MAGI-1: Autoregressive Video Generation at Scale

Sand AI
research@sand.ai

Abstract

We present MAGI-1, a world model that generates videos by *autoregressively* predicting a sequence of video chunks, defined as fixed-length segments of consecutive frames. Trained to denoise per-chunk noise that increases monotonically over time, MAGI-1 enables causal temporal modeling and naturally supports streaming generation. It achieves strong performance on image-to-video (I2V) tasks conditioned on text instructions, providing high temporal consistency and scalability, which are made possible by several algorithmic innovations and a dedicated infrastructure stack. MAGI-1 facilitates controllable generation via chunk-wise prompting and supports real-time, memory-efficient deployment by maintaining constant peak inference cost, regardless of video length. The largest variant of MAGI-1 comprises 24 billion parameters and supports context lengths of up to 4 million tokens, demonstrating the scalability and robustness of our approach. The code and models are available at `magi-source` and `magi-attention`. The product can be accessed at `magi-product`.

1 Introduction

World modeling and video generation have emerged as central challenges in artificial intelligence, requiring the synthesis of temporally coherent and photorealistic sequences conditioned on semantically rich inputs such as natural language, static imagery, or short video clips. This task resides at the intersection of spatial perception and temporal reasoning, with profound implications for fields including robotics, embodied artificial intelligence, interactive media, and scientific simulation. As video becomes a dominant modality for both human communication and machine understanding, the demand for generative models that are not only high-fidelity and computationally efficient, but also causally consistent and compatible with streaming applications, has become increasingly urgent.

Building on the remarkable success of diffusion (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) and flow-matching frameworks (Lipman et al., 2022; Liu et al., 2022a) in image generation, recent research has increasingly focused on extending these approaches to video synthesis. However, most large-scale video diffusion models continue to rely on globally conditioned denoising architectures that process the entire temporal sequence simultaneously. These models typically employ uniform noise levels and require full-sequence access during inference. Such designs disregard the causal structure inherent to temporal data, rendering them suboptimal for scenarios requiring streaming, real-time interaction, or autoregressive generation.

To overcome these limitations, we present MAGI-1: a large-scale diffusion-based generative model that produces video through the *autoregressive* generation of temporally segmented chunks, each consisting of a fixed-length sequence of consecutive frames. This chunk-wise approach offers a principled trade-off between causal modeling and temporal abstraction, enabling the model to capture mid-range temporal dependencies while maintaining strict left-to-right temporal consistency. Training is conducted at the chunk level with temporally *progressive* noise levels, resulting in a model that is both autoregressively structured and adaptable in its conditional generation capacity.

MAGI-1 adheres strictly to causal constraints and facilitates real-time, streaming-compatible video synthesis that approximates multi-step diffusion trajectories with reduced-step, chunk-level predictions. This is enabled by a Transformer (Vaswani et al., 2017) backbone specifi-

cally designed for bidirectional spatial and causal temporal denoising, supported by a carefully engineered training infrastructure. Central to this infrastructure is a novel distributed attention mechanism (MagiAttention) tailored for ultra-long autoregressive contexts, along with a scalable execution framework optimized for low-latency, parallelized inference. These core components are further augmented by a robust data curation pipeline that supports multi-stage training and dynamically adapts the data distribution based on ongoing model evaluation. Together, these architectural and algorithmic advances empower MAGI-1 to deliver efficient, scalable, and controllable video generation. Notably, the inference-time peak resource usage of MAGI-1 is independent of the total video length, as each chunk is processed with a fixed computational and memory footprint. This makes MAGI-1 particularly suitable for low-latency, memory-efficient applications. The largest variant of the model comprises 24 billion parameters and supports context lengths of up to 4 million tokens, demonstrating the scalability and robustness of the framework.

We evaluate MAGI-1 using both internal metrics and publicly available benchmarks, with a particular focus on the image-to-video (I2V) generation task. Our evaluation protocol assesses prompt fidelity, temporal coherence, and subject integrity. On VBench-I2V (Huang et al., 2024) and Physics-IQ Benchmark (Motamed et al., 2025), MAGI-1 achieves substantial improvements over previous models, especially in its ability to synthesize complex motion, preserve semantic alignment, and model physically plausible interactions.

In summary, MAGI-1 establishes a scalable and autoregressive foundation for diffusion-based video synthesis. By integrating architectural innovations, high-throughput inference techniques, and a comprehensive data processing framework, MAGI-1 bridges the gap between high-quality generative performance and real-time applicability. The complete inference codebase and pre-trained models are publicly accessible at `magi-source`, the distributed attention available at `magi-attention`, and a live demonstration available at `magi-product`.

2 MAGI-1

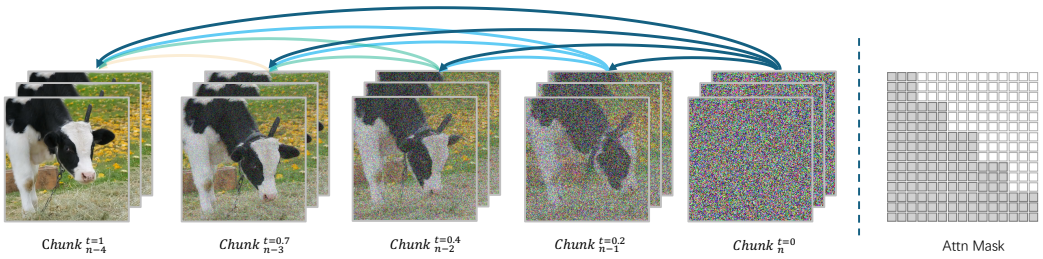


Figure 1: (Left) MAGI-1 performs chunk-wise autoregressive denoising. The video is generated in chunks of 24 frames, where each chunk attends to all previously denoised chunks. Once a chunk reaches a certain denoising level, the next chunk begins generation. (Right) A block-causal attention mask enforces temporal causality across chunks, enabling pipelined and parallel generation.

MAGI-1 is an autoregressive denoising video generation model operating in latent space. The generation process is illustrated in Fig. 1. Unlike other bi-directional denoising models (*e.g.*, Sora (OpenAI, 2024)) that generates the video as a whole, MAGI-1 generates the video chunk-by-chunk in a pipeline manner. Specifically, each chunk consists of multiple frames that are denoised holistically. As a chunk is denoised to a certain extent (not necessary completely clean), the next chunk begins generation, conditioned to all preceding chunks. This design allows multiple chunks to be processed concurrently. In our implementation, each chunk contains 24 raw frames (equivalent to one second video clip at 24 FPS), and up to four chunks can be inferred simultaneously.

Compared to fully denoising one chunk before starting subsequent chunks, our method leverages parallelism to better utilize computation, reducing the latency of obtaining sub-

sequent clean chunks and enabling real-time streaming video generation. Moreover, the auto-regressive design naturally supports video continuation without additional specific designs, and extends seamlessly to image-to-video generation. This unified framework enables us to cover text-to-video generation, video continuation, and image-to-video generation within a single pre-training process, eliminating the need for task-specific fine-tuning required by other methods. By maintaining consistency between pre-training and downstream tasks, our approach achieves superior performance in both video continuation and image-to-video generation.

In this section, we will systematically introduce the training, distillation, and inference of MAGI-1 in detail.

2.1 Transformer-based Variational Auto-Encoder

To improve the efficiency of both training and inference, MAGI-1 employs a variational autoencoder (VAE) to obtain a compressed latent space, over which denoising is performed. While most open-source VAEs are built upon convolutional architectures (*e.g.*, U-Net (Ronneberger et al., 2015)), they are considerably slower than the transformer-based counterparts (*e.g.*, ViT (Dosovitskiy et al., 2020)) of comparable model size on modern GPUs. To address this, we design our VAE architecture based on transformers.

The architecture of our VAE is illustrated in Fig. 2. In the encoder, the input is first processed by an embedding module based on a 3D convolution with a kernel size of $8 \times 8 \times 4$[1] and a stride of $8 \times 8 \times 4$, producing an output with 1024 channels. Absolute positional embeddings are then added to enrich spatial and temporal representations. Building on this, we stack 24 transformer blocks, where self-attention is stabilized through query, key and value normalization to improve training stability. The output of the final transformer block is normalized by a LayerNorm and then projected via a linear layer to 32 channels: the first 16 channels represent the predicted mean, and the remaining 16 channels represent the predicted log-variance. Compared to the raw video input, the encoded features are downsampled by a factor of 8 in the spatial dimensions and by a factor of 4 in the temporal dimension.

The decoder adopts a symmetric architecture to the encoder. To restore the original spatial and temporal resolution, we first apply a pixel shuffle operation to the output of the final transformer block, followed by a 3D convolution with a kernel size of $3 \times 3 \times 3$ and 3 output channels to generate the final output in pixel space. For image inputs consisting of a single frame, we replicate the frame four times along the temporal dimension, which yields better performance compared to padding with three empty frames.

VAE	PSNR	Params (M)	Avg Encode Time (ms)	Avg Decode Time (ms)
OpenSoraPlan-1.2 (Lin et al., 2024)	28.39	239	51.08	17.48
CogVideoX (Yang et al., 2025)	35.99	216	40.19	142.96
HunyuanVideo (Kong et al., 2024)	37.27	246	124.39	47.11
StepVideo (Ma et al., 2025)	33.75	499	30.47	18.12
Wan2.1 (Wang et al., 2025a)	35.95	127	51.91	79.43
Ours	36.55	614	36.68	12.28

Table 1: Comprehensive comparison of our VAE with other open-source approaches. Thanks to the optimized inference support of transformers, our VAE achieves the fastest decode speed under identical hardware conditions, despite having the largest model size.

The training process of the VAE consists of two stages. In the first stage, we use a fixed input resolution during training: 16-frame short clips with a spatial resolution of 256×256 pixels, to maximize training efficiency by avoiding unnecessary padding. In the second stage, two key modifications are introduced. First, both image data (single frame) and video

[1]The kernel size is specified in the order of height, width, and temporal dimensions.

data (16-frame clip) are jointly used during training. Second, we adopt variable spatial resolutions and aspect ratios by randomly sampling at each training step, enabling the VAE to generalize across different resolutions. Specifically, we constrain the total number of pixels (height $\times$ width) is approximately 256^2 or 384^2, while sampling the aspect ratio uniformly from the range [0.25, 4.0]. In both stages, we apply a combination of L1 loss, KL divergence loss, LPIPS loss, and GAN loss, following common practice.

During inference, we use sliding window approach to support arbitrary resolutions. In the spatial dimension, we adopt a window size of 256 $\times$ 256 pixels with a stride of 192 pixels, resulting in a 25% overlap between adjacent patches in spatial. In the temporal dimension, no overlap is applied.

Tab. 1 shows the comparison with other open-source VAEs. All models were evaluated on a single NVIDIA H800 GPU. To eliminate potential biases from varying slicing strategies at higher resolutions, we report the average processing speed measured across 169 test videos, each containing 25 frames with a spatial resolution of 256×256 pixels. Despite having the largest model size, our transformer-based VAE achieves the fastest average decoding time among all models and significantly outperforms most baselines in encoding speed. In terms of reconstruction quality (measured by PSNR), it remains highly competitive, ranking second overall.

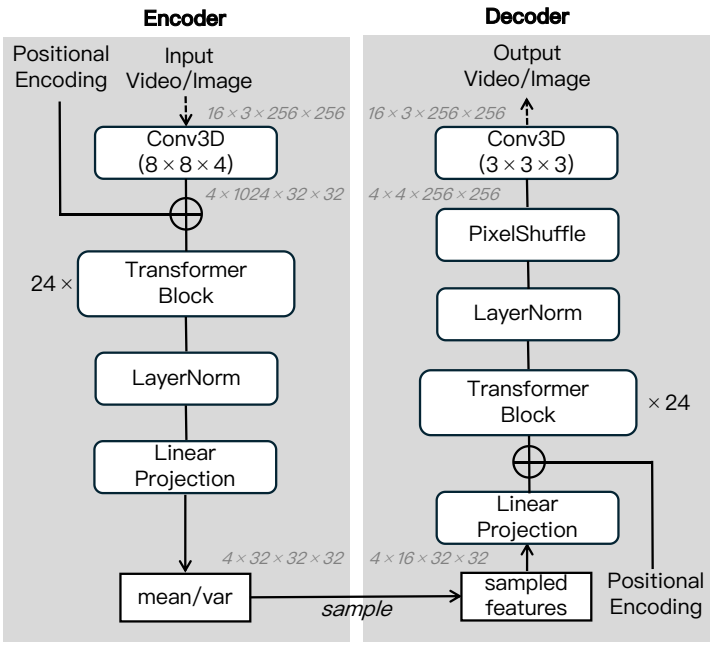


Figure 2: Model Architecture of Transformer-based VAE.

2.2 Auto-Regressive Denoising Model

2.2.1 *Training objective*

MAGI-1 employes flow-matching (Albergo & Vanden-Eijnden, 2022; Liu et al., 2022a; Lipman et al., 2022) as its training objective. Given a training video clip contains n chunks, we sample independent Gaussian noises for each chunk. The linear interpolation with respect to the denoising timestep t between the sampled noise and the clean latent of the i-th chunk is defined as:

$$x_i^t = (1-t)x_i^0 + tx_i^1, \tag{1}$$

where x_i^1 denotes the latent of i-th chunk and x_i^0 is the corresponding sampled Gaussian noise. The ground-truth velocity for each chunk is given by:

$$v^*(x_i^t) = \frac{dx_i^t}{dt} = x_i^1 - x_i^0. \tag{2}$$

In the auto-regressive model, earlier chunks are cleaner than later ones. For convenience, we define the noise timestep sampled assigned to each chunk as t_i, and impose the constraint $t_i < t_j$ whenever $i < j$.[2] The interpolation of the entire video clip is then defined as: $X_T = \{x_0^{t_0}, x_1^{t_1}, ..., x_n^{t_n}\}$. The model is trained to minimize the following objective:

$$\mathbb{E}_{c,X_T} \parallel v(x_i^{t_i}|t_i, c, \{x_{j<i}^{t_j}\}; \theta) - v^*(x_i^{t_i}) \parallel^2 . \tag{3}$$

where $v(\cdot;\theta)$ is the denoising model parameterized by θ, and c denotes the conditioning text inputs. Note that the prediction of velocity for x_i explicitly conditioned on all its preceding chunks x_j where $j < i$.

In contrast, typical bi-directional denoising video models do not enforce monotonicity of the noise timestep. Instead, they apply the equality constraint, where all chunks share the same noise timestep. Accordingly, their training objective is formulated as:

$$\mathbb{E}_{c,X_T} \parallel v(x_i^{t_i}|t_i, c, X_T; \theta) - v^*(x_i^{t_i}) \parallel^2 . \tag{4}$$

where the velocity prediction for x_i is conditioned on all chunks, regardless their temporal order.

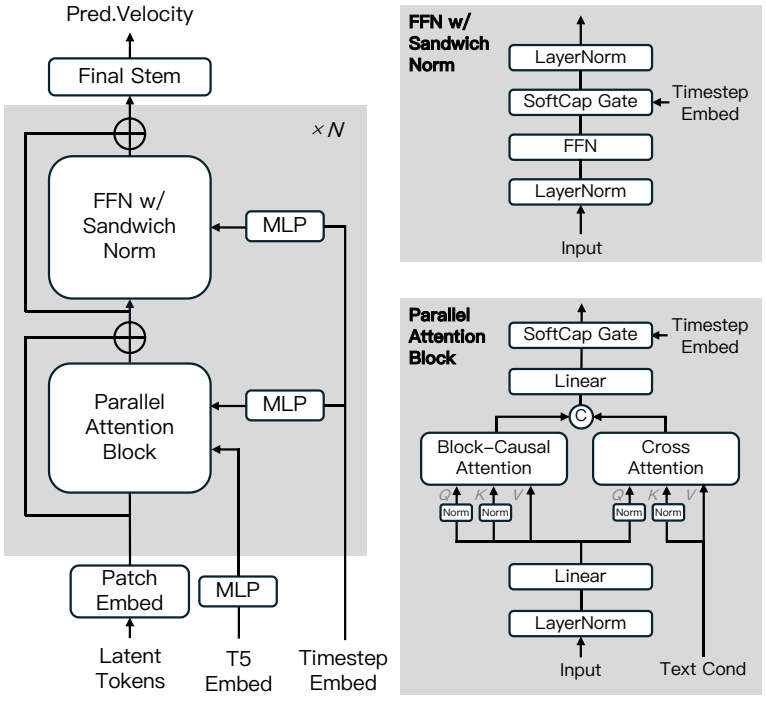


Figure 3: Model Architecture of Auto-Regressive Denoising Model.

[2]In completely denoising cases, $t_i = t_j = 0$, but we use the strict inequality here for simplicity.

2.2.2 *Model Architecture*

MAGI-1 is built upon the Diffusion Transformer (DiT) architecture. However, to better meet the requirements of auto-regressive modeling and to improve training efficiency and stability at scale, we introduce several key modifications. As shown in Fig. 3(a), MAGI-1 follows a high-level architecture similar to the standard DiT, consisting of four main components: patch embedding, attention, feed-forward network (FFN), and final stem. We employ T5 (Raffel et al., 2020) to extract text embeddings, while the timestep information is encoded using sinusoidal positional embeddings. Our primary modifications target the attention and FFN modules, which are illustrated in Fig. 3(b) and Fig. 3(c), respectively. In the following, we provide a detailed description of these modifications.

Block-Causal Attention MAGI-1 employs full attention within each chunk and causal attention across chunks. Spatial and temporal positional information is encoded using a learnable 3D RoPE (Su et al., 2024), in which the base frequency is learnable. However, existing attention implementations (Dao et al., 2022; Dao, 2023) do not efficiently support block-causal attention, therefore, we implemented a new kernel called Flexible-Flash-Attention on top of FlashAttention-3. Further details can be found in Sec. 4.1.2.

Parallel Attention Block MAGI-1 adopts a parallel design for spatial-temporal self-attention and cross-attention with external conditioning input, offering improved computational efficiency over the serial attention architecture. In the serial setup, each attention module independently computes query projections and incurs a separate round of Tensor Parallel (TP) communication. In contrast, the parallel block computes query projections once and applies them to both attention types concurrently, reducing TP communication from two rounds to one per block. This optimization lowers inter-GPU synchronization overhead and enhances scalability in large-scale models.

QK-Norm and GQA Earlier studies on vision transformers (Liu et al., 2022b; Dehghani et al., 2023) have shown that normalizing the queries and keys of attention can significantly improve training stability. Moreover, inspired by recent advances in large language models (LLMs), we replace the standard multi-head attention (MHA) with grouped-query attention (GQA) (Ainslie et al., 2023) to reduce memory consumption. Both techniques are applied to the spatial-temporal attention and cross-attention modules in our design.

Sandwich Normalization in FFN In practice, we have noticed that the numerical problems are more likely to appears in FFN modules as the model size increase. Therefore, we have added LayerNorm before and after the FFN input and output to alleviate the challenge.

SwiGLU SwiGLU (Shazeer, 2020) has been widely adopted in large language models and has been shown to consistently improve performance than ReLU. Therefore, we employ SwiGLU in the feed-forward network (FFN) of our 24B model.

Softcap Modulation The standard DiT incorporates timestep information via *adaLN*, where the denoising timestep is used to compute a scaling factor that modulate both the input and output activations of the attention and FFN. While this design works well for small models, we observed that in large models it tends to amplify activation magnitudes and exacerbate numerical instability. To address this issue, we apply a Softcap to the scaling factor, constraining its values within the range of $[-1, 1]$. Furthermore, since we adopt QK-Norm in attention modules, we remove the input modulation of *adaLN*.

2.2.3 *Training Recipes*

Training Configurations We train a 4.5B and 24B MAGI-1 models and their configurations is shown in Tab. 2. The training is organized into three stages. Take the 4.5B model as an example. In the first two stages, the resolution of training data is set to 360p and 480p, respectively, with video length is up to 8 seconds. In the third stage, the resolution is further increased to 720p, and the video length is extended up to 16 seconds. Throughout all three

	4.5B	24B
Layers	34	48
Model Dimension	3072	6144
FFN Activation	GLU	SwiGLU
FFN Dimension	12288	16384
Attention Type	GQA + QK-Norm	GQA + QK-Norm
Block-Casual Attention Head	128	128
Block-Casual Attention Group	8	8
Cross Attention Head	128	128
Cross Attention Group	8	8
Positional Embedding	Learnable 3d RoPE	Learnable 3d RoPE
Optimizer	AdamW	AdamW
Weight Decay	1×10^{-1}	1×10^{-1}
Peak LR	1×10^{-4}	1×10^{-4}
Warm-up	1000	10000
β_1	0.9	0.9
β_2	0.95	0.95

Table 2: Model Specification of MAGI-1.

stages, the image and video are trained jointly. At the beginning of training, we apply a learning rate warmup, gradually increasing the learning rate to 1e$-$4 in 1000 steps. Then, we adopt a stepwise learning rate scheduling strategy. In the first two stages, the learning rate remains constant, and the stage is switched when the visual assessment of generated video does not significantly improve. In the third stage, we gradually reduce the learning rate once the validation loss reaches a plateau, eventually reducing to 1e$-$5.

For the 24B model, we reduce the resolution in the first training stage from 360p to 256p, as this stage primarily serves to making the model learn global motion dynamics and semantic concepts. Lowering the resolution allows for more training iterations within the same computational budget, thereby improving training efficiency. In addition, we extend the learning rate warmup phase to 10,000 steps to enhance stability during the early training phase. Furthermore, since larger models typically require longer training to reach performance saturation, we proportionally increase the number of training steps at each stage, guided by empirical visual assessment on the validation set.

Multi-Task Training via Data Configurations Bi-directional denoising models typically support only text-to-video generation during pretraining, while tasks such as image-to-video generation often require dedicated architectural designs or additional finetuning. In contrast, within the auto-regressive framework, text-to-video, image-to-video, and video continuation tasks differ solely in the proportion of clean chunks present in the training data. As illustrated in Fig. 4, the early stage of text-to-video generation corresponds to the case of all chunks are noisy, while the inclusion of some clean chunks represents to video continuation. Image-to-video generation is a special case of video continuation, with only the first frame of the first chunk being clean.

Thanks to this property, our auto-regressive model enables unification of various generation tasks under a single training objective without additional task specific fine-tuning and requiring only adjustment of the proportion of clean chunks in the training data.

Furthermore, unlike bi-directional denoising models — where the text condition must be predefined for the entire video and remains fixed throughout generation — MAGI-1 allows for different text conditions to be provided for each chunk, enabling fine-grained, chunk-wise text control. To better support this capability, we design a dedicated auto-regressive captioning strategy (Data details are described in Sec. 3.4) that adapts training accordingly. Additional examples of this fine-grained control are provided in Sec. 2.6.

Timestep Sampler in Training Early studies have demonstrated that improving the design of timestep sampler (commonly known as SNR sampler) can facilitate training efficiency by

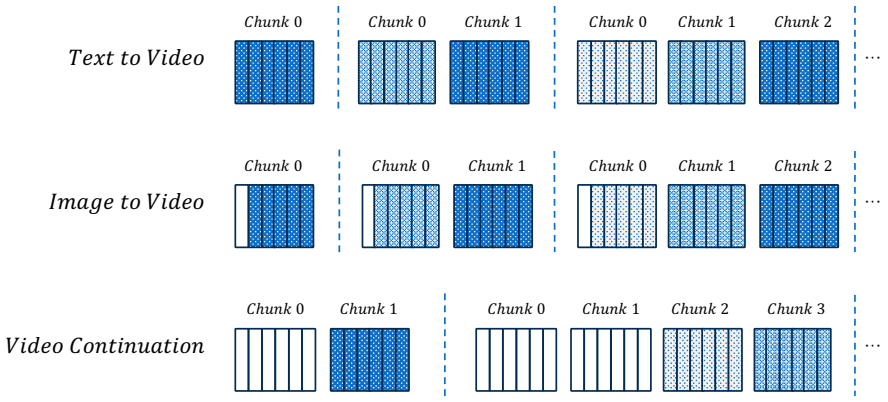


Figure 4: The figure shows how different tasks can be unified by varying the proportion of clean chunks. Each vertical bar represents a latent frame in a chunk, with darker bars indicating higher noise levels and the *white* bars denoting clean frames. The first row illustrates the early inference stage of T2V generation, starting from a single fully noisy chunk and progressing to multiple noisy chunks, before any clean chunk has been produced. The middle row depicts the case of I2V generation, treated as a special case of continuation in which only the first frame of the first chunk is clean. The last row describes a general stage where clean chunks are already available, applicable to video continuation and other scenarios involving prior denoised content.

