



# Open-Sora: Democratizing Efficient Video Production for All

Zangwei Zheng\*, Xiangyu Peng\*, Tianji Yang, Chenhui Shen, Shenggui Li,  
Hongxin Liu, Yukun Zhou, Tianyi Li, Yang You

HPC-AI Tech

## Abstract

Vision and language are the two foundational senses for humans, and they build up our cognitive ability and intelligence. While significant breakthroughs have been made in AI language ability, artificial visual intelligence, especially the ability to generate and simulate the world we see, is far lagging behind. To facilitate the development and accessibility of artificial visual intelligence, we created Open-Sora, an open-source video generation model designed to produce high-fidelity video content. Open-Sora supports a wide spectrum of visual generation tasks, including text-to-image generation, text-to-video generation, and image-to-video generation. The model leverages advanced deep learning architectures and training/inference techniques to enable flexible video synthesis, which could generate video content of up to 15 seconds, up to 720p resolution, and arbitrary aspect ratios. Specifically, we introduce Spatial-Temporal Diffusion Transformer (STDiT), an efficient diffusion framework for videos that decouples spatial and temporal attention. We also introduce a highly compressive 3D autoencoder to make representations compact and further accelerate training with an ad hoc training strategy. Through this initiative, we aim to foster innovation, creativity, and inclusivity within the community of AI content creation. By embracing the open-source principle, Open-Sora democratizes full access to all the training/inference/data preparation codes as well as model weights. All resources are publicly available at: <https://github.com/hpcatech/Open-Sora>.

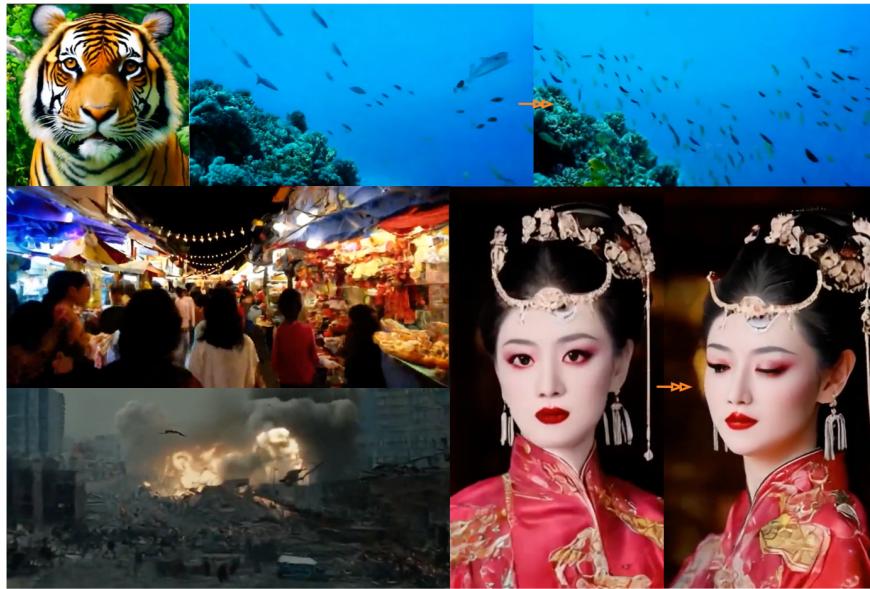


Figure 1: Open-Sora can generate high-fidelity videos. Images with arrows illustrate the motion.

# 1 Introduction

The field of video generation models has advanced rapidly, with UNet-based architectures demonstrating remarkable performance [2, 11, 27]. Following the impressive results showcased by OpenAI’s Sora [3], Transformer-based models such as DiT [22] have highlighted the potential of scaling diffusion models [12, 17, 20]. In this context, we introduce Open-Sora, one of the earliest projects to reproduce Sora, achieving strong results and garnering significant attention.

In the Open-Sora project, we provide comprehensive support for training video generation models, offering the complete suite of necessary code. This includes everything from processing and filtering training data to the training code and model weights, all made available to the community to foster its development. We have successfully reproduced nearly all the techniques mentioned in the Sora report, enabling the generation of videos up to 16 seconds in length, at multiple resolutions up to 720p, with controllable motion dynamics for text-to-video and image-to-video tasks. Figure 1 showcases examples of videos generated by our model.

**Overview** Open-Sora has undergone three major updates, corresponding to versions 1.0 (Mar 2024), 1.1 (April 2024), and 1.2 (June 2024). Each release has an online report in the repository. Unless otherwise specified, Open-Sora specifically refers to Open-Sora 1.2; elsewhere, Open-Sora is used as the default reference. This paper provides a comprehensive overview of the techniques behind these versions and highlights the performance of version 1.2. In Section 2, we discuss data composition and preprocessing methods. Section 3 focuses on the model architecture, and Section 4 details the training process.

