_j[:,:,(b_j-1)n+m,:] \quad \text{for} \quad b_j = 1,\ldots,\frac{s_k}{N_{cp}},$$

where $K_j = K[:,:,(j-1)\frac{s_k}{N_{cp}} : j\frac{s_k}{N_{cp}},:]$, $j \in [1, N_{cp}]$. Then we gather the pooled key representations and compute the pooled score matrix for rank i:

$$S_{\text{pool}_i} = \frac{Q_{\text{pool}_i}\left(\bigoplus_{j=1}^{N_{cp}} K_{\text{pool}_j}\right)^\top}{\sqrt{d}}$$

where $\bigoplus$ denotes concatenation along the sequence dimension and $Q_i = Q[:,:,(i-1)\frac{s_q}{N_{cp}} : i\frac{s_q}{N_{cp}},:]$, $i \in [1, N_{cp}]$. Based on S_{pool_i}, the mask M_i is constructed by selecting the top-r key blocks for each query block across all batches and heads.

To optimize efficiency, we employ a ring-attention communication pattern where the computation of local pooled scores overlaps with the communication of K_{pool_i} tensors between adjacent ranks.

Ring Attention with Local Block Selection Mask Once M_i is obtained, each rank computes its attention output O_i by the online softmax algorithm with $M_{ij} \in \mathbb{R}^{b\times n_h \times \frac{N_q}{N_{cp}} \times \frac{N_k}{N_{cp}}}$, which is the block of mask M_i corresponding to rank j. Ring-attention [Liu et al., 2023] is adopted to overlap the attention computation and the communication of K_j, V_j.

A.2.3 Implementation Details

Our hardware-aligned 3D Block Sparse Attention operator is implemented using Triton[Tillet et al., 2019], building upon the implementation of Flash Attention[Dao, 2023]. We implemented both forward and backward passes for both single-GPU and context-parallel configurations.

3D block size The 3D block size is set to $t = h = w = 4$. This configuration represents a trade-off between speed and flexibility. In our implementation, the fastest performance is achieved when $t_q \cdot h_q \cdot w_q = 128$ and $t_k \cdot h_k \cdot w_k = 1024$ (i.e., the default configuration of $t \cdot h \cdot w = 64$ is not the fastest due to the hardware alignment), but this comes at the cost of reduced flexibility in handling varying resolutions, especially N_{cp} is large. In our experiments, we observed no significant differences in post-training results across various tested configurations of 3D block sizes, with $t_q \cdot h_q \cdot w_q$ values in [64, 128] and $t_k \cdot h_k \cdot w_k$ values in [64, 128, 256, 512, 1024].

Sparsity The hyperparameter r controls the number of key blocks selected per query block. The computational complexity scales linearly with r. We set r to $\frac{1}{8}N_k$ during the distillation training phase and to $\frac{1}{16}N_k$ during the refinement-expert training phase.

Construction of the Block Selection Mask Regarding the construction of the block selection mask, two primary strategies are explored:
1) Top-r mode: As described earlier, this approach selects the top r key blocks based on their pooled attention scores.
2) CDF-p mode: This method selects key blocks in descending order of their pooled scores until the cumulative softmax of the scores reaches a threshold p.

In our experiments, the CDF-p mode yields better generation quality under high speedup ratios in a training-free setting. However, in trainable scenarios, it suffers from the time cost caused by different number of key blocks selected by the query blocks. Therefore, we adopted the top-r approach for our trainable implementation.

A.3 Appendix-C

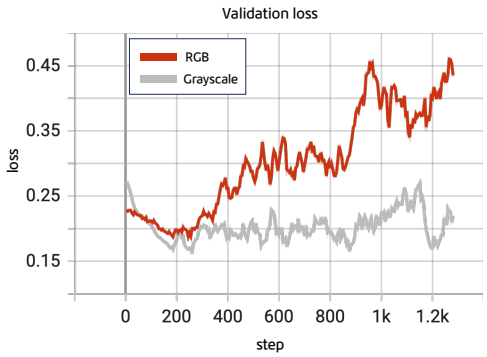

Figure 20: MQ Reward model validation loss curve