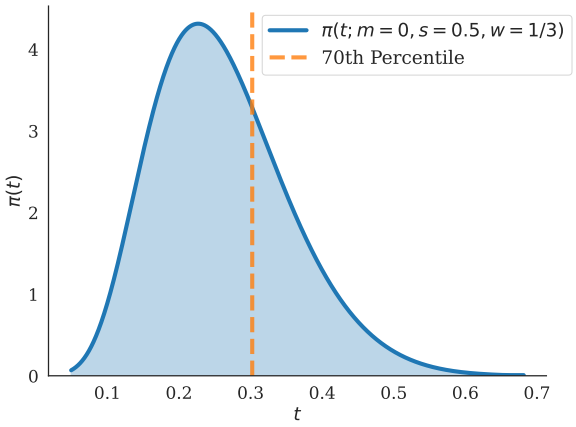


Figure 5: The probability density of training timestep. We generally aim to allocate 70% of training computation when t < 0.3.

better allocating computations across different noise levels. (Esser et al., 2024) introduce the Logit-Normal sampling strategy, which provides a flexible framework for controlling the distribution of sampled timestep, the transformed timestep density $\pi(t)$ is:

$$\pi(t; m, s) = \frac{1}{s\sqrt{2\pi}} \frac{1}{t(1-t)} \exp\left(-\frac{(\text{logit}(t) - m)^2}{2s^2}\right), \tag{5}$$

where $\text{logit}(t) = \log \frac{t}{1-t}$. In addition, (Esser et al., 2024) further introduce a timestep shift strategy to handle the resolution increasing:

$$t' = \frac{wt}{1 - (1-w)t} \tag{6}$$

In MAGI-1, we draw inspiration from these two sampling strategies but make adjustment for video data. Since videos typically contain more redundant information than images, we

aim to shift the overall sampling distribution further towards the noise side compared to images. In our preliminary experiments, as shown in Fig. 6, we observed that the model is capable of generating reasonably clear video outputs at $t = 0.3$. Based on this observation, we heuristically allocate approximately 70% of the training computation budget to the region where $t < 0.3$. Following this principle, we set the $m = 0$, $s = 0.5$, and $w = 1/3$ for all cases during training.

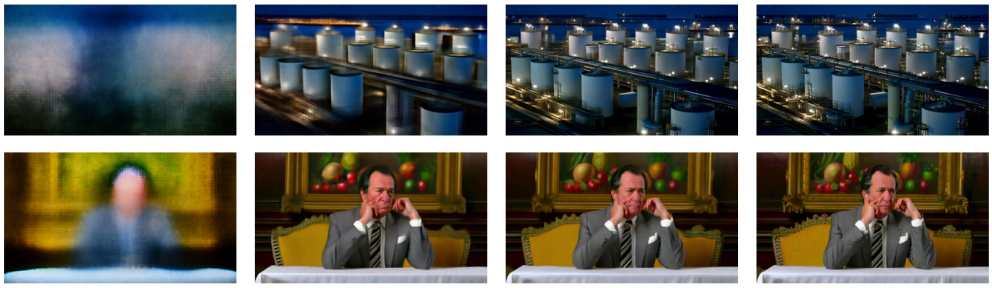


Figure 6: The generation results at the given timestep t. Through empirical experiments, we found that model is capable of generating quite clear video outputs at $t = 0.3$.

Design Choices for Clean Chunks There are two types of chunks in the training of MAGI-1: noisy chunks and clean chunks, and we adopt three key designs to handle clean chunks:

First, in practical video continuation scenarios, users typically upload an initial video clip and provide follow-up text descriptions or dynamically update the prompt during the continuation process. Considering this usage, we argue that clean chunks should not be conditioned on text inputs.

Second, exposure bias is a well-recognized challenge in training auto-regressive models (Bengio et al., 2015; Wiseman & Rush, 2016). A common mitigation strategy is to inject a small amount of noise into clean data during training. Following this practice, we inject up to 5% noise into clean chunks to alleviate exposure bias.

Finally, since clean chunks are relatively abundant in the pre-training data, they risk dominating the training signal. To address this, we apply the loss function exclusively to noisy chunks. Nevertheless, clean chunks still participate in training through the attention mechanism and continue to receive gradient updates. Empirically, we observe that blocking the gradients of clean chunks leads to a significant degradation in model performance.

2.3 Distillation Using Shortcut Model

Flow-matching formulates the generative process as an ODE that maps noise to data along high-dimensional, curved trajectories. Sampling from such models is computationally intensive, typically requiring dozens of function evaluations with sufficiently small step sizes to incrementally transform noise into data. This inefficiency motivates the development of diffusion distillation methods (Luhman & Luhman, 2021; Salimans & Ho, 2022) that can reduce the required number of inference steps without sacrificing sample quality.

This work adopts shortcut model (Frans et al., 2024) as the distillation target. Given a noise-data interpolation defined by $x_i^t = (1-t)x_i^0 + tx_i^1$, where x_i^1 is the clean data point at the i-th chunk and x_i^0 denotes Gaussian noise, the shortcut model uses a *single* neural network to predict a velocity field $v(x_i^t \mid t, s)$[3], conditioned not only on the current timestep t, but

[3]For clarity, we omit irrelevant variables and denote $v(x_i^t \mid t, s)$ as a shorthand for the full expression $v(x_i^{t_i} \mid t_i, s, c, \{x_{j<i}^{t_j}\}; \theta)$.

also on the *desired step size* $s \propto \Delta t$. Here, Δt denotes the interval between adjacent timesteps, while the reciprocal $1/s \in 2^{\mathbb{N}}$ specifies the number of function evaluations required to complete the denoising process[4].

The generation process of the shortcut model closely resembles the flow-matching formulation and can be expressed as $\hat{x}_i^{t+\Delta t} = x_i^t + \Delta t \cdot v(x_i^t, t, s)$, where $\hat{x}_i^{t+\Delta t}$ denotes the model-predicted next point in the denoising trajectory, explicitly indicated by the hat symbol over $x_i^{t+\Delta t}$. As $\Delta t \to 0$, this formulation recovers the standard flow-matching scenario, where the shortcut model approximates the instantaneous velocity.

During training, the shortcut model constructs distillation targets using a bootstrap procedure, leveraging the principle that a single shortcut step is equivalent to two consecutive steps of half the step size. Formally, the update rule $\hat{x}_i^{t+\Delta t_1+\Delta t_2} = x_i^t + (\Delta t_1 + \Delta t_2) \cdot v(x_i^t, t, 2s)$ can also be written as $\hat{x}_i^{t+\Delta t_1+\Delta t_2} = \hat{x}_i^{t+\Delta t_1} + \Delta t_2 \cdot v(\hat{x}_i^{t+\Delta t_1}, t + \Delta t_1, s) = x_i^t + \Delta t_1 \cdot v(x_i^t, t, s) + \Delta t_2 \cdot v(\hat{x}_i^{t+\Delta t_1}, t + \Delta t_1, s)$, leading to the relationship:

$$v(x_i^t, t, 2s) = \frac{\Delta t_1}{\Delta t_1 + \Delta t_2} v(x_i^t, t, s) + \frac{\Delta t_2}{\Delta t_1 + \Delta t_2} v(\hat{x}_i^{t+\Delta t_1}, t + \Delta t_1, s) \tag{7}$$

In practice, the smallest s utilized is $1/64$, corresponding to the standard flow-matching inference setting that requires 64 function evaluations. When training with this minimal step size, we incorporate classifier-free guidance (CFG) distillation (Meng et al., 2023) (see Sec. 2.4.1 for details). The step size s for distillation is cyclically sampled from the set $[1/64] \times 8 \cup [1/32, 1/16, 1/8]$. This sampling strategy enables a single distilled model to perform denoising with different computational budgets (64, 32, 16, or 8 steps), thus providing flexibility to dynamically balance generation quality and inference efficiency at test time.

2.4 Inference Approach

2.4.1 *Diffusion Guidance*

Classifier-free guidance (Ho & Salimans, 2022), a widely adopted low-temperature sampling technique in diffusion models, offers a principled approach to mediating the inherent trade-off between sample fidelity and diversity in generative modeling. This technique is particularly effective in text-to-video generation, where the objective is to synthesize temporally coherent video frames that conform to given textual prompts.

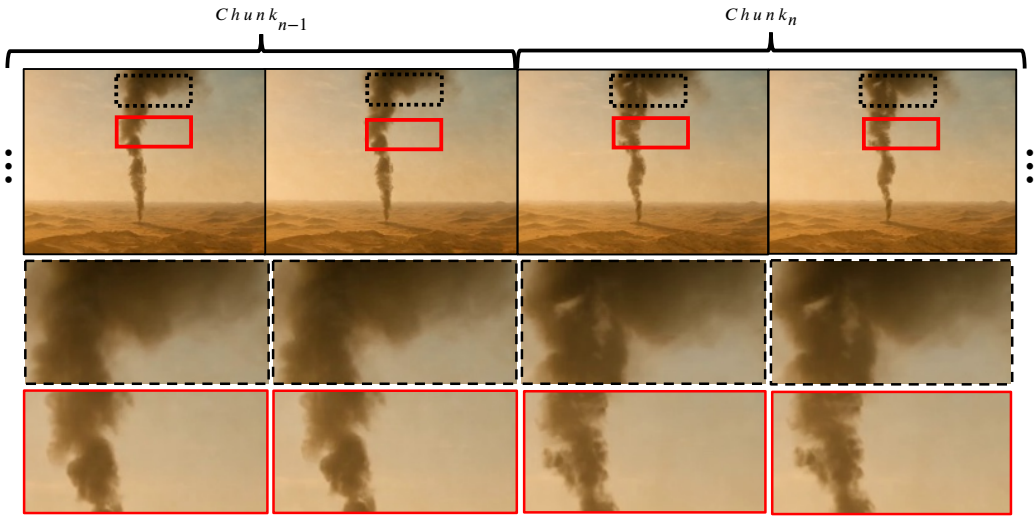
To improve clarity, we omit irrelevant variables and express the *guided* posterior distribution of the latent variable x_t given condition c using Bayes' rule as $p_{\text{guided}}(x_t \mid c) \propto p(x_t) \cdot p(c \mid x_t)^w$, where the exponent $w \geq 1$ serves as an inverse temperature parameter. Exponentiating the conditional likelihood $p(c \mid x_t)$ concentrates the distribution around modes better aligned with the conditioning signal, thereby reducing entropy and improving sample fidelity.

In our setting, the generation of the i-th video chunk x_i is conditioned not only on the textual prompt c_{text}, but also on a sequence of preceding chunks $x_{<i}$, which may include partially denoised or fully noised representations. The guided conditional distribution is thus formulated as:

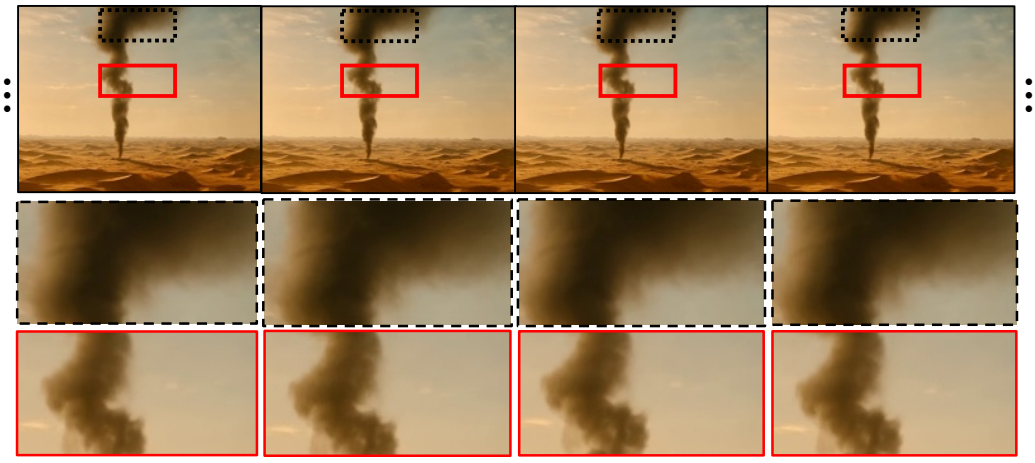
$$p_{\text{guided}}(x_i \mid x_{<i}, c_{\text{text}}) \propto p(x_i) \cdot p(x_{<i} \mid x_i)^{w_{\text{prev}}} \cdot p(c_{\text{text}} \mid x_{<i}, x_i)^{w_{\text{text}}}, \tag{8}$$

where w_{prev} and w_{text} are scalar weights modulating the influence of temporal and semantic signals, respectively. Taking the logarithm of both sides of Eq. 8, we obtain the guided score: $\nabla_{x_i} \log p_{\text{guided}}(x_i \mid x_{<i}, c_{\text{text}}) = \nabla_{x_i} \log p(x_i) + w_{\text{prev}} \cdot \nabla_{x_i} \log p(x_{<i} \mid x_i) + w_{\text{text}} \cdot \nabla_{x_i} \log p(c_{\text{text}} \mid x_{<i}, x_i)$. Applying Bayes' rule, we rewrite the gradients as $\nabla_{x_i} \log p(x_{<i} \mid x_i) = \nabla_{x_i} \log p(x_i \mid x_{<i}) - \nabla_{x_i} \log p(x_i)$, and $\nabla_{x_i} \log p(c_{\text{text}} \mid x_{<i}, x_i) = \nabla_{x_i} \log p(x_i \mid$

[4]In practice, note that s and Δt may differ due to nonlinearities in the denoising schedule.



(a) $w_{\text{prev}} = 1.0$



(b) $w_{\text{prev}} = 1.5$

Figure 7: This figure demonstrates the impact of w_{prev} on the generation results. (a) When $w_{\text{prev}} = 1.0$, there are perceptible misalignments between adjacent chunks (*e.g.*, the shape of the smoke). (b) When $w_{\text{prev}} = 1.5$, this phenomenon is significantly alleviated.

$x_{<i}, c_{\text{text}}) - \nabla_{x_i} \log p(x_i \mid x_{<i})$. Substituting and regrouping, we arrive at the final guided score:

$$\begin{aligned} \nabla_{x_i} \log p_{\text{guided}}(x_i \mid x_{<i}, c_{\text{text}}) = \; & (1 - w_{\text{prev}}) \cdot \nabla_{x_i} \log p(x_i) \\ & + (w_{\text{prev}} - w_{\text{text}}) \cdot \nabla_{x_i} \log p(x_i \mid x_{<i}) \\ & + w_{\text{text}} \cdot \nabla_{x_i} \log p(x_i \mid x_{<i}, c_{\text{text}}). \end{aligned} \tag{9}$$

This decomposition cleanly separates the contributions of the unconditional prior, temporal context, and prompt conditioning. It enables controllable trade-offs between coherence and semantic fidelity in autoregressive generation.

As a special case, when w_{prev} is 1, *i.e.*, the guidance from previous chunks is disabled, the score function in Eq. 9 simplifies to $\nabla_{x_i} \log p_{\text{guided}}(x_i \mid x_{<i}, c_{\text{text}}) = (1 - w_{\text{text}}) \cdot \nabla_{x_i} \log p(x_i \mid x_{<i}) + w_{\text{text}} \cdot \nabla_{x_i} \log p(x_i \mid x_{<i}, c_{\text{text}})$. This form recovers the standard classifier-free guidance formulation widely adopted in bidirectional text-to-video diffusion models, which interpolates between unconditional and prompt-conditioned signals.

However, during our chunk-wise generation process, we observed subtle yet perceptible misalignments between adjacent chunks, resulting in temporal artifacts. This observation underscores the necessity of reinforcing temporal guidance to maintain chunk-to-chunk coherence. To this end, we increase w_{prev} to 1.5, thereby amplifying the influence of preceding content. As shown in Fig. 7, this adjustment significantly enhances inter-chunk alignment and mitigates flickering artifacts, resulting in smoother and more temporally consistent video synthesis. Nevertheless, it should be noted that further increasing w_{prev} beyond this optimal range may lead to saturation artifacts or even cause the video to become static (*i.e.*, still frames) as playback progresses. We follow standard practice by setting w_{text} to 7.5.

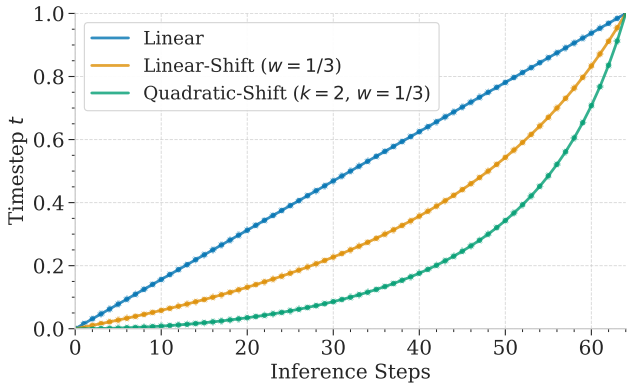


Figure 8: Inference timestep sampling of non-distilled model.

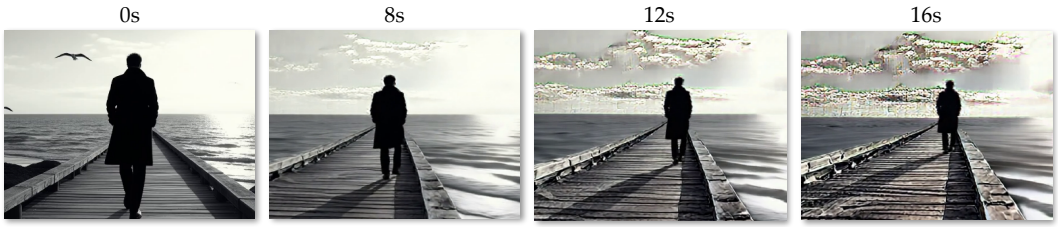
2.4.2 *Inference Timestep Sampler*

Previous video generation studies have demonstrated that applying targeted timestep sampling strategies during inference can significantly improve generation quality. In our work, we observed similar behavior in MAGI-1. To enable finer-grained control over the sampling process, we introduce an additional tunable power transformation based on the scaling formula (Eq. 6) $t' = \frac{wt^k}{1-(1-w)t^k}$. Through extensive experiments, we found that setting $w = 1/3$ and $k = 2$ yields the best visual quality, and visualization of the sampler is shown in Fig. 8.

2.4.3 *Fine-Grained Control of Guidance Strength*

Non-distilled Model. In the case of the non-distilled model, as described in Sec. 2.4.1, we set $w_{\text{prev}} = 1.5$ and $w_{\text{text}} = 7.5$ during generation. In practice, when synthesizing longer videos (typically exceeding 5 seconds), we observe noticeable saturation and checkerboard artifacts progressively emerging during playback. These artifacts are primarily attributed to excessively strong guidance. However, uniformly reducing the strength of w_{prev} and w_{text} often results in degraded content quality and increased flickering artifacts. This motivates a more fine-grained strategy in which the guidance scales are dynamically adjusted throughout the denoising process, that is, varying w_{prev} and w_{text} as the denoising timestep t progresses from 0 to 1.

To investigate when strong guidance is necessary, we analyze the evolution of latent representations throughout the denoising process (Fig. 6). As t approaches 0.3, just before the final denoising stage begins, we observe that the decoded latent representations already exhibit coherent video content, with both structural and semantic elements largely established. The remaining denoising steps, from $t = 0.3$ to $t = 1$, primarily serve to refine local details, resembling a super-resolution process. Based on this observation, we hypothesize that strong guidance from either the text or previous chunks is no longer necessary during this stage. Accordingly, for $t > 0.3$, we reduce the guidance scales to $w_{\text{prev}} = 1.0$ and $w_{\text{text}} = 0.0$, such that only the $\nabla_{x_i} \log p(x_i \mid x_{<i})$ term remains active. As illustrated in Fig. 9, this simple



(a) w/o Fine-Grained Control



(b) w/ Fine-Grained Control

Figure 9: (a) When the generation length exceeds 5 seconds, severe artifacts emerge and intensify over time. (b) By adjusting the guidance strength (*i.e.*, $w_{\text{prev}} = 1.0$ and $w_{\text{text}} = 0.0$ when $t > 0.3$), there are no serious artifacts in the entire generation.

yet effective adjustment significantly alleviates temporal artifacts and improves the overall coherence of longer video generations.

Distilled Model. A similar observation holds for the distilled model. Saturation artifacts progressively intensify as the video plays, motivating a comparable mitigation strategy. In the first three stages, we directly use the distilled model's output score, $\nabla_{x_i} \log p_{\text{distilled}}(x_i \mid x_{<i}, c_{\text{text}})$. In the final denoising range, we incorporate additional *guidance* to reduce the influence of the previous chunk, even though the model has already undergone classifier-free guidance distillation. Specifically, we adopt the following guided score in the final stage:

$$\begin{aligned}\nabla_{x_i} \log p_{\text{guided, distilled}}(x_i \mid c_{\text{text}}, x_{<i}) = (1 - w_{\text{prev}}) \cdot \nabla_{x_i} \log p_{\text{distilled}}(x_i \mid c_{\text{text}}) \\ + w_{\text{prev}} \cdot \nabla_{x_i} \log p_{\text{distilled}}(x_i \mid c_{\text{text}}, x_{<i}).\end{aligned} \tag{10}$$

This formulation is derived by switching the positions of $x_{<i}$ and c_{text} in Eq. 8 and Eq. 9, resulting in the form $\nabla_{x_i} \log p_{\text{guided}}(x_i \mid c_{\text{text}}, x_{<i}) = (1 - w_{\text{text}}) \cdot \nabla_{x_i} \log p(x_i) + (w_{\text{text}} - w_{\text{prev}}) \cdot \nabla_{x_i} \log p(x_i \mid c_{\text{text}}) + w_{\text{prev}} \cdot \nabla_{x_i} \log p(x_i \mid c_{\text{text}}, x_{<i})$. By setting $w_{\text{text}} = 1$, thereby disabling the text guidance term, the first component vanishes and the expression simplifies to Eq. 10. The rationale behind this modification is that we do not introduce a null text token during distillation, and therefore do not explicitly model $p_{\text{distilled}}(x_i)$ or $p_{\text{distilled}}(x_i \mid x_{<i})$. In our experiments, we set $w_{\text{prev}} = 0.7$, which effectively attenuates the influence of previous chunk guidance in the final denoising stage and helps mitigate temporal saturation artifacts.

2.4.4 KV Cache

Thanks to its auto-regressive nature, MAGI-1 can leverage the KV cache mechanism during inference, which is a widely adopted technique in language models to avoid redundant computations. Specifically, once a chunk has been sufficiently denoised, its features can be cached and reused by subsequent denoising chunks without the need for recomputation.

Furthermore, by constraining the KV range, MAGI-1 can easily support long video generation. For example, by setting the KV range to 8 for all chunks, each newly generated chunk depends only on the preceding 8 seconds of video content. This design ensures that the computational cost of generating long videos scales linearly with their duration.

In addition, many KV compression (Hooper et al., 2024; Xiao et al., 2023b; Sheng et al., 2023) techniques have recently been developed to reduce the computational overhead of

auto-regressive model while preserving the ability to reference the full history as much as possible. MAGI-1 is theoretically compatible with these advancements, although we leave their exploration in MAGI-1 for future work.

MAGI-1 also benefits from the unique characteristics of denoising models: at higher noise levels, the model focuses on capturing global structural information, whereas at lower noise levels, it produces fine details and textures. By dynamically adjusting the KV range at different denoising stages, we can unlock new capabilities that were previously challenging to achieve, such as enabling temporally controllable shot transitions while preserving subject identities, or allowing changes in object identities while maintaining consistent global layouts. More details and experimental results are provided in Sec. 2.6.

2.5 Prompt-Enhancement Strategy

MAGI-1 is trained with highly descriptive captions that follow a specific structure as text conditions. However, in real-world scenarios, user inputs vary widely: ranging from very brief prompts to overly elaborate descriptions. This mismatch between the training distribution and real user inputs often leads to suboptimal inference performance. To address this gap, we propose a Prompt Enhancement (PE) strategy during inference. We take the image-to-video (I2V) task as an example to illustrate our PE approach. In this setting, users typically provide an image along with an optional textual prompt. To enhance the user input, we employ a state-of-the-art multi-modal large language model (MLLM) to perform prompt refinement. Our PE pipeline consists of two parallel sub-processes:

- The first sub-process analyzes and describes the content of the uploaded image.
- The second sub-process predicts the temporal evolution of the scene or objects in the first frame, such as actions, motion trajectories, and object transitions.

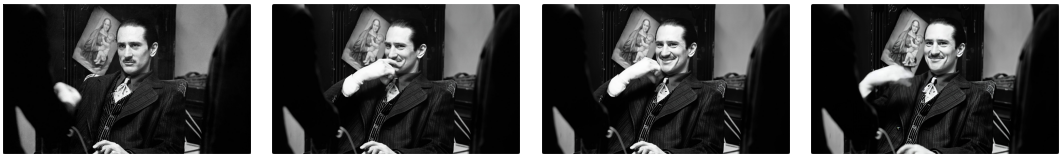
This structured enhancement strategy significantly improves generation quality. However, due to the large size of the state-of-the-art MLLM, it incurs high computational cost and latency, limiting its feasibility in real applications. To enable lightweight deployment, we distill the enhanced prompts generated by the large MLLM into a smaller, more efficient model (~7B). We construct a training corpus of approximately 2 million examples, filtering out samples with excessively long target texts to ensure controlled output length. Based on human evaluation, the distilled model achieves comparable video generation quality to its larger counterpart, while greatly reducing inference latency and computational resource usage.