## 2 Data

### 2.1 Data Source

The dataset used is all open-sourced to make the model training fully reproducible. In total, 30M video clips ranging from 2s to 16s are generated, with a total duration of 80k hours. **Webvid-10M** [1] contains 10M video-text pairs from the stock footage sites. The videos are low-resolution and have a watermark. **Panda-70M** [6] is a large-scale dataset with 70M video-caption pairs. We use a 20M high-quality subset for training. **HD-VG-130M** comprises 130M text-video pairs. The caption is generated by BLIP-2. We find the scene and the text quality are relatively poor. **MiraData** [32] a high-quality dataset with 77k long videos, mainly from games and city exploration. **Vripit** [36] a densely annotated dataset of 400k videos. **Inter4K** [28] is a dataset containing 1K video clips with 4K resolution.

In addition, we get free-licensed videos from **Pexels**, **Pixabay**, and **Mixkit**. Most videos from these websites are of high quality. We really appreciate these great platforms and the contributors.

The image dataset, which we use to train together with videos, contains around 3M images in total. **LAION** [26] is a large-scale open dataset, and we use a subset with an aesthetic score larger than 6.5. **Unsplash-lite** [31] dataset comprises 25k nature-themed Unsplash photos. This dataset covers a vast range of uses and contexts.

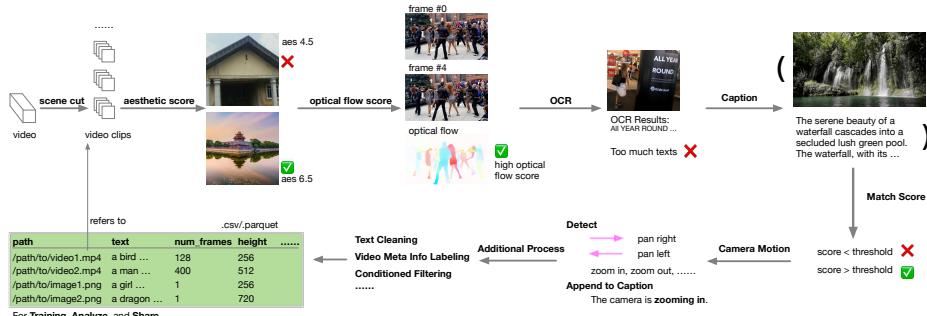


Figure 2: Open-Sora Data Labeling and Processing Pipeline

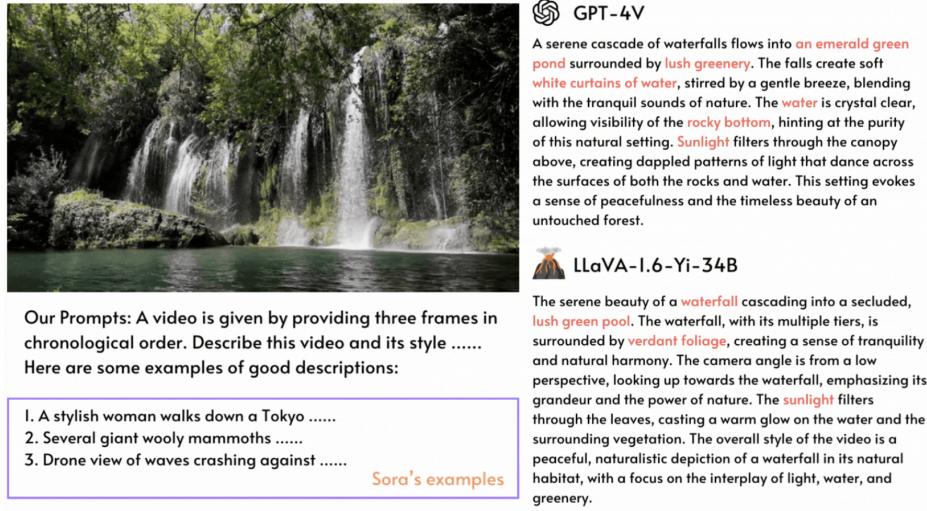


Figure 3: Open-Sora Video Captioning

## 2.2 Data Pre-processing

High-quality data is crucial for training good generation models. To this end, we establish a complete pipeline for data processing, which could seamlessly convert raw videos to high-quality video-text pairs. The pipeline is shown in Figure 2. To begin the data pre-process pipeline, we use PySceneCut [7] to detect scenes and cut videos into clips.

