Cosmos World Foundation Model Platform for Physical AI

NVIDIA[1]

Abstract

Physical AI needs to be trained digitally first. It needs a digital twin of itself, the policy model, and a digital twin of the world, the world model. In this paper, we present the Cosmos World Foundation Model Platform to help developers build customized world models for their Physical AI setups. We position a world foundation model as a general-purpose world model that can be fine-tuned into customized world models for downstream applications. Our platform covers a video curation pipeline, pre-trained world foundation models, examples of post-training of pre-trained world foundation models, and video tokenizers. To help Physical AI builders solve the most critical problems of our society, we make our platform open-source and our models open-weight with permissive licenses available via NVIDIA Cosmos.

1. Introduction

Physical AI is an AI system equipped with sensors and actuators: the sensors allow it to observe the world, and the actuators allow it to interact with and modify the world. It holds the promise of freeing human workers from physical tasks that are dangerous, laborious, or tedious. While several fields of AI have advanced significantly thanks to data and compute scaling in the recent decade, Physical AI only inches forward. This is largely because scaling training data for Physical AI is much more challenging, as the desired data must contain sequences of interleaved observations and actions. These actions perturb the physical world and may cause severe damage to the system and the world. This is especially true when the AI is still in its infancy when exploratory actions are essential. A World Foundation Model (WFM), a digital twin of the physical world that a Physical AI can safely interact with, has been a long-sought remedy to the data scaling problem.

In this paper, we introduce the Cosmos World Foundation Model (WFM) Platform for building Physical AI. We are mainly concerned with the visual world foundation model, where the observations are presented as videos, and the perturbations can exist in various forms. As illustrated in Fig. 2, we present a pre-training-and-then-post-training paradigm, where we divide WFMs into pre-trained and post-trained WFMs. To build a pre-trained WFM, we leverage a large-scale video training dataset to expose the model to a diverse set of visual experiences so it can become a generalist. To build a post-trained WFM, we fine-tune the pre-trained WFM to arrive at a specialized WFM using a dataset collected from a particular Physical AI environment for the targeted, specialized Physical AI setup. Fig. 1 shows example results from our pre-trained and post-trained WFMs.

Data determines the ceiling of an AI model. To build a high-ceiling pre-trained WFM, we develop a video data curation pipeline. We use it to locate portions of videos with rich dynamics and high visual quality that facilitate learning of physics encoded in visual content. We use the pipeline to extract about 100M clips of videos ranging from 2 to 60 seconds from a 20M hour-long video collection. For each clip, we use a visual language model (VLM) to provide a video caption per 256 frames. Video processing is computationally intensive. We leverage hardware implementations of the H.264 video encoder and decoder available in modern GPUs for decoding and transcoding. Our video data curation pipeline leverages many pre-trained image/video understanding models. These models have different throughputs. To maximize the overall throughput for generating trainable video data, we build a Ray-based orchestration pipeline (Moritz et al., 2017). The details are described in Sec. 3.

We explore two scalable approaches for building pre-trained WFMs discussed in Sec. 5. These approaches are

[1]A detailed list of contributors and acknowledgments can be found in App. A of this paper.

© 2025 NVIDIA. All rights reserved.

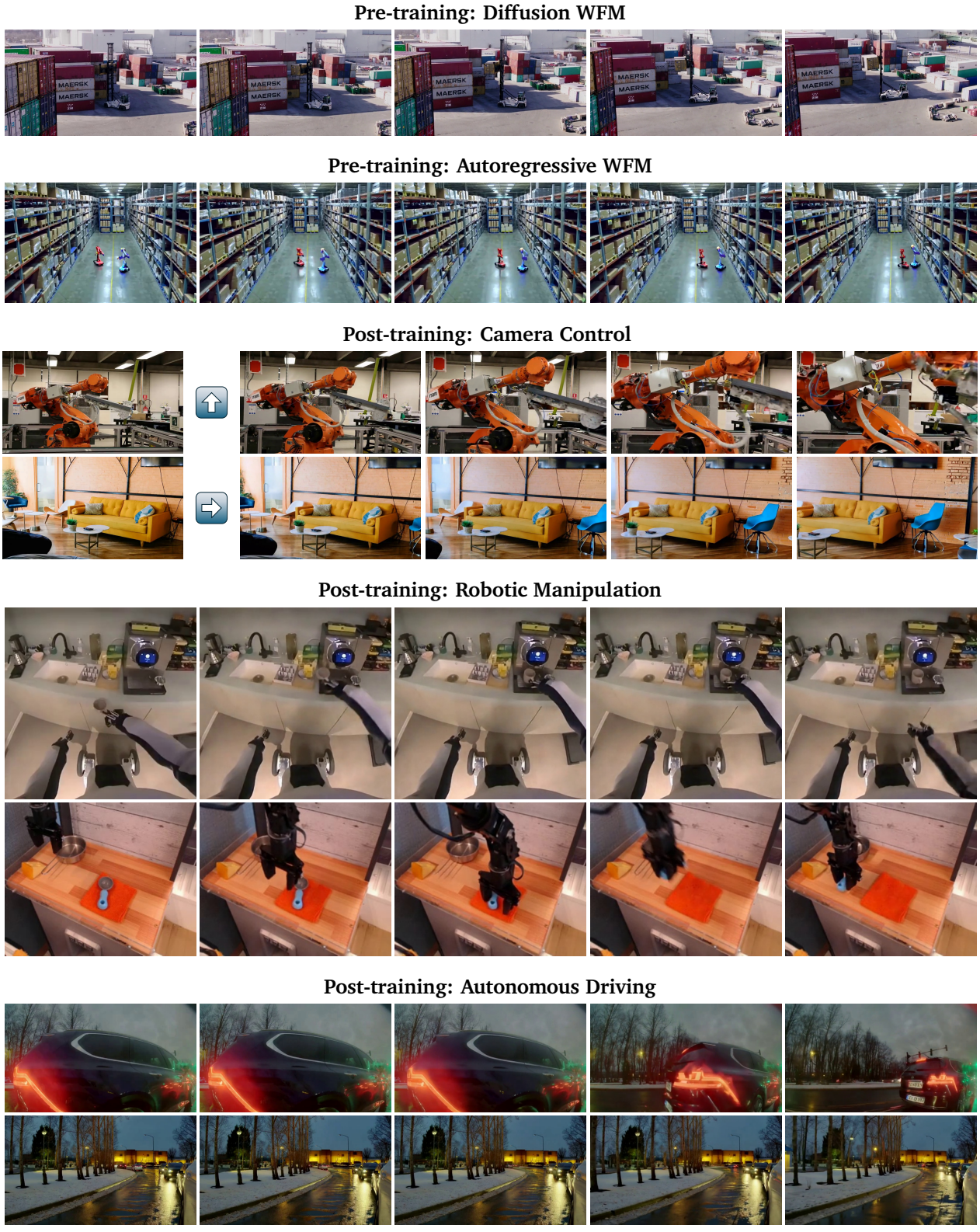


Figure 1: **Cosmos World Foundation Models**. Pre-trained Cosmos WFMs generate high-quality 3D consistent videos with accurate physics. The Cosmos suite of models includes both diffusion and autoregressive transformer models, which are trained using continuous and discrete latent representations of videos, respectively. Post-training these WFMs with specialized datasets enables them to be utilized in a wide range of Physical AI setups. Specifically, we present models with camera controllability, models capable of instruction-following for robotic manipulation, and models for autonomous driving scenarios. To check full videos and more video examples, please visit our website.

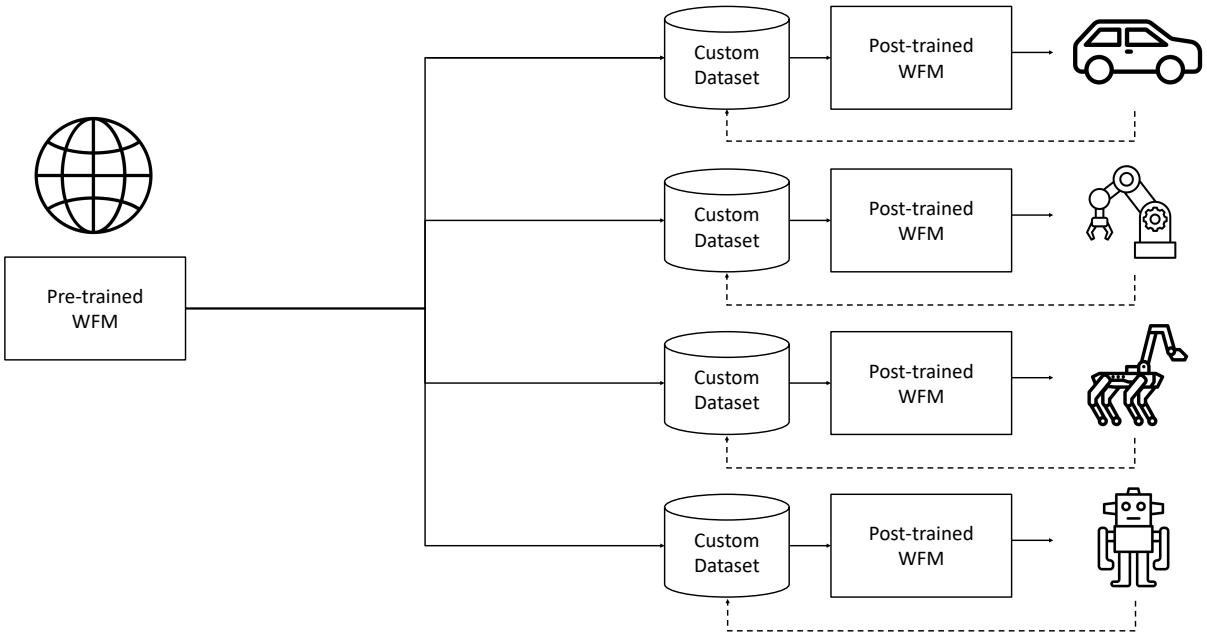


Figure 2: Pre-trained WFMs are world model generalists that are trained with large-scale, diverse video datasets capturing different aspects of real-world physics. These pre-trained world foundation models can be specialized to a target Physical AI setup through post-training. Usually, the datasets for post-training are "prompt"-video pairs collected from the target Physical AI setup. The prompt can be in the form of action commands, trajectory, instructions, *etc*. As the pre-trained WFM provides a great foundation, the dataset for post-training can be much smaller. This pre-training-and-post-training yields an efficient strategy for building a Physical AI system. In the figure, the dashed lines represent the data loop.

transformer-based diffusion models and transformer-based autoregressive models. A diffusion model generates videos by gradually removing noise from a Gaussian noise video. An autoregressive model generates videos piece by piece, conditioned on the past generations following a preset order. Both approaches decompose a difficult video generation problem into easier sub-problems, making it more tractable. We leverage state-of-the-art transformer architectures for their scalability. In Sec. 5.1, we present a transformer-based diffusion model design that exhibits strong world-generation capabilities. In Sec. 5.2, we present a transformer-based autoregressive model design for world generation.

Both the transformer-based diffusion model and transformer-based autoregressive model use tokens as representations of videos, where the former uses continuous tokens in the form of vectors, and the latter uses discrete tokens in the form of integers. We note that tokenization for videos—a process that transforms videos into a set of tokens—is highly nontrivial. Video contains rich information about the visual world. However, to facilitate learning of the WFMs, we need to compress videos into sequences of compact tokens while maximally preserving the original contents in the videos as the computation complexity of world foundation model training grows with the token counts. In many ways, building a video tokenizer is similar to building a video codec. We develop an attention-based encoder-decoder architecture to learn video tokenization for both continuous and discrete tokens described in Sec. 4.

We fine-tune the pre-trained WFMs to arrive at post-trained WFMs for various Physical AI tasks in Sec. 6. In Sec. 6.1, we fine-tune our pre-trained diffusion WFM to make it camera pose conditional. This post-training creates a navigable virtual world where users can explore the created world by moving the virtual viewpoint around. In Sec. 6.2, we fine-tune our WFMs on various robotic tasks, which consist of video-action sequences. We show that by leveraging the pre-trained WFMs, we can better predict the future state of the world based on

the action taken by the robot. In Sec. 6.3, we demonstrate how the pre-trained WFMs can be fine-tuned for various autonomous driving-related tasks.

Our intended use of the developed WFMs is for Physical AI builders. To better protect the developers when using the world foundation models, we develop a powerful guardrail system that consists of a pre-Guard to block harmful inputs and a post-Guard to block harmful outputs. The details are described in Sec. 7.

We aim to build a world foundation model platform to help Physical AI builders advance their systems. To achieve this goal, we make our pre-trained world foundation models and tokenizers available under the NVIDIA Open Model License at NVIDIA Cosmos and NVIDIA Cosmos Tokenizer respectively. The pre-training script and post-training script will be available at NVIDIA Nemo Framework together with the video data curation pipeline to help builders craft their fine-tuning datasets. While this paper makes several improvements in world foundation model design, the world foundation model problem is still far from being solved. Additional research is required to advance the state-of-the-art further.

2. World Foundation Model Platform

Let $x_{0:t}$ be a sequence of visual observations of the real world from time 0 to t. Let c_t be the perturbation to the world. As illustrated in Fig. 3, a WFM is a model $\mathcal{W}$ that predicts the future observation at time $t+1$, $\hat{x}_{t+1}$, based on the past observation $x_{0:t}$ and the current perturbation c_t. In our case, $x_{0:t}$ is an RGB video, while c_t is a perturbation that can take many forms. It can be an action taken by the Physical AI, a random perturbation, a text description of the perturbation, etc.



Figure 3: A world foundation model (WFM) $\mathcal{W}$ is a model that generates the future state of the world x_{t+1} based on the past observations $x_{0:t}$ and current perturbation c_t.

2.1. Future Cosmos

We believe a WFM is useful to Physical AI builders in many ways, including (but not limited to)

- **Policy evaluation.** This refers to evaluating the quality of a policy model in a Physical AI system. Instead of evaluating a trained policy by deploying it to a Physical AI system operating in the real world, one could instead let the digital copy of the Physical AI system interact with the world foundation model. The WFM-based evaluation is more cost-effective and time-efficient. With the WFM, builders can deploy the policy model in unseen environments that are otherwise unavailable. WFMs can help developers rule out incapable policies quickly and focus the physical resources on a few promising ones.
- **Policy initialization.** A policy model generates actions to be taken by the Physical AI system based on the current observations and the given task. A well-trained WFM, which models the dynamic patterns of the world based on the input perturbations, can serve as a good initialization of the policy model. This helps address the data scarcity problem in Physical AI.
- **Policy training.** A WFM paired with a reward model can be a proxy for the physical world to provide feedback to the policy model in a reinforcement learning setup. The agent can gain proficiency in solving tasks by interacting with the WFM.
- **Planning or model-predictive control.** A WFM can be used to simulate different future states following different action sequences taken by a Physical AI system. A cost/reward module can then be used to quantify the performance of these different action sequences based on the outcomes. The Physical AI can then execute the best action sequence based on the simulation results as a whole, as in planning

algorithms or in a receding horizon manner, as in model-predictive control. The accuracy of the world model upper-bounds the performance of these decision-making strategies.

- **Synthetic data generation.** A WFM can be used to generate synthetic data for training. It can also be fine-tuned to be conditioned on rendering metadata such as depth or semantic maps. One can use the conditional WFM for the Sim2Real use case.

While we list the possibilities, this paper does not include empirical results in applying Cosmos WFMs to them. We are eager to verify the claims in future work.

2.2. Current Cosmos

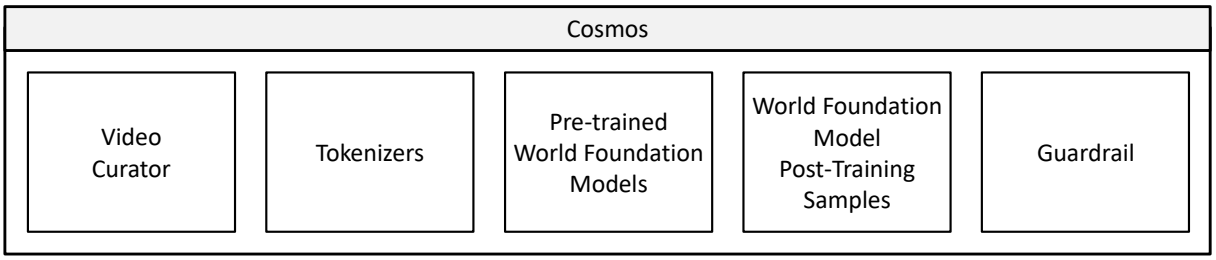


Figure 4: Cosmos World Foundation Model Platform consists of several major components: video curator, video tokenizer, pre-trained world foundation model, world foundation model post-training samples, and guardrail.

Fig. 4 visualizes what is available in the Cosmos WFM platform that is included in this paper, which includes video curator, video tokenization, world foundation model pre-training, world foundation model post-training, and guardrail.

Video curator. We develop a scalable video data curation pipeline. Each video is split into individual shots without scene changes. A sequence of filtering steps is then applied to the clips to locate high-quality and dynamic information-rich subsets for training. These high-quality shots are then annotated using a VLM. We then perform semantic de-duplication to construct a diverse but compact dataset.

Video tokenization. We develop a family of video tokenizers of different compression ratios. These tokenizers are causal. The token computation for the current frames is not based on future observation. This causal design has several benefits. On the training side, it makes joint image and video training possible since a causal video tokenizer is also an image tokenizer when the input is a single image. This is important for the video model to leverage image datasets for training, which contain rich appearance information of the worlds and tend to be more diverse. On the application side, causal video tokenizers are better aligned with Physical AI systems that live in the causal world.

WFM pre-training. We explore two scalable approaches for building pre-trained world foundation models—the diffusion model and the autoregressive model. We use the transformer architecture for its scalability.

For the diffusion-based WFM, the pre-training consists of two steps: 1) Text2World generation pre-training and 2) Video2World generation pre-training. Specifically, we train the model to generate a video world based on the input text prompt. We then fine-tune it to generate a future video world based on the past video and an input text prompt, which we refer to as the Video2World generation task.

For the autoregressive-based WFM, the pre-training consists of two steps: 1) vanilla next token generation and 2) text-conditioned Video2World generation. We first train the model to generate a future video world based on the input of past video—foresight generation. We then fine-tune it to generate a future video world based on the past video and a text prompt.

The video2world generation model is a pre-trained world model that generates the future based on the current

observation (the past video) and control input (prompt). For both diffusion-based and autoregressive-based WFMs, we build a family of models with different capacities and study their effectiveness on various downstream applications.

We further fine-tune our pre-trained diffusion WFM to arrive at a diffusion decoder to enhance the generation results of the autoregressive model. To better control the WFM, we also built a prompt upsampler based on a Large Language Model (LLM).

World model post-training. We show applications of the pre-trained WFMs on several downstream Physical AI applications. We fine-tune a pre-trained WFM with the camera pose as the input prompt. This allows us to navigate freely in the created world. We also demonstrate how our pre-trained WFMs might be fine-tuned for humanoid and autonomous driving tasks.

Guardrail. For safe usage of the developed world foundation models, we develop a guardrail system where harmful inputs and outputs are blocked.

3. Data Curation

We describe our video curation pipeline, which produces high-quality training datasets for both tokenizers and WFMs. As shown in Fig. 5, our pipeline consists of 5 main steps: 1) splitting, 2) filtering, 3) annotation, 4) deduplication, and 5) sharding. Every step is tailored to improve the data quality and accommodate the requirements of model training. We first present our raw dataset and then describe each step in detail.

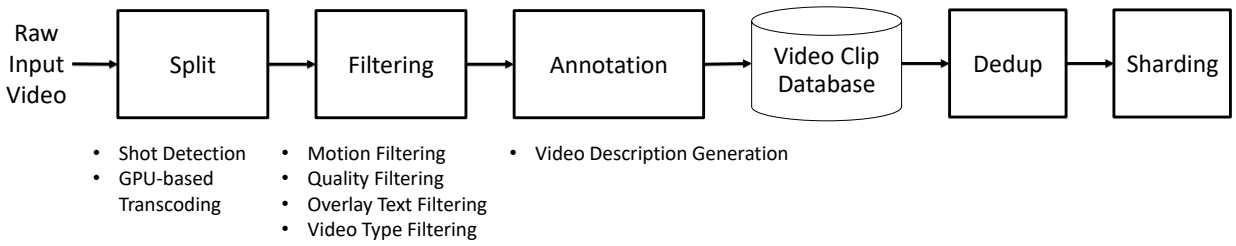


Figure 5: **Cosmos Video Curator contains five major steps: 1) split, 2) filtering, 3) annotation, 4) dedup, and 5) sharding**. The split step divides a long video into shots and transcribes them into clips. The filtering step removes clips that are of little value to world foundation model building. The annotation step adds a video description to each clip. The clips are then stored in a video clip database. To obtain the training dataset, one first performs semantic deduplication and then shards the video clips based on their resolutions and aspect ratios.

3.1. Dataset

We use both proprietary video datasets and publicly available open-domain Internet videos to train our models. Our goal is to enable Physical AI developers. To this end, we curate the video training dataset to cover various Physical AI applications and target the following video categories:

1. Driving (11%),
2. Hand motion and object manipulation (16%),
3. Human motion and activity (10%),
4. Spatial awareness and navigation (16%),
5. First person point-of-view (8%),
6. Nature dynamics (20%),
7. Dynamic camera movements (8%),
8. Synthetically rendered (4%), and
9. Others (7%).

These videos offer a broad coverage of different visual objects and actions. Their diversity improves the generalization of our WFMs and helps the models handle different downstream tasks. The unstructured nature of these videos and their sheer volume creates many challenges to processing them efficiently from both an algorithmic and an infrastructural perspective. The videos can be encoded with a wide variety of codecs and have different aspect ratios, resolutions, lengths, *etc*. Many videos have also been post-processed or edited with different visual effects, which may induce unwanted artifacts in the generated videos and hurt the performance of the world models if not appropriately handled.

In total, we accumulate about 20M hours of raw videos with resolutions from 720p to 4k. However, a significant amount of the video data is either semantically redundant or does not contain useful information for learning the physics of the world. Hence, we design a sequence of data processing steps to find the most valuable parts of the raw videos for training. We also collect image data as joint-image-and-video training has been shown to improve the visual quality of the generated videos and accelerate the model training. Thanks to the modular design of our data curation pipeline, we can use it to process both image and video data and generate datasets for both pre-training and fine-tuning. We generate about 10^8 video clips for pre-training and about 10^7 for fine-tuning.

3.2. Splitting

Our videos have arbitrary lengths, and modern deep-learning models cannot directly consume very long videos. Also, many videos contain shot transitions. They can start from one scene and then transition to a different scene where the two scenes can be disconnected entirely, *e.g.*, from two people talking in a modern kitchen in New York City to a scene of lions chasing zebra in an African savanna. It is important to segment each video based on its shot changes and generate visually consistent video clips so that the model can learn visual content transitions that are physically plausible instead of artificially edited.

3.2.1. Shot Detection

Splitting aims to temporally segment raw videos of arbitrary lengths into clips without shot changes. It takes the raw videos as input and generates each shot's start and end frame indices. Clips shorter than 2s are discarded, as they could be shot transitions or visual effects. Clips longer than 60s are further split to have a maximal length of 60s. The subsequent filtering steps can then determine whether a clip contains useful information for learning the physics of the world.

Shot boundary detection is a classical computer vision problem. Existing methods detect shot boundaries based on changes in the visual feature space, but they differ in how to learn visual features from video frames. We evaluate several algorithms for the task in Tab. 1: PySceneDetect (Castellano, 2024), Panda70M (Chen et al., 2024), TransNetV2 (Soucek and Lokoc, 2024), and AutoShot (Zhu et al., 2023).

PySceneDetect is a popular library that detects shot changes by thresholding the temporal change of color histogram in HSV space. Note that it is also adopted by the recent MovieGen work (Polyak et al., 2024). Panda70M augments PySceneDetect with CLIP-embedding-based stitching and filtering. TransNetV2 and AutoShot, on the other hand, are neural network-based, predicting a probability of each frame being a transition frame given a 100-frame rolling input window.

It is critical to select an algorithm that can handle heavily edited videos well, as they often have complex shot changes compounded with various visual effects. This motivates us to build a dedicated benchmark to evaluate whether the method can generate clips with clean shot cuts from videos. Our benchmark (named ShotBench[2]) includes existing datasets, such as RAI, BBC Planet Earth (AI Image Lab, University of Modena, 2016), ClipShots (Tang et al., 2018) and SHOT (Zhu et al., 2023). For ClipShots, we define the transition frame as the midpoint of the start and end of each shot annotation to be consistent with other datasets.

[2]ShotBench is available at https://github.com/NVlabs/ShotBench.

Table 1: Comparison of splitting algorithms on different datasets.

Dataset	Metric	PySceneDetect	Panda70M	TransNetV2	AutoShot
BBC	Precision ↑	0.894	0.959	0.983	**0.984**
	Recall ↑	0.884	0.653	**0.951**	0.922
	F1 ↑	0.889	0.777	**0.967**	0.952
RAI	Precision ↑	0.856	**0.933**	0.918	0.889
	Recall ↑	0.807	0.746	0.921	**0.923**
	F1 ↑	0.831	0.829	**0.919**	0.906
SHOT	Precision ↑	0.769	**0.949**	0.883	0.866
	Recall ↑	0.673	0.462	0.767	**0.804**
	F1 ↑	0.718	0.622	0.821	**0.834**
ClipShots	Precision ↑	0.395	0.649	**0.685**	0.653
	Recall ↑	0.602	0.424	0.772	**0.781**
	F1 ↑	0.477	0.513	**0.726**	0.711

Tab. 1 compares the different methods on ShotBench. We set the confidence threshold to 0.4 for both TransNetV2 and AutoShot. For Panda70M, we follow their implementation for splitting, excluding the filtering steps, for a fair comparison. End-to-end learning-based approaches (*e.g.*, TransNetV2 and AutoShot) perform much better than methods using hand-crafted features or heuristic rules (*e.g.*, PySceneDetect and Panda70M). Though TransNetV2 and AutoShot perform comparably on existing datasets, we found TransNetV2 works better on more challenging shot changes. Using an end-to-end neural network (*i.e.*, TransNetV2) also allows us to increase the throughput of splitting by leveraging modern GPUs for acceleration without the hurdle of hybrid approaches (such as Panda70M) that use complicated logic to combine PySceneDetect and ImageBind embeddings (Girdhar et al., 2023).

3.2.2. Transcoding

Our videos use many different codecs with various settings, which poses challenges to data curation. We re-encode each video clip from shot detection into a consistent, high-quality mp4 format. This simplifies the subsequent data curation process. With a unified video codec, the stability and efficiency of our dataloader for model training are also greatly improved. We use the h264_nvenc codec with a high bitrate and stress test our setting using videos with fast motion and high-frequency texture to ensure no perceptible visual degradation.

We thoroughly evaluate different hardware and software configurations for transcoding to maximize the throughput in Tab. 2. Modern GPUs provide hardware-accelerated video encoding and decoding capabilities. NVIDIA L40S has hardware accelerators for both decoding (NVDEC) and encoding (NVENC), whereas NVIDIA H100 only has NVDEC. We compensate H100 with the maximum available CPU cores (28 instead of 1) for a fair comparison with L40S in Tab. 2. L40S has about 17% higher throughput than H100 (0.0674 *vs*. 0.0574). For software configurations, switching from libx264 to h264_nvenc and transcoding multiple clips from the same video in batches significantly boost the throughput. We observe issues with ffmpeg fully utilizing NVDEC/NVENC accelerators, especially on multi-GPU nodes. Replacing ffmpeg with PyNvideoCodec for video stream transcoding leads to much higher accelerator utilization and the biggest throughput improvement (0.3702 *vs*. 0.1026). We only keep ffmpeg for audio remixing and use PyNvideoCodec to better leverage the computing power in the GPUs. We achieve a $\sim 6.5\times$ increase in throughput when combining all the improvements together.

3.3. Filtering

The video clips produced from the splitting step are noisy, with vastly different qualities covering various topics. We design the filtering step to 1) remove video clips whose visual quality fails to meet our minimal requirements, 2) select high-quality video clips suitable for fine-tuning, and 3) tailor the data distribution for building WFMs. We achieve the above goal by doing motion filtering, visual quality filtering, text filtering, and

Table 2: Transcoding performance with different software settings.

Method	GPU	CPU	Codec	Batch	NVDEC (#accelerator)	NVENC (#accelerator)	Throughput (videos/s)
ffmpeg	H100	28	libx264	1	7	0	0.0574
ffmpeg	L40S	1	h264_nvenc	1	3	3	0.0674
ffmpeg	L40S	1	h264_nvenc	16	3	3	0.1026
pynvc+ffmpeg	L40S	1	h264_nvenc	1	3	3	**0.3702**

video type filtering.

3.3.1. Motion Filtering

We have two main goals in motion filtering: 1) remove videos that are static or with random abrupt camera motion (usually from hand-held cameras) and 2) tag videos with different types of camera motion (*e.g.*, pan, zoom, tilt, *etc.*), which can provide additional information to guide model training.

We build a lightweight classifier for motion filtering. The input to the classifier is a sequence of motion vectors or optical flow extracted from a video clip. The classifier is based on the ViT architecture and is trained with labeled videos. We experiment with motion vectors from h264 codec, the Farneback optical flow algorithm(Farnebäck, 2003), and an NVIDIA TensorRT-accelerated optical flow estimation network. We find that the classifier built on top of the NVIDIA TensorRT-accelerated optical flow estimation works the best, producing high classification accuracy for motion filtering.

3.3.2. Visual Quality Filtering

We consider two criteria, distortion and appearance quality, for visual quality-based filtering. First, we remove video clips with distortions, such as artifacts, noise, blur, low sharpness, overexposure, underexposure, *etc*. We use a video quality assessment model trained on human-rated videos based on DOVER (Wu et al., 2023). This gives a perceptual quality score per clip, and we use the scores to remove clips that are in the bottom 15%. Second, we filter out video clips with low appearance quality. We apply an image aesthetic model (Schuhmann, 2022) on sampled frames from an input clip. We set a conservative threshold, *i.e.*, 3.5, since aesthetics are less important for Physical AI.

3.3.3. Text Overlay Filtering

Some of our videos are post-processed to add text to include additional information for the viewer. We also find that text tends to co-occur with different visual effects. Our goal is to learn the physics of the world. It is crucial to remove videos with such excessive text. Note that we focus on text added in post-processing instead of text in the original scene from which the video is created, such as the street names in driving videos.

We train an MLP-based binary classifier to detect such videos. The input to the classifier is a video embedding extracted using InternVideo2 (Wang et al., 2025). We use a proprietary VLM to build the training set to label positive and negative videos. Our trained model achieves high prediction accuracy in the validation set.