2.6 Model Capability Study

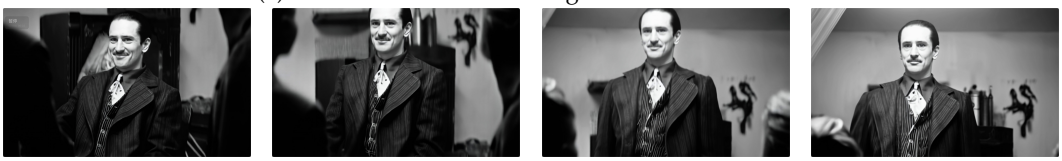
Real-time Streaming Video Generation The chunk-by-chunk pipelined inference of MAGI-1 offers two key advantages: (1) the time to display the first clear chunk is independent of the total generated video length; and (2) the generation latency between consecutive chunks is significantly reduced. Combined with a high-performance inference infrastructure, MAGI-1 enables real-time streaming video generation, unlocking new applications in interactive content and live streaming. More implementation details are in Sec. 4.2.1.

Chunk-wise Text Controllability Chunk-wise text controllability is one of the key features of MAGI-1, enabling us to decompose complex actions into simpler, shorter segments and significantly enhancing the model's ability to generate intricate action sequences. Furthermore, when combined with the capability of MAGI-1 for long video generation, this makes it possible to create videos with complex narrative structures, as illustrated in Fig. 10

Video Continuations Video continuation is a task that MAGI-1 natively supports. In the community, an alternative approach to video continuation relies on image-to-video generation (I2V), where the last frame of the given prefix video is used as the starting frame for the extended video. However, this approach often struggles to maintain consistent



(a) A man smiles while resting his chin on his hand.



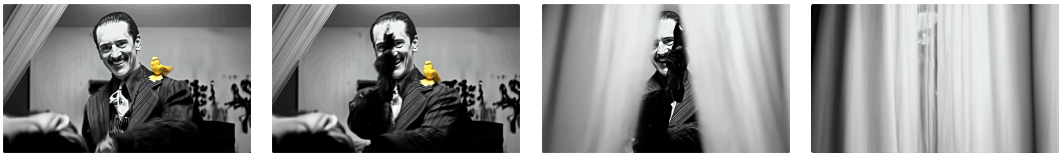
(b) He slowly rises from his seat.



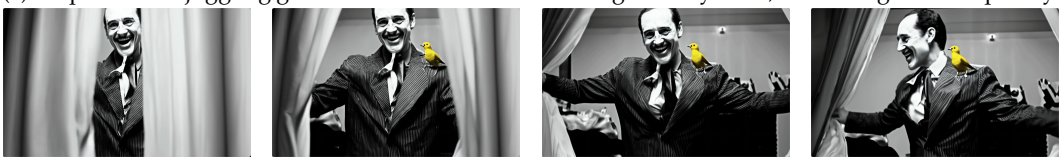
(c) Draws a pistol, from which a red rose is fired.



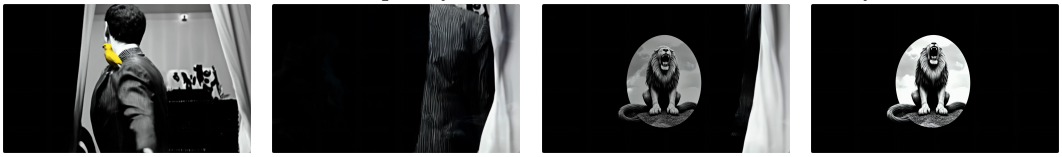
(d) The rose transforms into a yellow bird that lands on his shoulder as he makes a playful expression.



(e) He performs a juggling gesture as curtains on both sides gradually close, concealing him completely.

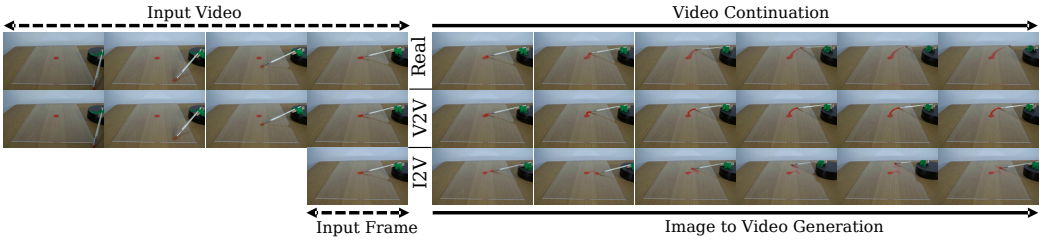


(f) The curtains reopen by the man, then he turns and walks away.

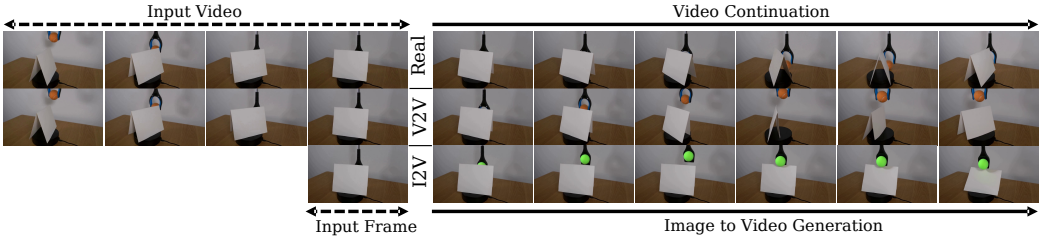


(g) As he departs, a roaring lion logo slowly fades into view on the screen.

Figure 10: This figure presents a near 30-second video generation example that demonstrates the capability of our model for complex actions and narrative structures through chunk-wise controllability and long-video generation. The sequence progresses from (a) to (g), with each sub-caption corresponding to the text prompt used during generation.



(a) Text Guidance: A clear acrylic sheet placed on a wooden table with a small dollop of red paint. A rotating paintbrush attached to a rotating platform rotates clockwise and goes through the paint. Static shot with no camera movement.



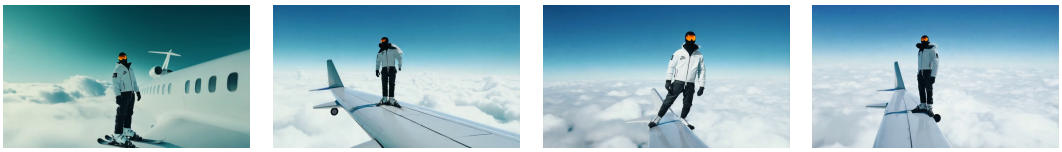
(b) Text Guidance: A grabber arm is holding a tennis ball above a piece of cardstock propped up on a rotating platform sitting on a table that rotates clockwise. The grabber lowers the ball and places is on the table as the cardstock rotates. Static shot with no camera movement.

Figure 11: Comparison between video-conditioned (V2V) and image-conditioned (I2V) video continuation. (a) MAGI-1 (V2V) accurately captures the pen's rotational trajectory by leveraging historical motion information, while I2V fails to reproduce the correct motion due to the absence of temporal context. (b) In an occlusion scenario, V2V successfully predicts post-occlusion behavior by utilizing information before the occlusion, whereas I2V shows poor temporal consistency. Each example presents the real-world scene (top row), MAGI-1 (V2V) generation (middle row), and MAGI-1 (I2V) generation (bottom row).

motion trajectories between the generated continuation and the prefix video, leading to motion discontinuities or generating implausible predictions due to the loss of essential historical information. Fig. 11 shows such cases. In the pen rotation example, I2V fails to capture the correct rotational velocity because it lacks access to preceding motion dynamics. Similarly, in the occlusion scenario, I2V cannot accurately predict the object's reappearance after occlusion due to missing temporal information. In contrast, conditioning on the full prefix video allows MAGI-1 to naturally preserve motion continuity by leveraging historical patterns and temporal dependencies, enabling seamless video continuation.

Controllable Shot Transition Another exciting feature of MAGI-1 is its ability to enable diverse and controllable transitions at any designated chunk by adjusting the KV range across different denoising stages. Specifically, by setting the KV range to 1 only at high-noise denoising stages (meaning the model cannot access the preceding video content) while keeping a normal KV range (*e.g.*, 8) at other stages, we can achieve shot transitions while preserve object identities unchanged, as shown in Fig. 12a. Conversely, by setting the KV range to 1 only at low-noise stages, we can produce transitions where the overall layout of the scene remains consistent, but the fine details of the objects change, as illustrated in Fig. 12b.

We believe the above capabilities can offer an entirely new level of creative control for video content creation.



(a) Shot transition with preserved identity.



(b) Transition with consistent scene layout but changing object details.

Figure 12: This figure illustrates two examples of realizing distinct shot transitions by modulating the KV range at different denoising stages. (a) demonstrates a case where the KV range is set to 1 only at the high-noise denoising stages, whereas (b) applies it at the low-noise denoising stages.

3 DATA

Training a high-performance video generation model demands massive, high-quality, and diverse data. To this end, we have developed a scalable data processing system that constructs the training dataset for MAGI-1 from tens of petabytes of raw videos and images collected from a wide range of sources.

An overview of the data processing pipeline is shown in Fig. 13. We utilize PySceneDetect[5] to cut long videos into short clips, ensuring that each clip contains only a single shot. Next, we apply a series of filters to remove low-quality data and eliminate duplicates. While this initial filtering stage effectively discards most of the low-quality data, some problematic cases still persist. To further improve data quality, we incorporate a multi-modal large language model (MLLM) as a stronger filter. Data that passes this filter is then captioned by the MLLM to provide accurate and detailed descriptions.

Through this process, we curate training data with satisfactory visual and motion quality. However, the distribution of the data — particularly in terms of semantic concept — still requires consideration. Specifically, we observed that the modeling difficulty varies significantly across different concepts. To address this, we use a dynamic distribution adjustment strategy based on evaluation results obtained during training. Additionally, we tailor the data distribution to accommodate the multi-stage training strategy.

In the sections that follow, we provide a detailed description of each component in our data processing pipeline.

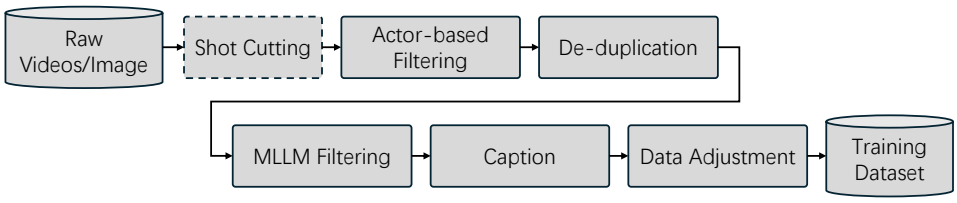


Figure 13: Overview of the our data processing pipeline. The shot cutting module is only applied for video data.

[5]PySceneDetect: https://github.com/Breakthrough/PySceneDetect

3.1 Filter Actors

We have developed a set of filtering actors to ensure the quality of the training data. These actors are described below in details:

Video Quality Assessment We adopt DOVER (Wu et al., 2023) to assess the visual quality of each video clip. DOVER provides three distinct quality scores: overall score, aesthetic score, and technical score. Through empirical evaluation, we found that the technical score alone is the most effective indicator for our use case.

Aesthetics We employ the LAION aesthetic model (Schuhmann et al., 2022) to predict aesthetic score for each image and video. Since the LAION aesthetic model is originally designed for images, we use the aesthetic score of the first frame to represent the quality of the entire video clip.

Overexposed and Underexposed Some videos suffer from overexposure or underexposure, which we have found to adversely affect training stability. To remove such data, we convert every frame of the video to the HSI color space and compute the average brightness across the entire video. Videos identified as either overexposed or underexposed, based on their average brightness, are excluded from the training set.

Motion Strength To quantify the motion strength of each video, we employ the RAFT optical flow model (Teed & Deng, 2020). To reduce computational overhead, all videos are first downsampled to 8 FPS before computing the optical flow between adjacent frames. The optical flow is calculated at the pixel level, and the overall motion strength is obtained by averaging the flow magnitudes across all pixels in the clip.

However, this approach tends to underestimate motion in cases where the background remains static while the foreground exhibits significant movement. To mitigate this issue, we additionally apply a saliency detection model (Zhao & Wu, 2019) to each frame. The resulting saliency maps enable us to distinguish between foreground and background regions, allowing us to compute the average optical flow separately for both.

As a result, we derive three motion statistics: overall motion strength, foreground motion strength, and background motion strength. To balance data quality and training difficulty, we prioritize video clips with moderate motion strength, avoiding both overly static and excessively dynamic videos. Specifically, we define lower and upper thresholds for all three motion statistics to guide data selection.

Camera Movement Stability A significant portion of collected videos is captured with handheld devices, which often results in erratic camera movements that are challenging for the model to learn. Since such cases are not effectively filtered by motion strength alone, we estimate camera stability by evaluating the consistency of optical flow between adjacent frames, filtering out clips with unstable camera motion.

Slides Movement Slide movements, such as floating photos or banners commonly found in screen recordings or slideshow presentations, are another undesirable case. To detect these, we analyze the divergence of the optical flow across all pixels in each frame. If the divergence remains consistently low over time, the clip is identified as containing slide movements and is removed.

Border Detection We perform edge detection on each frame and apply the Hough transform to identify persistent vertical and horizontal lines across frames. These lines are treated as potential borders, and the proportion of frames containing such borders serves as a confidence score for filtering.

Text Detection We perform text detection on video frames to identify and exclude clips containing excessive textual content. Specifically, if any frame within a clip contains an

overly large number of characters or if the detected text regions occupy a substantial portion of the frame, the corresponding clip is discarded.

A notable exception is subtitles, which typically consist of fewer characters and occupy relatively limited spatial regions, rendering them less likely to be filtered out by the aforementioned criteria. Nevertheless, subtitles exhibit distinctive spatiotemporal patterns: they consistently appear in fixed locations where most commonly at the top or bottom of the frame, and persist across multiple consecutive frames. By leveraging these characteristics, we are able to reliably detect and exclude video clips containing subtitles from the training data.

Logo Detection Many videos contain logos in the corners, which is an undesirable pattern for model training. To address this, we employ the Florence-2 model (Xiao et al., 2024), which supports open-vocabulary object detection. By providing a predefined set of keywords, Florence-2 accurately detects and localizes logos within video frames and providing confidence scores for filtering.

Corner Face Detection In commentary videos, narrators typically appear in a fixed corner of the screen, and we aim to exclude such patterns from our training data. To achieve this, we employ a face detection model, leveraging both face location and detection confidence to identify potential narrators. Specifically, we average the detection confidence of faces located in fixed corners across all frames to estimate the likelihood of a narrator's presence.

Transition Detection While PySceneDetect can segment raw videos into clips based on shot boundaries, it struggles to handle complex transitions, and as a result, the resulting clips may still contain multiple shots. To address this issue, we sparsely sample keyframes from each video and use CLIP (Radford et al., 2021) to compute the semantic similarity between adjacent keyframes. If the similarity falls below a predefined threshold, the clip is considered to contain multiple shots and is subsequently removed.

3.2 De-duplication

Recent studies on large language models (Lee et al., 2021; Hernandez et al., 2022) have shown that even small amounts of duplicate data can significantly degrade performance. Motivated by this, we conduct rigorous de-duplication. We compute pairwise similarity scores using both CLIP (Radford et al., 2021) and DINOv2 (Oquab et al., 2023), and treat any clip exceeding the threshold in either similarity as a duplicate to be removed.

3.3 MLLM as Advanced Filter

After the above filtering and de-duplication processes, most of the undesired data have been effectively removed. However, due to the limitations of the current filtering actors, a small portion of low-quality data still remains. As the remaining data size has been significantly reduced and to further improve data quality, we leverage a multi-modal large language model (MLLM) to perform an additional round of filtering. This enables us to detect more complex bad cases. Notably, this step can be seamlessly integrated into the subsequent caption procedure, thereby reducing overall costs and improving efficiency.

3.4 Caption

Highly Descriptive Caption Recent advances (Betker et al., 2023) have demonstrated that using MLLMs to generate highly descriptive captions is crucial for improving image generation quality, and we adopt this approach for captioning our data. Compared to images, videos have richer temporal information, including actions, camera movements, and scene changes. However, most mainstream MLLMs are primarily designed for images. To address this, we process each video by extracting a set of key-frames to form an image sequence. Through empirical analysis, we find that using 4 to 12 frames per video clip (depending on its duration) reaches the best trade-off between descriptive accuracy and

Attribute	Instruction
Scene Count	Identify the number of distinct scenes in the video.
Camera Transitions	Note any noticeable transitions between shots.
Camera Shot Type	Specify the type of camera shot used.
Camera Movement	Describe any camera movements.
Main Subject Identification	Determine who or what is the central focus of the video.
Subject Attributes	Describe the main subject's appearance.
Subject Position	Indicate where the main subject is within the frame.
Subject Action	Explain what the main subject is doing.

Table 3: Predefined attributes used in caption instruction.

computational efficiency. For video data, the captioning prompt is structured into two stages. In the first stage, the model is guided through a series of targeted questions aimed at eliciting responses on predefined attributes of the video clip (as summarized in Tab. 3). This step encourages the model to perform a structured analysis of the content. In the second stage, the model generates the final descriptive caption, which can incorporate salient observations identified in the preceding analysis of first stage. In contrast, for image data, we directly prompt the model to generate a caption without the attribute-based pre-analysis. Example captions are provided in Tab. 4.

Auto-Regressive Caption Unlike typical bi-directional denoising video generation models that produce an entire video as a whole, our model generates videos in an auto-regressive manner. This design allows our model to condition different parts of the video on distinct text prompts, offering greater controllability. To enable this capability, we provide fine-grained, second-by-second descriptions for each video clip. Tab. 4 shows example. Specifically, the caption of the first second is instruct to generates a detailed description. For caption of subsequent seconds, they focus on describing changes relative to the previous one.

3.5 Data Adjustment in Training

We have two different data adjustment scenarios during training. First, we use a multi-stage training strategy, with later stages having higher data quality; Second, we dynamically adjust the data distribution during training based on the evaluation results.

Multi-stage Adjustment MAGI-1 is trained in three stages, with the data resolution gradually increasing from 256p to 480p and ultimately to 720p. Alongside the resolution improvements, the data volume is progressively reduced, and more rigorous filtering strategies are employed to ensure higher data quality. Furthermore, in the final stage, the video duration is extended from a maximum of 8 seconds to a maximum of 16 seconds, allowing the model to capture richer temporal dynamics. The data specifications for each stage are summarized in Tab. 5.

Dynamic Distribution Adjustment An appropriate data distribution is crucial for training high-performance models. However, identifying the optimal distribution in advance is challenging. For instance, during training, we observed that landscape scenes are relatively easy for the model to learn, while human expressions are significantly more difficult. These insights are hard to predict beforehand. To address this, we adopt a dynamic distribution adjustment strategy. By continuously monitoring model performance throughout the training process, we can adaptively adjust the proportion of specific data subsets to strengthen the underperforming aspects of the model, thereby enabling a more effective learning process.

4 Infrastructure

In this section, we introduce our training infrastructure and inference infrastructure.

Caption Type	Example
Video Detail Caption	Medium shot of a hotel reception desk with two staff members. A woman stands on the left, and a man in a suit and red tie stands on the right. White orchids are in vases on either side of the desk. A painting hangs on the wall behind the desk. The man on the right picks up a telephone receiver and begins a phone conversation. The man is now more prominently featured in the frame, his upper body taking up a larger portion of the screen. The woman on the left is still visible, but less prominent. The man continues his phone conversation, his expression becoming more serious. The new arrival is now standing at the reception desk, slightly behind the man on the phone. The woman is still visible on the left. A man in a dark suit approaches the reception desk from the right side of the frame.
Image Detail Caption	A young woman with long dark hair stands on a rocky beach. She is wearing a light beige, strapless top and matching wide-legged pants. Her arms are crossed, and her hands are near her chest. She is barefoot. The rocks are various shades of brown and tan, some smooth and some rough. The rocks are wet in places. The ocean is visible in the background. The sky is light blue and mostly clear. A small child is partially visible in the lower left corner of the frame, seemingly playing near the water's edge. The woman is positioned in the center of the frame, slightly off-center towards the right. She is facing the camera directly. The rocks behind her are large and form a backdrop to her figure. The rocks in the foreground are smaller and scattered around her feet. The child is in the lower left corner, facing towards the center of the frame, and is partially obscured by rocks. The ocean is in the far background, above the rocks, and extends across the entire width of the frame. The sky is visible above the rocks and the ocean, occupying the upper portion of the frame. The lighting is natural, with sunlight illuminating the scene. The overall composition is balanced, with the woman as the focal point, surrounded by the natural elements of the beach.
AR Caption	**1st second:** A woman holds a lipstick tube, her expression changes subtly. The background is a simple, light brown wooden wall. The woman in the frames is wearing a beige lace sleeveless top and gold necklaces. She holds a gold lipstick tube in her right hand. Her makeup is subtle, and her expression changes slightly throughout the two frames. Her hair is dark brown and styled in a shoulder-length cut. The lighting is soft and even, creating a neutral mood. There are no other objects or people visible in the frames. **2nd second:** The woman's head tilts slightly, her expression shifts from a neutral to a slight smile. The lipstick remains in her hand. The camera remains static, focusing on the woman. **3rd second:** The woman's head is slightly turned to the left, her expression is more serious. The lipstick is still in her hand. The camera remains static, focusing on the woman. **4th second:** The woman's head is turned slightly to the right, her expression is neutral. The lipstick is still in her hand. The camera remains static, focusing on the woman.

Table 4: Caption examples used in MAGI-1.

	stage-1	stage-2	stage-3
Resolutions	256p/360p	480p	720p
Video Duration	$\leq$ 8s	$\leq$ 8s	$\leq$ 16s
Image-Video Ratio	4:1	4:1	4:1
AR Caption Ratio	0%	10%	10%

Table 5: Data configuration of different stages.

4.1 Training Infrastructure

Efficient training of large-scale autoregressive denoising models like MAGI-1 requires carefully tailored distributed training infrastructure. Existing distributed training frameworks, such as Megatron (Shoeybi et al., 2020) and DeepSpeed (Rajbhandari et al., 2020) are primarily designed for large language models (LLMs). However, MAGI-1 differs significantly from LLMs in both algorithmic side and data side.

On the algorithmic side, MAGI-1 integrates both autoregressive and denoising modeling paradigms, resulting in a model architecture that is notably more complex than that of typical LLMs. It incorporates components such as gating, cross-attention, and block-causal attention that are rarely used in language models.

On the data side, a single video training example typically contains tens to hundreds of times more tokens than a text example. Furthermore, ensuring the temporal and semantic integrity of video content imposes strict constraints, making it infeasible to directly apply common data processing strategies from LLMs, such as arbitrary sequence truncation or concatenation of multiple samples into a single training sequence offline, in the context of video generation.

These fundamental differences introduce unique challenges, necessitating a new, purpose-built distributed system design. In this section, we propose novel solutions to address these challenges to enable efficient and scalable training of MAGI-1.

Specifically, the training of MAGI-1 leverages a combination of data parallelism (DP), context parallelism (CP), and tensor parallelism (TP). To address the DP load imbalance caused by variable-length video sequence and the insufficient GPU utilization on short token sequences, we introduce a distributed Packing and Padding (PnP) during training, that performs online batching of video data in each training iteration. This strategy mitigates GPU bubbles thereby significantly improving overall training efficiency (Sec. 4.1.1).

Due to the use of PnP and the inherent demands of block-causal attention in MAGI-1, we require an efficient attention implementation capable of supporting highly flexible attention masks. Additionally, given the extremely long token sequences typical in video training data, native support for context parallelism is essential. To address these requirements, we propose MagiAttention: a scalable distributed attention mechanism that efficiently handles diverse attention masks and is optimized for ultra-long sequences (Sec. 4.1.2).

Through the above innovations, we enable the efficient training of MAGI-1. However, while developing the MAGI-1 training system, we identified several limitations in existing large-scale training frameworks. For instance, most current frameworks (including ours) do not treat verifiable numerical accuracy in distributed environments as a first-class design concern. Moreover, the tight coupling between algorithm development and infrastructure implementation often creates friction between algorithm researchers and infrastructure engineers, hindering efficient collaboration. To address these challenges, we discuss potential directions and design principles for next-generation training infrastructure in Sec. 4.1.3, with the goal of providing insights and practical guidance for the broader research and engineering community.

4.1.1 *Distributed Packing and Padding*

Due to the integrity constraints of video data and the variability in video lengths and resolutions, we adopt a Packing and Padding (PnP) strategy (Sirluk, 2024; Kundu et al., 2024) to batch video samples in a way that minimizes excessive padding and reduces unnecessary computational overhead in distributed training scenarios. Moreover, the data composition is frequently adjusted during the training of MAGI-1 (See Sec. 3.5), and to accommodate such flexibility, we employ an online PnP strategy instead of a offline approach.