For high-quality video filtering, we mainly follow the SVD data preprocessing pipeline [2]. **Aesthetic Score** measures the aesthetic preference of frames in the video. We use the scorer from LAION [26] and use the average score among three sampled frames. **Optical Flow Score** measures the dynamics scale of the video and can be used to filter videos with low motion. We use the UniMatch [34] model for this task. Some videos are of dense text scenes like news broadcasts and advertisements, which are not desired for training. **Optical Character Recognition (OCR)** recognizes texts available in the videos, and those with too much text are removed. We use the DBNet++ model implemented by MOCR [16].

To provide good captions for videos, we utilize GPT-4V and PLLaVA [35]. The former one provides API, and the latter one is open-source and can be deployed by us. Although with some level of hallucinations, the results are enough to train a text-to-video model. In practice, we use the pretrained PLLaVA 13B model and select 4 frames from each video for captioning with a spatial pooling shape of 2\*2. However, the captioning model can hardly produce information about camera movement. Thus, we detect camera motion with optical flow and append the text to the caption.

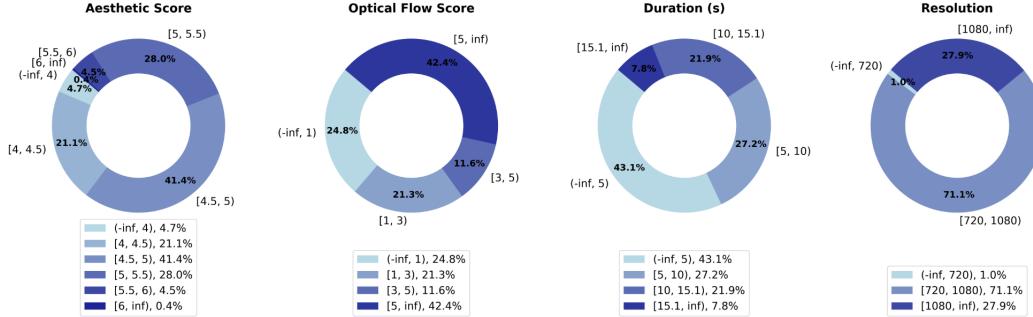


Figure 4: Distribution of data for last stage training.

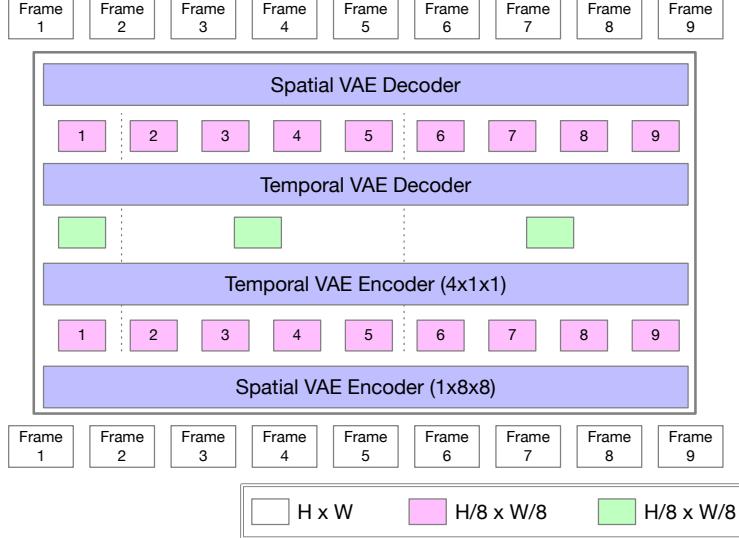


Figure 5: 3D autoencoder by utilizing a pretrained 2D autoencoder.

Some statistics of the video data used in the last stage are shown in Figure 4. We present basic statistics of duration and resolution, as well as aesthetic score and optical flow score distribution. We also extract tags for objects and actions from video captions and count their frequencies.

### 3 Model Architecture

Our video generation framework follows Sora’s report [3]. The videos are first compressed by a video compression network, namely a 3d autoencoder. A text encoder encodes texts. Then, a DiT-like transformer handles the video and text latent.

#### 3.1 3D Autoencoder

In Open-Sora 1.0 and 1.1, we utilized Stability-AI’s 2D VAE [25] (84M parameters), which compresses video spatially by a factor of 8x8. To reduce the temporal dimension, we downsampled by extracting one frame every three frames. However, this approach resulted in low temporal fluency due to a reduction in generated FPS. To address this limitation, Open-Sora 1.2 introduces a video compression network inspired by OpenAI’s Sora, achieving 4x compression in the temporal dimension. This eliminates the need for frame extraction, enabling video generation at the original FPS.