3.3.4. Video Type Filtering

To adjust the training data distribution and filter out unwanted video types, we design a comprehensive taxonomy that categorizes videos based on their content type and visual style. We train a classifier to label each video clip with categories from the taxonomy. We refine our data by excluding specific video types that could lead to poor generation quality or unrealistic dynamics, such as abstract visual patterns, video game footage, animated content, *etc*. We further adjust the data distribution by upsampling from categories that are more relevant to WFMs (*e.g.*, human action, human and object interaction, *etc.*) and downsampling on categories that are less important (*e.g.*, nature or landscape videos).

Given the absence of pre-existing labeled datasets matching our taxonomy, we leverage a proprietary VLM to create training and evaluation data for the classifier. For each video clip, we prompt the VLM with eight

uniformly sampled frames and query for the most appropriate taxonomy label. Using the annotated data, we train an MLP classifier on the same InternVideo2 embeddings from text filtering.

3.4. Annotation

Text descriptions are usually paired with image and video data to provide supervision and conditions for world model training. We use a VLM to generate high-quality and consistent captions for each video clip. We configure the VLM in a way such that it focuses on the material facts and details in the videos. Using this approach to provide descriptions of videos instead of relying on Alt text also eases the burden of learning for world models as we do not need to adapt to different text styles or formats during training.

We test several SOTA methods (*i.e.*, VFC (Ge et al., 2024), Qwen2-VL (Wang et al., 2024), VILA (Lin et al., 2024; Xue et al., 2024)) for caption generation on our videos, and find VILA generates more accurate descriptions based on a small-scale human evaluation. We use an internal VILA model with 13B parameters, fine-tuned for video captioning. It has an enlarged context window suitable for processing long, multi-frame contexts, with a max input and output token length of 5904 and 256, respectively. To improve the inference efficiency, we use an FP8-quantized TensorRT-LLM engine, resulting in a 10 $\times$ speed-up in throughput compared to a PyTorch half-precision baseline, as shown in Tab. 3. We prompt VILA with "*Elaborate on the visual and narrative elements of the video in detail*" and feed it 8 uniformly sampled frames from the input clip. The average length of captions is 559 characters or 97 words.

Table 3: Inference throughput comparison of VILA on a single H100 GPU.

Engine	Precision	Batch Size	Throughput (clips/s)	Throughput (tokens/s)
PyTorch	FP16	1	0.21	49.6
TRT-LLM	FP16	1	0.40	95.6
TRT-LLM	FP16	16	1.09	260.9
TRT-LLM	FP8	16	1.96	470.6

3.5. Deduplication

Given the sheer volume of our videos, there could be duplicated or near-duplicated samples in the training set. It is critical to deduplicate the data to create a more balanced and diverse data distribution. It also improves the efficiency of training and reduces the chance of memorizing specific training samples.

We adopt the approach from SemDeDup (Abbas et al., 2023) and DataComp (Gadre et al., 2024) for scalable semantic deduplication. We reuse the InternVideo2 embeddings computed during filtering and cluster the embeddings using a multi-node GPU-accelerated implementation of k-means (RAPIDS, 2023) with $k = 10,000$. We compute the pairwise distances within each cluster of embeddings to identify duplicates. When duplicated videos are detected, we choose the video with the highest resolution to ensure no quality is lost due to deduplication. To avoid storing the entire pairwise distance matrix in GPU memory, we calculate on-the-fly the necessary upper-triangular matrix and argmax reduction in blocks of 256. We remove about 30% of training data during deduplication.

We also leverage the extracted InternVideo2 embeddings and clustering results to build a visual search engine that supports querying the whole training dataset with free-form text and videos. The search engine is useful for debugging issues in our data and understanding the gap between the pre-training dataset and downstream applications.

3.6. Sharding

This step aims to package the processed video clips into webdatasets that our model trainer can directly consume for training. We shard the videos based on their resolution, aspect ratio, and length to align with our

training curriculum. Besides pre-training datasets, we also create fine-tuning datasets with even higher quality by leveraging the different filters described above.

3.7. Infrastructure

Our data processing infrastructure uses AnyScale Ray (Moritz et al., 2017) to implement a streaming pipeline system for geographically distributed clusters, addressing two key challenges in large-scale ML workflows: efficient resource utilization across homogeneous nodes and robust operation over high-latency connections to data sources. By decoupling data transfer from computation, pipelines operate efficiently with remote data storage while maintaining memory requirements that scale with pipeline complexity rather than dataset size, enabling unbounded stream processing.

Our architecture enables concurrent utilization of complementary hardware resources through parallel pipeline stages, for instance, simultaneously using network bandwidth for data ingestion, NVDEC units for video decoding, and GPUs for compute-intensive transformations. We extend the Fragmentation Gradient Descent algorithm (Weng et al., 2023) to optimize this multi-resource allocation, with our scheduler automatically scaling individual stages to maintain balanced throughput across specialized hardware accelerators.

4. Tokenizer

Tokenizers are fundamental building blocks of modern large-scale models. They transform raw data into more efficient representations by learning a bottle-necked latent space discovered in an unsupervised manner. Specifically, visual tokenizers map raw and redundant visual data—such as images and videos—into compact semantic tokens, making them crucial for handling high-dimensional visual data. This ability not only enables efficient training of large-scale transformer models but also democratizes their inference on limited computational resources. Fig. 6 schematically illustrates the tokenization training pipeline where the goal is to train the encoder and decoder so that the bottleneck token representation maximally preserves visual information in the input.

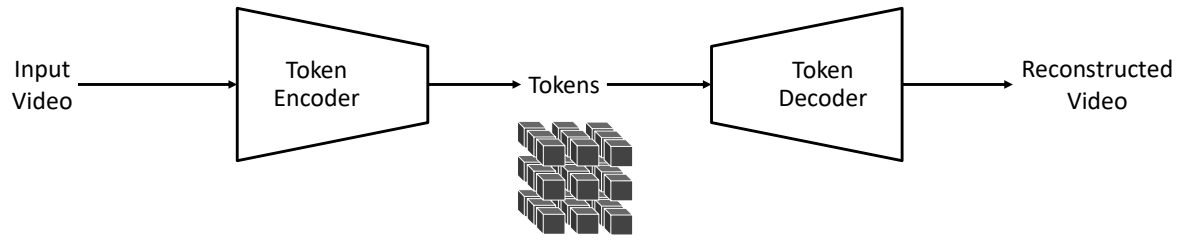
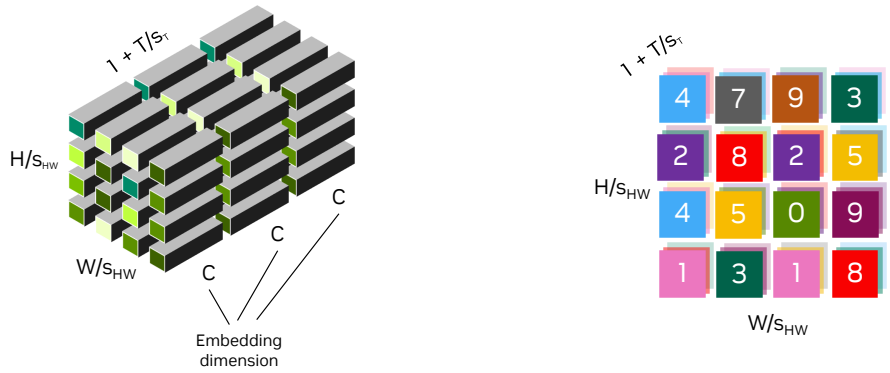


Figure 6: **Video tokenization pipeline.** An input video is encoded into tokens, which are usually much more compact than the input video. The decoder then reconstructs the input video from the tokens. Tokenizer training is about learning the encoder and decoder to maximally preserve the visual information in the tokens.

Tokenizers come in two types: continuous and discrete (see Fig. 7 for illustrations). Continuous tokenizers encode visual data into continuous latent embeddings, as in latent diffusion models like Stable Diffusion (Rombach et al., 2022) or VideoLDM (Blattmann et al., 2023). These embeddings are suitable for models that generate data by sampling from continuous distributions. Discrete tokenizers encode visual data into discrete latent codes, mapping them into quantized indices, as seen in autoregressive transformers such as VideoPoet (Kondratyuk et al., 2024). This discrete representation is necessary for models such as GPT that are trained with the cross-entropy loss. Fig. 7 illustrates the two types of tokens.

The success of tokenizers largely relies on their ability to deliver high compression rates without compromising their subsequent visual reconstruction quality. On one hand, high compression reduces storage and computa-



(a) Continuous tokens (b) Discrete tokens

Figure 7: **Visualization of continuous and discrete tokenizers**. Tokens along spatial ($\frac{H}{s_{HW}} \times \frac{W}{s_{HW}}$) and temporal ($1 + \frac{T}{s_T}$) dimensions, with a spatial compression factor of s_{HW} and a temporal compression factor of s_T. The first temporal token represents the first input frame, enabling joint image ($T = 0$) and video ($T > 0$) tokenization in a shared latent space. Left: Continuous latent embeddings with an embedding size of C. Right: Quantized indices, each color representing a discrete latent code.

tional demands. On the other hand, excessive compression can lead to the loss of essential visual details. This trade-off presents a significant challenge in tokenizer design.

We present Cosmos Tokenizer, a suite of visual tokenizers that includes both continuous and discrete tokenizers for images and videos. Cosmos Tokenizer offers exceptional visual reconstruction quality and inference efficiency. It offers a range of compression rates to accommodate diverse computational constraints and application needs. Tab. 4 presents a comparison of different visual tokenizers and their capabilities.

Table 4: Comparison of different visual tokenizers and their capabilities.

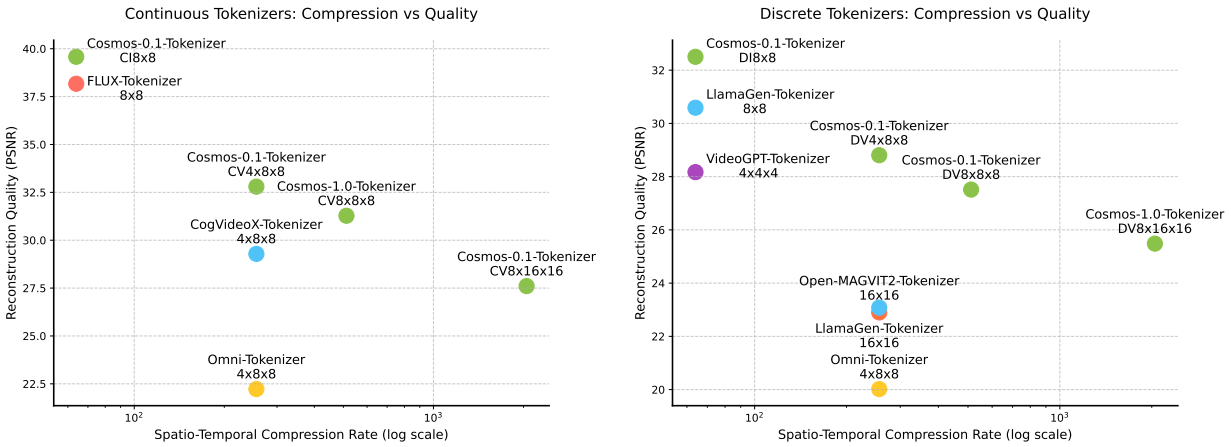
Model	Causal	Image	Video	Joint	Discrete	Continuous
FLUX-Tokenizer (FLUX, 2024)	-	✓	✗	✗	✗	✓
Open-MAGVIT2-Tokenizer (Luo et al., 2024)	-	✓	✗	✗	✓	✗
LlamaGen-Tokenizer (Sun et al., 2024)	-	✓	✗	✗	✓	✗
VideoGPT-Tokenizer (Yan et al., 2021)	✗	✗	✓	✗	✓	✗
Omni-Tokenizer (Wang et al., 2024)	✗	✓	✓	✓	✓	✓
CogVideoX-Tokenizer (Yang et al., 2024)	✓	✓	✓	✓	✗	✓
Cosmos-Tokenizer	✓	✓	✓	✓	✓	✓

We design Cosmos Tokenizer using a lightweight and computationally efficient architecture with a temporally causal mechanism. Specifically, we employ causal temporal convolution layers and causal temporal attention layers to preserve the natural temporal order of video frames, ensuring seamless tokenization of images and videos using a single unified network architecture.

We train our tokenizers directly on high-resolution images and long-duration videos without limiting the categories or aspect ratios. Unlike existing tokenizers that focus on specific data categories and sizes, the Cosmos Tokenizer operates across various aspect ratios—including 1:1, 3:4, 4:3, 9:16, and 16:9. They are temporally length-agnostic during inference, capable of tokenizing beyond the temporal length on which it was trained.

We also evaluate our tokenizers on standard image and video benchmarking datasets, including MS-COCO 2017 (Lin et al., 2014), ImageNet-1K (Deng et al., 2009), and DAVIS (Perazzi et al., 2016). To facilitate the video tokenization study for Physical AI applications, we curate a video dataset that covers many video categories for Physical AI, ranging from fish-eye, robotics, driving, human activities, and spatial navigation.

The dataset is available at github.com/NVlabs/TokenBench.



(a) Continuous tokenizers

(b) Discrete tokenizers

Figure 8: **Comparison of continuous (left) and discrete (right) tokenizers in terms of spatio-temporal compression rate (log scale) versus reconstruction quality (PSNR).** Each solid point represents a tokenizer configuration, illustrating the trade-off between compression rate and quality. Notably, our tokenizer demonstrates an excellent compression-quality trade-off, delivering superior quality even at higher compression rates compared to other methods. The evaluation was performed on the DAVIS dataset. We calculate the PSNR of image tokenizers on all the individual frames.

As shown in Fig. 8, our evaluation results demonstrate Cosmos Tokenizer significantly outperforms existing tokenizers by a large margin—for instance, achieving a +4 dB PSNR improvement in reconstruction quality on DAVIS videos. It runs up to $12\times$ faster and can encode videos up to 8 seconds at 1080p and 10 seconds at 720p in one shot without running out of memory on a single NVIDIA A100 GPU with 80GB memory. A suite of pre-trained models, with spatial compression of $8\times$ and $16\times$, and temporal compression factors of $4\times$ and $8\times$ is available at github.com/NVIDIA/Cosmos-Tokenizer.

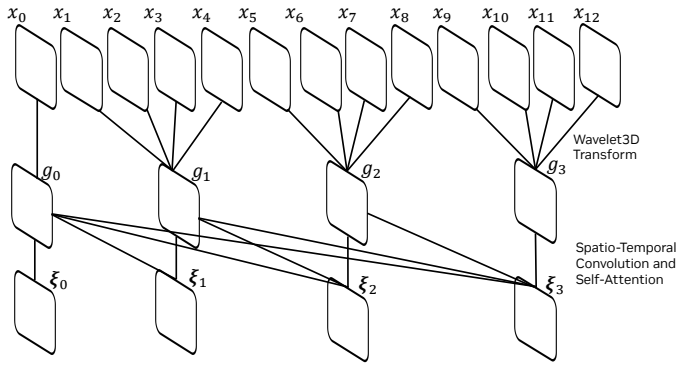
4.1. Architecture

Cosmos Tokenizer is designed as an encoder-decoder architecture. Given an input video $x_{0:T} \in \mathbb{R}^{(1+T)\times H\times W\times 3}$, with H, W, T being the height, width, and number of frames, the encoder ($\mathcal{E}$) tokenizes the inputs into a token video $z_{0:T'} \in \mathbb{R}^{(1+T')\times H'\times W'\times C}$, with a spatial compression factor of $s_{HW} = \frac{H}{H'} = \frac{W}{W'}$ and a temporal compression factor of $s_T = \frac{T}{T'}$. The decoder ($\mathcal{D}$) then reconstructs the input video from these tokens, resulting in the reconstructed video $\hat{x}_{0:T} \in \mathbb{R}^{(1+T)\times H\times W\times 3}$, mathematically given by:

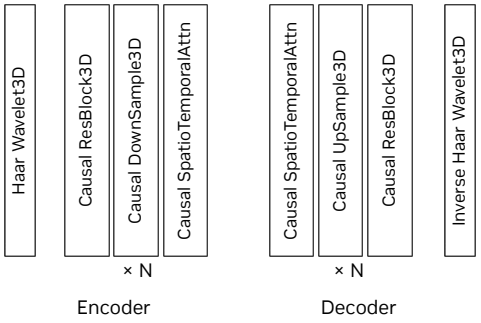
$$\hat{x}_{0:T} = \mathcal{D}\Big(\mathcal{E}\big(x_{0:T}\big)\Big). \tag{1}$$

Our architecture employs a temporally causal design, ensuring that each stage processes only current and past frames, independent of future frames. Unlike common approaches, our tokenizer operates in the wavelet space, where inputs are first processed by a 2-level wavelet transform. Specifically, the wavelet transform maps the input video $x_{0:T}$ in a group-wise manner to downsample the inputs by a factor of four along x, y, and t. The groups are formed as: $\{x_0, x_{1:4}, x_{5:8}, ..., x_{(T-3):T}\} \to \{g_0, g_1, g_2, ..., g_{T/4}\}$. Subsequent encoder stages process the frames in a temporally causal manner as $\{g_0, g_{0:1}, g_{0:2}, ...\} \to \{\xi_0, \xi_1, \xi_2, ...\}$. Successive encoder stages follow a similar scheme, finally outputting the tokens $z_{0:T'}$. The causal design helps adapt models built on top of the tokenizer to downstream Physical AI applications that often operate on the temporal causal setting. The wavelet transform allows us to operate on a more compact video representation that eliminates redundancies in pixel information, allowing the remaining layers to focus on more semantic compression.

Our encoder stages (post wavelet transform) are implemented using a series of residual blocks interleaved with downsampling blocks. In each block, we employ a spatio-temporal factorized 3D convolution, where we first apply a 2D convolution with a kernel size of $1 \times k \times k$ to capture spatial information, followed by a temporal convolution with a kernel size of $k \times 1 \times 1$ to capture temporal dynamics. We use left padding of $k - 1$ to ensure causality. To capture long-range dependencies, we utilize a spatio-temporal factorized causal self-attention with a global support region—for instance, $1 + T'$ for the last encoder block. We use the Swish activation function (Ramachandran et al., 2017) for non-linearity. We leverage Layer Normalization (LayerNorm) (Lei Ba et al., 2016) instead of Group Normalization (GroupNorm) (Wu and He, 2018), which prevents large magnitudes from appearing in specific regions of the latent space or reconstructed outputs (Karras et al., 2020; Sadat et al., 2024). The decoder mirrors the encoder, replacing the downsampling blocks with an upsampling block. Fig. 9 depicts an overview of the overall Cosmos Tokenizer architecture.



(a) Temporal Causality: Illustration of the temporal causality mechanism, where inputs $x_0, x_1, \ldots, x_{12}$ are processed through grouped intermediate outputs $g_0, g_1, \ldots$, and further refined by spatio-temporal convolution and attention operations.



(b) Network Architecture: The encoder-decoder network structure includes a 3D Haar wavelet, causal residual, causal downsampling, and causal spatio-temporal attention blocks. The decoder mirrors the encoder's structure, replacing downsampling with upsampling.

Figure 9: **Overall Cosmos Tokenizer architecture illustrating the integration of temporal causality and an encoder-decoder structure**. Temporal causality (left) processes sequential inputs, while the encoder-decoder (right) leverages wavelet transforms and causal operations to capture spatial and temporal dependencies in the data.

We employ the vanilla autoencoder (AE) formulation to model the continuous tokenizer's latent space. For discrete tokenizers, we adopt the Finite-Scalar-Quantization (FSQ) (Mentzer et al., 2023) as the latent space quantizer. The latent dimension for the continuous tokenizers is 16, whereas for the discrete tokenizers, it is 6, which represents the number of the FSQ levels, which are $(8, 8, 8, 5, 5, 5)$. This configuration corresponds to a vocabulary size of $64{,}000$.

4.2. Training Strategy

We employ a joint training strategy by alternating mini-batches of images and videos at a preset frequency. We only supervise the final output of our tokenizer's decoder. We do not use auxiliary losses tapped into the latent spaces, such as commitment or KL prior losses. For example, if a VAE (Kingma, 2013) formulation were used for continuous tokenizers instead of the vanilla AE, one would need to have the KL prior loss. If a VQ-VAE (van den Oord et al., 2017) were used for discrete quantization instead of the FSQ, one would need to have the commitment loss.

We employ a two-stage training scheme. In the first stage, we optimize with the L1 loss that minimizes the pixel-wise RGB difference between the input and reconstructed video ($\hat{x}_{0:T}$), given by

$$\mathcal{L}_1 = \|\hat{x}_{0:T} - x_{0:T}\|_1 , \tag{2}$$

and the perceptual loss based on the VGG-19 features (Simonyan and Zisserman, 2014), given by,

$$\mathcal{L}_{\text{Perceptual}} = \frac{1}{L}\sum_{l=1}^{L}\sum_{t}\alpha_l \left\|\texttt{VGG}_l(\hat{x}_t) - \texttt{VGG}_l(x_t)\right\|_1, \tag{3}$$

where $\texttt{VGG}_l(\cdot) \in \mathbb{R}^{H\times W\times C}$ is the features from the l-th layer of a pre-trained VGG-19 network, L is the number of layers considered, and α_l is the weight of the l-th layer.

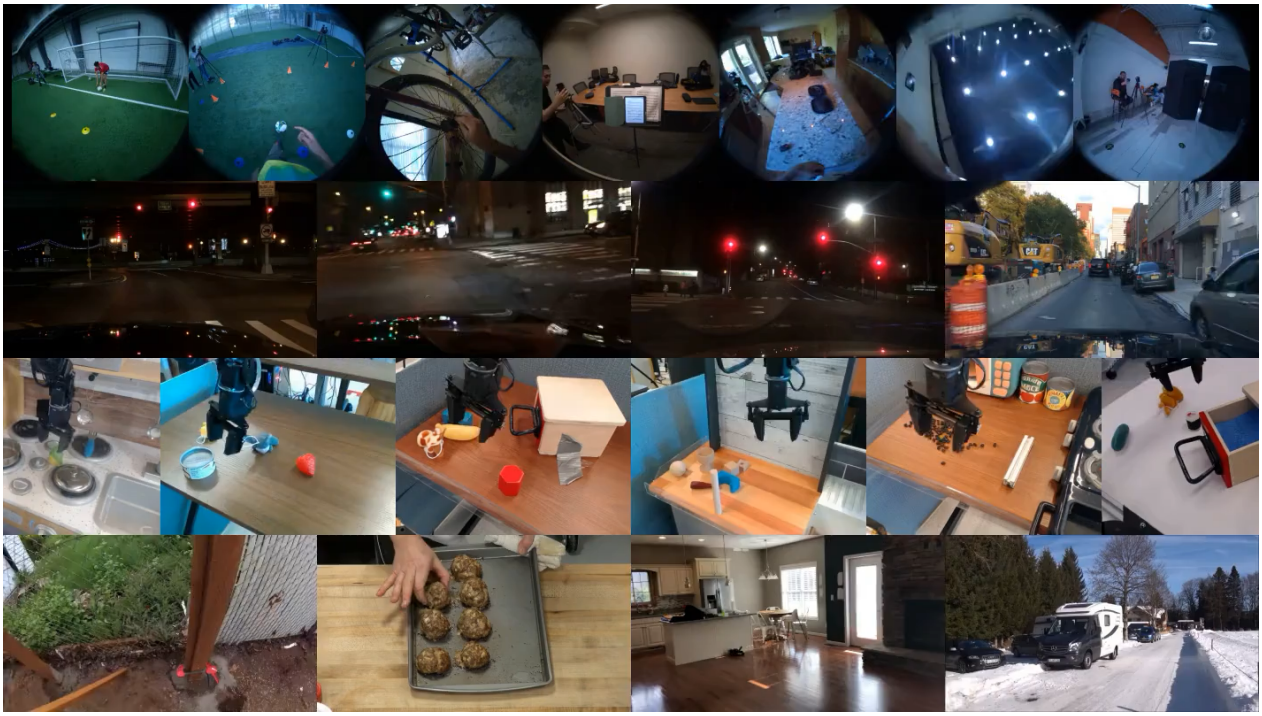


Figure 10: **Example videos from *TokenBench***. This figure shows diverse examples, including egocentric, driving, robotic manipulation, and web videos.

In the second stage, we use the optical flow (OF) loss (Teed and Deng, 2020) to handle the temporal smoothness of reconstructed videos,

$$\mathcal{L}_{\text{Flow}} = \frac{1}{T}\sum_{t=1}^{T}\left\|\texttt{OF}(\hat{x}_t,\hat{x}_{t-1}) - \texttt{OF}(x_t,x_{t-1})\right\|_1 + \frac{1}{T}\sum_{t=0}^{T-1}\left\|\texttt{OF}(\hat{x}_t,\hat{x}_{t+1}) - \texttt{OF}(x_t,x_{t+1})\right\|_1,$$

and the Gram-matrix (GM) loss (Gatys et al., 2016) to enhance the sharpness of reconstructed images,

$$\mathcal{L}_{\text{Gram}} = \frac{1}{L}\sum_{l=1}^{L}\sum_{t}\alpha_l \left\|\texttt{GM}_l(\hat{x}_t) - \texttt{GM}_l(x_t)\right\|_1. \tag{4}$$

Additionally, we use adversarial loss in the fine-tuning stage to further enhance reconstruction details, particularly at large compression rates.

We train the image tokenizers (denoted as **CI** and **DI**) at two compression rates: 8×8 and 16×16. Similarly, we train the video tokenizers (denoted as **CV** and **DV**) at three compression rates: $4\times 8\times 8$, $8\times 8\times 8$, and $8\times 16\times 16$. Here, the compression rates are expressed as $H\times W$ for images and $T\times H\times W$ for videos, where T represents the temporal dimension, and H and W represent the spatial dimensions.

For the video tokenizers, we create two variants:

Table 5: Evaluation of continuous video (CV) tokenizers on DAVIS and TokenBench.

			DAVIS			TokenBench		
Tokenizer	Frames	Formulation	PSNR ↑	SSIM ↑	rFVD ↓	PSNR ↑	SSIM ↑	rFVD ↓
CogVideoX-Tokenizer4×8×8	17	VAE	29.29	0.864	19.58	32.06	0.909	6.97
Omni-Tokenizer4×8×8	17	VAE	22.23	0.713	117.66	24.48	0.830	35.86
Cosmos-0.1-Tokenizer-CV4×8×8	49	AE	**32.80**	**0.900**	**15.93**	**35.45**	**0.928**	**6.85**
Cosmos-0.1-Tokenizer-CV8×8×8	49	AE	30.61	0.856	30.16	34.44	0.917	11.62
Cosmos-0.1-Tokenizer-CV8×16×16	49	AE	27.60	0.779	93.82	31.61	0.875	43.08
Cosmos-1.0-Tokenizer-CV8×8×8	121	AE	31.28	0.868	23.49	35.13	0.926	9.82

Table 6: Evaluation of discrete video (DV) tokenizers on DAVIS and TokenBench.

			DAVIS			TokenBench		
Tokenizer	Frames	Quantization	PSNR ↑	SSIM ↑	rFVD ↓	PSNR ↑	SSIM ↑	rFVD ↓
VideoGPT-Tokenizer4×4×4	-	VQ	28.17	**0.850**	72.33	**33.66**	**0.914**	**13.85**
Omni-Tokenizer4×8×8	17	VQ	20.02	0.703	188.60	25.31	0.827	53.55
Cosmos-0.1-Tokenizer-DV4×8×8	17	FSQ	**28.81**	0.818	**37.36**	31.97	0.888	19.67
Cosmos-0.1-Tokenizer-DV8×8×8	17	FSQ	27.51	0.789	100.15	30.95	0.873	43.86
Cosmos-0.1-Tokenizer-DV8×16×16	17	FSQ	25.09	0.714	241.52	28.91	0.829	113.48
Cosmos-1.0-Tokenizer-DV8×16×16	49	FSQ	25.49	0.719	259.33	29.33	0.838	107.43

1. **Cosmos-0.1-Tokenizer**: Trained using mini-batches sampling a smaller number of video frames (49 frames for **CV** and 17 frames for **DV**).
2. **Cosmos-1.0-Tokenizer**: Trained using mini-batches sampling a larger number of video frames (121 frames for **CV** and 49 frames for **DV**).

This approach ensures flexibility in handling varying temporal and spatial resolutions for image and video data.

4.3. Results

We extensively evaluate our Cosmos Tokenizer suite on various image and video benchmark datasets. For the evaluation of image tokenizers, we follow prior art to evaluate MS-COCO 2017 (Lin et al., 2014) and ImageNet-1K (Deng et al., 2009). We use the MS-COCO 2017 validation subset of 5,000 images, and ImageNet-1K validation subset of 50,000 images as image evaluation benchmark.

TokenBench. For video tokenizer evaluation, there is not yet a standard benchmark for high-resolution and long-duration videos. To this end, we introduce a benchmark called *TokenBench* to cover a wide variety of domains, including robotic manipulation, driving, egocentric, and web videos, and standardize the evaluation. We resort to existing video datasets that are commonly used for various tasks, including BDD100K (Yu et al., 2020), EgoExo-4D (Grauman et al., 2024), BridgeData V2 (Walke et al., 2023), and Panda-70M (Chen et al., 2024). We randomly sample 100 videos from each dataset and preprocess them by taking the first 10 seconds and resizing the short size to 1080. For Panda-70M, we manually filter out the videos with low-quality content and small motions. For EgoExo-4D, we randomly pick 100 scenes and sample one egocentric video and one exocentric video. This results in a total of 500 videos. Some examples of *TokenBench* can be found in Fig. 10. We release *TokenBench* at the github.com/NVlabs/TokenBench.

In addition to *TokenBench*, we also evaluate our video tokenizers on the DAVIS dataset at 1080p resolution.

Baselines and evaluation metrics. We evaluate our tokenizers at various compression rates to showcase their effectiveness for different computational needs. We compare each of these tokenizers with state-of-the-art image and video tokenizers. Tab. 4 presents the specific SOTA tokenizers we compared against in various settings. The evaluation metrics include Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM),

Table 7: Evaluation of continuous image (CI) tokenizers on various image datasets.

			(a) MS-COCO 2017			(b) ImageNet-1K		
Tokenizer	Height	Formulation	PSNR ↑	SSIM ↑	rFID ↓	PSNR ↑	SSIM ↑	rFID ↓
FLUX-Tokenizer8×8	-	VAE	24.00	0.682	2.501	20.09	0.518	1.229
Cosmos-0.1-Tokenizer-CI8×8	1024	AE	**28.66**	**0.836**	**1.760**	**28.83**	**0.837**	**0.689**
Cosmos-0.1-Tokenizer-CI16×16	1024	AE	23.63	0.663	3.823	23.72	0.655	1.031

Table 8: Evaluation of discrete image (DI) tokenizers on various image datasets.