The core idea of PnP is to efficiently utilize GPU resources by concatenating multiple short sequences into a batch while minimizing redundant filling. The offline formulation of this problem aligns with the classic bin-packing problem: given a set of input samples, the goal is to pack them into a set of bins, each with a fixed capacity `max_length`, while minimizing overall unused space. Although this problem is NP-complete, it can be efficiently approximated in practice using the First-Fit Decreasing (FFD) (Dósa, 2007) greedy algorithm.

In our online setting, we must process streaming data inputs while ensuring compatibility with the 3D parallelism strategy employed during training. To this end, we reformulate the problem as follows: given M candidate samples, we aim to pack them into N bins of size `max_length`, minimizing overall space waste. Here, M denotes the size of the candidate pool with $M \gg N$; N must be divisible by the `DP_SIZE`; and `max_length` must be divisible by `TP_SIZE`$\times$`CP_SIZE`.

In practice, we extend the FFD algorithm with custom heuristics to support efficient online packing under these constraints. This approach enables us to achieve a 99% capacity utilization rate and the differences between different DP groups can be neglected, thus substantially reducing computational overhead during training.

4.1.2 *MagiAttention: Towards Linear Scalability for Ultra-Long and Heterogeneous Mask Training.*

Training large-scale autoregressive diffusion models like MAGI-1 for video generation presents two major challenges:

- The extremely long context length of video tokens, which reaching up to 4 million during training, results in prohibitive computational and memory overhead. Context-Parallelism (CP) is designed for dealing such long context challenge, but existing state-of-the-art CP methods (Jacobs et al., 2023; Liu et al., 2023; Fang & Zhao, 2024; Gu et al., 2024; Chen et al., 2024b) face scalability limitations that face scalability limitations due to size constraints or the high communication overhead inherent in inefficient ring-style point-to-point (P2P) patterns. While recent efforts (Wang et al., 2024; Zhang et al., 2024; Ge et al., 2025) dynamically adjust CP sizes to avoid unnecessary sharding and redundant communication for shorter sequences, they still incur extra memory overhead for multiple NCCL process groups and involve complex scheduling to balance loads and synchronize across different subsets of ranks.
- The combination of block-causal attention and Packing-and-Padding introduces highly complex attention mask patterns (Sec.4.1.1), which cannot be efficiently handled by existing attention implementations.

To address the aforementioned challenges, we propose MagiAttention, which aims to support a wide variety of attention mask types (*i.e.*, kernel flexibility) while achieving linear scalability with respect to context-parallel (CP) size across a broad range of scenarios. Achieving this goal depends on meeting the following fundamental conditions:

- *Linearly Scalable Attention Kernel*: The performance of the attention kernel should not degradate as CP size increases. To this end, we introduce *Flex-Flash-Attention*, an extension of FlashAttention-3 (FA3), which native considers the efficiency impact of attention mask partitioning in distributed environments. It supports distributable mask representations with a tailored kernel implementation to ensure scalability while accommodating a broader range of attention mask types.

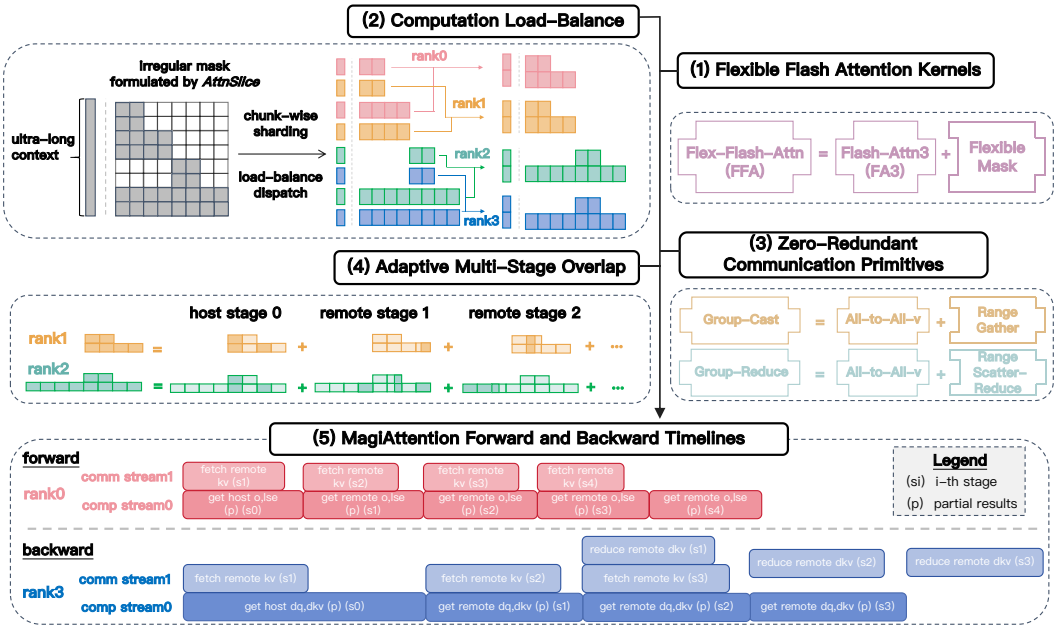


Figure 14: Overview of MagiAttention: (1) Flex-Flash-Attention(FFA), an efficient attention supports flexible mask patterns and native considers distribution requirements; (2) The *dispatch solver* shards and dispatches packed data with ultra-long contexts and heterogeneous masks, ensuring load-balanced computation; (3) Group-Cast and Group-Reduce primitives eliminate redundant communication; (4) The *adaptive multi-stage overlap* strategy effectively hides communication latency; (5) Forward and backward timelines of MagiAttention. With all techniques together, MagiAttention reach linear scalability under diverse scenarios.

- *Balanced Computational Workloads*: Imbalances in the computational load across CP ranks lead to unavoidable idle bubbles that hinder scalability. MagiAttention is natively designed to ensure *Computation Load Balancing*, mitigating such inefficiencies.
- *Full Overlap of Communication and Computation*: Without sufficient overlap, increasing CP size results in communication-induced idle time on GPUs, impairing scalability. MagiAttention introduces novel *Zero-Redundant Communication Primitives* to minimize communication overhead, along with an *Adaptive Multi-Stage Overlap* strategy that enables effective communication-computation overlap.

The overview of MagiAttention is shown in Fig. 14, and we will introduce key designs in the following, with comprehensive experimental results presented in Appendix B.2.

Flex-Flash-Attention. FlashAttention (Dao et al., 2022; Dao, 2023; Shah et al., 2024) is a foundational technique in large-scale model training for its superior performance and support for varlen-packed data with causal attention masks. However, it offers limited support for irregular attention masks, particularly when such patterns are distributed across CP ranks, resulting in increased complexity and underscoring the need for a more flexible attention kernel (PyTorch; Dong et al., 2024; Wang et al., 2025b) without compromising performance.

Therefore, we introduce Flex-Flash-Attention (FFA), which is natively designed for distribution scenarios and provides greater flexibility in handling diverse attention mask types. The core idea behind FFA is to generalize a distributable formulation for irregular attention masks by decomposing the entire mask into multiple computational units, each referred to as an `AttnSlice`. Each `AttnSlice` is defined by a triplet $QRange, KRange, MaskType$, which specifies a submask with a basic shape bounded by a contiguous 2D query-key region (see Fig. 20). Using this formulation, a wide variety of commonly used attention masks (Fig. 15)

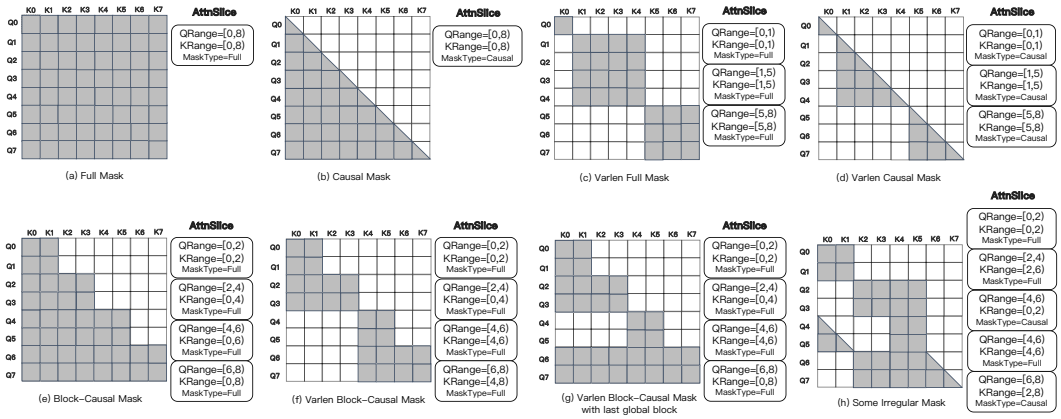


Figure 15: Examples of mask patterns formulated by `AttnSlice`. (a)-(d) Standard FA3-compatible patterns; (e)-(h) Irregular masks beyond FA3's capabilities, including our novel *varlen block-causal* design, which FFA supports seamlessly while maintaining performance comparable to FA3.

(including our varlen block-causal mask) can be expressed as a composition of multiple such triplets, making FFA highly suitable for distributed attention computation.

Built on FA3 kernels, Flex-Flash-Attention leverages NVIDIA Hopper GPUs' TMA feature (NVIDIA, 2024) and introduces slice-level parallelism with atomic operations for correctness (Fig 21), achieving comparable MFU to FA3 while supporting the flexible `AttnSlice` formulation [6] (see Appendix B.2 for benchmarks).

Computation Load-Balance. In context-parallelism (CP) settings, different CP ranks may be assigned heterogeneous attention masks, resulting in imbalanced computational workloads across ranks. Ring-Attention (zhuzilin, 2024) employs a specialized partitioning strategy designed specifically for causal attention, which limits its applicability to more general attention patterns. To overcome this limitation, we propose a generic and efficient `dispatch solver` that enables balanced workload distribution across CP ranks for a broad range of attention types.

First, to enable finer-grained control, we propose a chunk-wise permutable sharding strategy (Fig 14 (2)). Specifically, the entire mask is evenly partitioned along the query-dimension into dispatch chunks, each associated with a submask area: $\{(C_i, \text{Area}(C_i))\}_{i=1}^{n}$, where C_i indicates i-th dispatch chunk, $\text{Area}(C_i)$ is the mask area of C_i, n is $\frac{seqlen}{dispatch_chunk_size}$, and $dispatch_chunk_size$ is a hyperparameter controlling granularity. These dispatch chunks are then equally assigned to cp_size buckets, with each bucket containing the exact same number of dispatch chunks to ensure token-level load balance in non-attention modules, attaching with a summed submask area, denoted as $\{(B_j, \text{SumArea}(B_j))\}_{j=1}^{cp_size}$.

With above strategy, we could fine-grained control the computational workloads of each CP rank, and the load-balancing dispatch becomes a combinatorial optimization problem, defined as finding an optimal mapping function $f^* : \{C_i\}_{i=1}^{n} \to \{B_j\}_{j=1}^{cp_size}$ as follows

$$f^* = \arg\min_{f} \max_{j} \left\{\text{SumArea}(B_j)\right\} \tag{11}$$
$$\text{s.t. } |B_j| = \frac{n}{cp_size}, \;\; seqlen \;\%\; (cp_size \times dispatch_chunk_size) = 0$$

[6] Redundant computation from padding tokens is excluded by easily passing empty `QRange` or `KRange`.

However, this optimization is a known NP-hard problem, making it impractical to find an optimal solution on-the-fly during each training iteration, especially given the varying mask patterns across micro-batches. Thus, we propose an efficient greedy algorithm (as shown in Alg. 1) that provides a suboptimal yet effective solution within $O(n \log n)$ complexity.

Zero-Redundant Communication Primitives. The existing ring-style implementation uses point-to-point `send/recv` communication primitives, which cannot provide sufficient communication granularity, resulting in redundant communication. Take causal mask as an example, we analyze the redundant communication by recording the distribution of remote key-value (KV) requests and their gradients (dKV) under sparse attention masks. As shown in Fig 23, KV_0 is required by all queries and should be sent to all devices via Broad-Cast in the forward pass, with dKV_0 reduced via All-Reduce in the backward pass. In contrast, KV_7 is only needed by its host device but still circulates through all devices, and this redundancy intensifies in varlen scenarios.

To address this, we introduce two communication primitives: `group-cast` and `group-reduce`, which model the communication patterns of low-demand KV and dKV (Fig 24). For example, in the causal mask, KV_5 on rank_2 is required only by $\{\text{Q}_6, \text{Q}_7\}$ and should be sent exclusively to the target ranks $\{\text{rank}_0, \text{rank}_1\}$ via `group-cast`, while the partial dKV_5 is collected and reduced back to rank_2 via `group-reduce` accordingly.

As no existing communication kernels support these primitives, we prototype them using `all-to-all-v` (Fig 24), achieving zero-redundant communication in both forward and backward passes. However, this approach introduces extra pre-/post-processing overhead, similar to (un)permutation in expert parallelism (EP) (Gale et al., 2022). While kernel fusion mitigates the overhead, a dedicated implementation of `group-cast` and `group-reduce` remains a key direction for future work.

Adaptive Multi-Stage Overlap. Leveraging previous optimizations, we achieve high-performance computation through an efficient kernel and balanced workload dispatch, while minimizing communication overhead with our new primitives. To drive true linear scalability, we further improve end-to-end performance by introducing a multi-stage compute-communication overlap strategy, that effectively hides communication latency and adaptively optimizes overlap through manual or automatic tuning.

Similar to prior works (Liu et al., 2023; Zhao et al., 2023; He et al., 2024), we schedule pipeline stages to overlap computation with communication for both forward and backward passes (Fig 25). Each rank_i first partitions its remote KV/dKV communication into stages. In the forward pass, the scheduler first launches the `group-cast` kernel to prefetch the next remote KV, then asynchronously executes the FFA kernel for partial attention computation, hiding all communication behind computation [7]. In the backward pass, besides prefetching the next KV, the `group-reduce` kernel reduces the last dKV in a separate CUDA stream before launching the FFA kernel for the current stage, ensuring communication is overlapped across all stages except the final dKV reduction [8].

To adaptively control overlap granularity, we further introduce a tunable hyperparameter, *num_stages*, accounting for varying compute-to-communication ratios across training setups, microbatches, or between forward and backward passes. This parameter can be manually configured or automatically determined by our *overlap solver*, with a simple dynamic search algorithm (See Alg. 2 for more details).

[7] To prevent all SMs from being occupied by the attention kernel, we ensure the communication kernel picked first by setting `CUDA_DEVICE_MAX_CONNECTIONS=1` (User, 2023).

[8] Due to PyTorch's one-to-one mapping for process groups and collective communication streams including `all-to-all-v` (User, 2024), we internally use an additional CP group for `group-reduce` to enable full overlap between communication kernels in the backward pass.

4.1.3 *Rethinking System Design for Robust Distributed Training Frameworks with DTensor*

As large-scale models continue to evolve, the growing complexity of training procedures has exposed fundamental limitations in existing distributed training frameworks (Shoeybi et al., 2020; Rajbhandari et al., 2020). Two major bottlenecks are particularly prominent:

- Lack of testability by design. Most frameworks were not initially built with testability as a first-class feature, resulting in fragile infrastructure with limited maintainability and reliability;
- Tight coupling between model implementation and parallelization strategy. This entanglement prevents algorithm researchers and system engineers from working independently, hindering collaboration and modular development

We argue that next-generation distributed training frameworks must directly address these two pain points to support large-scale model research and deployment.

Inspired by early explorations (Xu et al., 2021; Yuan et al., 2022) and PyTorch's pioneering implementations (Zhao et al., 2023; Team, 2024b; Liang et al., 2024), we propose a blueprint for redesigning robust distributed training frameworks based on Pytorch Distributed Tensor (DTensor) (Team, 2024b) and Parallel Plan:

DTensor PyTorch DTensor introduces three parallel placements: `Replicated`, `Shard`, and `Partial`, alongside a distributed initialization strategy to maintain placement semantics (Contributors, 2025a), and a propagation mechanism that deduces output placements from input ones for supported ops, triggering communication as needed [9] (Contributors, 2025b). While it supports basic ops including naive distributed matmul, its current implementations lack the generality to handle more complex yet commonly scenarios in modern training workflows, as shown in Tab. 12.

Parallel Plan Parallel Plan provides a declarative interface for specifying parallelization strategies across model submodules. It works in conjunction with the `parallelize_module` function and is built on top of DTensor. However, its current capabilities are mostly limited to tensor parallelism (TP) and do not generalize well to other parallelism.

In our architecture design, we extend both DTensor and Parallel Plan to support a broader range of usages. These extensions enable the following key features:

Decoupling Modeling from Parallelization. This feature allows model researchers to concentrate on model design and algorithm development without needing to manage low-level parallelism details. At the same time, infrastructure engineers can independently optimize parallelization strategies without modifying model implementation. This clear separation of concerns enables more efficient collaboration and improved training throughput.

High-Precision Alignment with Non-Distributed Oracles. By disabling all parallel plans, we can seamlessly revert to non-distributed configurations, yielding "pure" model code that serves as a baseline or oracle for evaluating distributed correctness. To ensure alignment within a relative error of 10^{-8}, we upcast tensors to higher precision [10], enforce deterministic algorithms (Team, 2024a), and control randomness using consistent seed management. This design enables precise infrastructure testing, ultimately improving reliability and debuggability.

[9] In practice, DTensor selects communication patterns based on estimated redistribution cost, but these estimates are often inaccurate.

[10] In our experiments, `float32` is insufficient for fully alignment; `float64` suffices in most cases.

4.2 Inference Infrastructure

As an innovative large-scale autoregressive denoising model, MAGI-1 introduces two pivotal architectural innovations: multi-chunk parallel inference and KV cache, which unlock new possibilities for user experiences, such as real-time streaming video generation, and enables cost-effective deployment. However, these advancements also introduce new challenges to the inference infrastructure. In this section, we present our infrastructure design tailored to two major scenarios: real-time streaming inference on H100/H800, and cost-efficient deployment on RTX 4090 GPU.

4.2.1 Real-Time Streaming Video Generation

Our model adopts an auto-regressive architecture that supports real-time streaming video generation. To ensure a seamless user experience, we optimize for two key latency metrics: Time to First Chunk (TTFC), which measures the delay between task submission and starting to see the video, and Time Per Output Chunk (TPOC), which reflects the time required to generate each subsequent chunk. Maintaining a low TTFC enhances responsiveness, while keeping TPOC below 1 second is essential for uninterrupted playback.

We encountered three major challenges when designing the infrastructure:

- MAGI-1 consists of multiple sub-models: T5 for text embedding extraction, a VAE encoder for processing user-uploaded images and prefix videos, a VAE decoder for decode the denoised output, and a core auto-regressive denoising model. These components exhibit distinct computational characteristics: T5 and VAE are memory-bound, while the denoising model is compute-bound. Efficiently handling this heterogeneity is essential.
- To meet the TPOC target of under 1 second, MAGI-1 demands approximately 9 PFLOPS of compute per second of video, which far exceeds the capabilities of a single H100/H800 GPU. Achieving this requires serving models on multiple H100/H800 GPUs and a highly optimized parallelism strategy.
- First-chunk inference differs significantly from subsequent chunks. It is not compute-bound but CPU-bound, due to limited token workloads per GPU, resulting in a long TTFC.

To address these challenges, we propose a systematically optimized framework, enabling real-time streaming video generation for our largest 24B MAGI-1 model on 3-node, 24 H100 GPUs. Here, we briefly introduce our solutions.

Multi-Model Heterogeneous Serving Pipeline We designed a heterogeneous serving architecture that co-locates T5 and MAGI-1 on high-performance GPUs, while deploying the VAE to cost-efficient hardware. This approach enables concurrent execution of MAGI-1 inference and VAE decoding, minimizing idle time and improving overall throughput. Profiling-driven resource allocation strategies further enhance utilization efficiency. With this design, we could efficiently handling the heterogeneity of different models and achieve the best performance.

TPOC Optimization Given that the denoising model of MAGI-1 is compute-bound, we prioritized aggressive quantization and distributed inference optimizations:

- *Quantization.* We adopted W8A8 SmoothQuant Xiao et al. (2023a) to quantize both weights and activations to FP8 precision, except the first and last layers. The quantization delivered a 30% speedup without compromising generation quality.
- *Multi-Node Parallel Inference.* We adopt a Ulysses-based multi-node parallel inference strategy with sufficiently computation and communication overlapping (less than 3% of communication time remaining unoverlapped in the execution timeline). As

a result, the TPOC is optimized to be within 1 second when we generating 480p (3:4 aspect ratio) videos using 16 denoising steps and KV range of 5 on 24 H100/H800 GPUs.

TTFC Optimization For first-chunk inference, only a few hundred tokens need to be processed. In this scenario, the GPU workload is relatively light, and CPU-side bottlenecks become the primary constraint. To address this issue, we employ CUDA Graphs to minimize kernel launch overhead, reducing 30.4% latency. Additionally, we accelerate VAE decoding through a tile-based parallel mechanism and `torch.compile`, bringing latency down from 1 second to around 70 milliseconds. Collectively, these optimizations reduced TTFC to 2.3 seconds, ensuring a smooth real-time streaming experience. Tab. 6 summarizes the key optimizations and their corresponding latency gains[11].

Model	Optimization	TTFC(s)	Gain	TPOC(s)	Gain
Autoregressive Denoising Model	Baseline	73.34	-	45.49	-
	KV Cache	73.34	-	23.94	1.90X
	Ulysses	3.86	18.0X	1.26	18.0X
	Smooth Quant	3.00	1.29X	0.98	1.29X
	Cuda Graph	2.30	1.30X	0.98	-
Vae Decoder	Baseline	1.00	-	1.00	-
	Tile Parallel	0.20	5.00X	0.20	5.00X
	torch.compile	0.07	2.86X	0.07	2.86X
End-to-End	-	2.37	-	0.98	-

Table 6: Inference Optimization and Latency Gain

4.2.2 Cost-effective Inference on RTX 4090

The NVIDIA GeForce RTX 4090 is a highly cost-effective GPU with 24G memory. However, through in-depth memory profiling and analysis, we identified memory insufficiency as the primary bottleneck to serve our model on it. To address this challenge, we developed a highly memory-efficient inference architecture and performs systematically optimizations. As a result, we successfully deployed and ran our 4.5B-parameter model on a single RTX 4090 GPU, and also support our largest 24B model on an 8×RTX 4090 GPUs. In the following section, we briefly introduce the key optimization techniques.

Memory Optimization To address the memory constraints of the RTX 4090, we used a variety of techniques to do systematically optimization:

- *Quantization:* We adopt the same quantization strategy (WA8A SmoothQuant) as for streaming video generation.
- *KV-offload:* KV-offload is a technique that stores the KV cache in CPU memory by default and dynamically re-load it back to the GPU as needed. This approach significantly reduces peak GPU memory usage and is widely adopted in long-sequence processing for large language models (LLMs). In MAGI-1, we also adopt this technique to effectively address memory constraints.
- *Hybrid Parallelism and Communication Optimization:* The above two optimizations only enable 4.5B model deployment on a single RTX 4090 GPU. However, the largest 24B model further requires multi-GPU parallelism. Unlike the streaming setting where we primarily adopt a Ulysses-based context-parallelism (CP) approach, deployment

[11]While TPOC latency is expected to be the sum of autoregressive diffusion model and VAE decoder latencies—similar to TTFC—in our serving pipeline where autoregressive diffusion model and VAE run on separate machines, TPOC is instead computed as the maximum of the two.

on RTX 4090 employs a hybrid strategy combining pipeline-parallelism (PP) and context-parallelism.

Specifically, pipeline-parallelism is used to partition model weights, while context-parallelism is used to partition activations. However, since the RTX 4090 utilizes PCIe for inter-GPU communication, both PP and CP suffer from communication-induced bubbles that degrade compute utilization, as measured by Model FLOPs Utilization (MFU). For PP, we mitigate this by interleaving tasks to overlap GPU idle. For context-parallelism, we initially adopted the Ulysses approach, but found that communication could not be fully overlapped with computation under PCIe constraints.

Therefore, we propose an enhancement to Ulysses called Context Shuffle Overlap (CSO)(Details in Sec. A.3), which scatters each chunk evenly across all GPUs, enabling finer-grained overlap between computation and communication than plain Ulysses. This strategy significantly improves MFU under the limited interconnect bandwidth of the RTX 4090.

With the above optimizations, we constrained peak memory usage to 21.94 GB for the 4.5B model on a single RTX 4090 GPU, and 19.29 GB for the 24B model on 8×RTX 4090 GPUs. For the 24B model, the maximum MFU reached 58%.

5 Evaluation

Evaluation methods for video generation models in the research community are typically categorized into two complementary types: the first focuses on the perceptual quality of the generated videos, while the second evaluates the model's ability to faithfully capture underlying physics, which is often regarded as essential for modeling a realistic world. In MAGI-1, we adopt both evaluation types to obtain a comprehensive understanding of the model's strengths and limitations.

For perceptual quality evaluation, the inherently subjective nature of human preference, combined with the high-dimensional and diverse characteristics of video content (*e.g.*, motion continuity, aesthetic, and identity consistency), makes it challenging to rely solely on objective metrics. As a result, the community typically employs a hybrid evaluation protocol that integrates human subjective assessments with standardized automated metrics, ensuring a more robust and comprehensive evaluation.

There is currently no universally accepted human evaluation protocol or human evaluation platform within the community for perceptual quality evaluation. To address this, we design our own in-house evaluation benchmark based on a comprehensive review of existing human evaluation methodologies, combined with our understanding of both evaluation criteria and model capabilities. Human experts serve as evaluator in this system, comparing our model against other competitors under strict double-blind conditions, and providing assessments across multiple perceptual dimensions. For objective evaluation, we adopt VBench (Huang et al., 2024), which is currently the most widely used benchmark in the community. VBench consists of two evaluation tracks: text-to-video (T2V) and image-to-video (I2V). We primarily focus on the I2V track, as it more closely reflects real-world usage patterns: users typically generate videos from images rather than from text. For the same reason, we also allocate a larger proportion of I2V tasks during the training of MAGI-1, aiming to better align the model's capabilities with practical deployment scenarios.

Physics-IQ (Motamed et al., 2025) is one of the most representative benchmarks for evaluating a model's ability to capture physical dynamics in video. It presents a short video clip depicting real-world physical motion and asks the model to predict future frames. The predictions are then compared against ground-truth sequences to assess the model's understanding of physical rules.

The evaluation framework and the corresponding benchmark metrics are summarized in Tab. 7. The following sections present our evaluations in detail, and if not specified, we evaluate our 24B model by default.

Evaluation Category	Benchmark	Metrics
Perceptual Evaluation	In-house Human Evaluation	Overall Motion Quality Instruction Following Visual Quality
	VBench-I2V	Automated Quality Metrics
Physical Evaluation	Physics-IQ-Benchmark	Physics-IQ-Score

Table 7: Evaluation Benchmark Overview

5.1 Perceptual Evaluation

5.1.1 In-house Human Evaluation Benchmark

Our in-house evaluation benchmark is primarily designed for I2V task, and integrates three complementary components to ensure comprehensive and unbiased assessment. First, we design a hierarchical metric system that prioritizes completeness over simplicity, while enforcing orthogonality among metrics to enable fine-grained evaluation across multiple quality dimensions without redundancy. Second, we construct a benchmark dataset of 100 diverse image-prompt pairs through systematic selection. These pairs span a broad spectrum of scenarios, from simple object motions to complex human activities, and each curated to probe specific aspects of video generation capability. Third, we implement a double-blind comparison protocol with standardized output normalization, ensuring that each model operates under fair conditions for a meaningful comparison.

Evaluation Metrics. To ensure a comprehensive and reliable evaluation while avoiding unnecessary complexity, we adhere to three guiding principles in our metric design: comprehensiveness first, simplicity second, and orthogonality third. Unlike T2V, where both visual content and motion are generated from scratch, I2V starts with a fixed visual input provided by the user's uploaded image, while the subsequent dynamics are guided by the input text condition. This distinction shifts the evaluation focus toward assessing the motion and temporal quality of generated video while ensuring faithful preservation of the original visual elements.

Through preliminary analysis, we identified several common failure modes in I2V generation, including distortion, clipping, and temporal jittering. These typical issues guided the design of our evaluation framework, which emphasizes motion quality, temporal coherence, and the trade-off between source image fidelity and natural animation. Therefore, our evaluation framework organizes metrics into four primarily dimensions: *Overall*, *Motion Quality*, *Instruction Following*, and *Visual Quality*. Each dimension is further broken down into specific sub-metrics designed to capture particular aspects of video generation quality as shown in Tab. 8.

Dataset Construction. We construct a benchmark dataset consisting of 100 high-quality image-prompt pairs, each carefully selected to challenge different aspects of I2V generation. To ensure diversity and representativeness, we source data from four sources: 1) user-submitted inputs from existing video generation platforms, 2) synthetic images generated by FLUX (Labs, 2024), 3) authentic photographs from public repositories, and 4) professional cinematographic materials. Each sample is annotated with specific evaluation targets defined by our metric framework, enabling broad coverage of assessment dimensions while avoiding redundancy.

The dataset construction process follows a systematic multi-stage pipeline. We first establish a set of selection criteria focused on key challenges in I2V generation, including complex object deformation, multi-object interaction, dynamic camera motion, and lighting transitions. Based on these criteria, experts nominate candidate samples, which are then

Main Metric	Sub Metric	Description
Overall	-	General preference
Motion Quality	Motion Speed	Appropriate timing of movements
	Motion Amplitude	Natural range of movement
	Motion Smoothness	Continuous movement without jitter
	Movement Direction	Logical and consistent direction
Instruction Following	Subject Adherence	Following behavioral instructions
	Environment Adherence	Meeting contextual requirements
	Camera Adherence	Following camera movement requests
Visual Quality	Subject Features	Consistency of main subject
	Scene Features	Consistency of environment
	Lighting Changes	Quality of lighting transitions
	Texture Changes	Consistency of surface appearances

Table 8: Hierarchical Evaluation Framework

finalized through a collaborative voting procedure. This curated process ensures the resulting benchmark presents a diverse yet focused set of evaluation cases for rigorously testing I2V models.

Results and Analysis. Our evaluation methodology employs a paired comparison approach designed to directly measure relative model performance. Specifically, for each test case, we generate two videos (one from our model and one from a comparative model) using identical prompts and input images. Expert evaluators with strong aesthetic training then indicate their preference between each pair (Win/Tie/Lose) across multiple evaluation dimensions without knowledge of which model produced which video.

MAGI-1's autoregressive design enables generation of arbitrary-length videos. For fair comparison, we adapt our generation length to match each comparison model: for example, 5 seconds for Kling and 6 seconds for Hailuo. To avoid potential manipulation of visual quality, we maintain each model's native output without post-processing like resolution normalization. In addition, all models are evaluated using raw user inputs without any manual refinement from our side, relying solely on their built-in prompt enhancement (PE) mechanisms.

The evaluation results shown in Fig. 16 demonstrate MAGI-1's strong competitive position in the field. In terms of overall performance, our model shows advantages over the open-source model Wan-2.1 (Wang et al., 2025a), performs slightly behind the commercial model Kling1.6 (HD) (Kuaishou, 2024), but achieves clearly better results compared to both Hailuo(i2v-01) (MiniMax, 2024) and HunyuanVideo (Kong et al., 2024). Looking at specific capabilities, MAGI-1 excels particularly in instruction following and motion quality metrics, consistently receiving high scores across comparisons. However, in terms of visual quality, there remains room for improvement compared to top models.

5.1.2 VBench

VBench (Huang et al., 2024) is currently the most widely adopted benchmark in the community for automated and objective evaluation of video generation models. While its evaluation framework is still evolving and not without limitations, VBench remains a critical tool for model comparison due to its fully automated and reproducible assessment process, especially when contrasted with in-house human evaluations, which are often subjective and lack transparency.

VBench provides two primary evaluation tracks: text-to-video (T2V) and image-to-video (I2V). Given that I2V more closely reflects real-world usage patterns, where users typically input a static image to generate videos in existing product, and in line with our goal of

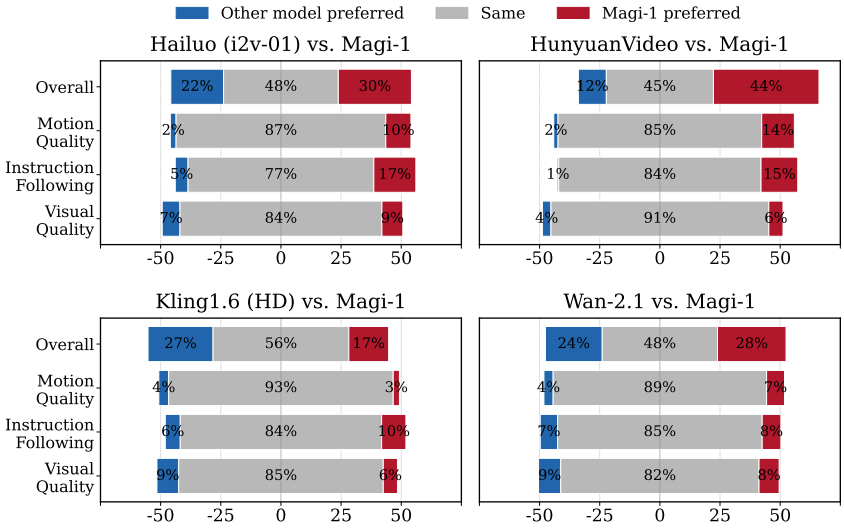


Figure 16: Comparative evaluation of our model against leading open-source and proprietary video generation models across multiple metrics. Each bar is divided into three sections: red, gray, and blue, representing Win-Tie-Loss percentages for each comparison. Blue sections indicate where users preferred the competitor model, gray sections represent ties, and red sections show where users preferred our model. The evaluation includes both API-based assessments like Kling1.6 (HD) (Kuaishou, 2024) and Hailuo (i2v-01) (MiniMax, 2024) and locally deployed models like Wan-2.1 (Wang et al., 2025a) and HunyuanVideo (Kong et al., 2024)), providing a comprehensive comparison across various implementation environments.

aligning evaluation with practical application scenarios, we focus our evaluation on the I2V track in VBench.

We evaluate the generation quality of MAGI-1 under two different configurations: MAGI-1 (1×decoder) and MAGI-1 (2×decoder). The only difference between them lies in the VAE decoder: MAGI-1 (2×decoder) employs an enhanced decoder capable of 2× upsampling, while the core autoregressive denoising model remains identical across both versions. For evaluation, both models generate 4-second videos at 24 FPS with a 16:9 aspect ratio.

The results are presented in Tab. 9. As shown, both of our models achieve outstanding performance, with MAGI-1 (2× decoder) reaching a top overall score of 89.28, ranking first among all models. Notably, the MAGI-1 models demonstrate a significant advantage in the *dynamic Degree* compared to other approaches, while simultaneously maintaining high visual quality, including strong performance in *aesthetic quality* and *motion smoothness*. This effectively addresses a common trade-off in other methods, where increasing motion amplitude often downgrade image quality. We attribute this strength to the autoregressive denoising architecture, which provides a stronger modeling capability for complex motion dynamics.

5.2 Physical Evaluation

Video generation models are increasingly recognized as a foundation toward building the world model, and the ability to accurately capture real-world physical dynamics has become a central focus within the research community. In contrast to perceptual evaluation, which inevitably involves subjective human preferences, physics-based evaluation aims to assess a model's ability to understand and simulate objective physical principles.

Currently, there are only a few established benchmarks (Bansal et al., 2024; Meng et al., 2024; Dash et al., 2011; Yi et al., 2019; Kang et al., 2024) in this area, and Physics-IQ (Motamed et al., 2025) stands out as the most comprehensive and state-of-the-art benchmark. Therefore,

	Metric (VBenchI2V)	MAGI-1 (2×decoder)	MAGI-1 (1×decoder)	VisualPi	StepFun (TI2V)
Quality Metrics	I2V-Camera	<u>50.85</u>	50.77	**51.20**	49.23
	I2V-Subject	<u>98.39</u>	98.36	**98.67**	97.86
	I2V-Background	**99.00**	<u>98.98</u>	98.87	98.63
	Subject Cons.	93.96	94.28	**96.87**	<u>96.02</u>
	Motion Smooth.	98.68	98.83	<u>99.18</u>	**99.24**
	Imaging Quality	69.71	69.68	**72.86**	<u>70.44</u>
	Dynamic Degree	**68.21**	<u>63.41</u>	49.93	48.78
	Background Cons.	96.74	96.90	**97.50**	<u>97.06</u>
	Aesthetic Quality	**64.74**	61.89	61.91	<u>62.29</u>
Agg. Scores	Quality Score	**82.44**	81.67	<u>81.95</u>	81.22
	I2V Score	<u>96.12</u>	96.08	**96.21**	95.50
	Total Score	**89.28**	88.88	<u>89.08</u>	88.36

Table 9: Quantitative evaluation results on VBench-I2V benchmark. MAGI-1 (1×decoder) denotes our baseline model (1280 × 720 resolution), while MAGI-1 (2×decoder) represents the enhanced variant with 2x VAE upsampling (2560 × 1440 resolution). Comparative data for other models are sourced from the top tier at latest Vbench leaderboard. Bold and underlined values indicate the highest and second-highest scores respectively across all metrics.

we adopt Physics-IQ to evaluate the physical understanding and reasoning capabilities of MAGI-1.

The Physics-IQ evaluation protocol uses 8-second real-world videos that depict objective physical phenomena. The first 3 seconds of each video are provided as conditional input to the model, which is then required to predict the remaining 5 seconds. The accuracy of the model's physical modeling capability is measured by comparing the predicted videos with the ground truth.

Since most existing video generation models do not natively support video-conditioned continuation, they typically approximate this task using image-to-video (I2V) generation, conditioning only on the last frame of the input video. To provide a comprehensive comparison, we report results for both two settings.

The results are presented in Tab. 10. When conditioned on video inputs, MAGI-1 outperforms all competing models by a substantial margin, reaches the score of 56.02. The previous state-of-the-art model VideoPoet (Kondratyuk et al., 2023), which also supports video-to-video (V2V) prediction, is outperformed by approximately 27 points. Even when using only image condition, MAGI-1 still achieves the highest score among all models, reaching 30.23, despite a noticeable drop compared to its video-conditioned version.

These results clearly demonstrate the strong capability of MAGI-1 in understanding and modeling real-world physical principles. We attribute this advantage to its autoregressive nature: modeling physical processes demands a focus on causality rather than mere correlation, and autoregressive models inherently promote causal reasoning. In contrast, bidirectional denoising models lack the algorithmic foundations necessary to effectively capture causality, which leads to inferior performance in such tasks. While VideoPoet is also an autoregressive model, its primary design objective is integration with language models, which limits its efficiency in modeling the video modality; In contrast, MAGI-1 is purpose-built for video generation, combining the strengths of autoregressive and denoising-based modeling. This targeted design enables it to achieve significantly superior performance.

Nevertheless, our model is not without limitations. Fig. 17 presents several representative results, revealing both its strengths and weaknesses. While MAGI-1 effectively captures primary dynamics—such as projectile motion, rotational behavior, and material deformation, it struggles with complex secondary effects, including precise collision responses, material-specific reactions, and post-deformation behavior. Notably, even when the predicted outcome deviates from the ground truth, the model often generates physically

Model	Phys. IQ Score↑	Spatial IoU ↑	Spatio Temporal↑	Weighted Spatial IoU ↑	MSE↓
MAGI-1 (V2V)	**56.02**	**0.367**	**0.270**	**0.304**	**0.005**
VideoPoet (V2V) (Kondratyuk et al., 2023)	29.50	0.204	0.164	0.137	0.010
Lumiere (V2V) (Bar-Tal et al., 2024)	23.00	0.170	0.155	0.093	0.013
MAGI-1 (I2V)	**30.23**	**0.203**	0.151	**0.154**	**0.012**
Kling1.6 (I2V) (Kuaishou, 2024)	23.64	0.197	0.086	0.144	0.025
VideoPoet (I2V) (Kondratyuk et al., 2023)	20.30	0.141	0.126	0.087	0.012
Gen 3 (I2V) (Runway, 2024)	22.80	0.201	0.115	0.116	0.015
Wan2.1 (I2V) (Wang et al., 2025a)	20.89	0.153	0.100	0.112	0.023
Lumiere (I2V) (Bar-Tal et al., 2024)	19.00	0.113	**0.173**	0.061	0.016
SVD (I2V) (Blattmann et al., 2023)	14.80	0.132	0.076	0.073	0.021
Pika 1.0 (I2V) (PikaLabs, 2024)	13.00	0.140	0.041	0.078	0.014
Sora (I2V) (OpenAI, 2024)	10.00	0.138	0.047	0.063	0.030
GroundTruth	100.0	0.678	0.535	0.577	0.002

Table 10: Quantitative comparison of video generation models evaluated on the Physics-IQ-Benchmark. Models are categorized by input modality: image-to-video (I2V) and video-to-video (V2V). Results were obtained through direct evaluation of model APIs, local deployment of open-source implementations, and as reported in Motamed et al. (2025). In the V2V task, models observe the first 3 seconds of an 8-second ground truth video and predict the remaining 5 seconds, while in the I2V task, models take only a single frame at the 3-second mark and predict the subsequent 5 seconds. Magi(V2V) utilizes the full 24 FPS video input (96 frames).

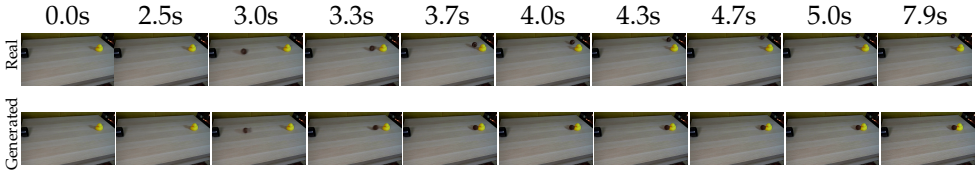
plausible alternatives. For example, in the second case (Fig. 17(b)), although the model fails to simulate the ignition of a match and the popping of a balloon, it instead produces a coherent sequence in which the rod rotates, contacts the object, and realistically bends upon impact. These results suggest that MAGI-1 has acquired a non-trivial physical intuition, capable of generating alternative yet physically consistent scenarios.

The influence of historical context length The benefit of utilizing historical context for more accurate predictions has already been demonstrated in the comparison between image-conditioned and video-conditioned MAGI-1 models. To more systematically evaluate the impact of historical information in physical modeling, we varied the length of accessible history by adjusting the KV range of MAGI-1 during inference. Fig. 18 presents the results. Overall, we observe that increasing the amount of historical context generally leads to better performance. However, the most significant gain occurs at KV range $= 2$, meaning that short-term history is often sufficient to support accurate predictions.

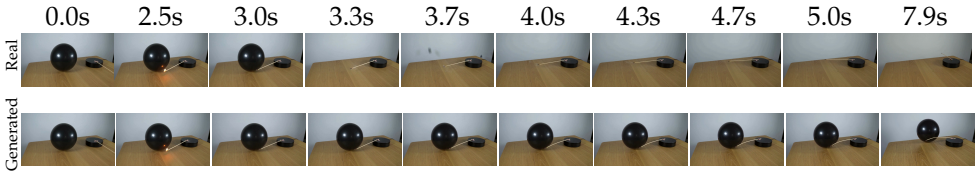
6 Related Works

This section reviews major developments in text-to-video generation, categorized by proprietary systems, open-source efforts, and recent trends in autoregressive and causal modeling. We highlight unresolved challenges in scalability, causality, and streaming compatibility—challenges that MAGI-1 is designed to address.

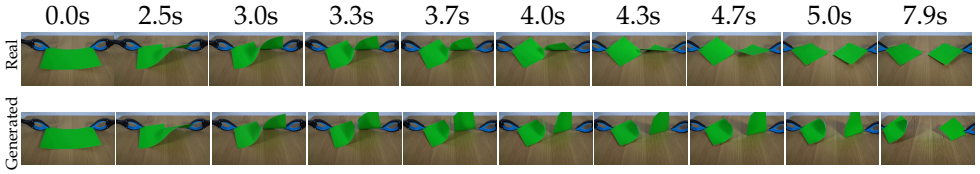
Proprietary Systems. Recent proprietary models have significantly advanced generation length, resolution, and semantic fidelity. OpenAI's Sora (OpenAI, 2024) introduced long-form, high-resolution generation with strong prompt consistency. Kuaishou's Kling (Kuaishou, 2024) and Runway's Gen-3 (Runway, 2024) emphasized temporal fidelity and fine-grained stylistic control, respectively. Luma AI's DreamMachine (LumaLabs, 2024) improved motion continuity and stylistic adherence. Pika Labs' Pika 1.5 (PikaLabs, 2024) enabled interactive control over visual attributes, while Meta's MovieGen (Polyak et al., 2024) offered transparency into foundational model training. Most recently, Google's Veo 2 (DeepMind, 2024) advanced physical realism and human motion modeling. Despite



(a) A light beige coffee table with a small yellow rubber ducky on it. A mustard yellow couch is in the background. There is a black pipe on one end of the table and a brown tennis ball rolls out of it towards the rubber ducky. Static shot with no camera movement.



(b) A black balloon is sitting on a wooden table next to a small rotating platform with a lit matchstick taped to it. The match rotates clockwise and touches the balloon. Static shot with no camera movement.



(c) Two black and blue gripping tools are pulling a piece of green paper from its two corners, causing it to tear. Static shot with no camera movement.

Figure 17: Case study results from the Physics-IQ Benchmark illustrate three distinct physical scenarios over time. Each scenario compares the ground truth (top row) with our model's predictions (bottom row), conditioned on the first 3 seconds and forecasting the next 5 seconds. The results highlight the model's ability to capture core physical interactions, as well as its limitations with complex material-specific effects: (a) The model correctly predicts the initial projectile motion but erroneously shows the ball deflecting off the duck instead of stopping upon impact. (b) Rotational dynamics are accurately captured, but the model fails to predict the match igniting and popping the balloon, instead showing the object being pushed back. (c) The model predicts the card tearing but struggles to model the motion of the torn pieces afterward.

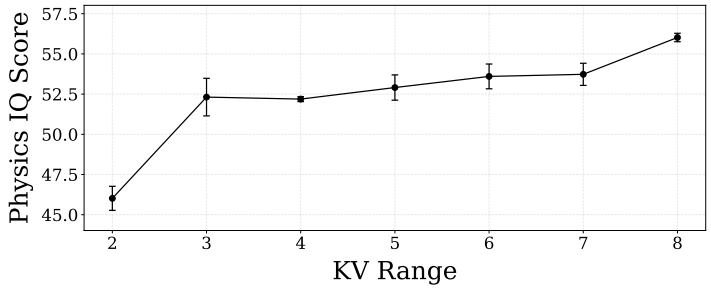


Figure 18: Physical IQ scores as a function of historical context. This visualization shows how performance changes with varying amounts of historical information, represented by the KV Range Value.

these innovations, most systems are closed-source and opaque in architecture, limiting reproducibility and extensibility.

Open-Source Ecosystem. The open-source community pioneered latent diffusion through Stable Diffusion (Esser et al., 2021), which integrated a variational autoencoder (Kingma, 2013) for latent representation, a CLIP-based text encoder (Radford et al., 2021), and a U-Net denoiser (Ronneberger et al., 2015). Temporal extensions such as VDM (Ho et al., 2022), AnimateDiff (Guo et al., 2024), and SVD (Blattmann et al., 2023) adapted the architecture for frame coherence. Transformer-based backbones like DiT (Peebles & Xie, 2023), PixArt-α (Chen et al., 2023), and Latte (Ma et al., 2024) demonstrated scalability and inspired early video adaptations. Recent open implementations—including Open-Sora (Zheng et al., 2024), Open-Sora-Plan (Lin et al., 2024), CogVideoX (Yang et al., 2025), Mochi 1 (GenmoTeam, 2024), HunyuanVideo (Kong et al., 2024), StepVideo (Ma et al., 2025), LTX-Video (HaCohen et al., 2024), and Wan (Wang et al., 2025a)—introduced modular advances in chunking, compression, and streaming. However, these systems largely retain bidirectional denoising and globally conditioned inference, limiting applicability to real-time or causal settings.

Autoregressive and Causal Modeling. An emerging trend is the integration of autoregressive modeling and causal constraints. Diffusion Forcing (Chen et al., 2024a) introduces independent per-token noise schedules that allow a causal model to denoise future tokens while keeping past tokens minimally perturbed, effectively unifying next-token prediction with full-sequence diffusion. FVDM (Liu et al., 2024) employed timestep vectorization for precise noise control. CausVid (Yin et al., 2024) combined causal inference with distillation for streaming scenarios. While promising, these models remain limited in scale, often lack chunk-wise abstraction, and do not unify video continuation with I2V/T2V generation.

MAGI-1: Scalable Autoregressive Diffusion. To our knowledge, MAGI-1 is the first large-scale, chunk-wise autoregressive diffusion model trained from scratch that unifies high-fidelity text-to-video, image-to-video, and video continuation tasks under strict causal constraints. It supports real-time streaming and long-horizon synthesis via efficient chunk-wise denoising, shortcut distillation, and KV-cached inference. By explicitly addressing scalability, causality, and streaming compatibility, MAGI-1 establishes a new foundation for unified and controllable video generation.

7 Conclusion

MAGI-1 introduces a scalable chunk-wise autoregressive diffusion framework for high-fidelity video synthesis. By progressively denoising fixed-length segments under strict causal constraints, it enables real-time, streaming-compatible generation with fixed computational overhead regardless of video length. The architecture builds upon a Transformer backbone enhanced with block-causal and parallel attention modules, and is supported by a distributed attention mechanism and a highly efficient training strategy for handling ultra-long contexts.

A key contribution lies in its unified design: MAGI-1 supports text-to-video, image-to-video, and video continuation tasks without requiring task-specific modifications, all under a shared training objective. Through chunk-wise text conditioning, it further achieves fine-grained semantic control across long-form video generation. A shortcut distillation strategy significantly reduces the number of diffusion steps required for inference, improving efficiency while maintaining temporal consistency and sample quality.