			(a) MS-COCO 2017			(b) ImageNet-1K		
Tokenizer	Height	Quantization	PSNR ↑	SSIM ↑	rFID ↓	PSNR ↑	SSIM ↑	rFID ↓
Open-MAGVIT2-Tokenizer16×16	-	LFQ	19.50	0.502	6.649	17.00	0.398	2.701
LlamaGen-Tokenizer8×8	-	VQ	21.99	0.616	4.123	19.64	0.498	1.403
LlamaGen-Tokenizer16×16	-	VQ	19.11	0.491	6.077	18.38	0.448	1.657
Cosmos-0.1-Tokenizer-CI8×8	1024	FSQ	**24.40**	**0.704**	**3.710**	**24.48**	**0.701**	**1.265**
Cosmos-0.1-Tokenizer-CI16×16	1024	FSQ	20.45	0.529	7.234	20.49	0.518	2.518

Table 9: Runtime performance comparison of tokenizers. Time is reported per image or per video frame.

Tokenizer	Type	Resolution	Frames	Parameters	Time (ms)
FLUX-Tokenizer8×8	Continuous-Image	1024 × 1024	-	84M	242
Cosmos-0.1-Tokenizer-CI8×8	Continuous-Image	1024 × 1024	-	77M	**62.7**
LlamaGen-Tokenizer8×8	Discrete-Image	1024 × 1024	-	70M	475
Cosmos-0.1-Tokenizer-DI8×8	Discrete-Image	1024 × 1024	-	79M	**64.2**
CogVideoX-Tokenizer4×8×8	Continuous-Video	720 × 1280	17	216M	414
Omni-Tokenizer4×8×8	Continuous-Video	720 × 1280	17	54M	82.9
Cosmos-0.1-Tokenizer-CV4×8×8	Continuous-Video	720 × 1280	49	105M	**34.8**
Omni-Tokenizer4×8×8	Discrete-Video	720 × 1280	17	54M	53.2
Cosmos-0.1-Tokenizer-DV4×8×8	Discrete-Video	720 × 1280	17	105M	**51.5**

reconstruction Fréchet Inception Distance (rFID) (Heusel et al., 2017) for images, and reconstruction Fréchet Video Distance (rFVD) (Unterthiner et al., 2019) for videos.

Quantitative results. Tabs. 5 and 6 summarize the average quantitative metrics of continuous and discrete video tokenizers on various benchmarks. As shown in both tables, Cosmos Tokenizer achieves state-of-the-art performance in all the metrics compared to prior arts on both the DAVIS video dataset and *TokenBench*, with a spatial-temporal compression ratio of $4 \times 8 \times 8$. Moreover, even with $2\times$ and $8\times$ higher compression ratios (*i.e.*, $8 \times 8 \times 8$ and $8 \times 16 \times 16$), Cosmos Tokenizer still achieves better quality than prior art, showcasing an excellent compression-quality trade-off.

Tabs. 7 and 8 summarize the average quantitative metrics of continuous and discrete image tokenizers on various image benchmarks, covering a wide range of image types. As shown, compared to prior arts, Cosmos Tokenizer consistently achieves state-of-the-art results with a compression ratio of 8×8. More importantly, at a $4\times$ larger compression ratio of 16×16, the image quality of Cosmos Tokenizer is often comparable or even better than prior art at 8×8 compression ratio, as shown in Tabs. 7 and 8.

These quantitative results on a variety of image and video benchmark datasets confirm that Cosmos Tokenizer is able to better represent visual content with large spatial-temporal compression.

Runtime performance. Tab. 9 shows the number of parameters and the averaged encoding and decoding times per image or per video frame, measured on a single A100 80GB GPU. In comparison, we also list the parameters and the average speeds of prior state-of-the-art tokenizers. As shown, for both image and video

tokenizers, Cosmos Tokenizer is $2\times \sim 12\times$ faster while maintaining the smallest model size compared to prior arts, showing that Cosmos Tokenizer has high efficiency for encoding and decoding visual content.

5. World Foundation Model Pre-training

Pre-trained WFMs are generalists that capture general knowledge of real-world physics and natural behaviors. We exploit two different scalable deep learning paradigms, diffusion models and autoregressive models, to build two families of WFMs. Both diffusion models and autoregressive models break a difficult generation problem into a sequence of easier sub-problems and have been turbo-charging the development of generative models. In the case of diffusion models, the difficult generation problem is divided into a sequence of denoising problems. In the case of autoregressive models, the difficult generation problem is divided into a sequence of next-token prediction problems. We discuss how we scale these deep learning paradigms using various parallelization techniques tailored for modern GPUs in our endeavor of building pre-trained WFMs. We train all of the WFM models reported in the paper using a cluster of $10{,}000$ NVIDIA H100 GPUs in a time span of three months.

Table 10: Maps of Cosmos World Foundation Model 1.0 Release. We have two sets of WFMs. One is based on diffusion models, while the other is based on autoregressive models. For each family, we build two base models and two derivative models. To achieve the best generation quality, we also build a prompt upsampler for the diffusion models and a diffusion decoder for the autoregressive models.

Type	Models		Tokenizer	Enhancer
Diffusion	Cosmos-1.0-Diffusion-7B-Text2World	→ Cosmos-1.0-Diffusion-7B-Video2World	Cosmos-1.0-Tokenizer-CV8x8x8	Cosmos-1.0-PromptUpsampler-12B-Text2World
	Cosmos-1.0-Diffusion-14B-Text2World	→ Cosmos-1.0-Diffusion-14B-Video2World		
Autoregressive	Cosmos-1.0-Autoregressive-4B	→ Cosmos-1.0-Autoregressive-5B-Video2World	Cosmos-1.0-Tokenizer-DV8x16x16	Cosmos-1.0-Diffusion-7B-Decoder-DV8x16x16ToCV8x8x8
	Cosmos-1.0-Autoregressive-12B	→ Cosmos-1.0-Autoregressive-13B-Video2World		

In Tab. 10, we present a map of our pre-trained WFMs and their companions. For the diffusion-based WFM family, we start by building two Text2World models of 7B and 14B, respectively, which render Cosmos-1.0-Diffusion-7B-Text2World and Cosmos-1.0-Diffusion-14B-Text2World. These models can map text prompts to videos of visual worlds. We then fine-tune the Text2World models to take additional video input, representing the current observation. The result is a Video2World model where the future video is predicted based on the current observation (input video) and the perturbation (text prompt). These diffusion models are latent diffusion models that take continuous tokens. We use Cosmos-1.0-Tokenizer-CV8x8x8 to produce the visual tokens. The training text prompts for the WFMs are produced by a VLM through video description generation. These descriptions follow a different distribution of human descriptions of videos. To mitigate the domain gap, we build Cosmos-1.0-PromptUpsampler-12B-Text2World based on the Mistral-NeMo-12B-Instruct model (Mistral and NVIDIA, 2024) to help convert human text prompts to those preferred by our diffusion-based WFMs.

For the autoregressive-based WFM family, we first build two base models that are 4B and 12B in size, respectively, to predict future videos purely based on the current video observation. We name them Cosmos-1.0-Autoregressive-4B and Cosmos-1.0-Autoregressive-12B, respectively. These are Llama3-style GPT models trained

from scratch for the video prediction task and bear no language understanding. To enable autoregressive-based WFMs to utilize textual information for next token prediction, we incorporate T5 embeddings of the input text prompt into the WFMs through cross-attention layers added to the transformer blocks. These autoregressive WFMs use Cosmos-1.0-Tokenizer-DV8x16x16, which maps an input video to a few integers. The heavy compression of the tokenizer can sometimes lead to undesired distortions. To address the problem, we build a diffusion decoder (Cosmos-1.0-Diffusion-7B-Decoder-DV8x16x16ToCV8x8x8) through fine-tuning the Cosmos-1.0-Diffusion-7B-Text2World model to map discrete tokens in the DV8x16x16 space to continuous tokens in the CV8x8x8 space.

5.1. Diffusion-based World Foundation Model

Our diffusion-based WFMs are latent diffusion models that operate within a learned latent space of a tokenizer, enabling a compact, reduced-dimensional representation of videos. This design choice offers several advantages: it reduces computational costs during both training and inference while simplifying the denoising task (Hoogeboom et al., 2024; Rombach et al., 2022). To tokenize videos into latent representations, we employ Cosmos-1.0-Tokenizer-CV8x8x8.

5.1.1. Formulation

To train our diffusion WFMs, we adopt the approach outlined in EDM (Karras et al., 2022, 2024). The denoising score matching loss for the denoiser D_θ, evaluated at a noise level σ, is defined as

$$\mathcal{L}(D_\theta, \sigma) \;=\; \mathbb{E}_{\mathbf{x}_0,\mathbf{n}}\Big[\big\|D_\theta(\mathbf{x}_0+\mathbf{n};\sigma)-\mathbf{x}_0\big\|_2^2\Big], \tag{5}$$

where $\mathbf{x}_0 \sim p_{\text{data}}$ is a clean image or video sampled from the training set, $\mathbf{n} \sim \mathcal{N}\big(\mathbf{0}, \sigma^2\mathbf{I}\big)$ is i.i.d. Gaussian noise, and D_θ is a noise-conditioned neural network tasked with denoising the corrupted sample $\mathbf{x}_0 + \mathbf{n}$. We adhere to the preconditioning design introduced in EDM for parameterizing D_θ. The overall training loss is defined as a weighted expectation of $\mathcal{L}(D_\theta; \sigma)$ over the noise levels:

$$\mathcal{L}(D_\theta) = \mathbb{E}_\sigma\left[\frac{\lambda(\sigma)}{e^{u(\sigma)}}\mathcal{L}(D_\theta,\sigma)+u(\sigma)\right], \tag{6}$$

$$\lambda(\sigma) = \big(\sigma^2+\sigma_{\text{data}}^2\big)\,/\,(\sigma\cdot\sigma_{\text{data}})^2, \tag{7}$$

$$\ln(\sigma) \sim\; \mathcal{N}\big(P_{\text{mean}}, P_{\text{std}}^2\big), \tag{8}$$

where the distribution of noise levels σ is controlled by hyperparameters P_{mean} and P_{std}. σ_{data} is the standard deviation of the training data, and the weighting function $\lambda(\sigma)$ ensures equal contribution of each noise level at the beginning of the training. However, as training progresses, this balance may deteriorate. To mitigate this issue, we treat the optimization over various noise levels as a form of multi-task learning. We utilize the uncertainty-based weighting approach by introducing $u(\sigma)$ as a continuous uncertainty function quantifying the uncertainty for the denoising objective $\mathcal{L}(D_\theta, \sigma)$ at noise level σ. We use a simple MLP to parameterize $u(\sigma)$ and minimize the overall loss $\mathcal{L}(D_\theta)$ during training. Intuitively, the contribution of loss at noise level σ is weighted down if the model is uncertain about the task, *i.e.*, if $u(\sigma)$ is high. At the same time, the model is penalized for this uncertainty, encouraging $u(\sigma)$ to be as low as possible.

Compared to recent video generative models that adopt the Gaussian flow matching formulation (Kong et al., 2024; Polyak et al., 2024), our work is derived from the diffusion score matching perspective (Ho et al., 2020; Song et al., 2020). However, as shown by Gao et al. (2024), these frameworks are theoretically equivalent, sharing fundamental similarities in their objectives and training procedures. Our EDM-based formulation aligns with these insights, mainly differing in the choice of preconditioning designs and hyperparameters. In practice, we have not encountered any performance limitations with the EDM formulation.

5.1.2. Architecture

In this section, we describe the design of our denoiser network D_θ that builds upon DiT (Peebles and Xie, 2023), which was originally designed for label-conditioned image generation. We adapt its architecture to better suit our goal of controllable video generation. We visualize the overall network design in Fig. 11.

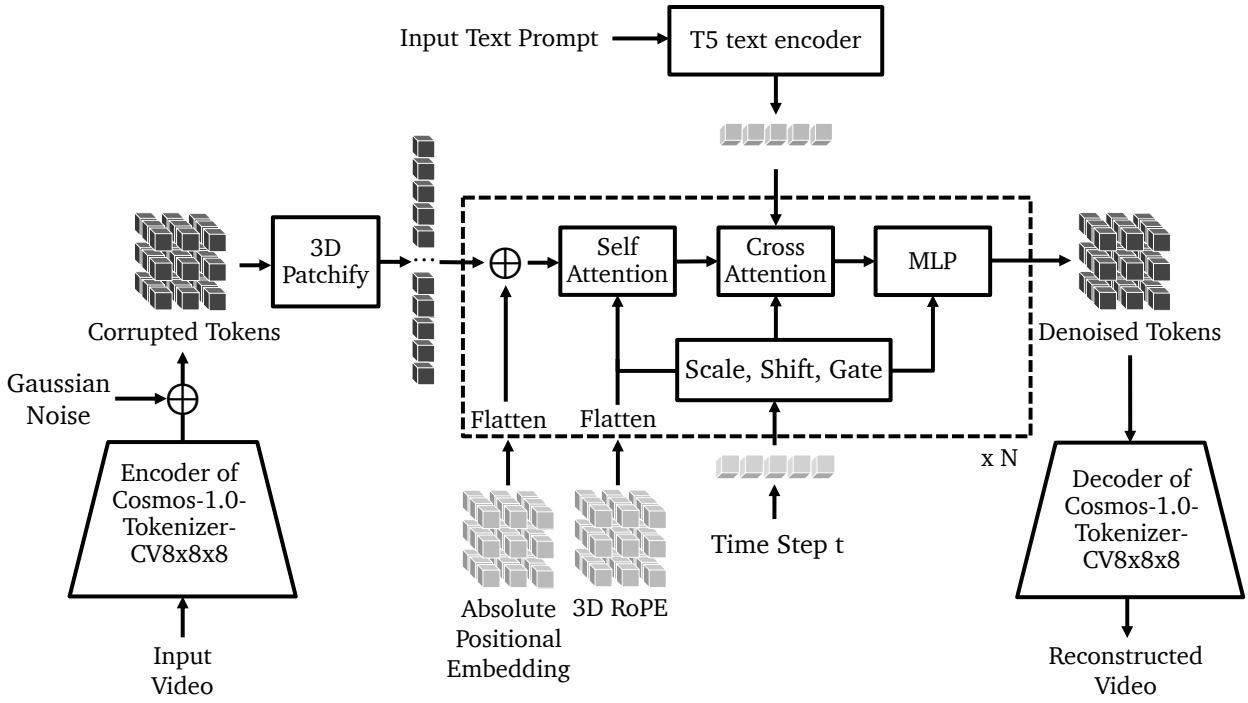


Figure 11: **Overall architecture of Cosmos-1.0-Diffusion World Foundation Model**. The model processes an input video through the encoder of the Cosmos-1.0-Tokenizer-CV8x8x8 to obtain latent representations, which are subsequently perturbed with Gaussian noise. These representations are then transformed using a 3D patchification process. In the latent space, the architecture applies repeated blocks of self-attention, cross-attention (integrating input text), and feed-forward MLP layers, modulated by adaptive layer normalization (scale, shift, gate) for a given time step t. The decoder of Cosmos-1.0-Tokenizer-CV8x8x8 reconstructs the final video output from the refined latent representation.

3D patchification. The input to our network is a latent representation of shape $T \times C \times H \times W$ for both image and video data, with images differentiated by a video with a single frame. To prepare inputs for our denoiser network, we first "patchify" the state using a linear layer and subsequently flatten it. This process involves projecting non-overlapping cubes of shape (p_t, p_h, p_w) into individual token inputs for the network. Consequently, after patchification, an image or video is reshaped into a one-dimensional, spatiotemporal sequence of length $THW/(p_t p_h p_w)$. We use $p_t = 1, p_h = p_w = 2$ for our denoiser network.

Hybrid positional embedding with FPS-aware 3D RoPE and learnable embedding. We employ a 3D-factorized Rotary Position Embedding (RoPE) (Su et al., 2024) to allow the generation of arbitrary size, aspect ratio, and video length. Specifically, we partition the feature dimension into three approximately equal chunks, each applying RoPE with positional information along the temporal, height, and width axes, respectively. In practice, this can be implemented efficiently without splitting and concatenation in each block by concatenating frequency embeddings in their respective axes and reusing RoPE kernels optimized for Large Language Models (LLMs). To further support video synthesis with varying frame rates, we rescale temporal frequencies based on the training video's Frames Per Second (FPS). Due to RoPE's relative positional encoding property and our 3D factorization design, the FPS-aware design is compatible with our joint image-video training. An additional benefit of RoPE is evident during progressive training when we alter resolution or video length. By leveraging

Neural Tangent Kernel (NTK)-RoPE (Peng and Quesnelle, 2023), we observe rapid model convergence, achieving reasonable performance even within $5{,}000$ training steps. Additionally, we find that adding an extra learnable absolute positional embedding per transformer block can further enhance the model, reduce training loss, and reduce morphing artifacts in generated videos.

Cross-attention for text conditioning. We rely on cross-attention layers in our network for incorporating linguistic information. Each transformer block consists of sequential self-attention, cross-attention, and feed-forward layers. While self-attention operates over spatiotemporal tokens, cross-attention integrates semantic context using T5-XXL (Raffel et al., 2020) embeddings as keys and values, enabling effective text conditioning.

Query-key normalization. In the early stages of training, we observe instability in the growth of attention logits, leading to a collapse of attention entropy. We follow existing literature (Dehghani et al., 2023; Esser et al., 2024; Wortsman et al., 2023) to normalize query Q and key K before the attention operation. We use Root Mean Square Normalization (RMSNorm) (Zhang and Sennrich, 2019) with learnable scales for all self-attention and cross-attention layers within our network.

AdaLN-LoRA. We find that DiT's adaptive layer normalization (AdaLN) layers (Peebles and Xie, 2023; Xu et al., 2019) account for a significant portion of the model parameters while contributing negligibly to the computational complexity in terms of FLOPs. Inspired by W.A.L.T (Gupta et al., 2024), we implement Low-Rank Adaptation (LoRA) (Hu et al., 2022) to decompose the dense linear projections in these layers into low-rank approximations. For Cosmos-1.0-Diffusion-7B, this architectural optimization achieves a 36% reduction in parameter count (from 11B to 7B parameters) while maintaining performance parity across all evaluation metrics, demonstrating the effectiveness of our parameter-efficient design.

Table 11: Configuration details of Cosmos-1.0-Diffusion models.

Configuration	7B-Text2World	14B-Text2World	7B-Video2World	14B-Video2World
Number of Layers	28	36	28	36
Model Dimension	4,096	5,120	4,096	5,120
FFN Hidden Dimension	16,384	20,480	16,384	20,480
AdaLN-LoRA Dimension	256	256	256	256
Number of Attention Heads	32	40	32	40
Number of Key / Value Heads	32	40	32	40
MLP Activation	GELU			
Positional Embedding	Hybrid positional embedding			
Conditional Information	Text; FPS	Text; FPS	Text; FPS; Frames	Text; FPS; Frames
Base Learning Rate	2^{-15}	2^{-16}	2^{-15}	2^{-16}
Weight decay	0.1	0.2	0.1	0.2
Learning Rate Warmup	Linear scheduler with $2{,}500$ iterations			
AdamW momentum and ϵ	$\beta_1, \beta_2 = 0.9, 0.99$; $\epsilon = 10^{-10}$			

5.1.3. Training Strategy

This section outlines the methodologies employed to train our models on datasets spanning multiple modalities, resolutions, aspect ratios, and conditioning inputs.

Joint image and video training. To leverage the vast abundance of high-quality, diverse image datasets in model training, we implement an alternating optimization strategy that interleaves batches of image and video data. To facilitate cross-modal knowledge transfer between image and video domains, we adopt a domain-specific normalization scheme that aligns the latent distributions using sufficient statistics estimated independently for image and video data. This approach is motivated by the observation that reducing the distributional shift between image and video latent representations improves generation quality. Furthermore, we observe non-stationary statistics across temporal and channel dimensions in video latent representations. To address this heterogeneity, we employ a normalization strategy that applies frame-wise and channel-wise

standardization to video latent representations, effectively encouraging them to better approximate an isotropic Gaussian prior distribution.

Beyond cross-modality knowledge transfer, our normalization scheme provides an important theoretical benefit: scale invariance in the signal-to-noise ratio during training. Consider two zero-mean latent representations with different scales: one standardized to unit variance, and another with variance 4. When adding Gaussian noise $\mathcal{N}(0, \sigma^2)$ to achieve a desired signal-to-noise ratio for the standardized representation, we must scale the noise to $\mathcal{N}(0, 4\sigma^2)$ for the unnormalized representation to maintain the same ratio. By standardizing all latent representations, we ensure consistent signal-to-noise ratios across different scales, facilitating model adaptation even when the underlying tokenizer is updated during training.

To maintain computational efficiency, we balance image and video batch sizes to ensure comparable memory utilization across GPUs. However, we observe that the video batch denoising loss exhibits slower convergence compared to the image batch loss. We attribute this to the inherent temporal redundancy in video frames, which results in smaller gradient magnitudes for video batches. Drawing inspiration from recent advances in multi-resolution image training (Atzmon et al., 2024; Chen, 2023; Hoogeboom et al., 2023), we address this convergence discrepancy by scaling the video batch noise levels by the square root of the frame count relative to image batch noise levels.

Table 12: Stages of progressive training and their specifications.

Stage	Resolution	Number of Frames	Context Length	FSDP Size	CP Size
Low-resolution Pre-training	512p (640×512)	57	10,240 [a]	64	2
High-resolution Pre-training	720p (1280×704)	121	56,320 [b]	64	8
High-quality Fine-tuning	720p (1280×704)	121	56,320 [b]	64	8

[a] 10,240 (the context length) is computed as: 640 (width) $\div 8$ (tokenize) $\div 2$ (patchify) $\times 512$ (height) $\div 8$ (tokenize) $\div 2$ (patchify) $\times[(57-1) \div 8 + 1]$ (tokenize frames).
[b] 56,320 (the context length) is computed as: 1280 (width) $\div 8$ (tokenize) $\div 2$ (patchify) $\times 704$ (height) $\div 8$ (tokenize) $\div 2$ (patchify) $\times[(121-1) \div 8 + 1]$ (tokenize frames).

Progressive training. We adopt a progressive training strategy, with the specifics of each stage detailed in Tab. 12. The initial stage involves training on videos and images at a resolution of 512 pixels, using videos composed of 57 frames. Subsequently, we transition to the target resolution of 720 pixels, increasing the video length to 121 frames. After pre-training on massive data, we fine-tune the model on a high-quality subset for $\mathcal{O}(10k)$ iterations with a linearly decaying learning rate. Consistent with findings from Dai et al. (2023), we also find that fine-tuning can improve the quality of the generated videos.

Multi-aspect training. To accommodate content with varying aspect ratios, we organize the data into five distinct buckets corresponding to ratios of 1:1, 3:4, 4:3, 9:16, and 16:9, assigning each image or video to the bucket with the closest aspect ratio. During training, each data parallel process group samples from one bucket, allowing different buckets across different parallel process groups. We implement longest-side resizing to maximally preserve the original content information described in the prompt. For batch processing, we apply reflection padding to missing pixels and supply the padding mask to the diffusion backbone, enabling precise control during inference.

Mixed-precision training. We maintain two copies of the model weights: one in BF16 and another in FP32. During the forward and backward passes, the BF16 weights are used to improve training efficiency, resulting in gradients and activations also in BF16 format. For parameter updates, the weights are updated in FP32 to ensure numerical stability. The updated FP32 parameters are then copied and cast to BF16 for the next iteration. To further stabilize training, we scale the loss of denoising score matching in Eq. (5) by a factor of 10. We also find that lower betas and eps coefficients in AdamW significantly reduce loss spikes. For our 14B diffusion model training, we rarely encountered loss spikes, and there were no non-recoverable loss spikes.

Text conditioning. For our Text2World models, we employ T5-XXL (Raffel et al., 2020) as the text encoder. We zero-pad T5 embeddings to maintain a fixed sequence length of 512. To enhance text-context alignment, we adopt classifier-free guidance (Ho and Salimans, 2022). Unlike prior works (Balaji et al., 2022; Saharia et al., 2022) that randomly zero out text embeddings, we omit this step due to the effectiveness of negative prompts during inference. Notably, as a text-to-image generator, our model excels in generating high-fidelity images even without guidance, a capability we attribute to the high-quality training dataset. While classifier-free guidance typically promotes mode-seeking behavior for preferred visual content, we find that careful data selection achieves a similar effect. However, for video generation, the lack of comparable high-quality data leads to suboptimal results under low guidance settings. Consequently, higher guidance values are required to produce satisfactory content in video-generation tasks.

Image and video conditioning. We extend our Text2World models to build Video2World models that support image and video conditioning by incorporating previous frame(s) into the generation process. Specifically, the conditional frame(s) are concatenated with the generated frames along the temporal dimension. To improve robustness against variations in input frame(s) during inference, we introduce augmented noise to the conditional frames during training. The sigma value for this augmented noise is sampled with $P_{\text{mean}} = -3.0, P_{\text{std}} = 2.0$. Additionally, the input to the diffusion model is concatenated along the channel dimension with a binary mask that distinguishes conditional frames from generated frames. The loss function excludes contributions from the locations of conditional frames, focusing exclusively on the generated output. To improve generalization, we randomly vary the number of conditional frames during training. During inference, the model can flexibly operate with either a single conditional frame (image) or multiple previous frames as input.

5.1.4. Scaling Up

Here, we outline the techniques that enable efficient scaling of our diffusion WFMs. We analyze the memory requirements of our models, discuss parallelism strategies, and compare our training setup against other video diffusion models and state-of-the-art LLMs.

Memory requirements. The four major components that consume the GPU memory are:

- **Model parameters**: 10 bytes per parameter. Our mixed precision training stores model parameters in both FP32 and BF16, alongside Exponential Moving Average (EMA) weights in FP32.
- **Gradients**: 2 bytes per parameter. We store the gradients in BF16.
- **Optimizer states**: 8 bytes per parameter. We use AdamW (Loshchilov and Hutter, 2019) as our optimizer and store the optimizer states (*i.e.*, first and second moments) in FP32.
- **Activations**: $(2 \times \text{number_of_layers} \times 15 \times \text{seq_len} \times \text{batch_size} \times \text{d_model})$ bytes. We store the activations in BF16. Tab. 13 provides details of the stored activations for major operations within the network. To optimize memory usage, we implement selective activation checkpointing (Chen et al., 2016; Korthikanti et al., 2023), recomputing activations for memory-limited layers such as normalization functions.

For instance, our 14B model (Cosmos-1.0-Diffusion-14B-Text2World) requires approximately 280 GB for model parameters, gradients, and optimizer states, alongside 310 GB for activations during high-resolution pre-training. Given the 80GB HBM3 limit of NVIDIA H100 GPUs, we employ Fully Sharded Data Parallelism (FSDP) and Context Parallelism (CP) to distribute memory demands across multiple GPUs.

Fully Sharded Data Parallelism (FSDP). FSDP improves memory efficiency by sharding model parameters, gradients, and optimizer states across devices. It gathers parameters only when needed during computation and releases them afterward. Unlike standard data parallelism, which duplicates parameters across devices, FSDP distributes parameters, gradients, and optimizer states, with each device managing only its shard. This approach minimizes memory usage to the largest temporarily unsharded parameter set alongside its shard of parameters, gradients, and optimizer states. For our implementation, we utilize a sharding factor of 32 for the 7B model and 64 for the 14B model to balance memory and communication latency.

Table 13: Cosmos-Diffusion transformer FLOPs and activation memory. The table provides the computational cost (FLOPs) and activation memory requirements for each operation. For FLOPs, we use a factor of 2 to describe the multiply accumulate cost. A "—" denotes cases where the value is either negligible due to its small magnitude or omitted because activation checkpointing is employed to recompute values instead of storing them, thus saving memory.

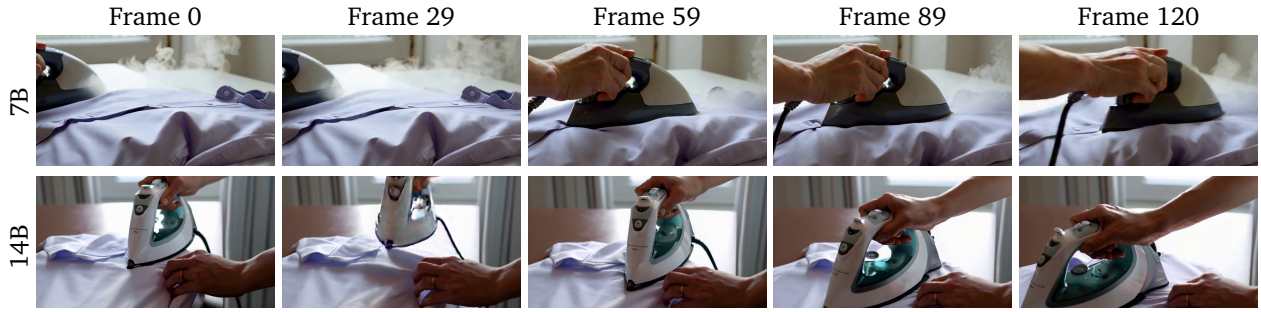
Layer	Operation	FLOPs	Activations (Tensor Shape)
Self-attention	Q, K, V Projections	$2 \times 3 \times \text{seq_len} \times \text{d_model}^2$	$\text{seq_len} \times \text{batch_size} \times \text{d_model}$ [a]
	QK Norm	—	$2 \times \text{seq_len} \times \text{batch_size} \times \text{d_model}$ [b]
	$A = Q@K^T$	$2 \times \text{seq_len}^2 \times \text{d_model}$	— [c]
	$A' = \text{Softmax}(A)$	—	— [d]
	$A'@V$	$2 \times \text{seq_len}^2 \times \text{d_model}$	$\text{seq_len} \times \text{batch_size} \times \text{d_model}$ [e]
	Final Projection	$2 \times \text{seq_len} \times \text{d_model}^2$	$\text{seq_len} \times \text{batch_size} \times \text{d_model}$
Cross-attention	Q, K, V Projections	$2 \times \text{seq_len} \times \text{d_model}^2$	$\text{seq_len} \times \text{batch_size} \times \text{d_model}$
	QK Norm	—	$\text{seq_len} \times \text{batch_size} \times \text{d_model}$ [f]
	$A = Q@K^T$	—	— [c]
	$A' = \text{Softmax}(A)$	—	— [d]
	$A'@V$	—	— [g]
	Final Projection	$2 \times \text{seq_len} \times \text{d_model}^2$	$\text{seq_len} \times \text{batch_size} \times \text{d_model}$
Feedforward	Up Projection	$4 \times \text{seq_len} \times \text{d_model}^2$	$\text{seq_len} \times \text{batch_size} \times \text{d_model}$
	GELU	—	$4 \times \text{seq_len} \times \text{batch_size} \times \text{d_model}$
	Down Projection	$4 \times \text{seq_len} \times \text{d_model}^2$	— [h]
AdaLN	LayerNorm	—	$\text{seq_len} \times \text{batch_size} \times \text{d_model}$
	Scale	—	— [i]
	Shift	—	—
	Gate	—	$\text{seq_len} \times \text{batch_size} \times \text{d_model}$

[a] The shared input is stored. [b] The query Q and key K are stored. [c] The normalized query Q and key K are recomputed. [d] The attention scores ($A = Q@K^T$) are recomputed. [e] The value V is stored. The normalized attention weights ($A' = \text{Softmax}(A)$) are recomputed. [f] In cross-attention, only query Q is counted; key K has much shorter sequence length and is thus negligible. [g] In cross-attention, the value V has much shorter sequence length and is thus negligible. [ht] The input is recomputed from GELU. [i] The input is recomputed from LayerNorm.