Empirical results on VBench-I2V and Physics-IQ benchmarks demonstrate that MAGI-1 outperforms existing large-scale video diffusion models in prompt adherence, physical plausibility, and temporal coherence. Taken together, these contributions establish MAGI-1 as a robust and extensible foundation for autoregressive video synthesis—offering both state-of-the-art performance and a fertile ground for future advancements in modularity, controllability, and multi-modal reasoning.

8 Limitation and Future Work

While MAGI-1 demonstrates strong generation quality and low-latency inference via chunk-wise autoregressive denoising, its current architecture remains tightly coupled. Specifically, a single large decoder-style Transformer is tasked with both (1) high-level temporal context fusion—integrating static conditioning signals with progressively noisier visual inputs—and (2) low-level denoising, which requires accurate reconstruction of fine-grained visual details. This conflation of heterogeneous objectives introduces several technical limitations:

- **Inference latency bottleneck**: The same large model is repeatedly invoked across all denoising steps, even when only minor refinements are required. This leads to inefficient utilization of compute, especially in streaming settings where low-latency frame delivery is critical.
- **Optimization conflict**: Jointly optimizing global semantic planning and pixel-level restoration within a single model exacerbates objective interference, often leading to suboptimal scaling behavior.
- **Limited controllability**: The monolithic architecture constrains the insertion of auxiliary control signals—such as confidence-based guidance modulation, or dynamic temporal constraints—due to entangled latent pathways and overlapping functional scopes.

Thus, a decoupled design that structurally separates high-level semantic reasoning from low-level visual synthesis is worth exploring. Looking ahead, as video generation evolves from producing isolated clips to constructing long-form content with coherent narratives, we anticipate a convergence between video generation and understanding. In this closed-loop setting, the quality of generated content will increasingly depend on the model's capacity to understand video content, making understanding the key bottleneck. Although we are still far from this frontier, we believe that a modular architecture represents a crucial step toward closing the loop between video understanding and generation.

Contributions and Acknowledgments

Names are presented alphabetically by first name.

Core Contributors:

Hansi Teng
Hongyu Jia
Lei Sun
Lingzhi Li
Maolin Li
Mingqiu Tang
Shuai Han
Tianning Zhang
W.Q. Zhang
Weifeng Luo
Xiaoyang Kang
Yuchen Sun
Yue Cao
Yunpeng Huang
Yutong Lin
Yuxin Fang
Zewei Tao
Zheng Zhang
Zhongshu Wang
Zixun Liu

Contributors:

Dai Shi
Guoli Su
Hanwen Sun
Hong Pan
Jie Wang
Jiexin Sheng
Min Cui
Min Hu
Ming Yan
Shucheng Yin
Siran Zhang
Tingting Liu
Xianping Yi
Xiaoyu Yang
Xin Song
Xuan Hu
Yankai Zhang
Yuqiao Li

References

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Hritik Bansal, Zongyu Lin, Tianyi Xie, Zeshun Zong, Michal Yarom, Yonatan Bitton, Chenfanfu Jiang, Yizhou Sun, Kai-Wei Chang, and Aditya Grover. Videophy: Evaluating physical commonsense for video generation. *arXiv preprint arXiv:2406.03520*, 2024.

Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, et al. Lumiere: A space-time diffusion model for video generation. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024a.

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart-alpha: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023.

Yukang Chen, Fuzhao Xue, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, Ethan He, Hongxu Yin, Pavlo Molchanov, Jan Kautz, Linxi Fan, Yuke Zhu, Yao Lu, and Song Han. Longvila: Scaling long-context visual language models for long videos, 2024b. URL https://arxiv.org/abs/2408.10188.

PyTorch Contributors. Pytorch distributed tensor offsetbasedrngtracker. https://github.com/pytorch/pytorch/blob/v2.7.0-rc10/torch/distributed/tensor/_random.py#L156, 2025a.

PyTorch Contributors. Pytorch distributed tensor shardingpropagator. https://github.com/pytorch/pytorch/blob/v2.7.0-rc10/torch/distributed/tensor/_sharding_prop.py#L51, 2025b.

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

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Debabrata Dash, Neoklis Polyzotis, and Anastasia Ailamaki. Cophy: A scalable, portable, and interactive index advisor for large workloads. *arXiv preprint arXiv:1104.3214*, 2011.

Google DeepMind. Veo 2. *https://deepmind.google/technologies/veo/veo-2/*, 2024.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pp. 7480–7512. PMLR, 2023.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels, 2024. URL https://arxiv.org/abs/2412.05496.

György Dósa. The tight bound of first fit decreasing bin-packing algorithm is ffd(i) <= 11/9opt(i) + 6/9. ESCAPE'07, pp. 1–11, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3540744495.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. pp. 12873–12883, 2021.

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.

Jiarui Fang and Shangchun Zhao. Usp: A unified sequence parallelism approach for long context generative ai, 2024. URL https://arxiv.org/abs/2405.07719.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts, 2022.

Hao Ge, Junda Feng, Qi Huang, Fangcheng Fu, Xiaonan Nie, Lei Zuo, Haibin Lin, Bin Cui, and Xin Liu. Bytescale: Efficient scaling of llm training with a 2048k context length on more than 12,000 gpus, 2025. URL https://arxiv.org/abs/2502.21231.

GenmoTeam. Mochi 1. https://github.com/genmoai/models, 2024.

Diandian Gu, Peng Sun, Qinghao Hu, Ting Huang, Xun Chen, Yingtong Xiong, Guoteng Wang, Qiaoling Chen, Shangchun Zhao, Jiarui Fang, Yonggang Wen, Tianwei Zhang, Xin Jin, and Xuanzhe Liu. Loongtrain: Efficient training of long-sequence llms with head-context parallelism, 2024. URL https://arxiv.org/abs/2406.18485.

Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *International Conference on Learning Representations*, 2024.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

Horace He, Less Wright, Luca Wehrstedt, Tianyu Liu, and Wanchao Liang. [distributed w/ torchtitan] introducing async tensor parallelism in pytorch. https://discuss.pytorch.org/t/distributed-w-torchtitan-introducing-async-tensor-parallelism-in-pytorch/209487, 2024.

Danny Hernandez, Tom Brown, Tom Conerly, Nova DasSarma, Dawn Drain, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Tom Henighan, Tristan Hume, et al. Scaling laws and interpretability of learning from repeated data. *arXiv preprint arXiv:2205.10487*, 2022.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. 33: 6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, Yaohui Wang, Xinyuan Chen, Ying-Cong Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench++: Comprehensive and versatile benchmark suite for video generative models, 2024. URL https://arxiv.org/abs/2411.13503.

Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023. URL https://arxiv.org/pdf/2309.14509.

Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How far is video generation from world model: A physical law perspective. *arXiv preprint arXiv:2411.02385*, 2024.

Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, et al. Videopoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023.

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.

Kuaishou. Kling ai. *https://klingai.kuaishou.com/*, 2024.

Achintya Kundu, Rhui Dih Lee, Laura Wynter, Raghu Kiran Ganti, and Mayank Mishra. Enhancing training efficiency using packing with flash attention, 2024. URL https://arxiv.org/abs/2407.09105.

Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, Sanket Purandare, Gokul Nadathur, and Stratos Idreos. Torchtitan: One-stop pytorch native solution for production ready llm pre-training, 2024. URL https://arxiv.org/abs/2410.06511.

Bin Lin, Yunyang Ge, Xinhua Cheng, Zongjian Li, Bin Zhu, Shaodong Wang, Xianyi He, Yang Ye, Shenghai Yuan, Liuhan Chen, et al. Open-sora plan: Open-source large video generation model. *arXiv preprint arXiv:2412.00131*, 2024.

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

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

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022a.

Yaofang Liu, Yumeng Ren, Xiaodong Cun, Aitor Artola, Yang Liu, Tieyong Zeng, Raymond H Chan, and Jean-michel Morel. Redefining temporal modeling in video diffusion: The vectorized timestep approach. *arXiv preprint arXiv:2410.03160*, 2024.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12009–12019, 2022b.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

LumaLabs. Dream machine. *https://lumalabs.ai/dream-machine*, 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*, 2025.

Xin Ma, Yaohui Wang, Gengyun Jia, Xinyuan Chen, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*, 2024.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14297–14306, 2023.

Fanqing Meng, Jiaqi Liao, Xinyu Tan, Wenqi Shao, Quanfeng Lu, Kaipeng Zhang, Yu Cheng, Dianqi Li, Yu Qiao, and Ping Luo. Towards world simulator: Crafting physical commonsense-based benchmark for video generation. *arXiv preprint arXiv:2410.05363*, 2024.

MiniMax. Hailuo ai. *https://hailuoai.com/video*, 2024.

Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do generative video models understand physical principles?, 2025. URL https://arxiv.org/abs/2501.09038.

NVIDIA. Accelerating transformers with nvidia cudnn 9. https://developer.nvidia.com/blog/accelerating-transformers-with-nvidia-cudnn-9/, 2024. Accessed: 2024-12-12.

OpenAI. Video generation models as world simulators, 2024. URL https://openai.com/index/video-generation-models-as-world-simulators/.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. pp. 4195–4205, 2023.

PikaLabs. Pika 1.5. *https://pika.art/*, 2024.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu, Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dimitry Vengertsev, Edgar Schonfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

PyTorch. torch.nn.functional.scaled_dot_product_attention - pytorch 2.6 documentation. https://pytorch.org/docs/stable/generated/torch.nn.functional.scaled_dot_product_attention.html.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models, 2020. URL https://arxiv.org/abs/1910.02054.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

Runway. Gen-3. *https://runwayml.com/*, 2024.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.

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.

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

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pp. 31094–31116. PMLR, 2023.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

Sirluk. Efficient llm pretraining: Packed sequences and masked attention. https://huggingface.co/blog/sirluk/llm-sequence-packing, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

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.

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

PyTorch Team. Reproducibility — pytorch 2.6 documentation. https://pytorch.org/docs/stable/notes/randomness.html, 2024a.

PyTorch Team. torch.distributed.tensor — pytorch 2.6 documentation. https://pytorch.org/docs/stable/distributed.tensor.html, 2024b.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 402–419. Springer, 2020.

GitHub User. [question] why should cuda_device_max_connections=1 should be set when using seq_parallel or async comm? https://github.com/NVIDIA/Megatron-LM/issues/533, 2023.

GitHub User. Allow passing cuda stream to the nccl collectives (specially the functional collectives). https://github.com/pytorch/pytorch/issues/137390, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. 30, 2017.

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

Guoxia Wang, Jinle Zeng, Xiyuan Xiao, Siming Wu, Jiabin Yang, Lujing Zheng, Zeyu Chen, Jiang Bian, Dianhai Yu, and Haifeng Wang. Flashmask: Efficient and rich mask extension of flashattention, 2025b. URL https://arxiv.org/abs/2410.01359.

Yujie Wang, Shiju Wang, Shenhan Zhu, Fangcheng Fu, Xinyi Liu, Xuefeng Xiao, Huixia Li, Jiashi Li, Faming Wu, and Bin Cui. Data-centric and heterogeneity-adaptive sequence parallelism for efficient llm training, 2024. URL https://arxiv.org/abs/2412.01523.

Sam Wiseman and Alexander M Rush. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*, 2016.

Haoning Wu, Erli Zhang, Liang Liao, Chaofeng Chen, Jingwen Hou, Annan Wang, Wenxiu Sun, Qiong Yan, and Weisi Lin. Exploring video quality assessment on user generated contents from aesthetic and technical perspectives. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20144–20154, 2023.

Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4818–4829, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023a.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023b.

Peng Xu, Wei Ping, Xianchao Wu, Zihan Liu, Mohammad Shoeybi, and Bryan Catanzaro. Chatqa 2: Bridging the gap to proprietary llms in long context and rag capabilities. *arXiv preprint arXiv:2407.14482*, 2024.

Yuanzhong Xu, HyoukJoong Lee, Dehao Chen, Blake Hechtman, Yanping Huang, Rahul Joshi, Maxim Krikun, Dmitry Lepikhin, Andy Ly, Marcello Maggioni, Ruoming Pang, Noam Shazeer, Shibo Wang, Tao Wang, Yonghui Wu, and Zhifeng Chen. Gspmd: General and scalable parallelization for ml computation graphs, 2021. URL https://arxiv.org/abs/2105.04663.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Xiaotao Gu, Yuxuan Zhang, Weihan Wang, Yean Cheng, Ting Liu, Bin Xu, Yuxiao Dong, and Jie Tang. CogVideoX: Text-to-Video Diffusion Models with An Expert Transformer. 2025.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast causal video generators. *arXiv preprint arXiv:2412.07772*, 2024.

Jinhui Yuan, Xinqi Li, Cheng Cheng, Juncheng Liu, Ran Guo, Shenghang Cai, Chi Yao, Fei Yang, Xiaodong Yi, Chuan Wu, Haoran Zhang, and Jie Zhao. Oneflow: Redesign the distributed deep learning framework from scratch, 2022. URL https://arxiv.org/abs/2110.15032.

Geng Zhang, Xuanlei Zhao, Kai Wang, and Yang You. Training variable sequences with data-centric parallel, 2024.

Ting Zhao and Xiangqian Wu. Pyramid feature attention network for saliency detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3085–3094, 2019.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, March 2024.

zhuzilin. [feature request] balancing computation with zigzag blocking. https://github.com/zhuzilin/ring-flash-attention/issues/2, Feb 2024.

A Inference Infra

A.1 W8A8 Quantization

We adopt the A8W8 SmoothQuant approach (Xiao et al., 2023a), which leverages a calibration dataset to pre-compute per-channel scaling factors s. This enables an equivalent transformation of the form $Y = (X \cdot \text{diag}(s)^{-1}) \cdot (\text{diag}(s)W)$, effectively mitigating the impact of outliers in channel-wise activations.

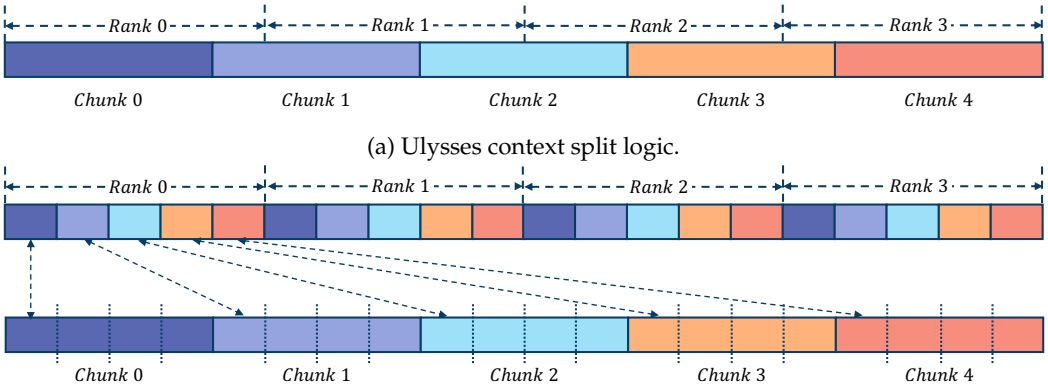
For calibration, we constructed a dataset encompassing a wide range of usage scenarios, including different task types (*e.g.*, T2V and I2V) and a uniformly sampled step size within the range [12, 32]. Notably, I2V samples constituted approximately 30% of the dataset. We employed the `FP8` data type for quantization, as `INT8` was found to introduce noticeable visual artifacts in the generated videos. Furthermore, we conducted a hyperparameter search over the range $\alpha \in (0.4, 0.6)$, and ultimately selected $\alpha = 0.45$ for SmoothQuant. All model weights were quantized except for the first and last layers. This quantization strategy led to a 30% performance improvement without compromising generation quality.

A.2 Multi-Node Parallel Inference

We adopted a multi-node Ulysses-based parallel inference framework across 3 nodes (24 GPUs), where inter-GPU communication and the high computational density of attention emerged as the primary bottlenecks. Ulysses performs four all-to-all communication steps for the q, k, v, and o tensors. To mitigate communication overhead, we carefully overlapped each communication stage with corresponding computations:

- v-communication overlaps with k-computation
- k-communication overlaps with q-computation
- q-communication overlaps with KV cache updates
- o-communication overlaps with cross-attention computation

This overlapping strategy effectively reduced communication overhead to less than 3% of total execution time.



(a) Ulysses context split logic.

(b) CSO context split logic.

Figure 19: Overview of Ulysses and CSO context split logic.

A.3 Context Shuffle Overlap

We optimized based on the Ulysses CP algorithm, striving to overlap communication with computation or data movement. Since the RTX 4090 GPUs communicate via PCIe, which has a relatively low bandwidth, we further proposed the **CSO (Context Shuffle Overlap)** algorithm to optimize communication more deeply.

The key difference between CSO and Ulysses lies in how the context is partitioned. In Ulysses, all chunks are distributed sequentially across different ranks. In contrast, CSO assigns each rank a partial view of every chunk. As illustrated in Fig. 19, this alternative partitioning strategy allows CSO to conveniently overlap computation and communication at the chunk level. Assuming we have 5 chunks, the complete process of CSO is as follows:

- k-communication and v-communication of all chunks overlaps with q-computation of all chunks
- q-communication of chunk 1 overlaps with KV cache updates
- q-communication of chunk 2 overlaps with o-computation of chunk 1
- q-communication of chunk 3 and o-communication of chunk 1 overlaps with o-computation of chunk 2
- q-communication of chunk 4 and o-communication of chunk 2 overlaps with o-computation of chunk 3
- q-communication of chunk 5 and o-communication of chunk 3 overlaps with o-computation of chunk 4
- o-communication of chunk 4 overlaps with o-computation of chunk 5
- o-communication of chunk 5 overlaps with cross-attention computation

In addition, the CSO partition pattern enables communication operations to be split into multiple balanced all-to-all communications. These balanced operations offer better performance compared to unbalanced all-to-all communication and also allow for efficient subsequent merging.

B Training Infrastructure

B.1 MagiAttention Materials

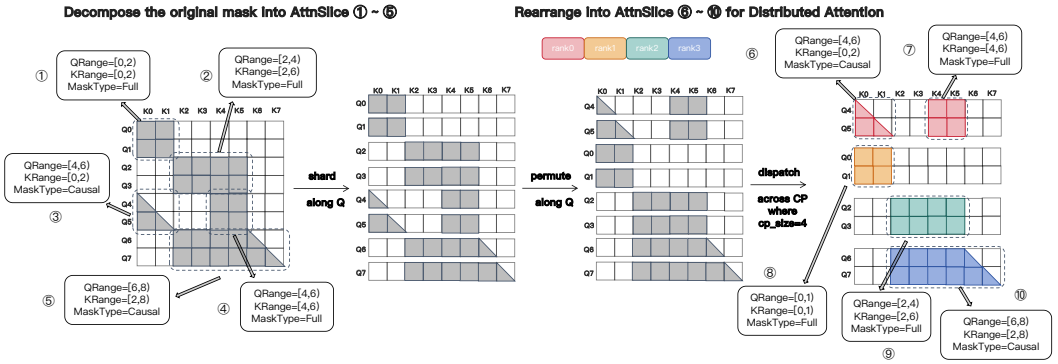


Figure 20: Illustration of `AttnSlice` formulation for some irregular mask (see 4.1.2). It decomposes the original mask into multiple `AttnSlices` and allows re-expression of fractal masks after rearrangement across CP ranks, making it suitable for distributed attention. Note that computation load balance across CP ranks is not considered in this illustration.

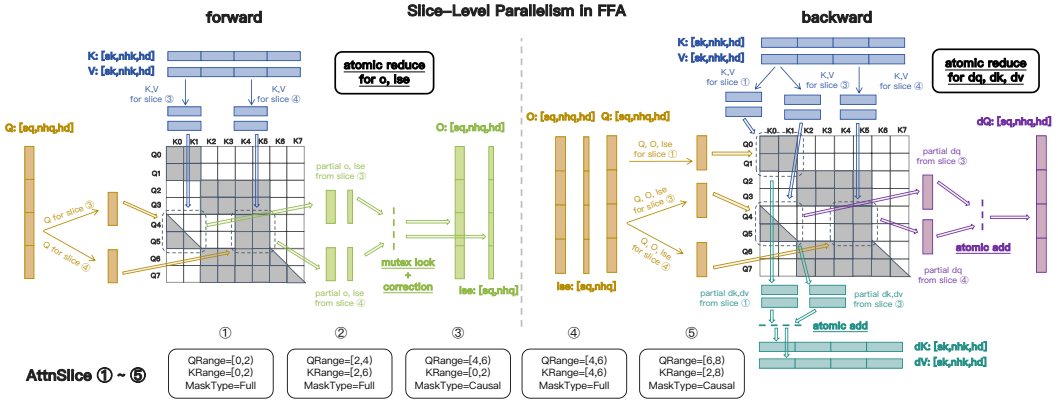


Figure 21: Illustration of slice-level parallelism in FFA for both forward and backward kernels (see 4.1.2). The overlapping nature across slices in both rows (`QRange`) and columns (`KRange`) necessitates *atomic reduce* operations in both kernels to ensure correct reduction.

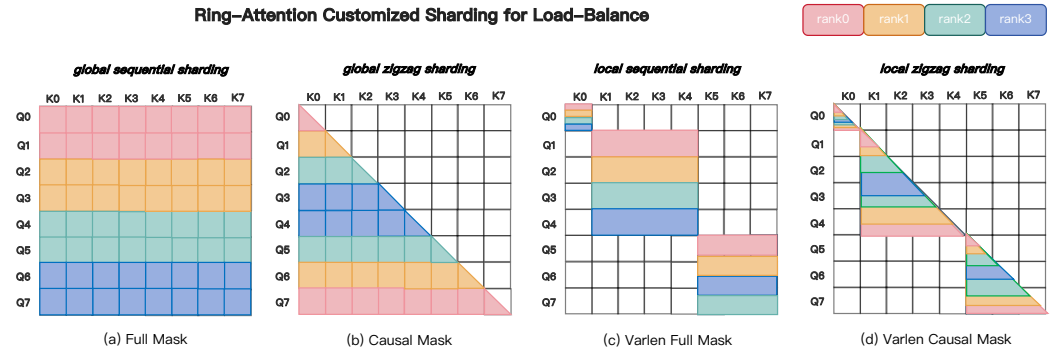


Figure 22: Illustration of Ring-Attention's customized sharding strategies for load balancing. (a) Full mask uses sequential sharding for the global mask; (b) Causal mask employs tailored *zigzag sharding* (zhuzilin, 2024); (c) Varlen full mask applies sequential sharding per local mask (one per packed sample); (d) Varlen causal mask uses *zigzag sharding* per local mask, causing performance degradation from fragmentation and padding.

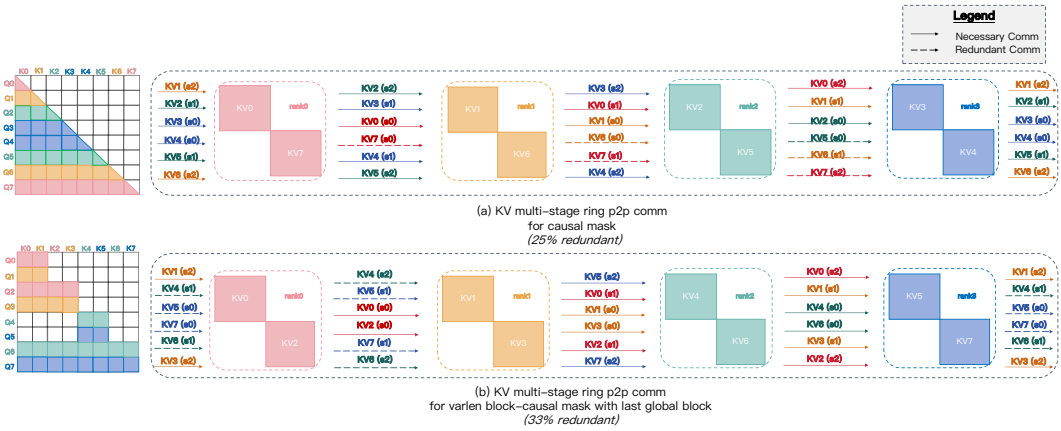


Figure 23: Examples illustrating redundant communication in Ring P2P patterns for distributed attention given heterogeneous masks (see 4.1.2).: (a) Even with a simple causal mask, Ring P2P incurs **25%** redundant communication; (b) For irregular mask patterns such as varlen block-causal mask with last global block, Ring P2P results in over **33%** redundancy.