Context Parallelism (CP). Scaling transformers for long-context settings introduces challenges with increased FLOPs and activation memory. CP addresses these challenges by distributing computation and activations across multiple GPUs. It works by splitting both the query Q and the key-value (K, V) along their sequence dimensions into CP_SIZE chunks, where CP_SIZE is the number of GPUs within a CP group. Each GPU processes one chunk of Q and iteratively accumulates partial attention outputs using blocks of (K, V) stored in the same CP group. Different implementations of CP utilize different communication primitives, including all-gather (Dubey et al., 2024), P2P (Liu et al., 2023), and all-to-all (Jacobs et al., 2023). We employ the P2P variant from TransformerEngine (NVIDIA, 2024), which overlaps computation and communication by transferring (K, V) blocks between GPUs while simultaneously processing attention. When block sizes are carefully chosen, this overlap effectively hides data transfer latency. We organize CP groups within NVLink-connected GPUs and overlap CP ranks with FSDP ranks for optimal utilization. For image iterations with shorter contexts, CP is disabled to improve throughput. Cross-attention layers do not use CP due to the shorter sequence lengths of (K, V), which results in insufficient computation to mask communication latency.

Using Cosmos-1.0-Diffusion-14B as an instance, employing FSDP with a sharding factor of 64 reduces memory requirements for parameters, gradients, and optimizer states, bringing them down from 280 GB to approximately $280 \,/\, 64 \approx 4$ GB per GPU. Similarly, employing CP with CP_SIZE $= 8$ decreases activation memory from 310 GB to roughly $310 \,/\, 8 \approx 40$ GB per GPU. It is important to note that these calculations are underestimations; in practice, additional memory is consumed by the tokenizer and unsharded parameters. Overlapping communication and computation in CP also necessitates each GPU to retain multiple chunks of (K, V).

Comparison with other video generative models. Our parallelism strategy is deliberately streamlined compared to approaches outlined in HunyuanVideo (Kong et al., 2024) and MovieGen (Polyak et al., 2024), which incorporate Tensor Parallelism (TP) and its extension, Sequence Parallelism (SP). Despite excluding TP/SP, our setup achieves comparable Model FLOPs Utilization (MFU). While TP/SP remains valuable in certain scenarios, such as larger models or alternative network topologies, a detailed analysis of tradeoffs is left for future work.



Prompt: Hands firmly grasp the handle of a steam iron, expertly gliding it over a wrinkled shirt. With each pass, the iron releases gentle clouds of steam, effortlessly smoothing the fabric and erasing wrinkles to reveal a crisp, neat finish. The iron moves with precision and care, transforming the shirt with each stroke. A subtle scent of fresh linen permeates the air, adding to the serene ambiance. Soft light filters through a nearby window, highlighting the fabric's newly smooth texture and creating a tranquil atmosphere as this meticulous task unfolds.

Figure 12: **Generated videos from Cosmos-1.0-Diffusion-7B-Text2world and Cosmos-1.0-Diffusion-14B-Text2world.** Both the Text2World models produce videos of high visual quality, motion dynamics and text alignment. Notably, compared to the 7B model, the 14B model demonstrates an enhanced ability to capture finer visual details and more intricate motion patterns. To check full videos and more video examples, please visit our website.

Comparison with large language models. Unlike LLMs, which are typically pre-trained with shorter context lengths, long-context settings significantly increase FLOPs due to the quadratic cost of self-attention. While FLOPs for LLMs are commonly calculated as $6 \times \text{seq_len} \times P$, where P is the number of parameters (Kaplan et al., 2020), we note that this formula is inaccurate for our diffusion WFMs. We provide the forward pass FLOPs of each key operation in Tab. 13.

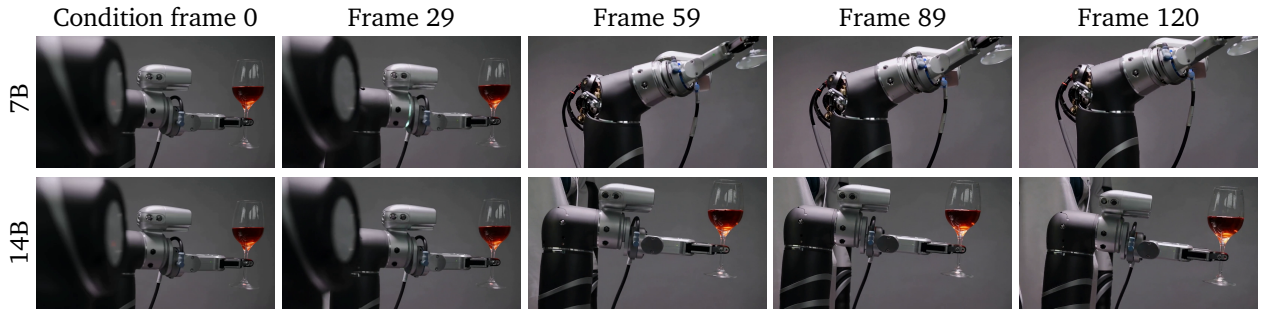
5.1.5. Prompt Upsampler

During training, our WFMs use detailed video descriptions as input text prompts to produce high-quality videos. However, during inference, user prompts may vary in length, structure, and style, often being much shorter. To bridge this gap between training and inference text prompts, we develop a prompt upsampler to transform original input prompts into more detailed and enriched versions. It can improve the prompts by adding more details and maintaining a consistent description structure, which leads to higher quality output.

The main requirements for the prompt upsampler are:

- **Fidelity to the input prompts**: The upsampled prompt must faithfully preserve the key elements of the original user input, including the main characters, actions or motions, key attributes, and overall intent.
- **Alignment with training distribution**: The upsampled prompt should closely resemble the distribution of training prompts of WFMs in terms of length, language structure, and style.
- **Enhanced visual details**: The upsampled prompt should be designed to prompt the WFMs to generate more accurate imagery.

Prompt upsampler for Text2World model. We fine-tune Mistral-NeMo-12B-Instruct (Mistral and NVIDIA, 2024) to build our prompt upsampler. To obtain paired data, that is, short prompts simulating user input and the corresponding long prompts reflecting the distribution of training prompts, we use a VLM to generate short



Prompt: The video depicts a robotic arm holding a wine glass filled with red wine. The robotic arm, equipped with multiple joints and mechanical components, appears to be designed for precision tasks. The glass is held delicately, showcasing the robot's capability to handle fragile objects. The background is minimalistic, emphasizing the interaction between the robot and the wine glass.



Prompt: The video depicts the interior of a large industrial facility, likely a factory or warehouse. The space is expansive with high ceilings and metal framework. Overhead cranes and various machinery are visible, indicating a setting for heavy manufacturing or assembly. The floor is mostly empty, with some scattered debris and marked lines. Safety signs and barriers are present, emphasizing the industrial environment. The lighting is natural, streaming through the high windows, illuminating the workspace.

Figure 13: **Generated videos from Cosmos-1.0-Diffusion-7B-Video2world and Cosmos-1.0-Diffusion-14B-Video2world.** The top two rows show 5-second videos generated by the models, conditioned on the first 9 frames. The bottom two rows show long video generation results. We generate long videos in an autoregressive manner, where the first generated video is conditioned on a single input image, and the subsequent five videos are conditioned on their previous nine frames. Both of the 7B and 14B models produce photorealistic videos with high visual fidelity. The 14B model demonstrates the ability to generate more complex scenes and exhibits superior motion stability. To check full videos and more video examples, please visit our website.

captions based on our training long prompts and corresponding videos. This long-to-short data creation strategy is effective in (1) preserving the authentic video content and distribution from detailed training prompts of WFMs and (2) ensuring fidelity between the short and long prompts. The resulting prompt upsampler is termed Cosmos-1.0-PromptUpsampler-12B-Text2World.

Prompt upsampler for Video2World model. For the Video2World model, the input consists of video conditions and a user text prompt. To enhance the user prompt, we utilize an open-source VLM, Pixtral-12B (Agrawal et al., 2024), combined with zero-shot prompt engineering, to upsample the prompt into a detailed description that considers both the video conditions and the user prompt. We found the vanilla Pixtral-12B model works well out of the box and did not proceed to perform a similar fine-tuning described above.

5.1.6. Results

In Fig. 12, we present qualitative results generated by our Cosmos-1.0-Diffusion-7B-Text2World and Cosmos-1.0-Diffusion-14B-Text2World models. Both models produce videos of high visual quality, motion dynamics, and text alignment. Compared to the 7B model, the 14B model is able to generate videos capturing more

complex visual details and intricate motions.

We show generated videos from Video2World 7B and 14B models in Fig. 13. The Video2World models support both image and video conditioning and can generate extended videos in an autoregressive manner. As demonstrated in Fig. 13, our Video2World models produce photorealistic videos with good motion dynamics and visual fidelity. The 14B model, again, generates better videos in terms of scene richness and motion stability.

5.2. Autoregressive-based World Foundation Model

In autoregressive WFMs, we formulate world simulation generation as a next-token prediction task similar to language modeling. We start by converting a video into a sequence of discrete video tokens $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ using the Cosmos Discrete Tokenizer introduced in Sec. 4. Then we train a Transformer decoder (Vaswani et al., 2017) to predict the next video token using past video tokens as context, similar to large language models (LLMs) (Brown et al., 2020; Dubey et al., 2024; Jiang et al., 2023). Specifically, the training objective is to minimize the following negative log-likelihood (NLL) loss:

$$\mathcal{L}_{NLL} = \sum_i -\log P(v_i | v_1, v_2, \ldots, v_{i-1}; \Theta), \tag{9}$$

where the conditional probability P of the predicted next video token v_i is modeled by a Transformer decoder with parameters Θ.

5.2.1. Architecture

Our autoregressive-based WFM architecture is illustrated in Fig. 14. We make several modifications to the standard transformer model architecture tailored for our video generation task, including adding 1) 3D-aware positional embeddings, 2) cross-attention to enable textual inputs for better control, and 3) QK-Normalization (Wortsman et al., 2023).

3D positional embeddings. Similar to our diffusion-based WFM (Sec. 5.1.2), we incorporate two complementary positional embedding mechanisms: 3D factorized Rotary Position Embedding (RoPE) for relative positions and 3D factorized absolute positional embedding (APE) for absolute coordinates. These mechanisms work in concert to provide comprehensive spatial and temporal information throughout the network.

- **3D Rotary Position Embedding (RoPE).** We apply 3D RoPE to our model to encode relative positional information across the temporal, height, and width dimensions. During training, we adopt a multi-stage training strategy in which the sequence length of videos increases as the training progresses. To adapt the 3D RoPE to the changing temporal duration, we use YaRN (Peng et al., 2023), a compute-efficient technique designed to extend the context window of RoPE. We apply YaRN extension only along the temporal axis as the video sequence length increases only along the temporal dimension. By utilizing YaRN, our model can extrapolate to context lengths longer than those encountered during the initial stages of training.
- **3D Absolute Positional Embedding (APE).** In addition to 3D RoPE, we incorporate a 3D APE within each transformer block to complement the relative positional encoding. This APE encodes positional information using sinusoidal embeddings factorized across temporal, height, and width dimensions, ensuring the model is aware of absolute positions. The embedding is added directly to the input tensor at each stage, enriching the positional context for the transformer. We find combining absolute and relative positional encodings enhances model performance, reduces training loss, and minimizes morphing artifacts in generated videos. Notably, while our diffusion-based WFM (Sec. 5.1.2) employs learnable embeddings, we adopt sinusoidal-based embeddings for APE in our autoregressive-based WFM.

Vocabulary. Tokenization is a crucial step that turns input text into a sequence of discrete tokens in large language models (LLMs). In LLMs, the vocabulary of possible tokens is determined by the LLM's tokenizer

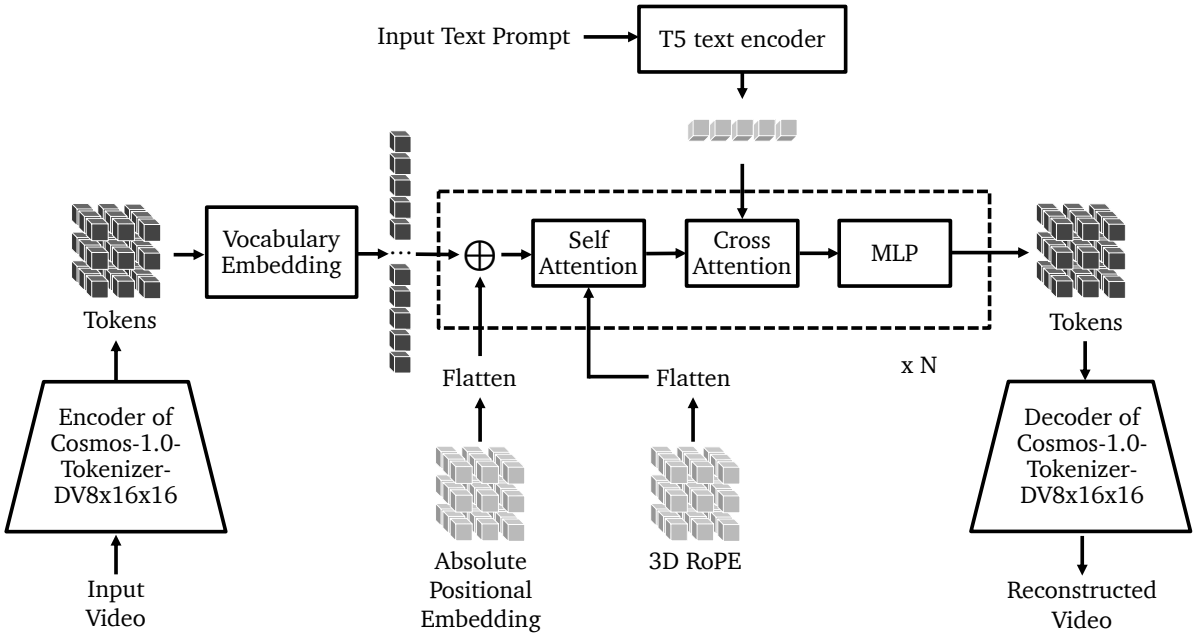


Figure 14: **Architecture of Cosmos-1.0-Autoregressive-Video2World Model.** The pipeline begins by encoding input video through the encoder of Cosmos-1.0-Tokenizer-DV8x16x16 to generate discrete tokens, which are transformed into learned embeddings. These embeddings are processed through repeated transformer blocks, each consisting of absolute positional embedding and 3D RoPE components that are flattened before entering the self-attention module. Each block also includes a cross-attention module that incorporates encoded text prompts (processed via a T5 text encoder), followed by a two-layer MLP. Finally, the decoder of Cosmos-1.0-Tokenizer-DV8x16x16 reconstructs the video from the output tokens.

(*e.g.*, tiktoken introduced by OpenAI (2022)) trained on a large corpus of text with algorithms such as Byte Pair Encoding (BPE) (Gage, 1994).

For our autoregressive models, we use our Cosmos-1.0-Tokenizer-DV8x16x16 as the tokenizer. As introduced in Sec. 4, we leverage the Finite-Scalar-Quantization (FSQ) (Mentzer et al., 2023) to quantize the 6-dimensional latent space into $(8, 8, 8, 5, 5, 5)$ levels. This quantization leads to a vocabulary size of $8\times8\times8\times5\times5\times5 = 64{,}000$.

Cross-attention for text conditioning. In addition to the self-attention blocks present in the transformer architecture, we add cross-attention layers to enable the model to condition on input text. Similar to diffusion-based WFM(Sec. 5.1.2), cross-attention is applied between the features of the transformer model and text embeddings obtained from a pre-trained text encoder (T5-XXL). In our experiments, we add cross-attention blocks after every self-attention layer.

Query-key normalization. In order to enhance training stability, we incorporate Query-Key Normalization (QKNorm) (Wortsman et al., 2023). QKNorm addresses instability in attention mechanisms by normalizing the query (Q) and key (K) vectors before computing their dot product, thereby preventing the softmax function from saturating and ensuring more effective learning. After normalization, the dot product is scaled by a learnable parameter γ instead of the fixed $1/\sqrt{d_k}$. This learnable scaling factor allows the model to adaptively control the magnitude of the attention scores, enhancing flexibility and expressivity.

Z-loss. To further improve training stability, we introduce a stabilization term known as the z-loss (de Brébisson and Vincent, 2016) into our training objective. The z-loss penalizes deviations of the logits from zero, effectively discouraging the model from generating excessively large logit values that could result in numerical instability or gradient explosions. The z-loss is defined as the sum of the squared logits as $\mathcal{L}_{\text{z-loss}} = \lambda \cdot \sum_i z_i^2$. We found z-loss to be critical in maintaining gradient norms to a healthy range, especially when scaling the training to a

large number of GPU nodes. Empirically, we found that the z-loss coefficient $\lambda = 3 \times 10^{-4}$ strikes an optimal balance, effectively stabilizing training without adversely affecting model performance.

5.2.2. Scaling Up

This section describes the techniques that enable efficient scaling of our autoregressive WFMs. We briefly analyze the memory consumption of our models, discuss parallelism strategies, and compare our training setup with other autoregressive models.

Memory requirements. During training, GPU memory is mainly consumed by:

- **Model parameters**: 6 bytes per parameter. We store the model parameters in both BF16 and FP32.
- **Gradients**: 2 bytes per parameter. We store the gradients in BF16.
- **Optimizer states**: 8 bytes per parameter. We store the first and second moments of AdamW (Loshchilov and Hutter, 2019) both in FP32.
- **Activations**: Approximately $(2 \times \text{number_of_layers} \times 17 \times \text{seq_len} \times \text{batch_size} \times \text{d_model})$ bytes. We refer readers to Korthikanti et al. (2023) for a detailed analysis of activation memory of state-of-the-art autoregressive models.

For instance, our 12B model (Cosmos-1.0-Autoregressive-12B) demands approximately 192 GB of memory for its parameters, gradients, and optimizer states combined. As this is beyond a single NVIDIA H100 GPU's 80GB HBM3 capacity, we leverage tensor parallelism (TP) (Shoeybi et al., 2019) and its extension, sequence parallelism (SP) (Korthikanti et al., 2023), to distribute the memory requirements and computation across multiple GPUs.

Tensor Parallelism (TP). Tensor Parallelism (TP) (Shoeybi et al., 2019) splits the weights of linear layers along either the input or output feature dimensions, with the choice guided by the goal of minimizing inter-GPU communication. For example, in a two-layer feedforward network, the weights of the first layer are partitioned along the output feature dimension, while those of the second layer are partitioned along the input feature dimension. This arrangement allows intermediate activations to be processed locally without requiring communication between GPUs. The final outputs are then combined using all-reduce communication. By employing TP, each GPU stores only a fraction, specifically $1 / \text{TP_SIZE}$, of the weights for linear layers. However, the default implementation of TP still replicates activations along the sequence dimension for operations like LayerNorm, resulting in redundancy.

Sequence Parallelism (SP). SP (Korthikanti et al., 2023) extends Tensor Parallelism by further partitioning the context along the sequence dimension. This approach is applicable to operators, such as LayerNorm and Dropout in self-attention layers, where each element in the sequence can be processed independently. With SP enabled. Each GPU stores only a fraction, specifically 1 / TP_SIZE, of the activations.

Comparison with other autoregressive models. Compared to popular LLMs, our model doesn't leverage memory-saving attention variants such as MQA or GQA. Otherwise, our autoregressive model is deliberately designed to closely resemble the architecture of LLMs (Adler et al., 2024; Brown et al., 2020; Dubey et al., 2024; Jiang et al., 2023; Team, 2024; Yang et al., 2024), as this alignment offers flexibility and scalability. Experiments that leverage more parallelisms, such as context parallelism and pipeline parallelism, to further scale up the model sizes and context lengths are left for future works.

5.2.3. Training Strategy

We perform pre-training of our autoregressive WFMs in multiple stages.

- **Stage 1**: In the first stage, the model is trained using the video prediction objective. Given the first frame as the input condition, the model is trained to predict future video frames. A context length of 17 frames is used for this task, *i.e.*, the model predicts 16 future frames with the first frame as input.

- **Stage 1.1**: This stage performs video prediction but with an increased context length of 34 frames. We use the YaRN extension on the temporal dimension to increase the context length of RoPE.
- **Stage 2**: In stage 2 of our training, we introduce text conditioning to our model. Text embeddings are incorporated using newly initialized cross-attention layers. The model is trained with a 34-frame context. To improve text-to-video generation ability, the model is trained using joint image and video data as described in Sec. 5.1.3. When image batches are used, we use a larger batch size as the context length for images is much smaller than that of videos.

All our models are trained with a fixed spatial resolution of 640×1024.

Cooling down. After pre-training, we conduct a "cooling-down" phase with high-quality data, similar to LLM training practices (Dubey et al., 2024). During this phase, we linearly decay the learning rate to 0 while training on high-quality image-video pairs. The cooling-down phase is carried out over $30{,}000$ iterations.

Table 14: Configuration details of Cosmos-1.0-Autoregressive models.

Configuration	4B	5B-Video2World	12B	13B-Video2World
Number of Layers	16	16	40	40
Model Dimension	4,096	4,096	5,120	5,120
Cross Attention Layers	✗	✓	✗	✓
Base Learning Rate	1×10^{-3}	3×10^{-4}	1×10^{-3}	5×10^{-4}
Weight Decay	0.01			
Learning Rate Warmup	Linear scheduler with 5,000 iterations			
Activation Function	SwiGLU			
FFN Hidden Dimension	14,336			
Number of Attention Heads	32			
Number of Key / Value Heads	8			
Number of Tokens	12,800			
Vocabulary Size	64,000			
Positional Embedding	3D RoPE ($\theta = 500{,}000$) + 3D APE			

We train two sets of autoregressive-based WFMs. We start by building two base models: one with a 4B capacity and the other with a 12B capacity. These are pure next-video token predictors that do not take text prompts as input. We then derive a Video2World version from each of the base models, where we add cross-attention layers to them to leverage text prompt inputs for next video token prediction.

- **Cosmos-1.0-Autoregressive-4B**: a 4B transformer model for next video token prediction. This model is trained using stage 1 and stage 1.1 of the multi-stage training objective.
- **Cosmos-1.0-Autoregressive-5B-Video2World**: a 5B transformer model derived from our Cosmos-1.0-Autoregressive-4B and trained additionally with stage 2 of the multi-stage training objective.
- **Cosmos-1.0-Autoregressive-12B**: a 12B transformer model for next video token prediction. This model is trained using stage 1 and stage 1.1 of the multi-stage training objective.
- **Cosmos-1.0-Autoregressive-13B-Video2World**: a 13B transformer model derived from Cosmos-1.0-Autoregressive-12B and trained additionally with stage 2 of the multi-stage training objective.

5.2.4. Inference Optimization Towards Real-Time Generation

Our Cosmos Autoregressive WFMs share architectural similarities with LLMs, enabling us to leverage established LLM inference optimization techniques to address the sequential decoding bottleneck. We implement a combination of key-value caching, tensor parallelism, and torch.compile, following the gpt-fast[3] implementation in PyTorch (Paszke et al., 2019).

Speculative decoding. To further accelerate our autoregressive WFMs, we apply the Medusa speculative decoding framework (Cai et al., 2024). Unlike common speculative decoding approaches that require a separate

[3] https://github.com/pytorch-labs/gpt-fast

draft model (Leviathan et al., 2023) or training-free methods with limited speedup (Teng et al., 2024), Medusa extends the transformer backbone with extra decoding heads to predict multiple subsequent tokens in parallel. It then verifies these speculated tokens with rejection sampling. The inference is thus accelerated by alleviating the bottleneck of one-token-at-a-time processing. We demonstrate the potential of the Medusa technique in visual autoregressive acceleration without compromising the quality of generated outputs.

In our implementation, we fine-tune our pre-trained autoregressive WFMs by introducing Medusa heads into the architecture. These heads are strategically inserted after the last transformer hidden states, where all backbone parameters and the final unembedding layer are shared across different heads. Each Medusa head is a single-layer FFN with SiLU activation and residual connection. We further merge the weight matrices of multiple Medusa heads into a unified FFN to maximize parallelism during token prediction. Note that we do not use the tree-based attention mechanism from Cai et al. (2024).

To investigate the optimal Medusa setup for our autoregressive WFMs, we conduct an in-depth study from two aspects: (1) which transformer layers to fine-tune and (2) how many Medusa heads to add. For the first problem, we compare between full fine-tuning and selective layer freezing. We observe that only fine-tuning the Medusa heads gives poor multi-token prediction, while full fine-tuning incurs quality degradation. We empirically identify that unfreezing the last two transformer layers and the final unembedding layer while keeping the backbone frozen yields the best performance. This strategy ensures our Medusa training achieves decent speculative decoding accuracy without suffering from catastrophic forgetting.

Table 15: Impact of Medusa head number on average token throughput and number of forward passes. The experiments are conducted on 8 $\times$ H100 GPUs and 50 unseen test videos of 640×1024 resolution.

Model	Medusa Head Number	0	3	6	9	12
4B	Token Throughput (tokens/s)	444.95	663.51	829.59	**894.67**	890.64
	# of Forward Passes	7680	2860	2073	1812	**1682**
5B	Token Throughput (tokens/s)	303.61	659.94	758.58	**982.77**	978.80
	# of Forward Passes	10240	2857	2382	1799	**1673**

To explore the optimal number of Medusa heads, we calculate the model token throughput and forward pass count with different numbers of Medusa heads. The ablation studies are conducted on 8 $\times$ H100 GPUs and evaluated on 50 unseen test videos of 640×1024 resolution. The results in Tab. 15 suggest that our Medusa framework can effectively accelerate inference, with up to $2.0\times$ token throughput and $4.6\times$ less forward pass for the 4B model, and up to $3.2\times$ token throughput and $6.1\times$ less forward pass for the 5B model. We show that though more Medusa heads can reduce the number of forward passes needed to generate, it may slow down the overall token throughput. We find that 9 Medusa heads yield the best trade-off between computational efficiency and model performance.

In Tab. 16, we show performance analysis of autoregressive WFMs with Medusa integration. This analysis was conducted on H100 GPUs and evaluated on test videos of 640×1024 resolution in the BF16 precision. Results show that the Medusa implementation consistently accelerates inference for both 4B and 5B models under different GPU configurations.

Low-resolution adaptation for real-time inference. We pursue real-time inference by adapting our model to a lower spatial resolution of 320×512, which results in a lower number of tokens per video. Specifically, we first fine-tune the discrete video tokenizer (Cosmos-1.0-Tokenizer-DV8$\times$16$\times$16 in Sec. 4) on 320p low-resolution videos using videos from the target Physical AI domain. Then, we fine-tune our autoregressive WFM that is pre-trained in 640×1024 resolution (Cosmos-1.0-Autoregressive-4B in Sec. 5.2.3) with this low-resolution tokenizer on videos of 320×512 resolution from the target Physical AI domain. Finally, we add the Medusa heads to the fine-tuned low-resolution autoregressive WFM.

Table 16: Performance analysis of Cosmos Autoregressive Models on test videos of 640×1024 resolution.

Model	GPU	No DD (s)	No DD+Medusa (s)	With DD (s)	With DD+Medusa (s)	VRAM (GB)
4B	1	31.04	23.52	61.49	53.08	29
	4	18.20	13.87	29.60	25.63	31
	8	17.62	9.91	30.30	22.83	34
5B	1	39.68	24.97	70.39	54.72	59
	4	25.59	20.96	37.29	33.35	51
	8	25.70	11.67	38.41	24.35	49
12B	1	84.78	—	116.66	—	45
	4	47.49	—	60.27	—	36
	8	45.69	—	58.81	—	37
13B	1	109.18	—	140.24	—	77
	4	67.80	—	80.76	—	55
	8	67.22	—	80.93	—	55

The table reports the average inference time (in seconds) and VRAM utilization of various Cosmos Autoregressive WFMs under different settings. Inference time is reported for generating 32 frames with a single conditioning frame as input. **No DD**: Time without diffusion decoder. **No DD+Medusa**: Time without diffusion decoder but with Medusa heads. **With DD**: Time with diffusion decoder. **With DD+Medusa**: Time with diffusion decoder and Medusa heads. **VRAM**: Video RAM usage in gigabytes.

Table 17: Decoding throughput of Cosmos-1.0-Autoregressive-4B with low-resolution adaptation, benchmarked on 8 $\times$ H100 80GB GPUs using 10-FPS videos of 320×512 resolution from the Physical AI domain.

Model (320×512)	Token Throughput (tokens/s)	Video Throughput (frames/s)
Cosmos-1.0-Autoregressive-4B (with Medusa)	806.61	10.08

We conducted inference benchmarking on 8 $\times$ H100 GPUs using torch.compile's "max-autotune" mode in BF16 precision, and evaluated with 10-FPS input videos from the target Physical AI domain. In Tab. 17, we report the average token throughput and frame generation speed achieved in this setup. We observe that our model can generate 10 video frames in less than 1 second, demonstrating that we can achieve real-time video generation at 10 FPS.

5.2.5. Diffusion Decoder

Our Cosmos tokenizer uses a lightweight encoder-decoder architecture to perform aggressive compression, which reduces the number of tokens for our WFM training. As a result of aggressive compression, it could sometimes lead to blurriness and visible artifacts in video generation, especially in the autoregressive WFM setting, where only a few integers are used to represent a rich video through discrete tokenization. We resort to the diffusion decoder design (OpenAI, 2024; Ramesh et al., 2022) to address the limitation. Specifically, we build a more powerful tokenizer decoder by fine-tuning Cosmos-1.0-Diffusion-7B-Text2Video in Sec. 5.1.

Fig. 15 illustrates how we train a diffusion decoder for our autoregressive WFMs. For each training video, we use Cosmos-1.0-Tokenizer-CV8x8x8 and Cosmos-1.0-Tokenizer-DV8x16x16 to compute a continuous token video and a corresponding discrete token video, respectively. We note that Cosmos-1.0-Tokenizer-CV8x8x8 can produce higher quality video outputs than Cosmos-1.0-Tokenizer-DV8x16x16 thanks to the more gentle continuous tokenization process and the less aggressive compression scheme ($8 \times 8 \times 8$ instead of $8 \times 16 \times 16$).

The discrete token video is treated as the conditional input to the denoiser of the Cosmos-1.0-Diffusion-7B model. To compute the conditional input, we first embed each discrete token of the discrete token video into a 16-dimensional vector based on a learnable vocabulary embedding layer. We then upsample the embedding $2\times$ along the x and y directions so that the conditional input will be of the same size as the noisy input to the denoiser from the continuous token video. We concatenate the noisy continuous inputs with the conditional

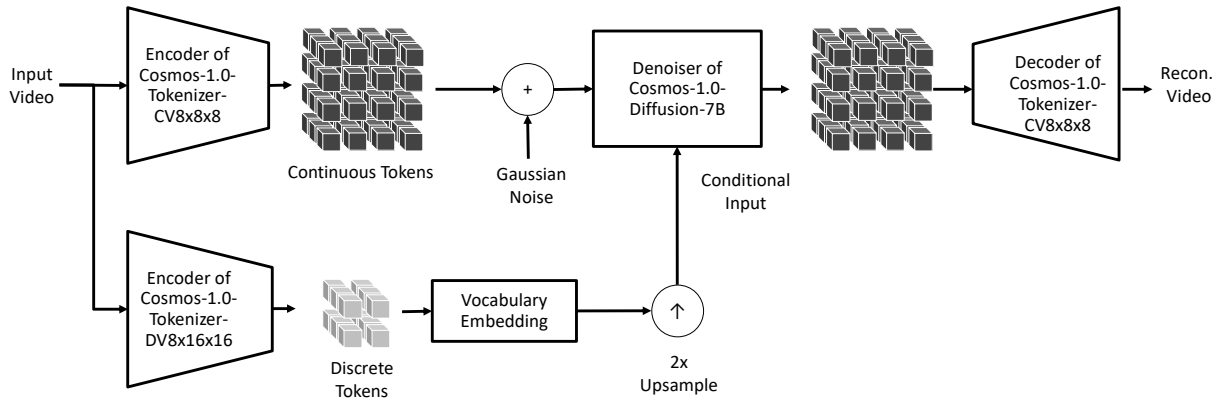


Figure 15: **Cosmos diffusion decoder training**. During training, each input video is tokenized twice: one through the target discrete tokenizer (DV8x16x16) and the other through a less-constrained continuous tokenizer (CV8x8x8). The discrete token video is used as the conditional input to the diffusion denoiser.

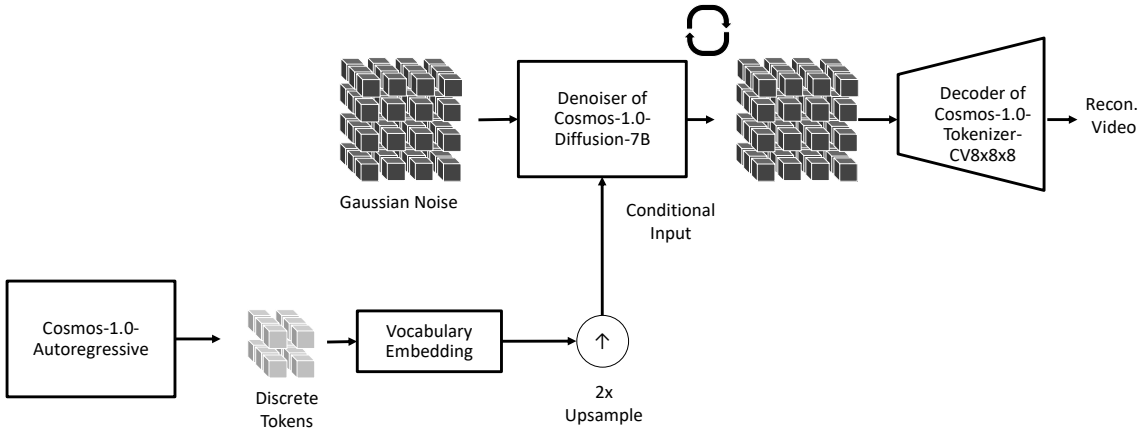


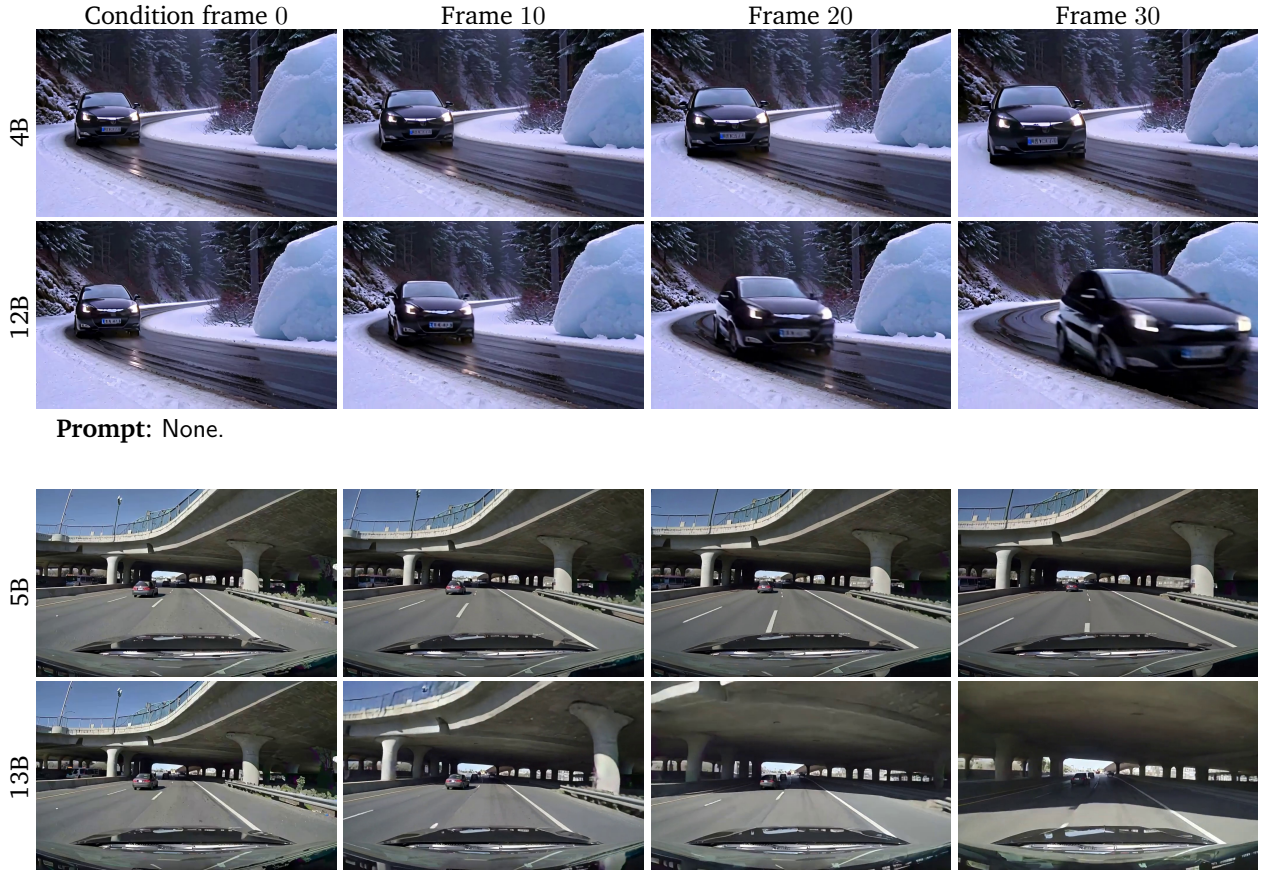
Figure 16: **Cosmos diffusion decoder inference**. During inference, the output video token from a Cosmos-1.0-Autoregressive model is conditionally inputted to the denoiser.

inputs along the channel dimension, which becomes the input to the diffusion denoiser. The first layer of the denoiser is channel-dimension expanded to accommodate the new input shape. We fine-tune the updated Cosmos-1.0-Diffusion-7B by removing the added noise. As the discrete token video is not noise-corrupted, the denoiser learns to leverage the residing information in the conditional input for denoising. The result is a higher-quality decoder for the tokenizer that decodes the discrete token by solving a reserve diffusion problem.

Fig. 16 illustrates the inference. The output discrete token video (under $8 \times 16 \times 16$ discrete compression) from our autoregressive WFM is decoded into a video through two steps. First, we roll out the conditional denoiser to generate a continuous token video (under $8 \times 8 \times 8$ continuous compression) based on the autoregressive WFM output. Next, the continuous token video is decoded by Cosmos-1.0-Tokenizer-CV8x8x8 to produce the resulting RGB video.

5.2.6. Results

In Fig. 17, we show qualitative results of our autoregressive WFMs using different model sizes. In the unprompted setting, comparing Cosmos-1.0-Autoregressive-4B and Cosmos-1.0-Autoregressive-12B model,



Prompt: None.

Prompt: The video of a car moving forward, passing under a large overpass. The road is clear, and there are a few other cars visible in the distance. The weather appears to be sunny, and the time of day is daytime. The scene is set on a busy highway with concrete structures and greenery on the sides.

Figure 17: **Generated videos from Cosmos Autoregressive World Foundation Models.** The top two rows are the video generation results with 4B and 12B models, while the bottom two rows are the Video2World results with a text prompt. We observe that the 12B and 13B models shows sharper videos and better motion than 4B and 5B models in both prompted and unprompted settings. To check full videos and more video examples, please visit our website.

we observe that the 12B model generates videos with better motion and sharper details. Similarly, in the prompted setting, comparing Cosmos-1.0-Autoregressive-5B-Video2World and Cosmos-1.0-Autoregressive-13B-Video2World reveals that the 13B model gets better motion than the 5B model.

In Fig. 18, we show the enhancements obtained when using the diffusion decoder. The outputs of the autoregressive model are blurry mainly due to the lossy compression in our discrete tokenizer. The use of the diffusion decoder can enhance details while preserving the content.

We empirically find the outputs of the autoregressive-based Text2World WFMs do not improve with upsampled prompts from the prompt upsampler discussed in Sec. 5.1.5. We hypothesize this is possibly due to the fact that these WFMs are pretrained with pure video generation tasks for most of the training. They are not forced hard enough to leverage text inputs.

5.2.7. Limitations

One notable failure case observed in the generated videos of our autoregressive WFMs is objects unexpectedly appearing from below. Fig. 19 illustrates an example of this issue. To understand the failure rate of our models, we conduct a systematic study by creating an evaluation set of 100 Physical AI inputs to our autoregressive

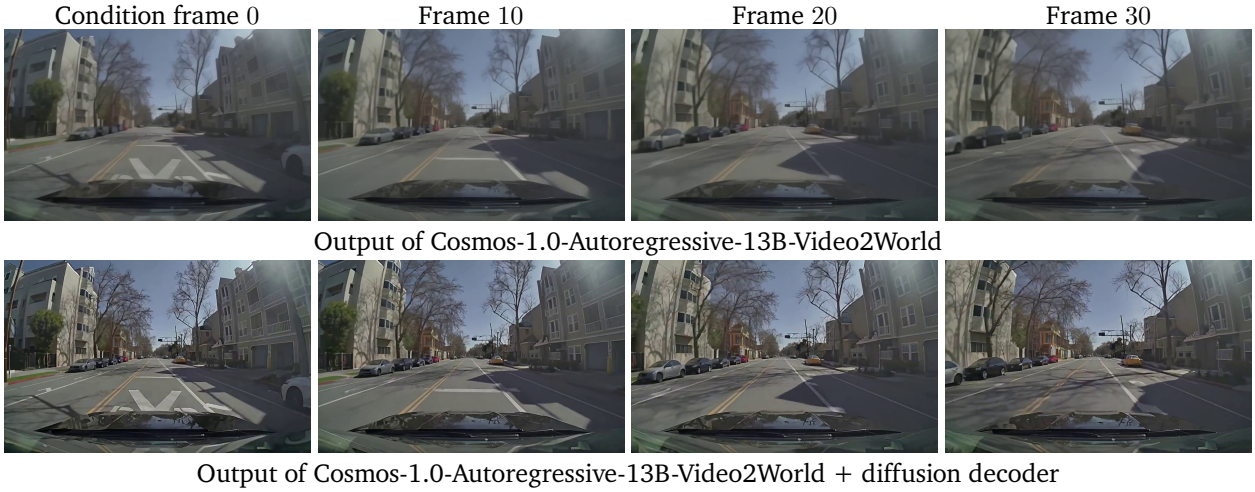


Figure 18: **Diffusion decoder comparison**. In the top panel, we show the video generation results with Cosmos-1.0-Autoregressive-13B-Video2World model. In the bottom panel, we show the enhanced video after the output from the autoregressive model is passed through the diffusion decoder. We observe that the autoregressive model alone produces blurry results, while the diffusion decoder can enhance the sharpness of videos while preserving the content.



Figure 19: **Failure cases of Cosmos Autogressive WFMs**. We observe failure cases in generated videos in which some objects (shown in red) unexpectedly appear from below.

WFMs. We generate videos with all our models using two input modes—image (single-frame) conditioning and video (9-frame) conditioning. For all generated videos, we manually inspect the failure cases and report the failure rate in Tab. 18. We observe that the smaller models Cosmos-1.0-Autoregressive-4B and Cosmos-1.0-Autoregressive-5B-Video2World show a higher corruption rate in single frame conditioning, while the larger models Cosmos-1.0-Autoregressive-12B and Cosmos-1.0-Autoregressive-13B-Video2World are more robust. Generation with 9-frame video conditioning is stable for all models, with a failure rate lower than 2%.

5.3. Evaluation

Pre-trained WFMs are generalists of visual world simulation. Their capabilities should be measured across multiple aspects. Here, we evaluate our models on two aspects. First, we evaluate the 3D consistency of the generated videos. An ideal WFM should generate video simulations from geometrically plausible 3D worlds. Second, we evaluate the physics alignment of the generated videos. We calculate how well the rendered dynamics adhere to the laws of physics. Evaluation of WFMs is a highly nontrivial task. We acknowledge that there are several other important aspects required for evaluation. We leave a more comprehensive evaluation as future work.

5.3.1. 3D Consistency

WFMs are designed to simulate 3D worlds through video generation, and it is essential to evaluate how well the generated videos are consistent with the 3D structure of the visual world. In addition to appearing realistic,

Table 18: Failure rate analysis of Cosmos Autoregressive models.

Model	Image Conditioning	Video Conditioning (9 frames)
Cosmos-1.0-Autoregressive-4B	15%	1%
Cosmos-1.0-Autoregressive-5B-Video2World	7%	2%
Cosmos-1.0-Autoregressive-12B	2%	1%
Cosmos-1.0-Autoregressive-13B-Video2World	3%	0%

the generated videos should maintain coherence with the physical principles of scenes through time, a key requirement for downstream Physical AI applications.

Test data and baseline model. We focus on the scenario of static scenes in order to effectively measure 3D consistency of videos with existing tools based on multi-view geometry. We curate a dataset of 500 videos randomly chosen from the test set of the RealEstate10K dataset (Zhou et al., 2018). We additionally caption the videos using a proprietary VLM to obtain text prompts that describe the videos as static scenes, so one does not need to consider scene motions for metric computation. We compare against VideoLDM (Blattmann et al., 2023) as the baseline method.

Metrics. Generated videos are effectively 2D projections of the underlying 3D visual worlds. We design the following metrics to measure the 3D consistency of generated videos.

1. **Geometric consistency**. We evaluate the 3D consistency of our generated worlds by quantifying how the epipolar geometry constraints are satisfied, including the Sampson error (Hartley and Zisserman, 2003; Sampson, 1982) and the success rate of camera pose estimation algorithms (Schönberger et al., 2016; Schönberger and Frahm, 2016) on the generated videos.
2. **View synthesis consistency**. We evaluate the ability of world foundation models to synthesize images at interpolated novel viewpoints while maintaining coherence with the underlying 3D structure.

The Sampson error is the first-order approximation of the distance from one interest point to its corresponding epipolar line in another view. Given N point correspondences (represented in homogeneous coordinates) $\{(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i)\}_{i=1}^N$ in a given frame pair, we define the Sampson error as

$$\epsilon_{\text{samp}} = \frac{1}{N}\sum_{i=1}^{N} \frac{|\bar{\mathbf{y}}_i^\top \mathbf{F} \bar{\mathbf{x}}_i|}{\sqrt{\|\mathbf{S}\mathbf{F}\bar{\mathbf{x}}_i\|_2^2 + \|\mathbf{S}\mathbf{F}^\top \bar{\mathbf{y}}_i\|_2^2}}, \quad \text{where } \mathbf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{10}$$

and $\mathbf{F}$ is the fundamental matrix estimated from the correspondences. We use the square root version of the error function to make the metric more intuitive in pixel units. We use a combination of SuperPoint (DeTone et al., 2018) and LightGlue (Lindenberger et al., 2023) to detect and match keypoint correspondences from a frame pair and estimate $\mathbf{F}$ using OpenCV's 8-point RANSAC algorithm. We normalize the average error by the diagonal length of the frame with respect to a 960×540 canvas.

We also evaluate 3D consistency of a generated video with its ability to self-synthesize novel viewpoints. Following the common practice of novel view synthesis literature (Mildenhall et al., 2020), we hold out every 8 frames as the test frames and fit a 3D Gaussian splatting model (Kerbl et al., 2023) with the rest of the training frames using the default settings from the Nerfstudio library (Tancik et al., 2023). We report the Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and LPIPS (Zhang et al., 2018) as the metrics to quantify the quality of the synthesized test views.

Results. We present the quantitative evaluation results in Tab. 19. The Cosmos WFMs achieve significantly better 3D consistency than our baseline model in terms of both geometric and view synthesis consistency. Not only are the interest points from Cosmos WFMs more 3D-consistent, but the camera pose estimation

Table 19: Evaluation of 3D consistency on base Cosmos models.

	Geometric Consistency		View Synthesis Consistency		
Method	Sampson error ↓	Pose estimation success rate (%) ↑	PSNR ↑	SSIM ↑	LPIPS ↓
VideoLDM (Blattmann et al., 2023)	0.841	4.4%	26.23	0.783	0.135
Cosmos-1.0-Diffusion-7B-Text2World	**0.355**	62.6%	**33.02**	**0.939**	**0.070**
Cosmos-1.0-Diffusion-7B-Video2World	0.473	**68.4%**	30.66	0.929	0.085
Cosmos-1.0-Autoregressive-4B	0.433	35.6%	32.56	0.933	0.090
Cosmos-1.0-Autoregressive-5B-Video2World	0.392	27.0%	32.18	0.931	0.090
Real Videos (Reference)	0.431	56.4%	35.38	0.962	0.054

success rate is also notably higher, reflecting both improved overall quality and enhanced 3D consistency, even reaching the level of real-world videos. Among the cases where camera poses were successfully estimated, the synthesized held-out views demonstrate higher quality across all image synthesis metrics. These results highlight the capability of our Cosmos WFMs to generate 3D-consistent videos, establishing them as effective world simulators.

5.3.2. Physics Alignment

An ideal WFM should exhibit a strong understanding of the laws of physics and produce future observations that respect them. While our pre-trained WFMs exhibit a certain level of physics understanding and advance the state-of-the-art, one can still easily generate examples that do not obey the law of physics. We believe additional steps in data curation where physically implausible videos are removed are required, as well as improved model design. While we leave a strong physics-aligned WFM as future work, we are still interested in measuring how much intuitive physics naturally emerges from large-scale data-driven pre-training.

To explore this, we design a controlled benchmark dataset using a physics simulation engine, taking inspiration from (Kang et al., 2024). We generate physics-grounded simulations to test the adherence of our pre-trained WFMs to Newtonian physics and rigid body dynamics. Specifically, we use simulation to generate physically correct photorealistic videos of test scenarios specific to physical laws of interest. These reference "ground truth" videos are then compared with "predicted" videos produced by a WFM given shared context (past observations and perturbation).

Synthetic data generation. Using PhysX (NVIDIA, 2024) and Isaac Sim (NVIDIA, 2024), we design eight 3D scenarios aimed at evaluating different physical effects:

1. **Free-falling object(s)**: objects dropping on a plane (gravity, collision, *etc*.)
2. **Tilted planar slope**: objects rolling down an incline (gravity, moment of inertia, *etc*.)
3. **U-shaped slope**: objects rolling down a U-shaped slope (potential, kinetic energy, *etc*.)
4. **Stable stack**: a stack of objects in equilibrium (balanced forces)
5. **Unstable stack**: a stack of objects in imbalance (gravity, collision, *etc*.)
6. **Dominoes**: sequence of rectangular bricks falling in sequence (transfer of momentum, collision, *etc*.)
7. **Seesaw**: objects on either side of a seesaw (torque, rotational inertia, *etc*.)
8. **Gyroscope**: a spinning top on a flat surface (angular momentum, precession, *etc*.)

For each scenario, we randomize the number and type of dynamic objects (varying sizes, textures, shapes), selecting from Omniverse assets (NVIDIA, 2024), as well as the background appearance. We simulate the kinematic state of objects over time and render the output videos from 4 different static camera views. In total, we render 800 1080p videos of 100 frames in length. The objects in each simulation roll-out are positioned so that they are all visible from the first frame to avoid any existence ambiguity.

Metrics. We are interested in assessing the adherence to physical laws by comparing the simulated ground-truth

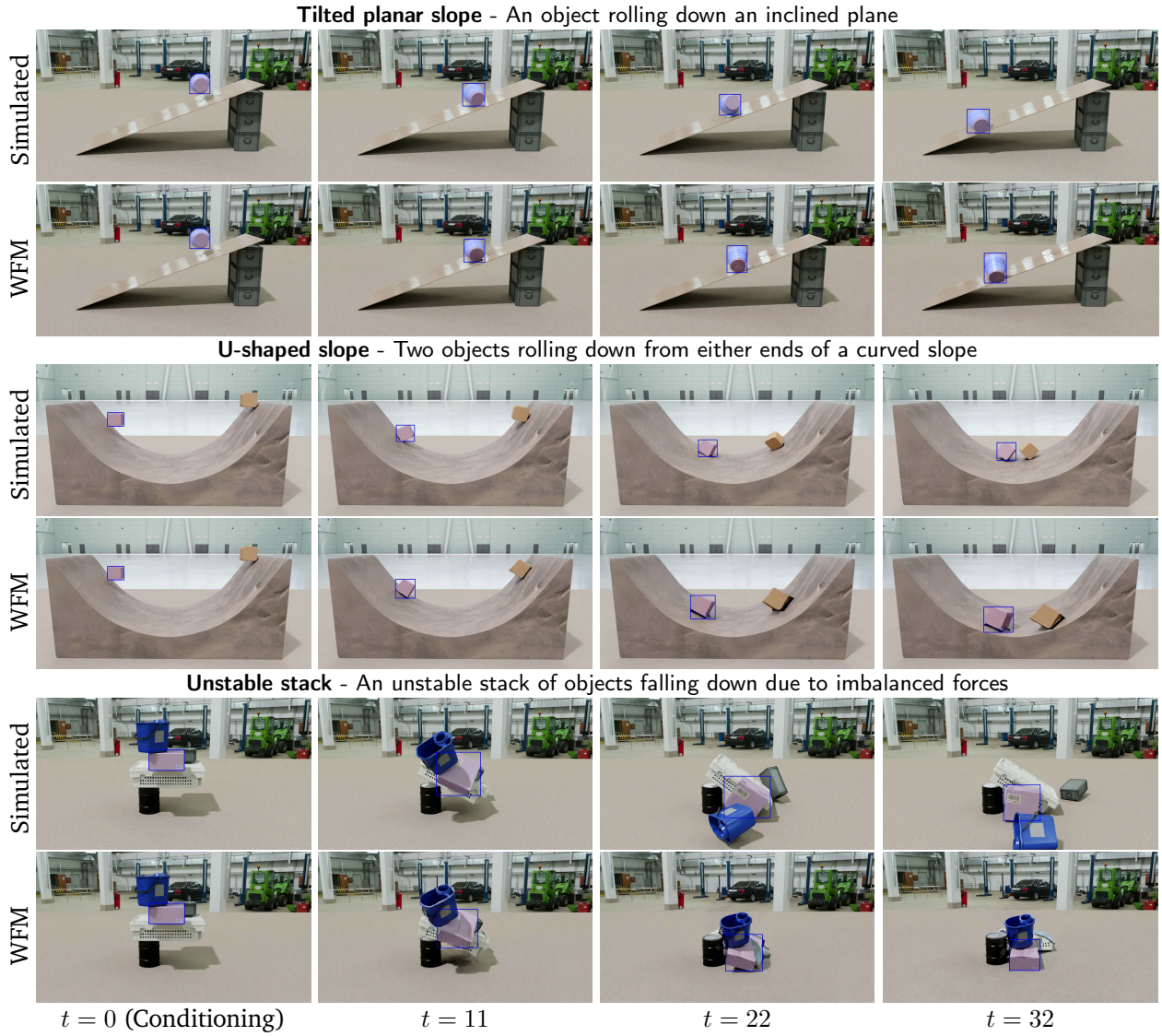


Figure 20: **Physics-scenario rollouts in simulation *vs*. pre-trained WFM.** We demonstrate three exemplar scenarios of increasing complexity as obtained from the reference (physically correct) simulation (first row in each group) and Cosmos-1.0-Diffusion-7B-Video2World rollouts (second row in each group). We condition the WFM on 9 frames and a prompt focusing on the kinematic state of the simulated objects. We show one tracked object (blue bounding box and mask) per example used to compute our object-level metrics (average IOU).

Table 20: Physics alignment results. We compare different variants of Cosmos WFMs in terms of accurate future prediction of a physical scenario using the pixel-level, feature-level, and object-level metrics. Metrics are calculated over 33 frames, the maximum length supported by the autoregressive variants of the Cosmos WFMs.

		Pixel-level		**Feature-level**	**Object-level**
Model	Condition(s)	PSNR ↑	SSIM ↑	DreamSim ↑	Avg. IoU ↑
Cosmos-1.0-Diffusion-7B-Video2World	prompt + 1 frame	17.34	0.538	0.836	0.332
Cosmos-1.0-Diffusion-7B-Video2World	prompt + 9 frames	**21.06**	**0.691**	0.859	0.592
Cosmos-1.0-Diffusion-14B-Video2World	prompt + 1 frame	16.81	0.521	0.836	0.338
Cosmos-1.0-Diffusion-14B-Video2World	prompt + 9 frames	20.21	0.635	0.860	**0.598**
Cosmos-1.0-Autoregressive-4B	1 frame	17.91	0.486	0.827	0.394
Cosmos-1.0-Autoregressive-4B	9 frames	18.13	0.482	0.859	0.481
Cosmos-1.0-Autoregressive-5B-Video2World	prompt + 1 frame	17.67	0.478	0.818	0.376
Cosmos-1.0-Autoregressive-5B-Video2World	prompt + 9 frames	18.29	0.481	0.864	0.481
Cosmos-1.0-Autoregressive-12B	1 frame	17.94	0.486	0.829	0.395
Cosmos-1.0-Autoregressive-12B	9 frames	18.22	0.487	**0.869**	0.487
Cosmos-1.0-Autoregressive-13B-Video2World	prompt + 1 frame	18.00	0.486	0.830	0.397
Cosmos-1.0-Autoregressive-13B-Video2World	prompt + 9 frames	18.26	0.482	0.865	0.482

video to the output directly generated by the WFM. Therefore, to produce future observations, we condition our WFMs on the first few frames (either 1 or 9 frames) of the ground truth video. When applicable, we additionally condition a WFM on a text prompt (obtained using a proprietary VLM by captioning the conditioning frames), focusing on the kinematic state of the objects being simulated in the past observations. Please refer to Fig. 20 for some examples of simulated versus predicted scenarios. For evaluation, we use the following metrics:

1. **Pixel-level metrics**. For a pixel-level comparison, we compute the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) to compare a predicted frame from the WFM rollout with the reference frame from the ground truth video.
2. **Feature-level metrics**. For a slightly higher-level semantic comparison, we calculate DreamSim similarity scores (Fu et al., 2023), a feature similarity metric, between the predicted and reference frames.
3. **Object-level metrics**. Finally, since we care most about how objects of interest are impacted by the ongoing physical phenomenon, we use tracking to compute object-level metrics that eliminate confounders (background changes, visual quality, *etc.*). Since the test conditions are synthetically generated, we have access to the ground-truth instance segmentation masks of the dynamic objects in the scenes. Using SAMURAI (Yang et al., 2024), we propagate the ground-truth instance masks in the first frame through the rest of the predicted video frames to extract tracks, allowing us to quantify object-level metrics. We compute the intersection-over-union (IoU) between ground truth and predicted object masks for each frame and object of interest.

We average these metrics across frames in a video, across videos in the evaluation set, and across four random seeds for rollouts. PSNR and SSIM are computed on all frames, excluding the ones used for conditioning.

Results. Quantitative results on physical alignment are outlined in Tab. 20. Based on quantitative and qualitative results, we make the following observations. Unsurprisingly, the models are able to better predict the overall object kinematics with more frames as conditioning input (which allows us to better infer 1st and 2nd order quantities such as speed and acceleration).

From the table, we also find that our diffusion WFMs perform better in pixel-level prediction than our autoregressive WFMs on the 9-frame conditional setting. This correlates with our visual observation that the diffusion-based WFMs render videos with higher visual quality. We also note that our results do not suggest that the larger model performs better on our physics alignment. While we observe larger models render videos with

higher visual quality, all the WFMs equally struggle with physics adherence and require better data curation and model design.

More generally, we observe that the rigid-body simulations described above already test the limits of our WFMs, serving as valuable tools for identifying specific failure cases. These range from low-level issues like object impermanence (spontaneous appearance and disappearance of objects) and deformation (shape changes) to more complex problems such as implausible kinematics, violation of gravity, *etc*. We believe such structured simulations offer a useful methodology to test physics alignment. We, therefore, intend to improve them over time by incorporating more complex scenarios, enhancing photorealism to bridge the sim-to-real gap (since WFM pre-training data consists of real videos), and refining our evaluation metrics for a more comprehensive assessment of physical understanding.

6. Post-trained World Foundation Model

In this section, we demonstrate how our Cosmos WFMs can be fine-tuned to support diverse Physical AI applications. We include examples from post-training our WFM with camera control to achieve 3D navigable visual world generation, post-training our WFM with action control on two different robotic setups for two different robotic manipulation tasks, and post-training our WFM with multi-view support for training autonomous driving agents.

Table 21: A map of Post-trained WFMs discussed in Sec. 6.

Section	Model	Condition(s)
Sec. 6.1	Cosmos-1.0-Diffusion-7B-Video2World-Sample-CameraCond	Text + Image + Cameras
Sec. 6.2	Cosmos-1.0-Autoregressive-7B-Video2World-Sample-Instruction	Text + Video
Sec. 6.2	Cosmos-1.0-Diffusion-7B-Video2World-Sample-Instruction	Text + Video
Sec. 6.2	Cosmos-1.0-Autoregressive-7B-Video2World-Sample-ActionCond	Action + Video
Sec. 6.2	Cosmos-1.0-Diffusion-7B-Video2World-Sample-ActionCond	Action + Video
Sec. 6.3	Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView	Text
Sec. 6.3	Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView-TrajectoryCond	Text + Trajectory
Sec. 6.3	Cosmos-1.0-Diffusion-7B-Video2World-Sample-MultiView	Text + Video

Tab. 21 provides a list of the discussed post-trained WFMs in different subsections of this section. We also list the conditional inputs to highlight the operation mode. Note that for each model, we add "-Sample" to emphasize our goal is to provide sample applications of our pre-trained WFMs. Those models are by no means a complete system or a production model for any real-world applications. The developer would need to fine-tune the WFMs on their custom datasets for their Physical AI setups for their target applications.