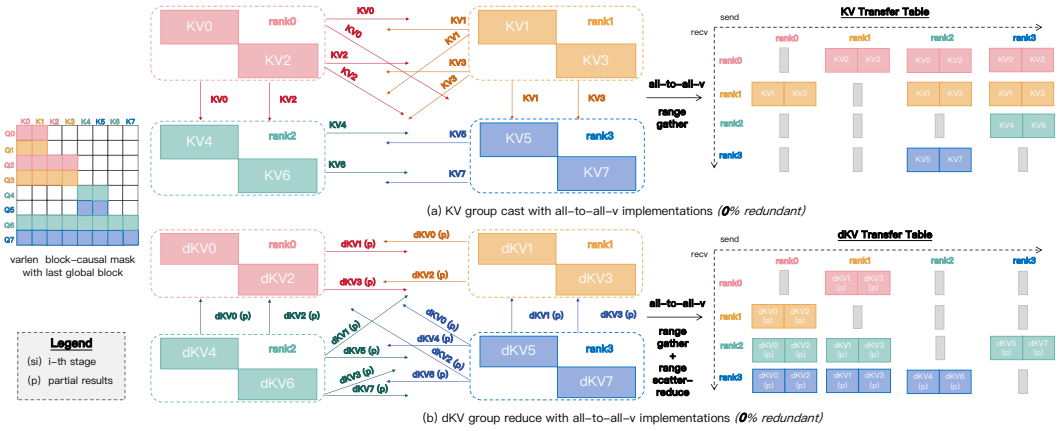


Figure 24: Illustration of `group-cast`/`group-reduce` primitives for zero redundancy, using the varlen block-causal mask with the last global block as an example for irregular patterns (see 4.1.2). (a) In both forward and backward passes, the `group-cast` primitive internally analyzes and generates a transfer table for KV send/receive buffers, and launches the underlying `all-to-all-v` to complete communication with our custom *Range Gather* kernel for pre-/post-processing. (b) In the backward pass, `group-reduce` similarly handles the partial dKV communication for reduction, using `all-to-all-v` with the *Range Gather* kernel for pre-processing and the *Range Scatter-Reduce* kernel for post-processing.

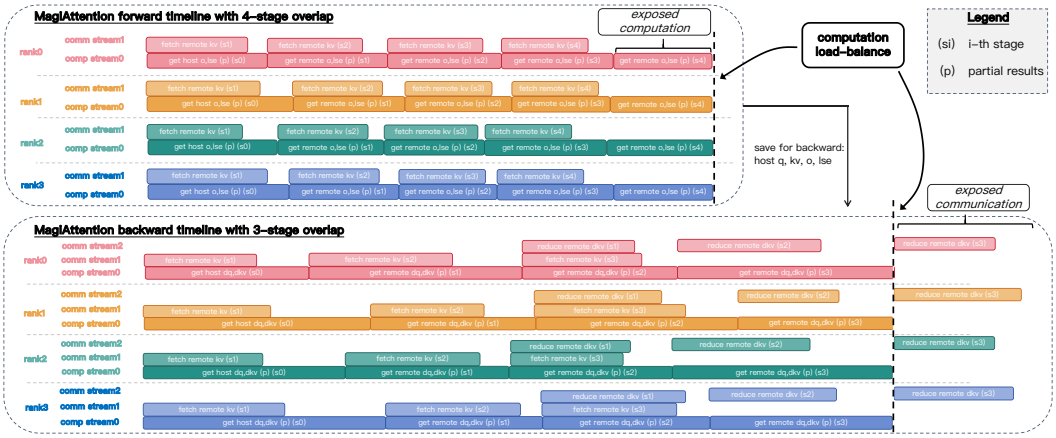


Figure 25: Schematic of MagiAttention's multi-stage overlap scheduling (see 4.1.2). (a) Forward pass: 4-stage scheduling overlaps computation (partial attention outputs and *lse* factors) with prefetching of next-stage KV requests (where applicable), hiding all communication overhead with the final stage's computation exposed. (b) Backward pass: 3-stage scheduling overlaps computation (partial dQ, dKV) with prefetching of next-stage KV requests and reduction of prior dKV requests, hiding all communication overhead except the dKV reduction of the final stage.

Algorithm 1 Greedy Load-Balance Dispatch Algorithm via Min-Heap w.r.t. 4.1.2

Require: Dispatch chunk and area pairs $\{(C_i, \text{Area}(C_i))\}_{i=1}^{n}$, number of buckets cp_size ($cp_size \mid n$)

Ensure: A dispatch mapping minimizing maximum bucket workload, satisfying Eq. (11)

1: Sort dispatch chunks $\{C_i\}_{i=1}^{n}$ in descending order by $\text{Area}(C_i)$
2: Compute chunk capacity per bucket: $chunk_per_bucket \leftarrow n / cp_size$
3: Initialize buckets: workload counters $W \leftarrow [0]^{cp_size}$, chunk counters $count \leftarrow [0]^{cp_size}$, and mapping $B_j \leftarrow \emptyset, \forall j$
4: Initialize min-heap $\mathcal{H}$ with tuples $(0, j)$ for each bucket $j \in [1, cp_size]$
5: **for** each chunk C_i in sorted order **do**
6: **repeat**
7: $(w_{\min}, j_{\min}) \leftarrow \text{extract-min}(\mathcal{H})$
8: **until** $count[j_{\min}] < chunk_per_bucket$ {Select least-loaded bucket that is not full}
9: Assign chunk: $B_{j_{\min}} \leftarrow B_{j_{\min}} \cup \{C_i\}$
10: Update workload and count:

$$W[j_{\min}] \leftarrow W[j_{\min}] + \text{Area}(C_i), \quad count[j_{\min}] \leftarrow count[j_{\min}] + 1$$

11: Insert updated tuple $(W[j_{\min}], j_{\min})$ back into $\mathcal{H}$
12: **end for**
13: **return** Bucket assignments $\{B_j\}_{j=1}^{cp_size}$ and maximum bucket workload $\max_j W[j]$

Algorithm 2 Dynamic Overlap Stage Search Algorithm w.r.t. 4.1.2

Require: Remote KV/dKV requests per rank; kernel cost models for FFA ($\mathrm{C}_{ffa}(\cdot)$), `group-cast` ($\mathrm{C}_{gc}(\cdot)$), and `group-reduce` ($\mathrm{C}_{gr}(\cdot)$), estimated via offline profiling; hyper-parameters to control search range: $(min_chunk_size, max_num_chunks)$

Ensure: Optimal number of overlap stages for both forward and backward passes

1: **for** each rank_i in parallel **do**

2: **[Step 1: Communication Partition]** Partition remote KV/dKV requests into p_i fine-grained communication packages, with sizes bounded by hyperparameters $(min_chunk_size, max_num_chunks)$

3: **[Step 2: Cost Evaluation]**

4: **for** $s_i \leftarrow p_i$ to 1 **do**

5: Randomly assign p_i communication packages into s_i stages, and pack them

6: Estimate the overall timeline costs for both forward and backward as:

$$C^{(fwd)}(s_i) \leftarrow \sum_{j=0}^{s_i-1} \underbrace{\max\left\{\mathrm{C}_{gc}(j{+}1),\ \mathrm{C}_{ffa}(j)\right\}}_{\text{overlapped}} + \underbrace{\mathrm{C}_{ffa}(s_i)}_{\text{exposed}}$$

$$C^{(bwd)}(s_i) \leftarrow \sum_{j=0}^{s_i} \underbrace{\max\left\{\mathrm{C}_{gc}(j{+}1), \mathrm{C}_{ffa}(j), \mathrm{C}_{gr}(j{-}1)\right\}}_{\text{overlapped}} + \underbrace{\mathrm{C}_{gr}(s_i)}_{\text{exposed}}$$

{Where $\mathrm{C}_{ffa}(0)$ denotes the host computation cost and $\mathrm{C}_{gc}(s_i{+}1), \mathrm{C}_{gr}(0), \mathrm{C}_{gr}(-1)$ are all assigned to 0 to handle boundary conditions}

7: **end for**

8: **[Step 3: Local Selection]**

9: $s_i^{*(fwd)} \leftarrow \arg\min_{s_i} C^{(fwd)}(s_i)$ {Optimal forward stage count for rank_i}

10: $s_i^{*(bwd)} \leftarrow \arg\min_{s_i} C^{(bwd)}(s_i)$ {Optimal backward stage count for rank_i}

11: **end for**

12: **[Step 4: Global Synchronization]**

13: $num_stages^{(fwd)} \leftarrow \max_i s_i^{*(fwd)}$ via All-Reduce across all CP ranks

14: $num_stages^{(bwd)} \leftarrow \max_i s_i^{*(bwd)}$ via All-Reduce across all CP ranks

15: **return** $num_stages^{(fwd)}, num_stages^{(bwd)}$

B.2 MagiAttention Experiments

B.2.1 Benchmarking MagiAttention kernel-level performance and flexibility

To demonstrate FFA kernels' state-of-the-art performance and flexibility in handling ultra-long, heterogeneous mask training, we measure the throughput (in TFLOPs/s) on Hopper GPUs for both forward and backward passes of prevalent attention kernels across standard mask patterns (see Fig.27, Fig.28) with their varlen variants (see Fig.29 and Fig.30), and some irregular mask patterns (see Fig.31).

Benchmark settings: for each mask pattern, we vary the sequence length *seqlen* from $4k, 8k, 16k, ...$, up to $128k$ ($seqlen_q = seqlen_k = seqlen$) while measuring throughput (in TFLOPs/s) for forward and backward passes of different attention kernels. Other configurations are fixed using common training settings (see Tab.11) to focus on the impact of sequence length and mask pattern. For the varlen packed data, we simply follow the variable sequence length distribution in the open-sourced dataset (Xu et al., 2024) (see Fig.26), from which we sample to pack and pad to the required *seqlen*.

To calculate the TFLOPs/s for various mask patterns during both forward and backward passes, we use the subsequent equations, following Flash-Attention (Dao, 2023):

$$\begin{aligned} \text{FLOPs}^{(fwd)} = \underbrace{2}_{\text{2 matmul}} \times \underbrace{2}_{\text{2 flops per matmul}} \times \text{MaskArea}(seqlen, mask_type) \\ \times batch_size \times num_heads_q \times head_dim \end{aligned} \tag{12}$$

$$\text{FLOPs}^{(bwd)} = \underbrace{2.5}_{\text{5 matmul due to recomputation}} \times \text{FLOPs}^{(fwd)} \tag{13}$$

$$\begin{aligned} where \quad & \text{MaskArea}(seqlen, full) = seqlen^2, \\ & \text{MaskArea}(seqlen, causal) = \frac{seqlen(seqlen+1)}{2}, \quad ... \end{aligned}$$

$$\text{TFLOPs/s}^{(wd)} = \frac{\text{FLOPs}^{(wd)}}{\text{Runtime}^{(wd)}}, \quad wd \in \{fwd, bwd\} \tag{14}$$

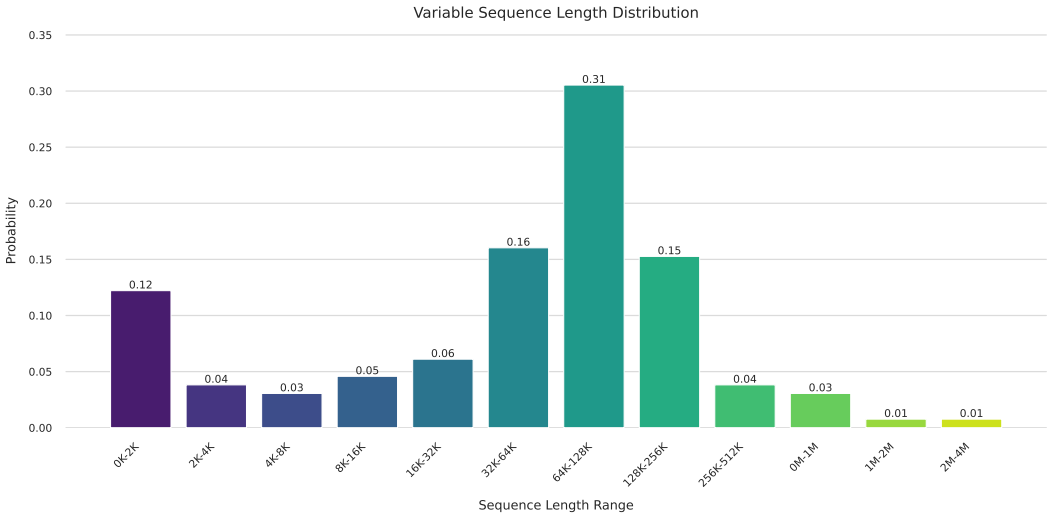
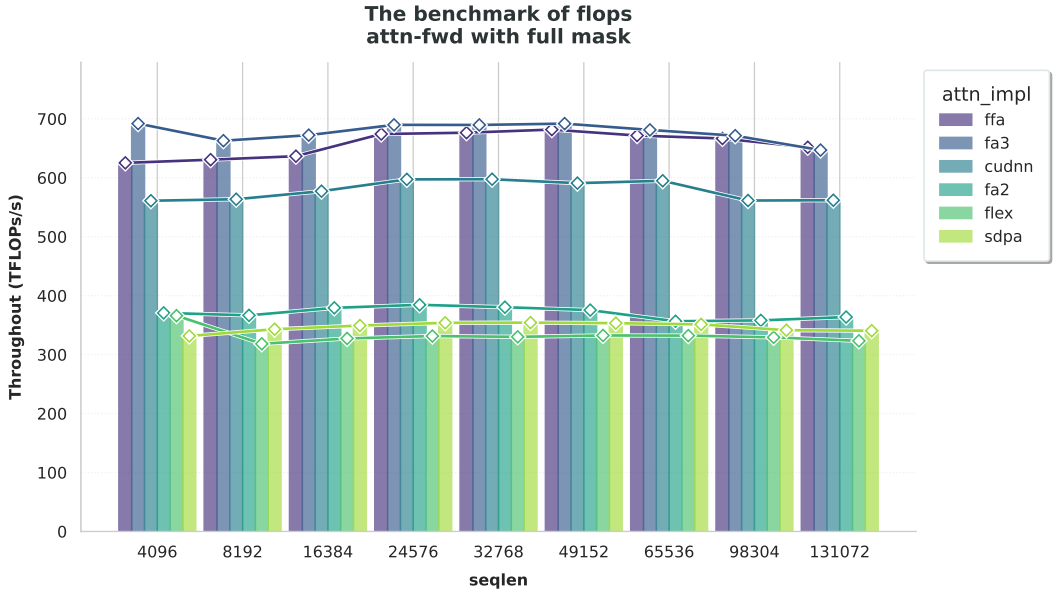


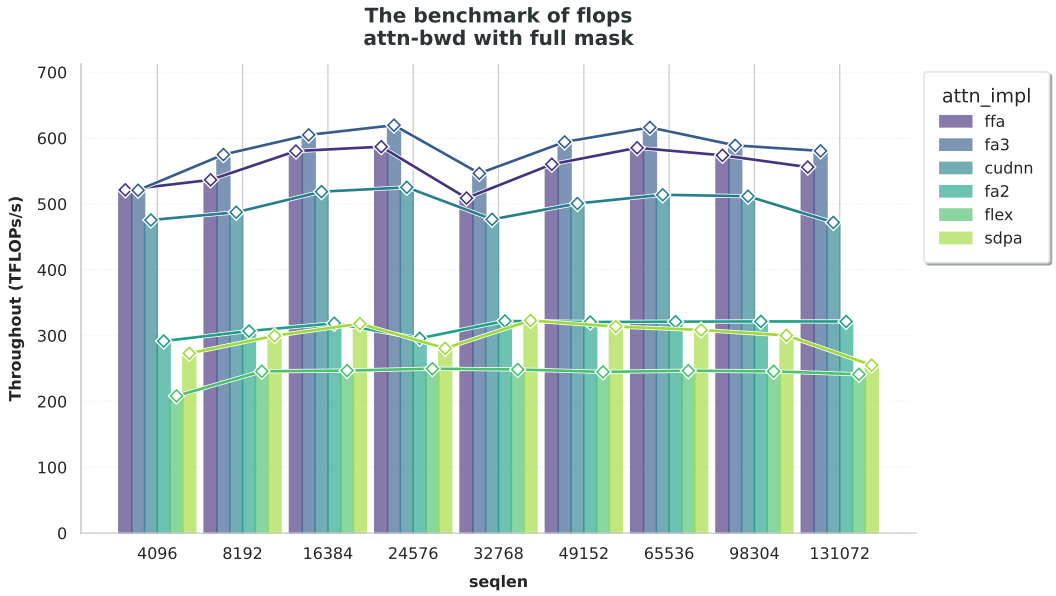
Figure 26: Distribution of sequence lengths in the dataset (Xu et al., 2024), used to sample and construct the variable-length data for both kernel-level and module-level experiments of MagiAttention.

Settings	Value
Batch Size (b)	1
Number of Heads (nh)	nhq:nhk:nhv = 64:8:8 (*GQA*)
Head Dimension (hd)	128
Dtype	torch.bfloat16
Window Size	1024 (*for sliding window masks only*)

Table 11: The fixed settings of FFA performance and flexibility benchmark

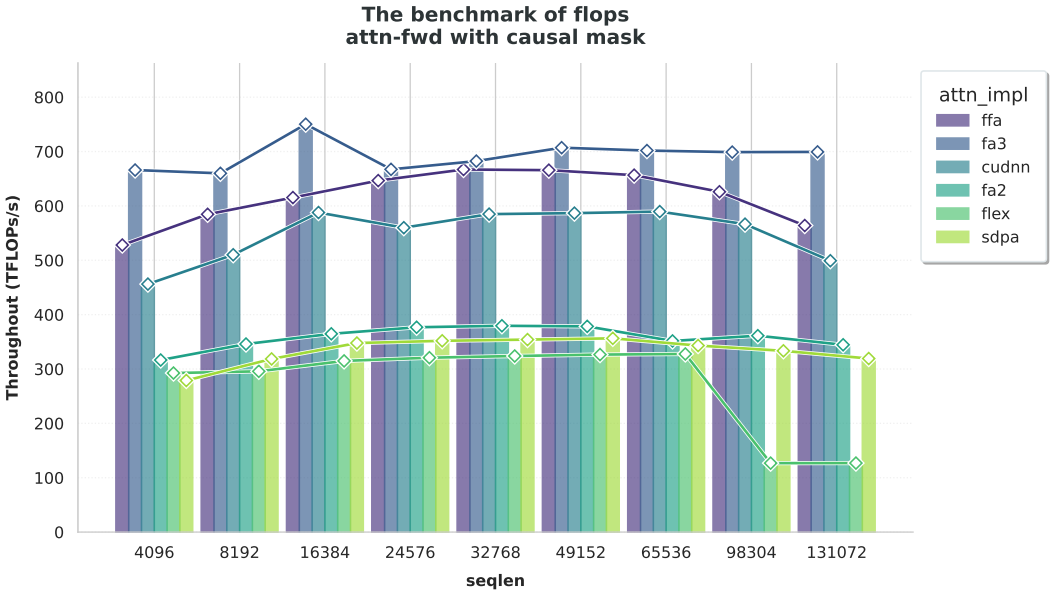


(a) Forward pass for full mask.

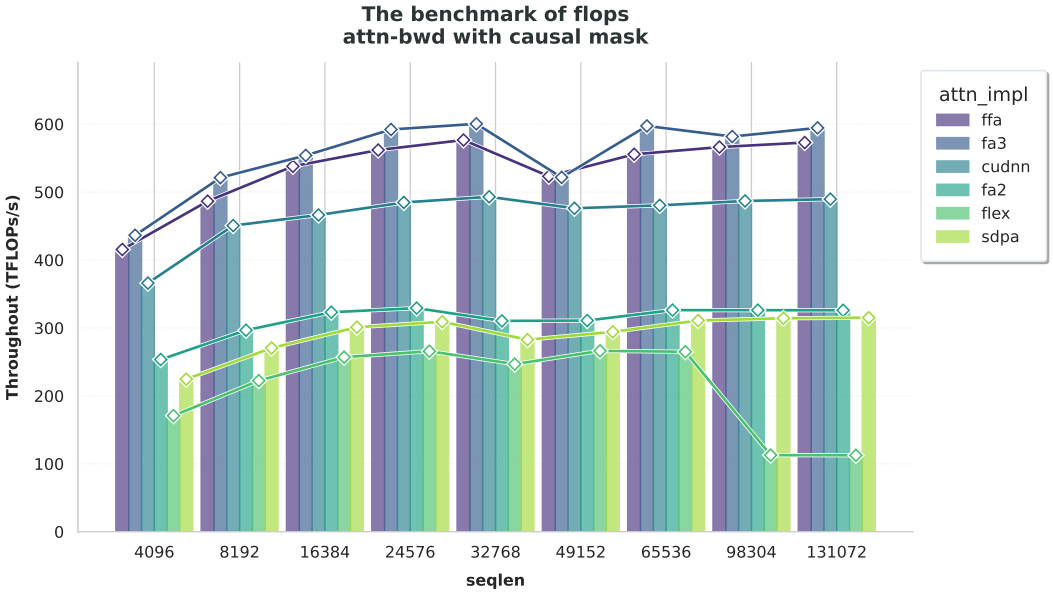


(b) Backward pass for full mask.

Figure 27: Benchmarking FFA's performance and flexibility against other leading attention kernels for full mask scenarios.

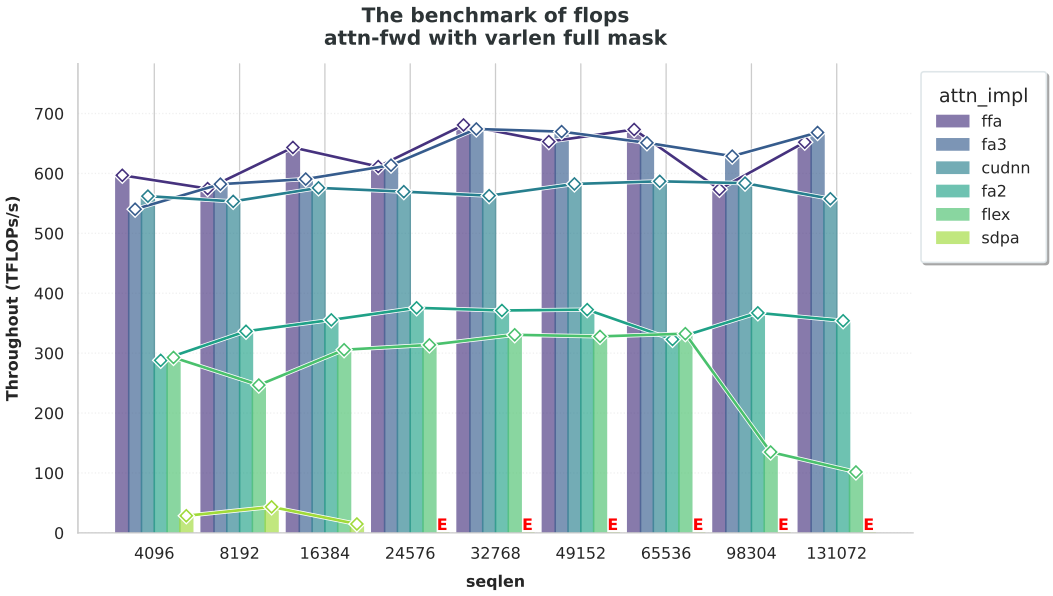


(a) Forward pass for causal mask.

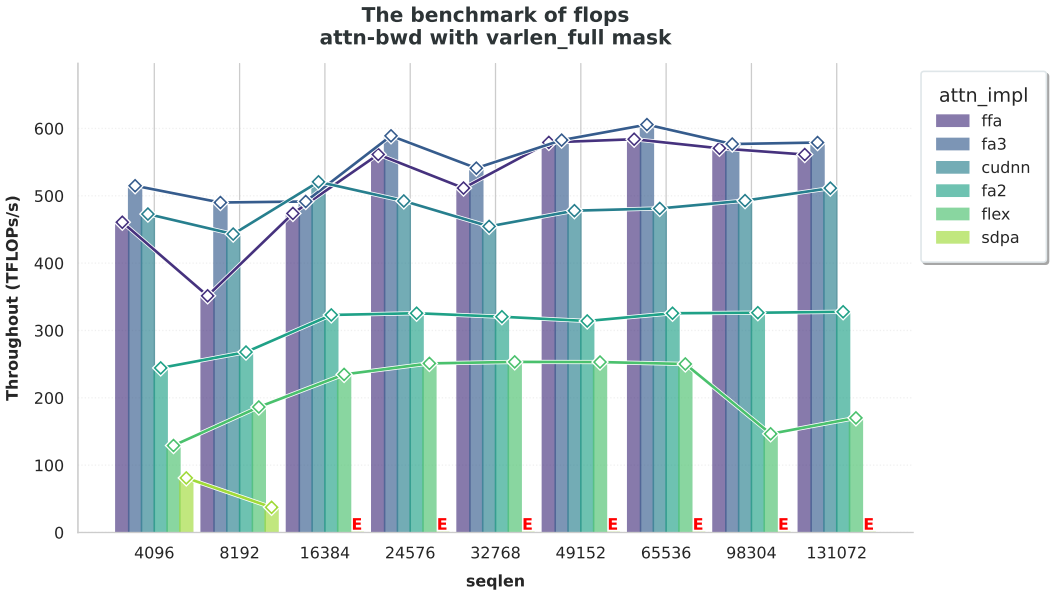


(b) Backward pass for causal mask.

Figure 28: Benchmarking FFA's performance and flexibility against other leading attention kernels for causal mask scenarios.