6.1. Post-training WFM for Camera Control

Through camera pose conditioning, we integrate camera control into Cosmos-1.0-Diffusion-7B-Video2World, making it an effective 3D world simulator. We term the result post-trained WFM as Cosmos-1.0-Diffusion-7B-Video2World-Sample-CameraCond. We focus on generating 3D worlds from a single reference input image, leveraging camera control to produce temporally coherent and 3D-consistent video simulations from the specified camera trajectories, where changes in perspective align with the underlying 3D structure of the scene.

6.1.1. Dataset

We use DL3DV-10K (Ling et al., 2024), a large-scale video dataset of static scenes, for this task. As a preprocessing step, we chunk all videos into clips with 256 frames. To obtain camera pose annotations densely for all frames within a clip, we run structure-from-motion on the chunked clips using GLOMAP (Pan et al., 2025). We set the camera pose of the first frame to be the identity transform and compute the relative camera poses for all

subsequent frames. We also use a proprietary VLM to caption the videos to obtain text prompts that describe the videos as static scenes.

6.1.2. Fine-tuning

We add camera control conditioning by concatenating the sampled latent embeddings with Plücker embeddings (Sitzmann et al., 2021), which has the same spatial dimensions as the latent embeddings. Specifically, given the camera pose, we compute the Plücker coordinates via

$$\mathbf{r} = (\mathbf{d}, \mathbf{m}) \in \mathbb{R}^6 \quad \text{where} \quad \mathbf{m} = \mathbf{c} \times \mathbf{d} \,, \tag{11}$$

where $\mathbf{c}$ is the camera center location and $\mathbf{d}$ is the unit ray direction of each latent pixel (where the latent embedding is treated as a downsampled image). All the camera poses are relative with respect to the initial frame. The Cosmos-1.0-Tokenizer-CV8x8x8 used by Cosmos-1.0-Diffusion-7B-Video2World models has a temporal compression rate of $8\times$, and thus for every 8 frames, we use the Plücker embedding at the 4th frame to concatenate with the corresponding latent representation.

We resized the input frames of our training videos to 704×1252 and padded them to 704×1280 with reflection. We sample 57 frames during training. The training objective and other hyper-parameters are the same as the base Diffusion WFM training (Sec. 5.1.3).

6.1.3. Evaluation

We assume a single reference image of the world is given and generate the future rollout as a video from the input image. We compare against CamCo (Xu et al., 2024), the state-of-the-art model for camera-controllable video generation under this setup. For a fair comparison, we use the CamCo model that was also fine-tuned on the DL3DV-10K (Ling et al., 2024) training set. As our post-trained WFM generates 57 frames and CamCo can only generate 14 frames, we compare the same 57-frame trajectories where we temporally downsample by $4\times$ for CamCo. The video resolution from CamCo is limited to 256×256. We additionally maximally center-crop the input image and test frames for evaluation.

For the test data, we use the same 500 samples from the RealEstate10K (Zhou et al., 2018) test set previously described in Sec. 5.3.1. We use the initial frame as the reference image and camera trajectories provided by the dataset as the camera control input, which we additionally rescale such that the distance between two ends of trajectories is normalized to 1.

Metrics. Following Xu et al. (2024), we evaluate the camera controllability of the post-trained world model in two aspects: video generation quality and 3D consistency. For video quality, we use the Fréchet Inception Distance (FID) (Heusel et al., 2017) and the Fréchet Video Distance (FVD) (Unterthiner et al., 2019) to assess the qualities at the frame and video levels, respectively. We use the same test data as the reference videos to compute the metrics (note that they are not used for pixel-level comparisons).

For 3D consistency, we evaluate via the ability of structure-from-motion (Pan et al., 2025; Schönberger et al., 2016; Schönberger and Frahm, 2016) libraries to re-estimate the camera poses, and we compare the results against the input camera control trajectories. Given N frames in the video, we quantify the camera trajectory error into two terms: the average rotation error ϵ_{rot} and translation error ϵ_{trans}, defined respectively as

$$\epsilon_{\text{rot}} = \frac{1}{N}\sum_{i=1}^{N} \cos^{-1}\left(\frac{\text{trace}(\hat{\mathbf{R}}_i^\top \mathbf{R}_i) - 1}{2}\right) \qquad \text{and} \qquad \epsilon_{\text{trans}} = \frac{1}{N}\sum_{i=1}^{N} \left\|\hat{\mathbf{t}}_i - \mathbf{t}_i\right\|_2 \,, \tag{12}$$

where $\mathbf{R}_i$ and $\mathbf{t}_i$ are the input rotation and translation of the i-th frame (serving as ground truth), and $\hat{\mathbf{R}}_i$ and $\hat{\mathbf{t}}_i$ are the re-estimated quantities. To account for ambiguities from camera pose estimation results up to a similarity transformation, we follow Lin et al. (2021) and run Procrustes analysis on the predicted camera trajectories to align against the ground truth.



Figure 21: **Qualitative comparison of camera control models**. Given the input frame and camera trajectory (color-coded temporally from red to violet), we compare Cosmos-1.0-Diffusion-7B-Video2World-Sample-CameraCond against CamCo (Xu et al., 2024) on the generated future frames as well as the re-estimated camera poses. CamCo suffers from the data distribution shift and often generates inaccurate trajectories or even out-of-distribution image syntheses that lead to un-estimatable camera poses. In contrast, the Cosmos camera control model can successfully generate future frames aligned with the camera control input while also maintaining high video quality and 3D consistency.

Table 22: Quantitative comparison of post-trained WFM with camera control.

	Camera Trajectory Alignment			Video Generation Quality	
Method	Pose estimation success rate (%) ↑	Rotation error (°) ↓	Translation error ↓	FID ↓	FVD ↓
CamCo (Xu et al., 2024)	43.0%	8.277	0.185	57.49	433.24
Cosmos-1.0-Diffusion-7B-Video2World-Sample-CameraCond	**82.0%**	**1.646**	**0.038**	**14.30**	**120.49**

Comparisons. We present the results in Tab. 22. First, our post-trained WFM can generate realistic and coherent 3D worlds. This is evidenced by the lower FID/FVD scores (higher visual quality) and the higher camera pose estimation success rate. Cosmos-1.0-Diffusion-7B-Video2World-Sample-CameraCond demonstrates better camera control, as the camera trajectory re-estimation is significantly closer to the original control input.

We also provide visual comparisons in Fig. 21. While CamCo struggles to generate content beyond the input image, Cosmos-1.0-Diffusion-7B-Video2World-Sample-CameraCond effectively generates visuals that adhere to the structure of a 3D world. Note that both models were post-trained on DL3DV-10K and evaluated on the RealEstate10K dataset, which introduces a significant distribution shift between training and testing. The Cosmos model successfully overcomes this distribution shift while also demonstrating its capability to generalize to unseen input camera trajectories.

Qualitative results. Fig. 22 shows our results from joystick-like control input on the camera, including *moving forward*, *moving backward*, *rotating left*, and *rotating right*. This demonstrates the use case where one can navigate the simulated world using a joystick to control the model in generating future video frames. A Physical AI agent could also use such control to predict the future of the world under different scenarios.

To show the diversity of the generation, we show generation results from the same input image and camera control with different random seeds in Fig. 23. Cosmos-1.0-Diffusion-7B-Video2World-Sample-CameraCond is able to generate different worlds while still maintaining 3D spatial and temporal coherence in the videos. This could be used to simulate different possible futures given the current states.

6.2. Post-training WFM for Robotic Manipulation

A world model has the potential to serve as a powerful planner and simulator for robotic manipulation. Here, we demonstrate how we fine-tune our pre-trained WFMs for two tasks: (1) instruction-based video prediction and (2) action-based next-frame generation. For **instruction-based video prediction**, the input is the current video frame of a robot as well as a text instruction, and the output is a predicted video of the robot following the instruction. For **action-based next-frame prediction**, the input is the current video frame of a robot as well as an action vector between the current and next frame, and the output is the predicted next frame showing the result of the robot performing the specified action. Given a sequence of actions, the model can be run autoregressively to predict a video of the robot executing the given actions.

6.2.1. Datasets

We curate two datasets for the two tasks described above. For **instruction-based video prediction**, we created an internal dataset called the Cosmos-1X dataset. It comprises approximately 200 hours of egocentric videos captured by EVE, a humanoid robot from 1x.Tech (Technologies, 2024) performing a variety of tasks, including navigation, folding clothes, cleaning tables, picking up objects, *etc*. From the raw videos, we selected approximately $12{,}000$ episodes ranging from 1 to 9 seconds. Each episode is labeled with a one-sentence instruction, which is later upsampled with a proprietary VLM. The videos are captured at 30 FPS with a resolution of 512×512.

For **action-based next-frame generation**, we used a public dataset called Bridge (Ebert et al., 2022), with the

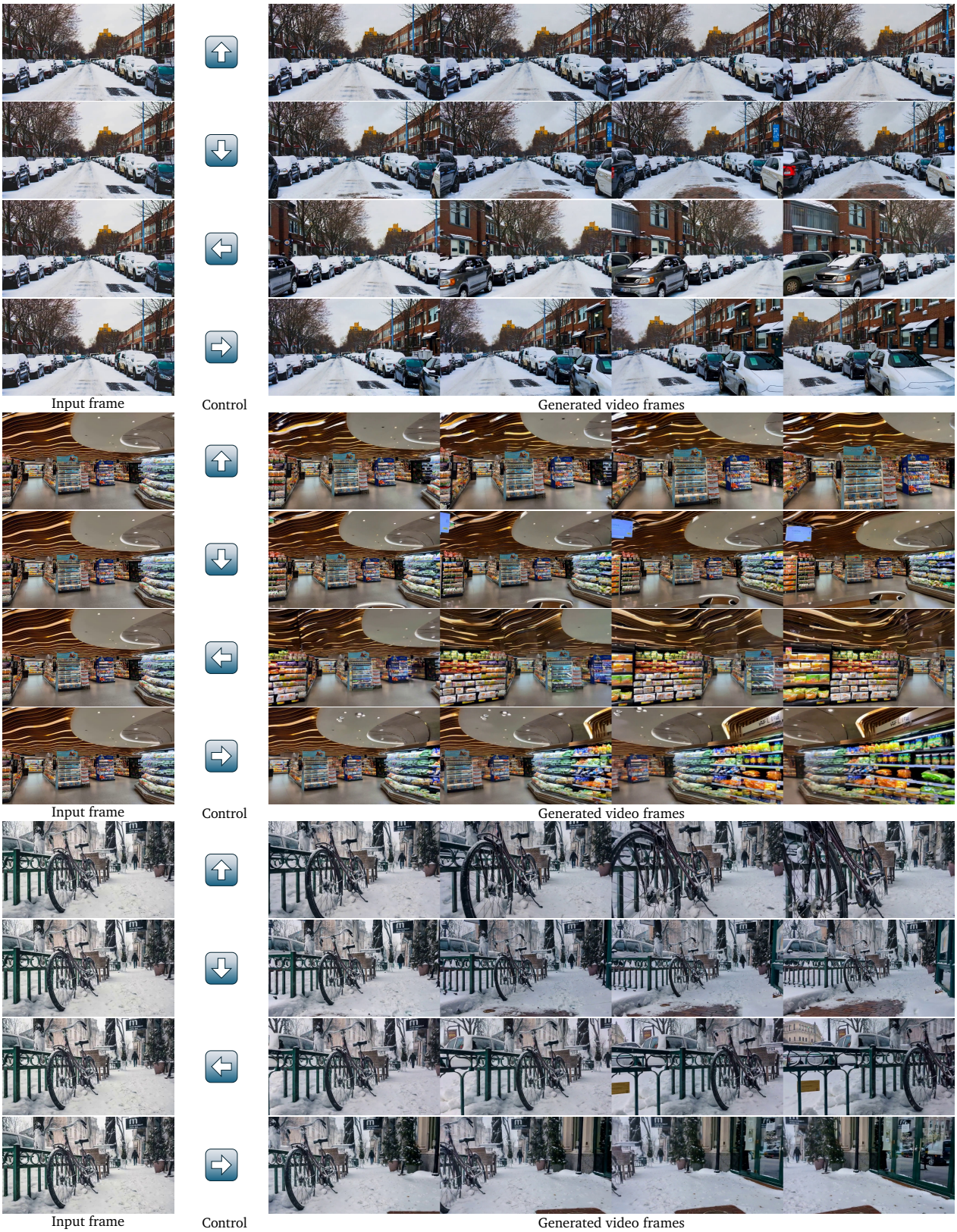


Figure 22: **Cosmos-1.0-Diffusion-7B-Video2World-Sample-CameraCond results with joystick control**. For each input frame (left-most column), we apply 4 different camera trajectories created with joystick-like control: *moving forward*, *moving backward*, *rotating left*, and *rotating right*. We visualize frames 14, 28, 42, and 57 from the generated videos.

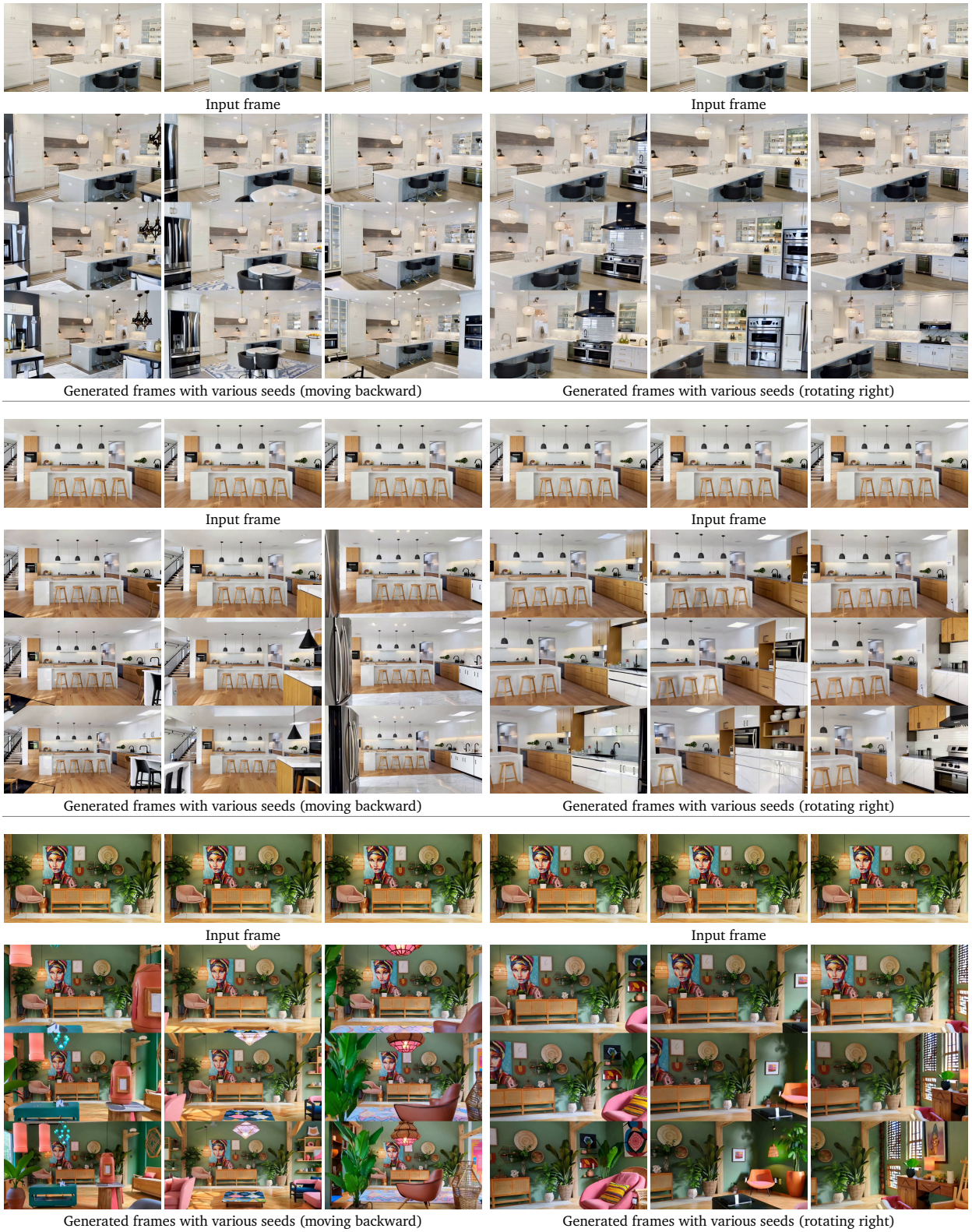


Figure 23: **Cosmos-1.0-Diffusion-7B-Video2World-Sample-CameraCond results with different seeds.** We show the capability of simulating diverse futures with our camera control model given the same input image and camera condition. For each group, we apply the same input frame and camera condition created with joystick. The first group shows *moving backward* and second group shows *rotating right*. Within each group, we show the generated videos with 3 different random seeds in each column. We visualize frames 19, 38, and 57 from the generated videos.

same configuration as a prior work (Zhu et al., 2024) for comparison. The Bridge dataset includes approximately $20{,}000$ episodes of third-person views of a robot arm performing different tasks in a kitchen environment, with videos of 320×256 resolution captured at 5 FPS. For each video frame, the corresponding action is defined as a 7-dimensional vector in the gripper coordinate space $(\Delta x, \Delta y, \Delta z, \Delta\theta_r, \Delta\theta_p, \Delta\theta_y, \Delta\text{Gripper})$ as in OpenVLA (Kim et al., 2024).

6.2.2. Fine-tuning

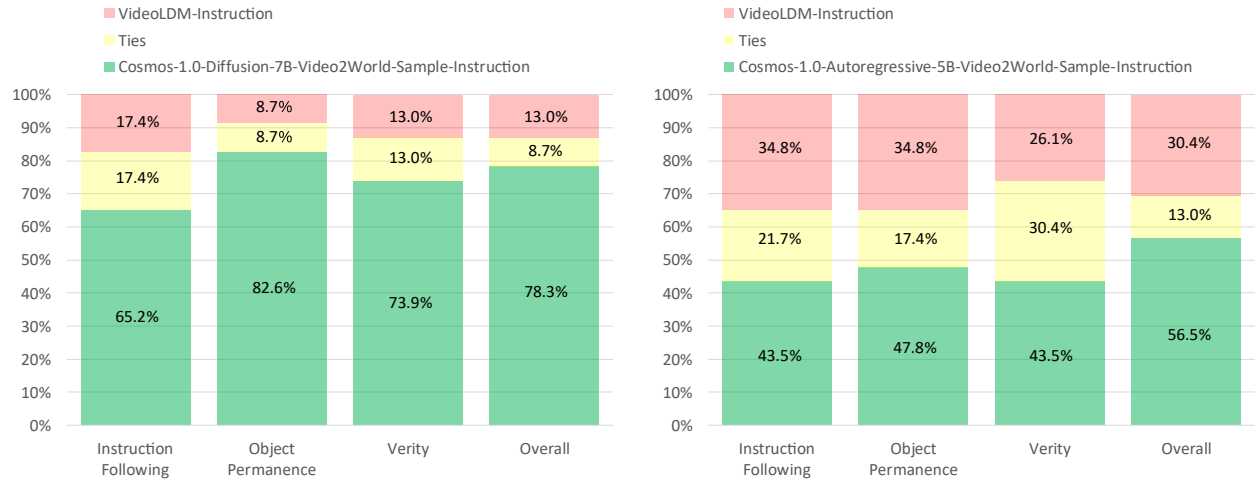
We fine-tune both our Cosmos-1.0-Diffusion-7B-Video2World (Sec. 5.1) and Cosmos-1.0-Autoregressive-5B-Video2World (Sec. 5.2) for instruction-based video prediction and action-based next-frame prediction tasks.

For **instruction-based video prediction**, we build two models based on the base WFMs. The first is called Cosmos-1.0-Diffusion-7B-Video2World-Sample-Instruction, and the second is called Cosmos-1.0-Autoregressive-5B-Video2World-Sample-Instruction. We compute the T5 embedding of the instruction, which is added to the finetuing of the base model via cross-attention.

For **action-based next-frame prediction**, we also build two models based on the base WFMs. The first one is called Cosmos-1.0-Diffusion-7B-Video2World-Sample-ActionCond, and the second one is called Cosmos-1.0-Autoregressive-5B-Video2World-Sample-ActionCond.

Since action is a new modality not encountered during pre-training, we introduce additional modules inside our models for conditioning. For Cosmos-1.0-Autoregress-5B-Video2World-Sample-ActionCond, we add an action embedder MLP to project the action vector into a tensor, which is then incorporated into the model via cross-attention. For Cosmos-1.0-Diffusion-7B-Video2World-Sample-ActionCond, we also add an action embedder MLP to predict the action into a tensor but instead, incorporate it into the model by adding it to the timestamp embedding of the DiT modules.

6.2.3. Evaluation



(a) Cosmos-1.0-Diffusion-7B-Video2World-Sample-Instruction *vs*. VideoLDM-Instruction.

(b) Cosmos-1.0-Autoregressive-5B-Video2World-Sample-Instruction *vs*. VideoLDM-Instruction.

Figure 24: **Human evaluation results for instruction-based video prediction on Cosmos-1X dataset**. The results show that compared with the baseline model (VideoLDM-instruction), our fine-tuned models (Cosmos-1.0-Diffusion-7B-Video2World-Sample-Instruction and Cosmos-1.0-Autoregressive-5B-Video2World-Sample-Instruction) have higher preferences in the four evaluation dimensions.

For **instruction-based video prediction**, we fine-tune VideoLDM (Blattmann et al., 2023) on the Cosmos-1X dataset and obtained VideoLDM-Instruction as a baseline for comparison. To evaluate the video generation performance of the models, we define the following dimensions:

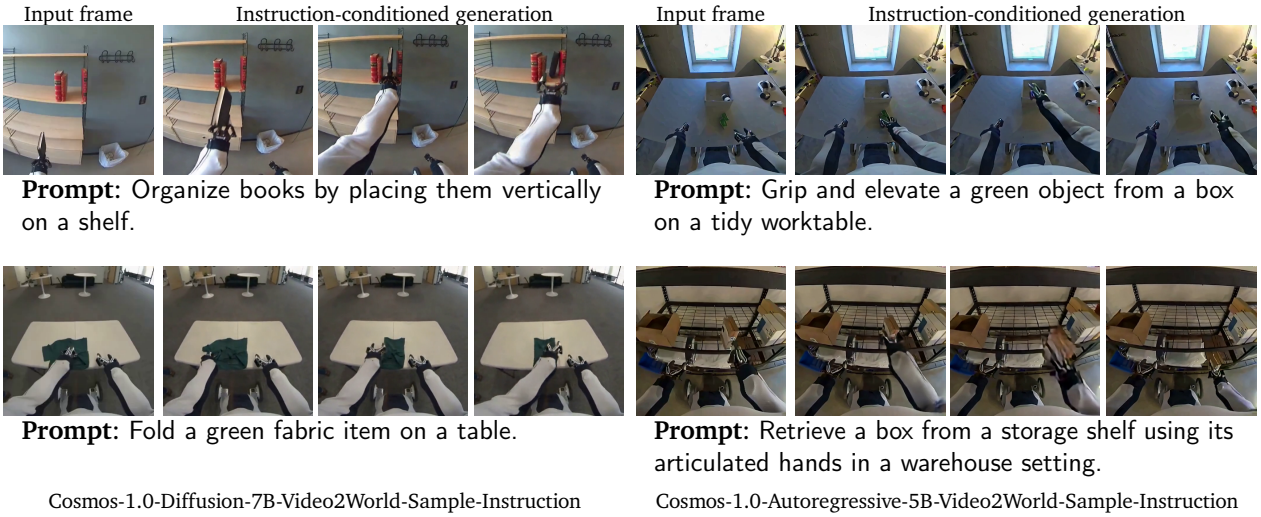


Figure 25: **Instruction-based video prediction samples on the Cosmos-1X dataset**. The left are the results of Cosmos-1.0-Diffusion-7B-Video2World-Sample-Instruction model, and the right are the results of Cosmos-1.0-Autoregressive-5B-Video2World-Sample-Instruction model.

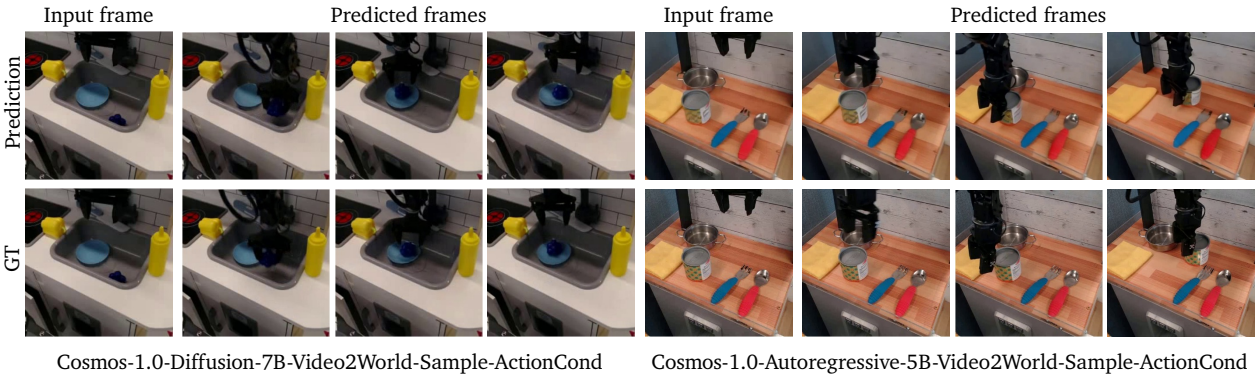


Figure 26: **Action-based next-frame prediction samples on the Bridge dataset**. The left is the results of the Cosmos-1.0-Diffusion-7B-Video2World-Sample-ActionCond model, and the right is the results of the Cosmos-1.0-Autoregressive-5B-Video2World-Sample-ActionCond model. As shown, the predicted video frames closely match the GT video frames for both models.

- **Instruction following**: Is the generated video aligned with the input language instruction?
- **Object permanence**: Do objects present in the scene remain throughout the generated video?
- **Verity**: Does the generated video faithfully represent the real world without unexpected imaginary objects?
- **Overall**: Is the generated video reasonable for the robot to plan accordingly?

Human evaluators are tasked to observe a pair of anonymous videos generated by different models but with the same language instruction and compare them along the dimensions listed above. A group of ten human evaluators performed the evaluation over 23 test episodes. The statistical results are summarized in Fig. 24.

As shown, we find that both Cosmos-1.0-Diffusion-7B-Video2World-Sample-Instruction and Cosmos-1.0-Autoregressive-5B-Video2World-Sample-Instruction perform better than VideoLDM-Instruction along the four evaluation dimensions. Cosmos-1.0-Diffusion-7B-Video2World-Sample-Instruction achieved 78.3% overall preference compared to 13.0% for VideoLDM-Instruction. Cosmos-1.0-Autoregressive-5B-Video2World-Sample-Instruction has also achieved better performance than diffusion-based VideoLDM-Instruction. Some predicted video frames for both fine-tuned WFMs are presented in Fig. 25, which shows the quality of the predicted

videos.

For **action-based next frame prediction**, we fine-tuned our models on the Bridge dataset. As a baseline, we fine-tune IRASim (Zhu et al., 2024) to derive an action-based next-frame prediction model IRASim-Action. We perform the next-frame prediction autoregressively to generate videos. To evaluate video generation quality, we compare the generated videos against ground truth videos over 100 episodes randomly selected from the official Bridge test set.

Table 23: Evaluation of action-based next-frame prediction on Bridge dataset.

Method	PSNR ↑	SSIM ↑	Latent L2 ↓	FVD ↓
IRASim-Action	19.13	0.64	0.38	593
Cosmos-1.0-Autoregressive-5B-Video2World-Sample-ActionCond	19.95	0.80	0.36	434
Cosmos-1.0-Diffusion-7B-Video2World-Sample-ActionCond	**21.14**	**0.82**	**0.32**	**190**

The computed metrics are summarized in Tab. 23, including PSNR, SSIM, Latent L2 (Zhu et al., 2024), and FVD. As shown, both Cosmos-1.0-Autoregressive-5B-Video2World-Sample-ActionCond and Cosmos-1.0-Diffusion-7B-Video2World-Sample-ActionCond models outperform the baseline model (IRASim-Action). Some predicted video frames are presented in Fig. 26, which shows the quality of the predicted videos compared to the ground truth.

6.3. Post-training WFM for Autonomous Driving

A world model for in-the-wild driving scenes has the potential to serve as a powerful simulation engine for training autonomous driving agents. As most autonomous vehicles are equipped with multiple cameras viewing different directions, an ideal world model for an autonomous vehicle should also be a multi-view one, preferably matching the precise setup of the sensors in the target vehicle. Here, we demonstrate how we fine-tune our pre-trained WFM to create a multi-view world model for autonomous driving tasks.

6.3.1. Dataset

We curate an internal dataset called the Real Driving Scene (RDS) dataset. It comprises approximately 3.6 million 20-second surround-view video clips (equivalent to approximately $20{,}000$ hours of data) captured using an NVIDIA internal driving platform. Each clip is recorded from six camera views: front, left, right, rear, rear-left, and rear-right. In addition, the dataset includes ego-motion information that we use to construct the trajectory data. We use the recorded timestamps of the front camera video to synchronize the frames of all other views.

This dataset was selected from a large labeled data corpus to match a target distribution of data attributes. The specific attribute tags include:

- Contender vehicle density (*e.g.*, none, low, medium, high)
- Weather (*e.g.*, clear, raining, snowing, fog)
- Illumination (*e.g.*, day, night)
- Ego vehicle speed (*e.g.*, standing, low, local, highway speeds)
- Ego vehicle behavior (*e.g.*, high, medium, low curvature trajectories and accelerations)
- Road type/population density (based on OpenStreetMap definitions: rural, residential, urban).

Additionally, the dataset was augmented through a second data-mining run to ensure a minimum number of clips containing rare road structures (*e.g.*, tollbooths, bridges, tunnels, speed bumps, *etc.*). Finally, videos from each camera view are captioned separately, starting with a template text string: "The video is captured from a camera mounted on a car. The camera is facing forward|left|right|backward|rear-left|rear-right."

6.3.2. Fine-tuning

We fine-tune our Cosmos-1.0-Diffusion-7B-Text2World (Sec. 5.1) into a multiple-view world model using the RDS dataset. To ensure consistent video generation across multiple views, we slightly modify the architectural design described in Sec. 5.1 and fine-tune the WFM to generate videos from all six cameras simultaneously.

We build three multi-view world models, summarized in Tab. 21. The first one is called Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView, which is a multi-view world model that can generate six camera views based on a text prompt input. The second one is called Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView-TrajectoryCond. This model is built on top of Cosmos-1.0-Diffusion-7B-Text2World-Sample-AV-MultiView and takes an additional trajectory input as the conditional input signal. The final model, Cosmos-1.0-Diffusion-7B-Video2World-Sample-MultiView, is fine-tuned from the Diffusion-7B-Video2World-Sample-MultiView model to support video-based conditioning. It achieves this by incorporating previous frames into the generation process. Cosmos-1.0-Diffusion-7B-Video2World-Sample-MultiView can take the video output from Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView and generate its extension. All three models output 6 views of 57 frames of video at a resolution of 848×480.

View-independent positional embedding and view embedding. Instead of extending the FPS-aware 3D RoPE Positional Embedding to include an additional view dimension, we opt to use the same positional embedding described in Sec. 5.1 independently to each view. To represent view differences, we modify the denoising function D_θ to take an additional view embedding as input. That is, the camera view information is supplied through global view embeddings instead of positional embedding.

View-dependent cross-attention. In our multi-view setting, each of the six views of the same scene would have a different video description. While we treat all six views as a whole as the state of the diffusion process and perform self-attention among all the elements in the six views for denoising, we find it beneficial to employ view-dependent cross-attention for textual inputs. Specifically, the cross-attention operation for each view only attends to the textual description for the specific view. Note that each view has a different video description in our dataset. With the view embedding and view-dependent cross-attention, we derive Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView from fine-tuning Cosmos-1.0-Diffusion-7B-Text2World.

Trajectory control condition. Optionally, in addition to the text condition, we fine-tune the model to produce videos that conform to the given future trajectory paths to enable more precise control of the agent. This enables the generation of unique driving scenarios that adhere to both driving trajectories recorded by real-world data and driving environments specified by the input text descriptions. The fine-tuned model is Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView-TrajectoryCond.

We define a trajectory as a sequence of 64 points in the 3D space, representing a sequence of translations of the agent from the initial position $(0, 0, 0)$ to the final destination, with each point separated by a 0.1-second interval. We compute the embedding of the trajectory input and make the result a conditional input to the denoiser of the fine-tuned Cosmos-1.0-Diffusion-7B-Video2World model. We note that it is possible to achieve more fine-grained control signals by giving a per-interval action vector, following prior works (Hu et al., 2023; Kim et al., 2020, 2021) or as in the robotic manipulation task (Sec. 6.2). We leave such extensions for future work.

6.3.3. Evaluation

We first present text-conditioned qualitative results in Fig. 27. Using Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView, we generate a 57-frame video with six views, which is then extended to 201 frames using the Cosmos-1.0-Diffusion-7B-Video2World-Sample-MultiView model. In Fig. 28, we demonstrate how the pre-trained world model enhances generalization, enabling the generation of rare or out-of-domain scenes from the RDS dataset, such as driving on a river. Lastly, Fig. 29 showcases the results from Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView-TrajectoryCond, where the ego car accurately follows the input



Prompt: The video captures a highway scene with a white truck in the foreground, moving towards the camera. The truck has a large cargo area and is followed by a motorcyclist wearing a full-face helmet. The road is marked with white lines and has a metal guardrail on the right side. The sky is partly cloudy, and there are green trees and bushes visible on the roadside. The video is taken from a moving vehicle, as indicated by the motion blur and the changing perspective of the truck and motorcyclist.



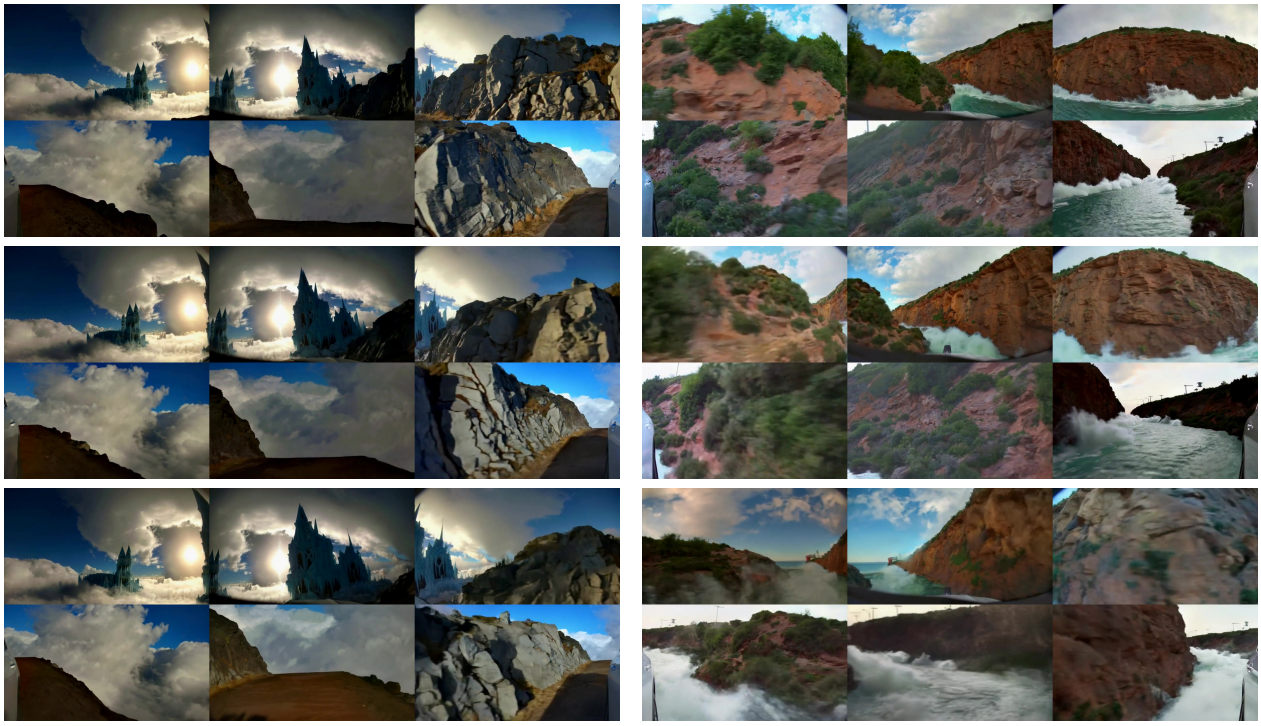
Prompt: The footage shows a multi-car pile-up on a foggy highway. Visibility is severely reduced due to thick fog, with only the taillights of vehicles ahead visible. Suddenly, brake lights flash, and cars begin to swerve and stop abruptly. The highway is cluttered with stopped and crashed vehicles. The surroundings are obscured by fog, adding to the chaos and confusion of the scene.

Figure 27: **Text-conditioned samples generated by Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView, extended to 8 seconds by Cosmos-1.0-Diffusion-7B-Video2World-Sample-MultiView**. This figure visualizes all six camera views in a group, with each row corresponding to a specific timestamp. The left example depicts a highway scene where a motorcycle is riding alongside a large truck. The right example shows the ego car following a sedan as it takes a right turn in a heavy snowy day.

trajectory.

For quantitative results, as a baseline, we followed the same fine-tuning recipe to fine-tune VideoLDM (Blattmann et al., 2023) to derive a multi-view world model called VideoLDM-MultiView. We use a set of evaluation metrics measuring video generation quality, multi-view consistency, and trajectory following accuracy. To evaluate video generation quality, we use 1000 samples to compute the scores. For consistency-related metrics, to better understand different models' behaviors under different scenarios, we categorize the ground-truth trajectories into four types: moving forward, turning left, turning right, and others (including static or complex movements). For each category, we gathered 200 samples and their corresponding prompts and conditions, totaling 800 samples. Below, we provide detailed descriptions of the metrics and the results.

Generation quality. We utilize Fréchet Inception Distance (FID) (Heusel et al., 2017) and Fréchet Video Distance (FVD) (Unterthiner et al., 2019) to measure the quality of the generated videos relative to the real ones. We first calculate a score per view by extracting 16 frames from each video. We then report the average score across all views per method. As shown in Tab. 24, we find both Cosmos-1.0-Diffusion-7B-Text2World-



Prompt: There are towering, intricately designed ice castle illuminated from within. Ahead, the sky showcases a vibrant sunset and a luminous moon positioned to the left of the castle, casting a blue hue across the scene. Fast-moving, dark, and dramatic clouds add to the otherworldly atmosphere. The 3D realistic art style focuses on lighting and texture, creating a striking visual effect.

Prompt: The footage captures a coastal area where the ocean meets a rugged, rocky cliff. Ahead, vibrant blue waves crash against the rocks, creating white foam. The cliff is a mix of brown and green hues, indicating vegetation and possibly moss or algae.

Figure 28: **Text-conditioned samples generated by Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView are extended to 8 seconds by Cosmos-1.0-Diffusion-7B-Video2World-Sample-MultiView**. The post-trained world model effectively preserves its generalization ability. In the left example, the ego car is driving towards an ice castle, while in the right example, the ego car is shown driving on a river.

Sample-MultiView and Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView-TrajectoryCond significantly outperform VideoLDM-MultiView on both metrics, demonstrating the superior quality of our pre-trained 7B Diffusion-based WFM over the VideoLDM-MultiView baseline.

Multi-view consistency. We use an extended version of the Sampson error (Hartley and Zisserman, 2003; Sampson, 1982) formulated in Sec. 5.3.1 to quantify the geometry consistency of the generated multi-view videos. As the ground-truth videos in our RDS dataset share similar fisheye camera intrinsic parameters, we use the median calibration to undistort the keypoints to a regular pinhole camera with a uniform size of 960×540 and 120 degrees of horizontal FoV. Under this setting, two metrics are computed for the generated multi-view video:

1. *Temporal Sampson Error (**TSE**)* measures whether the content generated for each camera is consistent over time. It is the median Sampson error of adjacent frames for each of the views.
2. *Cross-view Sampson Error (**CSE**)* measures whether multi-view consistency is preserved over time. It is the Sampson error across different generated views averaged in time. The fundamental matrix used in CSE is estimated using the keypoints accumulated across all temporal frames.

As shown in Tab. 24, we find that both Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView and Cosmos-

Table 24: Evaluation on post-trained multi-view world models for multi-view driving video generation.

	Generation Quality		Multi-View Consist.	
Method	FID ↓	FVD ↓	TSE ↓	CSE ↓
VideoLDM-MultiView	60.84	884.46	1.24	6.48
Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView	**32.16**	**210.23**	0.68	2.11
Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView-TrajectoryCond	-	-	**0.59**	**2.02**
Real Videos (Reference)	-	-	0.69	1.71

Table 25: Trajectory consistency evaluation on post-trained multi-view world models for multi-view driving video generation. The numbers of TAE are scaled by 10^2 for convenience, and the unit for TFE is cm.

Method	TAE-ATE ↓	TAE-RPE-R ↓	TAE-RPE-t ↓	TFE ↓
VideoLDM-MultiView	0.88	22.94	0.77	-
Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView	0.77	**4.25**	0.29	-
Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView-TrajectoryCond	**0.54**	4.31	**0.18**	**20.20**
Real Videos (Reference)	0.49	4.60	0.14	13.49

1.0-Diffusion-7B-Text2World-Sample-MultiView-TrajectoryCond render better multi-view geometry consistency over VideoLDM-MultiView. The overall geometric plausibility of the generated videos is much better for a world model fine-tuned from our WFM. We also note that, with the trajectory control condition, such consistency is further improved thanks to the explicit 3D guidance as Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView-TrajectoryCond is ranked the best.

Trajectory consistency: Trajectory Agreement Error (TAE). We design a robust multi-view camera pose estimation pipeline similar to the one used in Liang et al. (2024) based on the formulation of Teed and Deng (2021). Such a pose estimation pipeline features an online dynamic mask generation module and a highly efficient dense bundle adjustment module, reaching a robust and real-time performance for estimating multi-view camera poses. We use this pipeline to estimate the camera poses of the front camera, separately using two multi-view camera configurations that consider "front + left-front" cameras and "front + right-front" cameras. We then calculate their trajectory errors to show their agreement, reflecting the consistency of the multi-view generation. Specifically, we compute the Absolute Trajectory Error (ATE) and Relative Pose Error for both the translational component (RPE-t) and the rotational component (RPE-R). We normalize the length of the trajectories to 1.0 for fair comparison and exclude the cases with minor camera movements (*e.g.*, a car stopped at a red light).

As shown in Tab. 25, the results echo the findings from the multi-view geometry consistency, where the trajectory consistency of the world models fine-tuned from the Cosmos WFM is much better than that from the VideoLDM-MultiView. We note that the post-trained Cosmos world models have trajectory consistency that is close to real-world videos.

Trajectory consistency: Trajectory Following Error (TFE). Furthermore, for the model where we have trajectory control conditions being fed into the model, we use the same camera pose estimation pipeline used above to compute the poses of the front camera using multi-view information and compare the predicted trajectory with respect to the ground truth trajectory condition. This measures how well the model follows the



Figure 29: **Trajectory-conditioned generated samples from Cosmos-1.0-Diffusion-7B-Text2World-Sample-MultiView-TrajectoryCond**. Given the trajectory inputs on the left-most column, we generate multiview videos that follow the given trajectory. We visualize the front camera view in this figure.

given trajectory path. As shown in Tab. 25, the trajectory error estimated using the generated videos from our Cosmos post-trained world models is only <7cm less precise than the ground-truth oracle. Such a considerably slight margin shows that our model is able to accurately follow the given trajectory path, which is crucial for training autonomous driving agents.

Objects tracking consistency. Finally, we applied object detection and tracking using YOLOv11x (Khanam and Hussain, 2024) on the generated 8-second videos. Human annotators were tasked with identifying instances where the tracking algorithm misinterpreted physically impossible scenarios, such as two distinct objects (*e.g.*, a person and a car) merging incorrectly into a single tracked entity. To evaluate this, we provided annotators with a random sample of 20 generated videos containing 157 objects. Remarkably, none of the 157 objects exhibited any physically impossible scenarios, demonstrating the physical consistency and object permanence of our generated driving videos.

7. Guardrails

For the safe use of our WFMs, we develop a comprehensive guardrail system. It consists of two stages: the pre-Guard stage and the post-Guard stage. The pre-Guard stage leverages Aegis (Ghosh et al., 2024) and a keyword list to block harmful prompts. The post-Guard stage blocks harmful visual outputs using a video content safety classifier and a face blur filter. The pipeline is illustrated in Fig. 30.

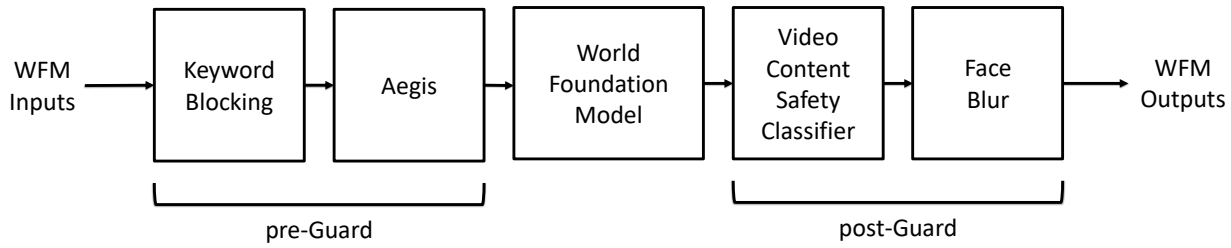


Figure 30: **Cosmos Guardrail overview**. Cosmos Guardrail contains pre-Guard and post-Guard, where pre-Guard blocks inputs based on Aegis (Ghosh et al., 2024) and keywords, while post-Guard blocks outputs based on a video content safety classifier and blurs output faces.

7.1. Pre-Guard

Our pre-Guard is a text-domain guardrail comprising an LLM-based guardrail for semantically complex prompts and a simple blocklist-based checker for explicitly unsafe keywords.

7.1.1. Keyword Blocking

The blocklist heuristic will act as the first line of defense to mitigate the risk of generating unsafe content. This is designed to block explicitly harmful generations by doing a keyword search on the prompt against a hard-coded blocklist of a large corpus of explicit and objectionable words. Input words are lemmatized using *WordNetLemmatizer*, a tool that uses a lexical database of the English language (Miller, 1995) to extract the root word from its variants. For example, the root word of "*abacii*" is "*abacus*". These lemmatized words are then compared to the words in the hard-coded blocklist, and the entire prompt is rejected if any profanity is found. We use a comprehensive set of keywords to maximally protect our users.

7.1.2. Aegis Guardrail

As the second line of defense, we use *Aegis-AI-Content-Safety-LlamaGuard-LLM-Defensive-1.0* (Ghosh et al., 2024), which is a fine-tuned version of *Llama-Guard* (Inan et al., 2023) trained on NVIDIA's Aegis Content Safety Dataset covering NVIDIA's broad taxonomy of 13 critical safety risk categories. There are two versions of AEGIS 1.0, the defensive version and the permissive version. The defensive version adopts a tighter permission boundary than the permissive version. Cosmos uses the defensive version of Aegis to block potentially harmful user prompts that attempt to generate harmful content. If the input prompt is categorized as unsafe by this prompt filter, the video is not generated, and an error message is displayed.

For using Aegis as a prompt filter, we classify the prompt as unsafe if it falls into the following categories: violence, sexual, criminal planning, weapons, substance abuse, suicide, child sexual abuse material, hatred, harassment, threat, and profanity. Any prompt that does not fall into the above categories is considered safe from the prompt-filtering standpoint.

7.2. Post-Guard

Our post-Guard is a vision-domain guardrail comprising a video content safety filter and a face blur filter for the generated output.

7.2.1. Video Content Safety Filter

The Video Content Safety Filter is a frame-level multi-class classifier trained on our video dataset and generation results. Among the classes, some are considered safe, while others are unsafe. A major challenge in training the classifier is in balancing false positives, where safe content is mistakenly flagged as unsafe, and false negatives, where unsafe content is wrongly classified as safe. To minimize classification errors, we carefully balanced the data during training.

We collect three kinds of ground truth annotated data. First, we sample a large set of videos from our dataset, extract frames, and determine its class using a VLM. Next, we generate synthetic videos with our WFMs using a set of prompts to ensure coverage of corner cases and least-represented content categories. Finally, human annotators provide the "gold standard" labels for a portion of our dataset, adding a vital layer of validation and helping us continuously refine the accuracy of our classifier. We extract the SigLIP (Zhai et al., 2023) embedding for each video frame and train a simple MLP classifier on the embeddings.

During inference, we generate a SigLIP embedding for every frame and then apply the classifier. The entire video is flagged as unsafe if any frame is classified as unsafe.

7.2.2. Face Blur Filter

We use RetinaFace (Deng et al., 2020), a state-of-the-art face detection model, to identify facial regions with high confidence scores. For any detected face region larger than 20×20 pixels, we apply pixelation to obscure the regions while preserving the overall scene composition for Physical AI applications.

7.3. Red Team Effort

We employ a dedicated red team to actively probe the system using both standard and adversarial examples that are collected in an internal attack prompt dataset. These video outputs are annotated by a team of expert annotators, who were specially trained for our task, to classify the generated video on a scale of 1-5 on multiple categories of harm related to the taxonomy in Sec. 7.1.2. These annotations also specify the start and end-frames where the unsafe content is detected, thereby generating high-quality annotations. The red team also probed each guardrail component independently with targeted examples to identify weaknesses and improve performance in edge cases. As of the date of publication, the red team has tested and annotated over $10{,}000$ distinct prompt-video pairs that were carefully crafted to cover a broad range of unsafe content.

8. Related Work

World models. The concept of "world models" originated from the seminal work of Ha and Schmidhuber (2018), which proposed learning a representation of the real world using neural network models to predict future states given current states and inputs. An accurate representation of the physical world model enables not only reliable prediction of future states but also informed decision-making. This concept of modeling the physical world is not new; traditional automation and robotics industries have long employed mathematical models based on physics laws and system identification in planning and control algorithms (Murray et al., 2017). However, these system-specific models, typically confined to low-dimensional state spaces, restrict generalization and knowledge transfer across different systems, limiting model reuse when applied to new tasks or environments. Recent advances in deep learning, particularly generative AI, have made it possible to learn world models directly from visual observations.

Modern world model pipelines can be categorized based on their backbone architecture. Most works (Hafner et al., 2019, 2021, 2023; Hansen et al., 2024; Kim et al., 2020, 2021), including the original paper (Ha and Schmidhuber, 2018) by Ha and Schmidhuber, employ a recurrent neural network to model system state evolution in a latent space learned via an autoencoder. More recent trends view world models as generative models in visual observation space, often in the form of conditional video generative models (*e.g.*, action-to-video, text-to-video). These models can be either autoregressive (Bruce et al., 2024; Liu et al., 2024; Micheli et al., 2023; Robine et al., 2023; Yang et al., 2023) or diffusion-based (Alonso et al., 2024; Ding et al., 2024; Valevski et al., 2024), as considered in this work. Another promising approach is generative simulation (Hua et al., 2024; Nasiriany et al., 2024), which combines generative AI and physical simulators to model the real world.

A well-trained world model can be applied in various ways, including verification (Hu et al., 2023), planning-based model predictive control (Bar et al., 2024; Hansen et al., 2024), and model-based reinforcement

learning (Alonso et al., 2024; Ding et al., 2024; Robine et al., 2023; Yang et al., 2023; Zhang et al., 2024). The effectiveness of world models has been demonstrated in domains such as computer games (Alonso et al., 2024; Bruce et al., 2024; Hafner et al., 2021; Kim et al., 2020; Valevski et al., 2024), real-world robots (Wu et al., 2023; Yang et al., 2023), and autonomous driving (Blattmann et al., 2023; Hu et al., 2023; Kim et al., 2021; Zhao et al., 2024). We envision that foundational world models will have transformative impacts on these industries.

Video generative models. The field of video generative models has undergone rapid development in recent years. From the initial models that produced short, low-resolution videos, the field has evolved significantly, with video generative models now at the forefront of generative AI research (Ho et al., 2022; Huang et al., 2024). Recent years have seen the emergence of impressive video generative models, such as Sora, Dream Machine, Gen 3 and Kling, capable of producing realistic, high-resolution videos (KuaiShou, 2024; Luma, 2024; OpenAI, 2024; Runway, 2024). These advancements have been made in just a few years since the release of the first video generative model.

Most existing work on video generative models focuses on text-to-video tasks, which generate videos based on text prompt inputs (Blattmann et al., 2023; Ge et al., 2023; Girdhar et al., 2024; Lin et al., 2024; Ma et al., 2024; Yang et al., 2024). These models enable users to create impressive videos using carefully designed text prompts. Other popular tasks include Image-to-Video that generates videos starting from a given image frame (Blattmann et al., 2023; Gururani et al., 2023; Mallya et al., 2022; Ren et al., 2024; Wang et al., 2021, 2024), Video-to-Video that generates new videos given a reference video (Ku et al., 2024; Liu et al., 2024; Mallya et al., 2020; Wang et al., 2018, 2019) and Action-to-Video that generates videos based on actions driven by the development of world models and embodied AI (Alonso et al., 2024; Bruce et al., 2024; Tulyakov et al., 2018; Valevski et al., 2024).

The majority of video generative models adopt the diffusion model framework (Blattmann et al., 2023; Ge et al., 2023; Lin et al., 2024; Ma et al., 2024; Yang et al., 2024) to gradually transform noise into video sequences. Autoregressive models have also been employed for video generation, offering the advantage of handling video and other modalities in a unified manner (Deng et al., 2024; Kondratyuk et al., 2024; Liu et al., 2024). While autoregressive models have shown promise, diffusion-based video models still excel in terms of visual quality. Our goal is to help Physical AI developers advance their applications. We believe that the diffusion-based and autoregressive-based models both have their pros and cons. Diffusion-based models could render videos with better visual quality. Autoregressive-based models can better leverage all sorts of techniques developed by the LLM community. We build both diffusion-based (Cosmos-Diffusion) and autoregressive-based (Cosmos-Autoregressive) WFMs and make them available to the Physical AI builders.

Video generation with camera control. 3D-consistent video generation traces back to early works in view synthesis and 3D reconstruction, where the community sought to create 3D-consistent videos using neural rendering applied to various 3D representations (Kerbl et al., 2023; Li et al., 2023; Mildenhall et al., 2020; Wang et al., 2021; Zhou et al., 2018). Within this line of research, single-image 3D view synthesis (Charatan et al., 2024; Lin et al., 2023; Tucker and Snavely, 2020; Wiles et al., 2020; Yu et al., 2021) is particularly challenging, typically requiring the learning of a strong 3D prior model from multi-view image datasets. As such 3D prior models often suffer to scale well, learning-based view synthesis has also been explored through a purely data-driven approach using scalable Transformer architectures (Dosovitskiy et al., 2021; Vaswani et al., 2017). This bypasses the need for explicit 3D prior knowledge (Rombach et al., 2021; Sajjadi et al., 2022): instead of relying on neural rendering applied to 3D representations, novel views are synthesized directly by neural networks conditioned on camera inputs (Tatarchenko et al., 2016). This paradigm has been successfully scaled up with diffusion models (Liu et al., 2023), finding broad applications in 3D asset generation (Li et al., 2024; Lin et al., 2023; NVIDIA, 2024; Poole et al., 2023; Qian et al., 2024; Shi et al., 2023). Recent advances in video generation quality suggest the potential for achieving full 3D consistency through the scaling of training video data (Brooks et al., 2024). Camera controllability on such models has since become an active area of

investigation (He et al., 2024; Wang et al., 2024; Xu et al., 2024) for its great potential applications to robotics and autonomous navigation.

Generative models for robotic control. Recent advances in deep generative models have sparked significant interest in their application to robotic control. Several approaches have emerged, with one line of work directly employing diffusion models as visuomotor policies, demonstrating substantial improvements in imitation learning in various robotic tasks (Chi et al., 2023; Ke et al., 2024; Prasad et al., 2024; Wang et al., 2024). Two other threads more related to this work are the use of pre-trained image and video generation models as motion planners and the use of image and video data for generative pre-training. The generative motion planning approach (Black et al., 2023; Du et al., 2024; Finn and Levine, 2017; Ko et al., 2024; Zhou et al., 2024) aims to enhance generalization to unseen environments by generating intermediate visual sub-goals rather than explicit action sequences. This visual representation strategy proves more robust, as image and video sub-goals can generalize across diverse environmental setups, unlike action sequences that are typically environment- and task-specific. The generative pre-training approach (Cheang et al., 2024; Gupta et al., 2024; He et al., 2024) leverages large-scale image and video datasets for pre-training. While Gupta et al. (2024) extract and utilize features from pre-trained text-to-image diffusion models to guide subsequent policy learning, Cheang et al. (2024) and He et al. (2024) use a two-stage framework: first pre-training the model to predict future frames, then fine-tuning it to jointly predict both actions and future frames.

Generative models for autonomous driving. Video generative models have the potential to revolutionize autonomous driving simulation by enabling the generation of realistic driving videos conditioned on diverse input modalities, such as text, images, trajectories, 3D data, or maps (Blattmann et al., 2023; Gao et al., 2024,,; Hu et al., 2023; Jia et al., 2023; Kim et al., 2021; Lu et al., 2025; Wang et al., 2023, 2024; Yang et al., 2024). Despite their potential, existing approaches have been limited by constraints in data scale (Gao et al., 2024,; Jia et al., 2023; Lu et al., 2025; Wang et al., 2023, 2024), resolution (Hu et al., 2023; Yang et al., 2024), and the number of camera views (Blattmann et al., 2023; Gao et al., 2024), restricting their effectiveness as comprehensive driving world simulators. To overcome these limitations, we leverage the capabilities of a powerful pre-trained WFM to develop a flexible and scalable driving simulator. Our model achieves high resolution, elevated frame rates, and multi-view consistency.

Tokenizer. There has been a fairly long history of learning latent features that reproduce the input visual data (He et al., 2022; Hinton et al., 1995; Kingma, 2013; van den Oord et al., 2017). Recently, such models, also known as tokenizers, have been widely incorporated as essential components to improve the efficiency of training large-scale generative models (Esser et al., 2021; Rombach et al., 2022).

Continuous visual tokenizers, often including Autoencoder (AE) and Variational Autoencoder (VAE), compress visual data into a continuous latent space where diffusion-based models can be efficiently trained on (Ho et al., 2020; Lipman et al., 2022; Song et al., 2020). At inference time, the generated latents are decoded back to RGB space with the tokenizer decoder. Various diffusion models have been trained in such a way for image (Betker et al., 2023; Dai et al., 2023; FLUX, 2024; Gafni et al., 2022; Podell et al., 2024; Ramesh et al., 2022; Rombach et al., 2022) and video generation (An et al., 2023; Blattmann et al., 2023,; Brooks et al., 2024; Ge et al., 2023; Girdhar et al., 2024; Wang et al., 2023; Yu et al., 2023; Zeng et al., 2024).

Discrete visual tokenizers additionally involve a quantizer (Lee et al., 2022; Mentzer et al., 2023; van den Oord et al., 2017; Yu et al., 2024,; Zhao et al., 2024) that further discretizes the continuous latents into a discrete space, allowing for easy integration into large language models (LLMs) and vision language models (VLMs) alongside other modalities, such as text and audio. Thus, discrete tokenizers are deployed in various visual understanding (Sun et al., 2024; Team, 2024; Wang et al., 2024; Wu et al., 2024) as well as image (Chang et al., 2022; Esser et al., 2021; Ramesh et al., 2021; Sun et al., 2024; Yu et al., 2022) and video generation tasks (Ge et al., 2022; Hong et al., 2023; Kondratyuk et al., 2024; Luo et al., 2024; Villegas et al., 2023; Wu et al., 2022; Yan et al., 2021; Yu et al., 2023).

Cosmos tokenizers are extensively built based on previous studies, *e.g.*, FSQ (Mentzer et al., 2023) and causal architecture (Yu et al., 2023), with the goal of creating a suite of efficient and high-quality tokenizers.

9. Conclusions and Discussions

The Cosmos World Foundation Models mark a significant step towards building general-purpose simulators for the physical world. This work outlines our comprehensive approach, including the data curation pipeline, the design of continuous and discrete tokenizers, the architecture of diffusion and autoregressive world foundation models, and the fine-tuning process for diverse downstream Physical AI tasks. Notably, we demonstrate the adaptability of our pre-trained world models to critical applications, including 3D world navigation, robotic manipulation, and autonomous vehicle systems, which demand both 3D consistency and action controllability.

Limitations. Despite the progress, the development of world foundation models is still in the early stages. Current models, including ours, fall short as reliable simulators of the physical world. We observe that our models still suffer from issues, including the lack of object permanence, inaccuracies in contact-rich dynamics, and inconsistency in instruction following. Additionally, the realism of the generated videos does not always reflect adherence to fundamental physical principles, such as gravity, light interactions, and fluid dynamics.

Evaluation presents another significant challenge. Defining robust rubrics for humans to evaluate physical fidelity is hard as such assessments are often influenced by personal biases, backgrounds, and other subjective factors. Moreover, these evaluations may not align positively with metrics used in downstream Physical AI tasks. In order to address these challenges, promising directions include the development of automated evaluators powered by multi-modal LLMs and leveraging existing physical simulators to enable reproducible and interactive evaluation, thereby reducing dependence on human evaluation.