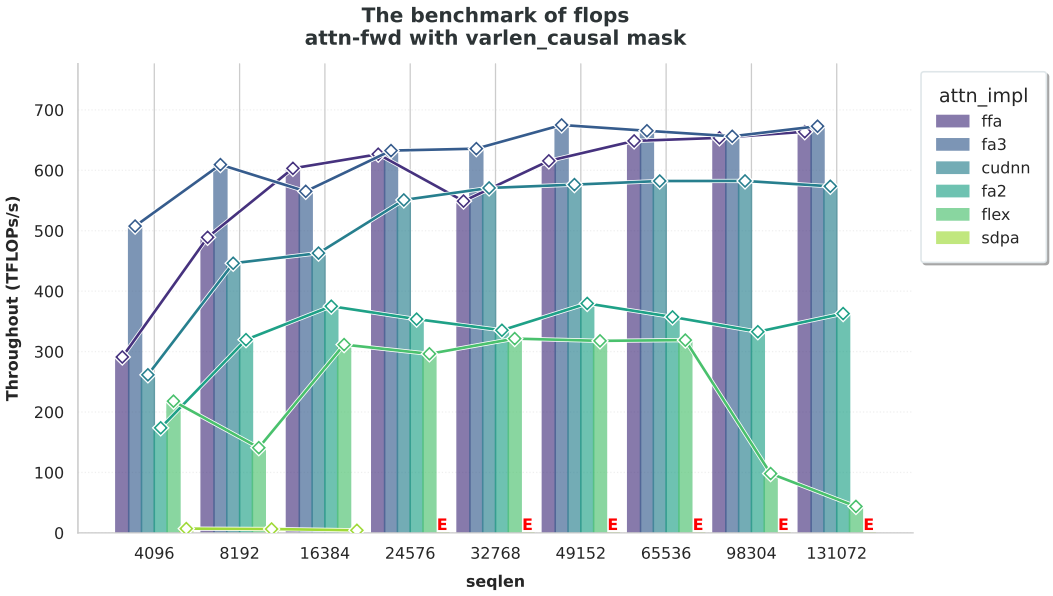


(a) Forward pass for varlen full mask.

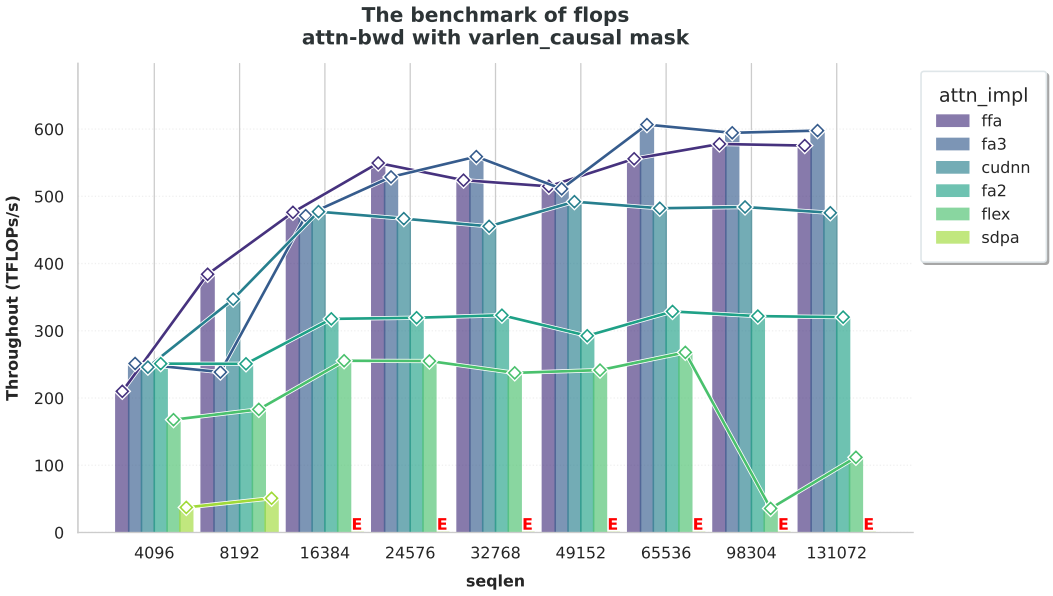


(b) Backward pass for varlen full mask.

Figure 29: Benchmarking FFA's performance and flexibility against other leading attention kernels for varlen full mask scenarios. (Note that: the **E** symbol indicates the corresponding distributed attention implementation raises *Cuda Out of Memory* error in that specific configuration.)

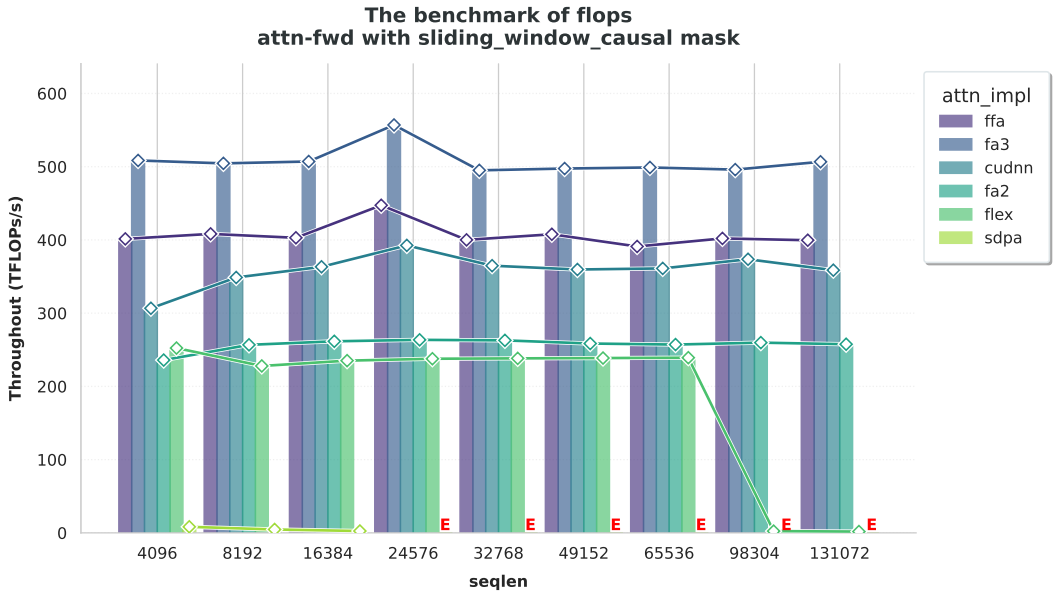


(a) Forward pass for varlen causal mask

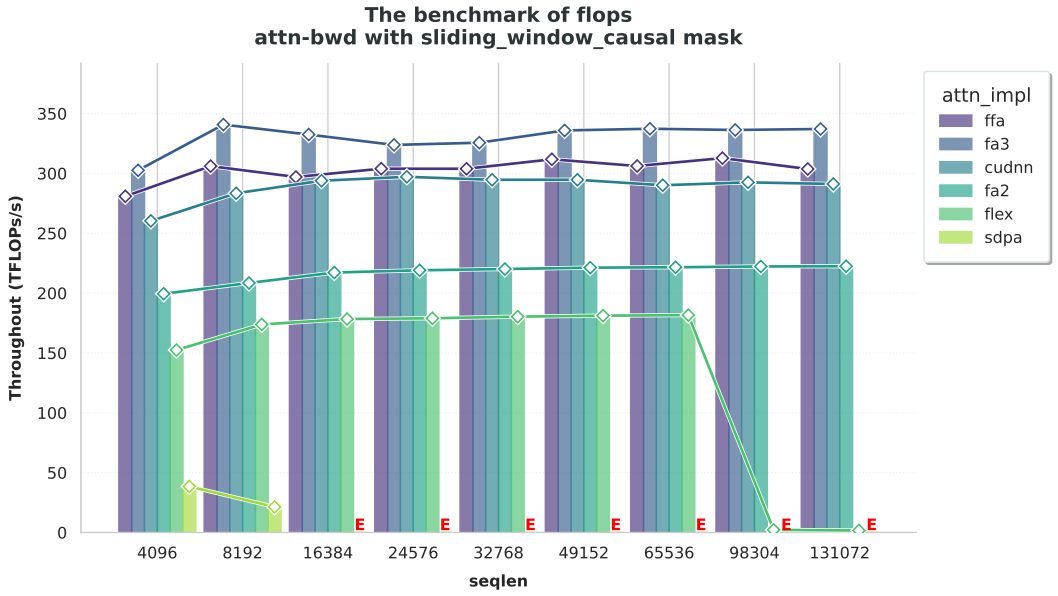


(b) Backward pass for varlen causal mask

Figure 30: Benchmarking FFA's performance and flexibility against other leading attention kernels for varlen causal mask scenarios. (Note that: the **E** symbol indicates the corresponding distributed attention implementation raises *Cuda Out of Memory* error in that specific configuration.)



(a) Forward pass for sliding-window causal mask.



(b) Backward pass for sliding-window causal mask.

Figure 31: Benchmarking FFA's performance and flexibility against other leading attention kernels for sliding-window causal mask scenarios. (Note that: the **E** symbol indicates the corresponding distributed attention implementation raises *Cuda Out of Memory* error in that specific configuration.)

B.2.2 Benchmarking MagiAttention module-level scalability

To validate the scalability of MagiAttention, we assess the per-GPU throughput (in TFLOPs/s/GPU) of the attention module during both forward and backward propagation, as the sequence length and parallel size increase. This assessment is compared against common CP strategies including Ring-Attention and Ulysses (Liu et al., 2023; Jacobs et al., 2023). Due to the complexity of supporting irregular masks for baselines, our experiments are limited to the full mask and varlen full mask scenarios. And the distribution of variable sequence lengths still follow the one in kernel-level experiments B.2.1. The results are presented in Fig.32 and Fig.33.

Our experiments are conducted on a large-scale productive GPU cluster [12]. We jointly scale the total sequence length *seqlen*, the context-parallel size *cp_size*, and the node size *nnodes* from (seqlen:64k, cp_size:1, nnodes:1) up to (seqlen:3072k (3M), cp_size:48, nnodes:48). The tensor-parallel size *tp_size* is kept constant at 8, with sequence-parallel enabled. Other data and model configurations for different mask types are consistent with those detailed in Tab.11.

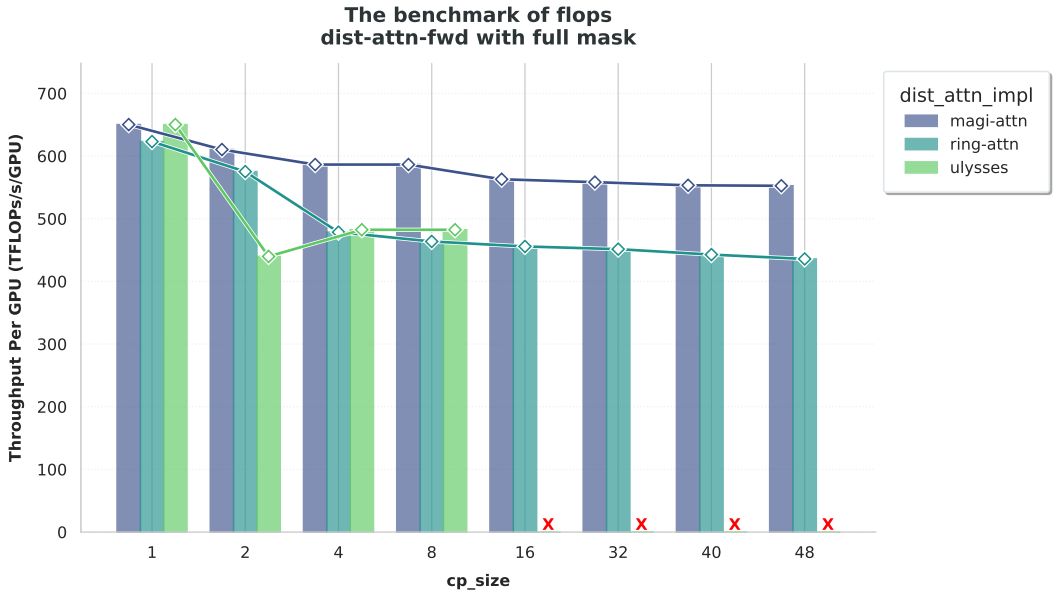
Therefore, in every training setting, each rank is assigned constantly with $seqlen = 64k$, $num_heads_q = 8$ and $num_heads_k = 1$ for attention propagation, while the remaining activations stays $seqlen = 8k$, $num_heads_q = 64$ and $num_heads_k = 8$ with SP enabled. This setup simulates a common training configuration.

To calculate the TFLOPs/s/GPU for various mask patterns during both forward and backward passes, similarly, we first calculate the `FLOPs` as Eq.12 and Eq.13 and apply the following equation:

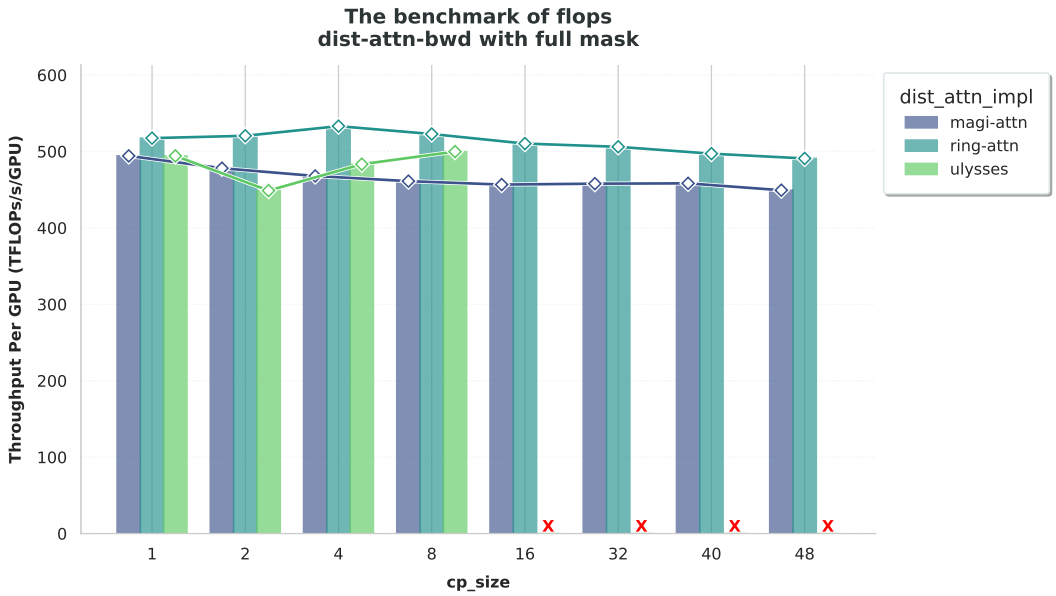
$$\text{TFLOPs/s/GPU}^{(wd)} = \frac{\text{FLOPs}^{(wd)}}{\text{Runtime}^{(wd)} \times cp_size}, \quad wd \in \{fwd, bwd\} \tag{15}$$

As demonstrated, MagiAttention exhibits linear scalability as the context length and CP size increase, in both full mask and varlen full mask configurations, for both forward and backward passes. In contrast, baseline methods either face strict limitations in scaling up or experience performance degradation with ultra-long contexts, which worsens with varlen mask patterns.

[12] Due to business and confidentiality reasons, specific details about the productive cluster, such as the number and type of GPUs, are withheld.

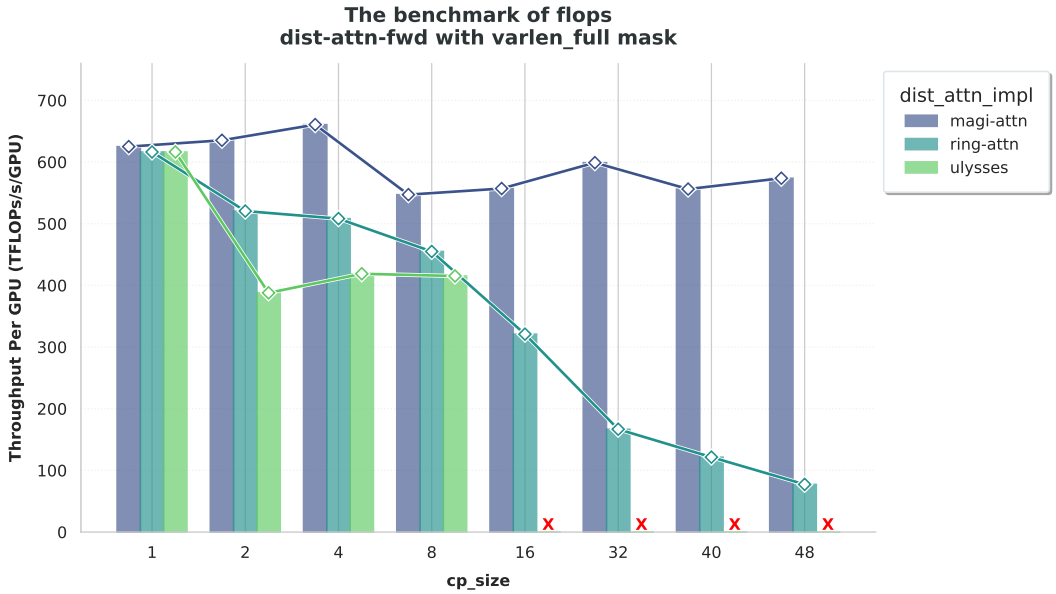


(a) Forward pass for full mask.

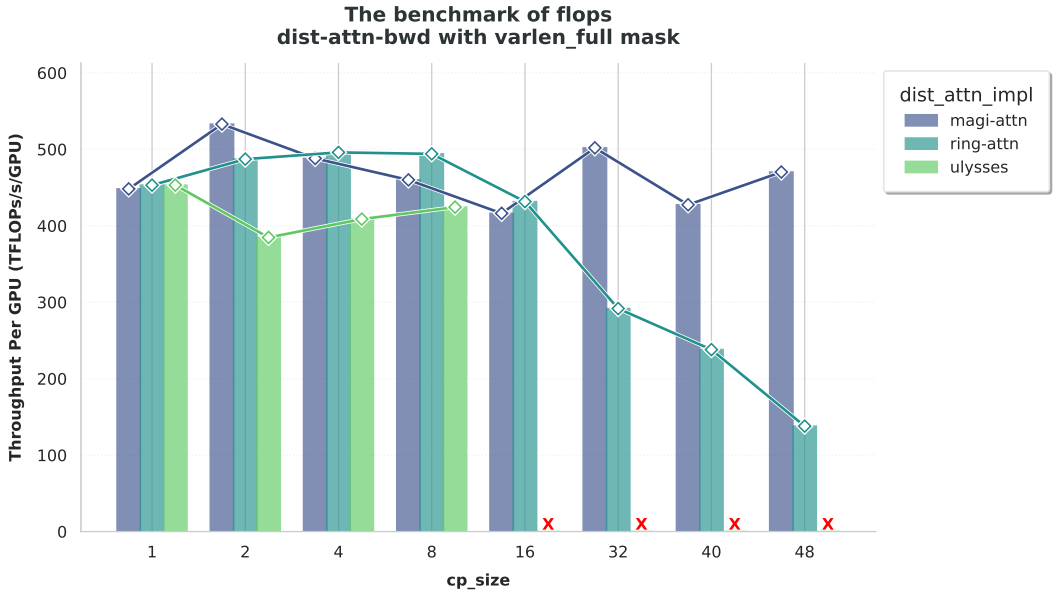


(b) Backward pass for full mask.

Figure 32: Benchmarking MaiAttention's scalability against other leading CP strategies for full mask scenarios. (Note that: the **X** symbol indicates the corresponding distributed attention implementation is not supported in that specific configuration.)



(a) Forward pass for varlen-full mask.



(b) Backward pass for varlen-full mask.

Figure 33: Benchmarking MaiAttention's scalability against other leading CP strategies for varlen full mask scenarios. (Note that: the **X** symbol indicates the corresponding distributed attention implementation is not supported in that specific configuration.)

B.3 Other Materials

Table 12: Examples of DTensor's parallel placements and propagation common in modern training workflows w.r.t. 4.1.3

Basic Placements	Specific Placements	Examples
Replicate	Replicate $^{(\mathrm{R})}$	LayerNorm w/o SP: $y^{(\mathrm{R})} = \frac{x^{(\mathrm{R})} - \mathrm{E}[x^{(\mathrm{R})}]}{\sqrt{\mathrm{Var}[x^{(\mathrm{R})}] + \epsilon}} * \gamma^{(\mathrm{R})} + \beta^{(\mathrm{R})}$
Partial	SumPartial $^{(\mathrm{P}_{sum})}$	RowLinear w/o SP: $O^{(\mathrm{R})} = \mathrm{AllReduce}((X^{(\mathrm{S}(1))} \times W^{(\mathrm{S}(0))})^{(\mathrm{P}_{sum})})$ ZERO-1 Grad Sync: $grad^{(\mathrm{R})} = \mathrm{AllReduce}(grad^{(\mathrm{P}_{sum})})$ FSDP Grad Sync: $grad^{(\mathrm{S}(0))} = \mathrm{ReduceScatter}(grad^{(\mathrm{P}_{sum})})$ MSE Loss Sync: $loss^{(\mathrm{R})} = \mathrm{AllReduce}(loss^{(\mathrm{P}_{sum})})$
	AvgPartial $^{(\mathrm{P}_{avg})}$	Parallel RMSNorm: $y^{(\mathrm{R})} = \frac{x^{(\mathrm{S}(1))}}{\mathrm{AllReduce}(\mathrm{RMS}(x^{(\mathrm{S}(1))})^{(\mathrm{P}_{avg})})} * \gamma^{(\mathrm{S}(1))}$
	MaxPartial $^{(\mathrm{P}_{max})}$	Grad L_∞-Norm : $grad_norm^{(\mathrm{R})} = \mathrm{AllReduce}(grad_norm^{(\mathrm{P}_{max})})$
	MinPartial $^{(\mathrm{P}_{min})}$	FP8 Scaling: $\tilde{x}^{(\mathrm{R})} = x^{(\mathrm{R})} \times \mathrm{AllReduce}((scaling_factor^{-1})^{(\mathrm{P}_{min})})$
	NormPartial $^{(\mathrm{P}_{norm})}$	Grad L_p-Norm : $grad_norm^{(\mathrm{R})} = \mathrm{AllReduce}(\mathrm{pow}(grad_norm^{(\mathrm{P}_{norm})}, p)^{(\mathrm{P}_{sum})})^{\frac{1}{p}}$
	LSEPartial $^{(\mathrm{P}_{lse})}$	LSE Correction : $lse^{(\mathrm{R})} = \log(\mathrm{AllReduce}(\exp(lse^{(\mathrm{P}_{lse})})^{(\mathrm{P}_{sum})}))$
	AttnPartial $^{(\mathrm{P}_{attn})}$	Attention Forward Correction: $o^{(\mathrm{R})} = \mathrm{AllReduce}((\exp(lse^{(\mathrm{P}_{lse})} - lse^{(\mathrm{R})}) \cdot o^{(\mathrm{P}_{attn})})^{(\mathrm{P}_{sum})})$
Shard	(Even)Shard $^{(\mathrm{S}(dim))}$	ColLinear with SP : $O^{(\mathrm{S}(1))} = \mathrm{AllGather}(X^{(\mathrm{S}(0))}) \times W^{(\mathrm{S}(1))}$ LayerNorm with SP: $y^{(\mathrm{S}(0))} = \frac{x^{(\mathrm{S}(0))} - \mathrm{E}[x^{(\mathrm{S}(0))}]}{\sqrt{\mathrm{Var}[x^{(\mathrm{S}(0))}] + \epsilon}} * \gamma^{(\mathrm{R})} + \beta^{(\mathrm{R})}$
	(Even)StridedShard $^{(\mathrm{SS}(dim))}$	FSDP + ColLinear w/o SP : $O^{(\mathrm{S}(1))} = X^{(\mathrm{R})} \times \mathrm{AllGather}(W^{(\mathrm{SS}(0),S(1))})^{(\mathrm{S}(1))}$ MHA QKV-fused ColLinear w/o SP : $Q^{(\mathrm{S}(1))}, K^{(\mathrm{S}(1))}, V^{(\mathrm{S}(1))} = \mathrm{split}(QKV^{(\mathrm{SS}(1))})$, $QKV^{(\mathrm{SS}(1))} = X^{(\mathrm{R})} \times W_{qkv}^{(\mathrm{SS}(1))}$
	UnevenStridedShard $^{(\mathrm{USS}(dim))}$	GQA QKV-fused ColLinear w/o SP : $Q^{(\mathrm{S}(1))}, K^{(\mathrm{S}(1))}, V^{(\mathrm{S}(1))} = \mathrm{split}(QKV^{(\mathrm{USS}(1))})$, $QKV^{(\mathrm{USS}(1))} = X^{(\mathrm{R})} \times W_{qkv}^{(\mathrm{USS}(1))}$

Note: the specific placements with the **bold text** are the ones we've complemented.