Autoregressive *vs*. Diffusion WFMs. Our evaluation results in 3D consistency (Sec. 5.3.1) and video generation for robotics (Sec. 6.2) indicate that diffusion-based WFMs currently deliver better generation quality. Through fine-tuning, diffusion-based WFMs are able to incorporate diverse control signals, including camera pose, end-effector positions, or autonomous vehicle trajectories, and generate outputs of novel formats like multi-view videos. However, autoregressive-based WFMs possess significant untapped potential. They could (1) leverage pre-trained weights from large language models (LLMs) to inherit extensive world knowledge and (2) enable faster generation through the use of advanced inference optimization techniques designed for causal attention. If these capabilities are fully realized, autoregressive WFMs may become particularly well-suited for applications requiring interactive control or real-time processing, such as planning and simulation in robotics. Importantly, the boundary between diffusion and autoregressive models is not rigid. Recent advancements have shown that diffusion transformers with bidirectional attention can be distilled into student transformers with causal attention, enabling support for key-value caching during inference (Yin et al., 2024). Similarly, autoregressive models can incorporate locally bidirectional attention to generate images via diffusion heads (Zhou et al., 2024). Exploring these hybrid approaches and their trade-offs remains an active and promising area of research. We plan to investigate these formulations further and provide a comprehensive analysis in future work.

A. Contributors and Acknowledgements

A.1. Core Contributors

- **Data Curation**
  Jacob Huffman, Francesco Ferroni, Alice Luo, Niket Agarwal, Hao Wang, Jing Zhang, David Page, Vasanth Rao Naik Sabavat, Sriharsha Niverty, Erik Barker, Lindsey Pavao, Stella Shi, Prithvijit Chattopadhyay, Shitao Tang, Yin Cui, Yunhao Ge, Qianli Ma, Yifan Ding, Seungjun Nah, Siddharth Gururani, Jiashu Xu, Grace Lam, Tiffany Cai, Jibin Varghese, Pooya Jannaty, Jay Zhangjie Wu, Yuxuan Zhang, Huan Ling, Hanzi Mao, Heng Wang
- **Tokenizer**
  Jinwei Gu, Xian Liu, Songwei Ge, Ting-Chun Wang, Haoxiang Wang, Fitsum Reda
- **Diffusion-based World Foundation Model Pre-training**
  Qinsheng Zhang, Lin Yen-Chen, Xiaohui Zeng, Huan Ling, Shitao Tang, Maciej Bala, Ting-Chun Wang, Yu Zeng, Seungjun Nah, Qianli Ma, Hanzi Mao
- **Autoregressive-based World Foundation Model Pre-training**
  Haoxiang Wang, Yifan Ding, Xian Liu, Jiaojiao Fan, Xiaohui Zeng, Yogesh Balaji
- **Prompt Upsampler**
  Yunhao Ge, Haoxiang Wang, Jiashu Xu, Yin Cui
- **Diffusion Decoder**
  Huan Ling, Jiaojiao Fan, Fitsum Reda, Yogesh Balaji, Hanzi Mao, Qinsheng Zhang
- **3D Consistency Pre-training Evaluation**
  Jiahui Huang, Chen-Hsuan Lin
- **Physics Alignment Pre-training Evaluation**
  Francesco Ferroni, Prithvijit Chattopadhyay, Xinyue Wei, Qianli Ma, Gergely Klár, Chen-Hsuan Lin
- **Camera Control Post-training Evaluation**
  Xiaohui Zeng, Tsung-Yi Lin, Jingyi Jin, Chen-Hsuan Lin
- **Robotics Post-training Evaluation**
  Lin Yen-Chen, Wei-Cheng Tseng, Yunhao Ge, Xian Liu, Shitao Tang, Fangyin Wei, Lyne Tchapmi, Yu Zeng, Qingqing Zhao, Yin Cui, Zhaoshuo Li, Jinwei Gu
- **Autonomous Driving Post-training Evaluation**
  Seung Wook Kim, Jay Zhangjie Wu, Jiahui Huang, Francesco Ferroni, Michele Fenzi, Daniel Dworakowski, Despoina Paschalidou, Ed Schmerling, Shiyi Lan, Laura Leal-Taixe, Sanja Fidler, Huan Ling
- **Guardrail**
  Jibin Varghese, Arslan Ali, Grace Lam, Pooya Jannaty
- **Platform Architect**
  Ming-Yu Liu

A.2. Contributors

Anqi Li, Arsalan Mousavian, Artur Zolkowski, Bartosz Stefaniak, Dieter Fox, Ethan He, Kaichun Mo, Morteza Ramezanali, Przemek Tredak, Wei Yang, Xiaowei Ren, Yongxin Chen, Zeeshan Patel

A.3. Acknowledgments

We thank 1X Technologies for generously providing humanoid robot data and offering invaluable support for the post-training for robotic manipulation in this technical report.

We thank Aarti Basant, Akan Huang, Alex Qi, Alexis Bjorlin, Amanda Moran, Amol Fasale, Ankit Patel, Aryaman Gupta, Ashna Khetan, Ashwath Aithal, Bor-Yiing Su, Bryan Catanzaro, Charles Hsu, Chris Pruett, Christopher Horvath, Clark Doan, Coulten Holt, Dane Aconfora, Deepak Narayanan, Dennis Chang, Dheeraj Kapur, Dong Ahn, Ebrar Erdem, Elmar Haussmann, Gandhi Vaithilingam, Henry Estela, Henry Vera, Herb Woodruff, Imad El Hanafi, Jashojit Mukherjee, Jason Sewall, Jensen Huang, John Dickinson, John Dickinson, Jonah Alben, Jonah Philion, Josh Abbott, Jun Gao, Kumar Anik, Lee Ditiangkin, Luke Alonso, Madison Huang, Marek Dabek, Mark Arnold, Max Ehrlich, Michele Ferretti, Misbah Mubarak, Misha Smelyanskiy, Paniz Karbasi, Mohamed Fawzy, Mohammad Harrim, Mohammad Shoeybi, Omkar Mehta, Pallab Bhattacharya, Pasha Shamis, Raju Wagwani, Rick Izzo, Robert Hero, Sharon Clay, Songyan Tang, Sophia Huang, Sridhar Bhuvanapalli, TJ Galda, Thomas Volk, Tobias Lasser, Vaibhav Ranglani, Vijay Anand Korthikanti, Yazdan Aghaghiri, Yugi Guvvala, and Zekun Hao for their feedback and engineering support.

We thank Iain Cunningham, Jim Fan, Marco Pavone, Meredith Price, Nikki Pope, Scott Reed, and Yuke Zhu for their feedback on the early draft of this technical report.

References

[1] Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023. 10

[2] Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, et al. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*, 2024. 29

[3] Pravesh Agrawal, Szymon Antoniak, Emma Bou Hanna, Baptiste Bout, Devendra Chaplot, Jessica Chudnovsky, Diogo Costa, Baudouin De Monicault, Saurabh Garg, Theophile Gervet, et al. Pixtral 12b. *arXiv preprint arXiv:2410.07073*, 2024. 26

[4] AI Image Lab, University of Modena. Bbc planet earth dataset, 2016. URL https://aimagelab.ing.unimore.it/imagelab/researchActivity.asp?idActivity=19. Accessed: 2024-10-17. 7

[5] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *NeurIPS*, 2024. 55, 56

[6] Jie An, Songyang Zhang, Harry Yang, Sonal Gupta, Jia-Bin Huang, Jiebo Luo, and Xi Yin. Latent-shift: Latent diffusion with temporal shift for efficient text-to-video generation. *arXiv preprint arXiv:2304.08477*, 2023. 57

[7] Yuval Atzmon, Maciej Bala, Yogesh Balaji, Tiffany Cai, Yin Cui, Jiaojiao Fan, Yunhao Ge, Siddharth Gururani, Jacob Huffman, Ronald Isaac, et al. Edify image: High-quality image generation with pixel space laplacian diffusion models. *arXiv preprint arXiv:2411.07126*, 2024. 22

[8] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022. 23

[9] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. *arXiv preprint arXiv:2412.03572*, 2024. 55

[10] 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*, 2023. 57

[11] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *NeurIPS Workshops*, 2023. 57

[12] 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. 56, 57

[13] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023. 11, 36, 37, 46, 50, 56, 57

[14] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators, 2024. URL https://openai.com/research/video-generation-models-as-world-simulators. 56, 57

[15] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 27, 29

[16] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *ICML*, 2024. 55, 56

[17] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *ICML*, 2024. 30, 31

[18] Brandon Castellano. Pyscenedetect, 2024. URL https://www.scenedetect.com. Video Cut Detection and Analysis Tool. 7

[19] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *CVPR*, 2022. 57

[20] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024. 56

[21] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024. 57

[22] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. 23

[23] Ting Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023. 22

[24] Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang-wei Chao, Byung Eun Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, et al. Panda-70m: Captioning 70m videos with multiple cross-modality teachers. In *CVPR*, 2024. 7, 16

[25] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *RSS*, 2023. 57

[26] Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, et al. Emu: Enhancing image generation models using photogenic needles in a haystack. *arXiv preprint arXiv:2309.15807*, 2023. 22, 57

[27] Alexandre de Brébisson and Pascal Vincent. The z-loss: a shift and scale invariant classification loss belonging to the spherical family. *arXiv preprint arXiv:1604.08859*, 2016. 28

[28] 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 *ICML*, 2023. 21

[29] Haoge Deng, Ting Pan, Haiwen Diao, Zhengxiong Luo, Yufeng Cui, Huchuan Lu, Shiguang Shan, Yonggang Qi, and Xinlong Wang. Autoregressive video generation without vector quantization. *arXiv preprint arXiv:2412.14169*, 2024. 56

[30] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 12, 16

[31] Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-stage dense face localisation in the wild. *CVPR*, 2020. 55

[32] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR Workshops*, 2018. 36

[33] Zihan Ding, Amy Zhang, Yuandong Tian, and Qinqing Zheng. Diffusion world model: Future modeling beyond step-by-step rollout for offline reinforcement learning. *arXiv preprint arXiv:2402.03570*, 2024. 55, 56

[34] 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. In *ICLR*, 2021. 56

[35] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *NeurIPS*, 2024. 57

[36] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 24, 27, 29, 30

[37] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. In *RSS*, 2022. 43

[38] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 57

[39] 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 *ICML*, 2024. 21

[40] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image Analysis*, 2003. 9

[41] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *ICRA*, 2017. 57

[42] FLUX. FLUX.1: Image generation, 2024. URL https://huggingface.co/black-forest-labs/FLUX.1-dev. 12, 57

[43] Stephanie Fu, Netanel Yakir Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. In *NeurIPS*, 2023. 39

[44] Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, et al. Datacomp: In search of the next generation of multimodal datasets. In *NeurIPS*, 2024. 10

[45] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. In *ECCV*, 2022. 57

[46] Philip Gage. A new algorithm for data compression. *The C Users Journal*, 1994. 28

[47] Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin P. Murphy, and Tim Salimans. Diffusion meets flow matching: Two sides of the same coin, 2024. URL https://diffusionflow.github.io/. 19

[48] Ruiyuan Gao, Kai Chen, Bo Xiao, Lanqing Hong, Zhenguo Li, and Qiang Xu. Magicdrivedit: High-resolution long video generation for autonomous driving with adaptive control. *arXiv preprint arXiv:2411.13807*, 2024. 57

[49] Ruiyuan Gao, Kai Chen, Enze Xie, Lanqing HONG, Zhenguo Li, Dit-Yan Yeung, and Qiang Xu. Magicdrive: Street view generation with diverse 3d geometry control. In *ICLR*, 2024. 57

[50] Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. Vista: A generalizable driving world model with high fidelity and versatile controllability. In *NeurIPS*, 2024. 57

[51] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 15

[52] Songwei Ge, Thomas Hayes, Harry Yang, Xi Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. Long video generation with time-agnostic vqgan and time-sensitive transformer. In *ECCV*, 2022. 57

[53] Songwei Ge, Seungjun Nah, Guilin Liu, Tyler Poon, Andrew Tao, Bryan Catanzaro, David Jacobs, Jia-Bin Huang, Ming-Yu Liu, and Yogesh Balaji. Preserve your own correlation: A noise prior for video diffusion models. In *ICCV*, 2023. 56, 57

[54] Yunhao Ge, Xiaohui Zeng, Jacob Samuel Huffman, Tsung-Yi Lin, Ming-Yu Liu, and Yin Cui. Visual fact checker: Enabling high-fidelity detailed caption generation. In *CVPR*, 2024. 10

[55] Shaona Ghosh, Prasoon Varshney, Erick Galinkin, and Christopher Parisien. Aegis: Online adaptive ai content safety moderation with ensemble of llm experts. *arXiv preprint arXiv:2404.05993*, 2024. 53, 54

[56] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In *CVPR*, 2023. 8

[57] Rohit Girdhar, Mannat Singh, Andrew Brown, Quentin Duval, Samaneh Azadi, Sai Saketh Rambhatla, Akbar Shah, Xi Yin, Devi Parikh, and Ishan Misra. Emu video: Factorizing text-to-video generation by explicit image conditioning. In *ECCV*, 2024. 56, 57

[58] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-exo4d: Understanding skilled human activity from first-and third-person perspectives. In *CVPR*, 2024. 16

[59] Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Li Fei-Fei, Irfan Essa, Lu Jiang, and José Lezama. Photorealistic video generation with diffusion models. In *ECCV*, 2024. 21

[60] Gunshi Gupta, Karmesh Yadav, Yarin Gal, Dhruv Batra, Zsolt Kira, Cong Lu, and Tim GJ Rudner. Pre-trained text-to-image diffusion models are versatile representation learners for control. In *ICLR Workshops*, 2024. 57

[61] Siddharth Gururani, Arun Mallya, Ting-Chun Wang, Rafael Valle, and Ming-Yu Liu. SPACE: Speech-driven Portrait Animation with Controllable Expression. In *ICCV*, 2023. 56

[62] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. 55

[63] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019. 55

[64] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *ICLR*, 2021. 55, 56

[65] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023. 55

[66] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *ICLR*, 2024. 55

[67] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 36, 51

[68] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation. *arXiv preprint arXiv:2404.02101*, 2024. 57

[69] Haoran He, Chenjia Bai, Ling Pan, Weinan Zhang, Bin Zhao, and Xuelong Li. Learning an actionable discrete diffusion policy via large-scale actionless video pre-training. In *NeurIPS*, 2024. 57

[70] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 57

[71] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 17, 41, 50

[72] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 1995. 57

[73] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 23

[74] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 19, 57

[75] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *NeurIPS*, 2022. 56

[76] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. In *ICLR*, 2023. 57

[77] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. Simple diffusion: End-to-end diffusion for high resolution images. In *ICML*, 2023. 22

[78] Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion (sid2): 1.5 fid on imagenet512 with pixel-space diffusion. *arXiv preprint arXiv:2410.19324*, 2024. 19

[79] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023. 49, 55, 56, 57

[80] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. 21

[81] Pu Hua, Minghuan Liu, Annabella Macaluso, Yunfeng Lin, Weinan Zhang, Huazhe Xu, and Lirui Wang. Gensim2: Scaling robot data generation with multi-modal and reasoning llms. *arXiv preprint arXiv:2410.03645*, 2024. 55

[82] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *CVPR*, 2024. 56

[83] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023. 54

[84] 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. 24

[85] Fan Jia, Weixin Mao, Yingfei Liu, Yucheng Zhao, Yuqing Wen, Chi Zhang, Xiangyu Zhang, and Tiancai Wang. Adriver-i: A general world model for autonomous driving. *arXiv preprint arXiv:2311.13549*, 2023. 57

[86] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. 27, 29

[87] 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. 37

[88] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 25

[89] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 14

[90] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022. 19

[91] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *CVPR*, 2024. 19

[92] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. In *CoRL*, 2024. 57

[93] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 2023. 36, 56

[94] Rahima Khanam and Muhammad Hussain. Yolov11: An overview of the key architectural enhancements. *arXiv preprint arXiv:2410.17725*, 2024. 53

[95] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 46

[96] Seung Wook Kim, Yuhao Zhou, Jonah Philion, Antonio Torralba, and Sanja Fidler. Learning to Simulate Dynamic Environments with GameGAN. In *CVPR*, 2020. 49, 55, 56

[97] Seung Wook Kim, , Jonah Philion, Antonio Torralba, and Sanja Fidler. DriveGAN: Towards a Controllable High-Quality Neural Simulation. In *CVPR*, 2021. 49, 55, 56, 57

[98] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 14, 57

[99] Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B Tenenbaum. Learning to act from actionless videos through dense correspondences. In *ICLR*, 2024. 57

[100] 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. In *ICML*, 2024. 11, 56, 57

[101] 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. 19, 25

[102] Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems*, 2023. 23, 29

[103] Max Ku, Cong Wei, Weiming Ren, Huan Yang, and Wenhu Chen. Anyv2v: A tuning-free framework for any video-to-video editing tasks. *TMLR*, 2024. 56

[104] KuaiShou. Kling, 2024. URL https://klingai.com/. 56

[105] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, 2022. 57

[106] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 14

[107] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023. 31

[108] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. In *ICLR*, 2024. 56

[109] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023. 56

[110] Hanxue Liang, Jiawei Ren, Ashkan Mirzaei, Antonio Torralba, Ziwei Liu, Igor Gilitschenski, Sanja Fidler, Cengiz Oztireli, Huan Ling, Zan Gojcic, et al. Feed-forward bullet-time reconstruction of dynamic scenes from monocular videos. *arXiv preprint arXiv:2412.03526*, 2024. 52

[111] 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. 56

[112] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, 2021. 41

[113] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *CVPR*, 2023. 56

[114] Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. Vila: On pre-training for visual language models. In *CVPR*, 2024. 10

[115] Kai-En Lin, Yen-Chen Lin, Wei-Sheng Lai, Tsung-Yi Lin, Yi-Chang Shih, and Ravi Ramamoorthi. Vision transformer for nerf-based view synthesis from a single input image. In *WACV*, 2023. 56

[116] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 12, 16

[117] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. In *ICCV*, 2023. 36

[118] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *CVPR*, 2024. 40, 41

[119] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 57

[120] Chang Liu, Rui Li, Kaidong Zhang, Yunwei Lan, and Dong Liu. Stablev2v: Stablizing shape consistency in video-to-video editing. *arXiv preprint arXiv:2411.11045*, 2024. 56

[121] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023. 24

[122] Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. World model on million-length video and language with blockwise ringattention. *CoRR*, 2024. 55

[123] Haozhe Liu, Shikun Liu, Zijian Zhou, Mengmeng Xu, Yanping Xie, Xiao Han, Juan C Pérez, Ding Liu, Kumara Kahatapitiya, Menglin Jia, et al. Mardini: Masked autoregressive diffusion for video generation at scale. *arXiv preprint arXiv:2410.20280*, 2024. 56

[124] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *ICCV*, 2023. 56

[125] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 23, 29

[126] Jiachen Lu, Ze Huang, Zeyu Yang, Jiahui Zhang, and Li Zhang. Wovogen: World volume-aware diffusion for controllable multi-camera driving scene generation. In *ECCV*, 2025. 57

[127] Luma. Dream machine, 2024. URL `https://lumalabs.ai/dream-machine`. 56

[128] Zhuoyan Luo, Fengyuan Shi, Yixiao Ge, Yujiu Yang, Limin Wang, and Ying Shan. Open-magvit2: An open-source project toward democratizing auto-regressive visual generation. *arXiv preprint arXiv:2409.04410*, 2024. 12, 57

[129] 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. 56

[130] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *ECCV*, 2020. 56

[131] Arun Mallya, Ting-Chun Wang, and Ming-Yu Liu. Implicit Warping for Animation with Image Sets. In *NeurIPS*, 2022. 56

[132] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023. 14, 28, 57, 58

[133] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *ICLR*, 2023. 55

[134] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 36, 56

[135] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 1995. 54

[136] Mistral and NVIDIA. Mistral-nemo-12b-instruct: A 12b parameter large language model, 2024. URL https://mistral.ai/news/mistral-nemo/. 18, 25

[137] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging AI applications. *CoRR*, abs/1712.05889, 2017. URL http://arxiv.org/abs/1712.05889. 1, 11

[138] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017. 55

[139] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024. 55

[140] NVIDIA. Isaac sim, 2024. URL https://developer.nvidia.com/isaac/sim. 37

[141] NVIDIA. Omniverse, 2024. URL https://www.nvidia.com/en-us/omniverse/. 37

[142] NVIDIA. Physx, 2024. URL https://github.com/NVIDIA-Omniverse/PhysX. 37

[143] NVIDIA. Edify 3d: Scalable high-quality 3d asset generation. *arXiv preprint arXiv:2411.07135*, 2024. 56

[144] NVIDIA. Transformer engine, 2024. URL https://github.com/NVIDIA/TransformerEngine. 24

[145] OpenAI. Tiktoken, 2022. URL https://github.com/openai/tiktoken. 28

[146] OpenAI. Dall·e 3, 2024. URL https://openai.com/dall-e. Accessed: [Insert access date here]. 32

[147] OpenAI. Sora, 2024. URL https://openai.com/sora/. 56

[148] Linfei Pan, Dániel Baráth, Marc Pollefeys, and Johannes L Schönberger. Global structure-from-motion revisited. In *ECCV*, 2025. 40, 41

[149] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 30

[150] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 20, 21

[151] Bowen Peng and Jeffrey Quesnelle. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation, 2023. 21

[152] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023. 27

[153] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 12

[154] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *ICLR*, 2024. 57

[155] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024. 7, 19, 25

[156] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023. 56

[157] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024. 57

[158] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, et al. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. In *ICLR*, 2024. 56

[159] 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. *JMLR*, 2020. 21, 23

[160] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 14

[161] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021. 57

[162] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 32, 57

[163] RAPIDS. Rapids: Libraries for end to end gpu data science, 2023. URL https://rapids.ai. 10

[164] Weiming Ren, Huan Yang, Ge Zhang, Cong Wei, Xinrun Du, Wenhao Huang, and Wenhu Chen. Consisti2v: Enhancing visual consistency for image-to-video generation. *arXiv preprint arXiv:2402.04324*, 2024. 56

[165] Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. *arXiv preprint arXiv:2303.07109*, 2023. 55, 56

[166] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. In *ICCV*, 2021. 56

[167] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 11, 19, 57

[168] Runway. Gen 3, 2024. URL https://runwayml.com/research/introducing-gen-3-alpha. 56

[169] Seyedmorteza Sadat, Jakob Buhmann, Derek Bradley, Otmar Hilliges, and Romann M Weber. Litevae: Lightweight and efficient variational autoencoders for latent diffusion models. *arXiv preprint arXiv:2405.14477*, 2024. 14

[170] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022. 23

[171] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *CVPR*, 2022. 56

[172] Paul D Sampson. Fitting conic sections to "very scattered" data: An iterative refinement of the bookstein algorithm. *Computer graphics and image processing*, 1982. 36, 51

[173] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 36, 41

[174] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 36, 41

[175] Christoph Schuhmann. Improved Aesthetic Predictor, 2022. URL https://github.com/christophschuhmann/improved-aesthetic-predictor. 9

[176] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 56

[177] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019. 29

[178] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 15

[179] Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *NeurIPS*, 2021. 41

[180] 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. 19, 57

[181] Tomás Soucek and Jakub Lokoc. Transnet v2: An effective deep network architecture for fast shot transition detection. In *ACM MM*, 2024. 7

[182] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024. 20

[183] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024. 12, 57

[184] Quan Sun, Yufeng Cui, Xiaosong Zhang, Fan Zhang, Qiying Yu, Yueze Wang, Yongming Rao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Generative multimodal models are in-context learners. In *CVPR*, 2024. 57

[185] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH*, 2023. 36

[186] Shitao Tang, Litong Feng, Zhanghui Kuang, Yimin Chen, and Wei Zhang. Fast video shot transition localization with deep structured models. In *ACCV*, 2018. 7

[187] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *ECCV*, 2016. 56

[188] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *URL https://arxiv.org/abs/2405.09818*, 2024. 57

[189] Gemma Team. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118. 29

[190] 1X Technologies. 1xgpt, 2024. URL https://github.com/1x-technologies/1xgpt. 43

[191] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 15

[192] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. In *NeurIPS*, 2021. 52

[193] Yao Teng, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Accelerating auto-regressive text-to-image generation with training-free speculative jacobi decoding. *arXiv preprint arXiv:2410.01699*, 2024. 31

[194] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020. 56

[195] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing motion and content for video generation. In *CVPR*, 2018. 56

[196] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Fvd: A new metric for video generation. In *ICLR Workshops*, 2019. 17, 41, 50

[197] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. *arXiv preprint arXiv:2408.14837*, 2024. 55, 56

[198] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017. 14, 57

[199] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 27, 56

[200] Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description. In *ICLR*, 2023. 57

[201] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *CoRL*, 2023. 16

[202] Junke Wang, Yi Jiang, Zehuan Yuan, Binyue Peng, Zuxuan Wu, and Yu-Gang Jiang. Omnitokenizer: A joint image-video tokenizer for visual generation. *arXiv preprint arXiv:2406.09399*, 2024. 12

[203] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 56

[204] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 10

[205] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *NeurIPS*, 2018. 56

[206] Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. Few-shot video-to-video synthesis. In *NeurIPS*, 2019. 56

[207] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. In *CVPR*, 2021. 56

[208] Xiang Wang, Shiwei Zhang, Changxin Gao, Jiayu Wang, Xiaoqiang Zhou, Yingya Zhang, Luxin Yan, and Nong Sang. Unianimate: Taming unified video diffusion models for consistent human image animation. *arXiv preprint arXiv:2406.01188*, 2024. 56

[209] Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, Jiagang Zhu, and Jiwen Lu. Drivedreamer: Towards real-world-driven world models for autonomous driving. *arXiv preprint arXiv:2309.09777*, 2023. 57

[210] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024. 57

[211] Yaohui Wang, Xinyuan Chen, Xin Ma, Shangchen Zhou, Ziqi Huang, Yi Wang, Ceyuan Yang, Yinan He, Jiashuo Yu, Peiqing Yang, et al. Lavie: High-quality video generation with cascaded latent diffusion models. *arXiv preprint arXiv:2309.15103*, 2023. 57

[212] Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Zun Wang, Yansong Shi, et al. Internvideo2: Scaling foundation models for multimodal video understanding. In *ECCV*, 2025. 9

[213] Yuqi Wang, Jiawei He, Lue Fan, Hongxin Li, Yuntao Chen, and Zhaoxiang Zhang. Driving into the future: Multiview visual forecasting and planning with world model for autonomous driving. In *CVPR*, 2024. 57

[214] Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024. 57

[215] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation. In *ACM SIGGRAPH*, 2024. 57

[216] Qizhen Weng, Lingyun Yang, Yinghao Yu, Wei Wang, Xiaochuan Tang, Guodong Yang, and Liping Zhang. Beware of fragmentation: Scheduling {GPU-Sharing} workloads with fragmentation gradient descent. In *USENIX ATC*, 2023. 11

[217] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 56

[218] Mitchell Wortsman, Peter J Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, et al. Small-scale proxies for large-scale transformer training instabilities. *arXiv preprint arXiv:2309.14322*, 2023. 21, 27, 28

[219] Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Daxin Jiang, and Nan Duan. Nüwa: Visual synthesis pre-training for neural visual world creation. In *ECCV*, 2022. 57

[220] 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 *ICCV*, 2023. 9

[221] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *CoRL*, 2023. 56

[222] Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, et al. Vila-u: a unified foundation model integrating visual understanding and generation. *arXiv preprint arXiv:2409.04429*, 2024. 57

[223] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 14

[224] Dejia Xu, Weili Nie, Chao Liu, Sifei Liu, Jan Kautz, Zhangyang Wang, and Arash Vahdat. Camco: Camera-controllable 3d-consistent image-to-video generation. *arXiv preprint arXiv:2406.02509*, 2024. 41, 42, 43, 57

[225] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. In *NeurIPS*, 2019. 21

[226] Fuzhao Xue, Yukang Chen, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, et al. Longvila: Scaling long-context visual language models for long videos. *arXiv preprint arXiv:2408.10188*, 2024. 10

[227] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021. 12, 57

[228] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. 29

[229] Cheng-Yen Yang, Hsiang-Wei Huang, Wenhao Chai, Zhongyu Jiang, and Jenq-Neng Hwang. Samurai: Adapting segment anything model for zero-shot visual tracking with motion-aware memory. *arXiv preprint arXiv:2411.11922*, 2024. 39

[230] Jiazhi Yang, Shenyuan Gao, Yihang Qiu, Li Chen, Tianyu Li, Bo Dai, Kashyap Chitta, Penghao Wu, Jia Zeng, Ping Luo, et al. Generalized predictive model for autonomous driving. In *CVPR*, 2024. 57

[231] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023. 55, 56

[232] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024. 12, 56

[233] 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. 58

[234] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 56

[235] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, 2020. 16

[236] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *TMLR*, 2022. 57

[237] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, and Lu Jiang. MAGVIT: Masked generative video transformer. In *CVPR*, 2023. 57, 58

[238] Lijun Yu, Jose Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A Ross, and Lu Jiang. Language model beats diffusion - tokenizer is key to visual generation. In *ICLR*, 2024. 57

[239] Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. *arXiv preprint arXiv:2406.07550*, 2024. 57

[240] Sihyun Yu, Kihyuk Sohn, Subin Kim, and Jinwoo Shin. Video probabilistic diffusion models in projected latent space. In *CVPR*, 2023. 57

[241] Yan Zeng, Guoqiang Wei, Jiani Zheng, Jiaxin Zou, Yang Wei, Yuchen Zhang, and Hang Li. Make pixels dance: High-dynamic video generation. In *CVPR*, 2024. 57

[242] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023. 55

[243] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, 2019. 21

[244] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 36

[245] Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. In *NeurIPS*, 2024. 56

[246] Guosheng Zhao, Xiaofeng Wang, Zheng Zhu, Xinze Chen, Guan Huang, Xiaoyi Bao, and Xingang Wang. Drivedreamer-2: Llm-enhanced world models for diverse driving video generation. *arXiv preprint arXiv:2403.06845*, 2024. 56

[247] Yue Zhao, Yuanjun Xiong, and Philipp Krähenbühl. Image and video tokenization with binary spherical quantization. *arXiv preprint arXiv:2406.07548*, 2024. 57

[248] Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039*, 2024. 58

[249] Siyuan Zhou, Yilun Du, Jiaben Chen, YANDONG LI, Dit-Yan Yeung, and Chuang Gan. Robodreamer: Learning compositional world models for robot imagination. In *ICML*, 2024. 57

[250] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 2018. 36, 41, 56

[251] Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. Irasim: Learning interactive real-robot action simulators. *arXiv preprint arXiv:2406.14540*, 2024. 46, 48

[252] Wentao Zhu, Yufang Huang, Xiufeng Xie, Wenxian Liu, Jincan Deng, Debing Zhang, Zhangyang Wang, and Ji Liu. Autoshot: A short video dataset and state-of-the-art shot boundary detection. In *CVPR Workshops*, 2023. 7