Given the high computational demands of training a 3D VAE, we aimed to leverage the knowledge embedded in the 2D VAE. Post-compression by the 2D VAE, we observed that temporally adjacent features remain highly correlated. Based on this insight, we developed a simple yet effective video compression network that first compresses spatially by 8x8 and subsequently temporally by 4x. The network architecture is illustrated in Figure 5.

We initialized the 2D VAE using SDXL’s pre-trained VAE [23]. For the 3D VAE, we adopted the architecture of Magvit-v2’s VAE [37], comprising 300M parameters. Combined with the 2D VAE, the total parameter count of the video compression network is 384M. The 3D VAE was trained for 1.2M steps with a local batch size of 1, using videos from Pexels and Pixabay. The training data primarily consisted of 17-frame clips at a resolution of 256x256. To improve image reconstruction accuracy, causal convolutions were employed within the 3D VAE.

Our training process consists of three stages: **Stage 1** (0–380k steps) Trained on 8 GPUs with the 2D VAE weights frozen. Objectives included reconstructing 2D VAE-compressed features and applying an identity loss to align features from the 3D VAE with those of the 2D VAE. The identity loss enabled faster convergence and improved initial image reconstruction quality. **Stage 2** (380k–640k steps) The identity loss was removed, and the 3D VAE was trained to refine its temporal understanding. **Stage 3** (640k–1.2M steps) Reconstruction of 2D VAE features was found insufficient for further

Table 1: Performance of VAE on validation dataset from pixabay.

Model	SSIM↑	PSNR↑
Open-Sora-Plan 1.1	0.882	29.890
Open-Sora 1.2	0.880	30.590

improvement, so the loss was replaced with a direct reconstruction of original videos. This stage utilized 24 GPUs and incorporated mixed video-length training by randomizing video lengths (up to 34 frames) with appropriate zero-padding, improving robustness to varying video durations.

During the first two stages, the dataset comprised 80% video and 20% image data. For video training, 17-frame clips were used, while image data was zero-padded to match the input format. However, we found this approach led to blurriness in videos of non-standard lengths. Mixed-length training in Stage 3 effectively resolved this issue.

The stacked VAE architecture requires minimal memory during inference, as inputs are already compressed. For efficiency, input videos are segmented into 17-frame clips. Compared to another open-source 3D VAE [17], our model achieves comparable performance with significantly reduced computational cost.

### 3.2 Architecture

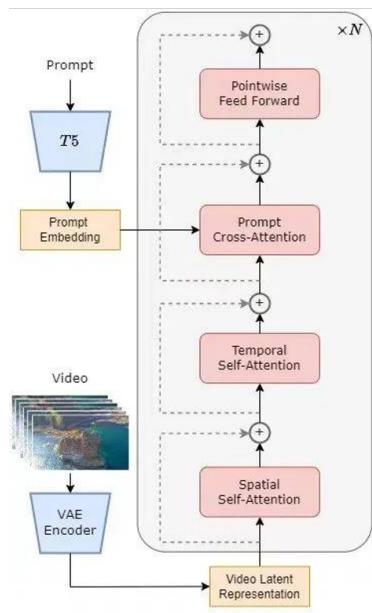


Figure 6: Open-Sora diffusion transformer architecture.

Our model architecture builds upon PixArt [4], an image diffusion transformer, where text is encoded using a T5 text encoder [24], and cross-attention is applied between video and text latents. To enable efficient video generation, we employ a spatial-temporal attention mechanism, Spatial-Temporal Diffusion Transformer (STDiT), inspired by Latte [20], replacing full attention on all tokens. Specifically, spatial self-attention is applied within each frame, while temporal attention is applied across frames at the same spatial location.

To focus on video generation, we designed the model to build upon a robust pre-trained image generation model. The model is initialized with PixArt- $\alpha$ , a T5-conditioned DiT structure optimized for high-quality and efficient image generation. The projection layers for the newly introduced temporal attention are initialized to zero, preserving the model’s original image generation capabilities at the start of training. The inclusion of temporal attention doubles the parameter count from 580M to 1.1B.

Several modifications were introduced to enhance training stability and performance. Following best practices in large language models, we replaced sinusoidal positional encoding with rotary positional embeddings (RoPE) [29] for temporal attention, as this aligns better with the sequence prediction nature of video generation tasks. Additionally, inspired by SD3 [10], we applied QK-normalization to all attention mechanisms to improve training stability, particularly under half-precision training. We used a small epsilon ( $1 \times 10^{-15}$ ) for QK-normalization, which proved effective in avoiding training spikes and ensuring smoother optimization [21]. These enhancements, combined with the spatial-temporal attention mechanism, enable our model to transition seamlessly from high-quality image generation to efficient video generation while maintaining stability and performance throughout the training process.

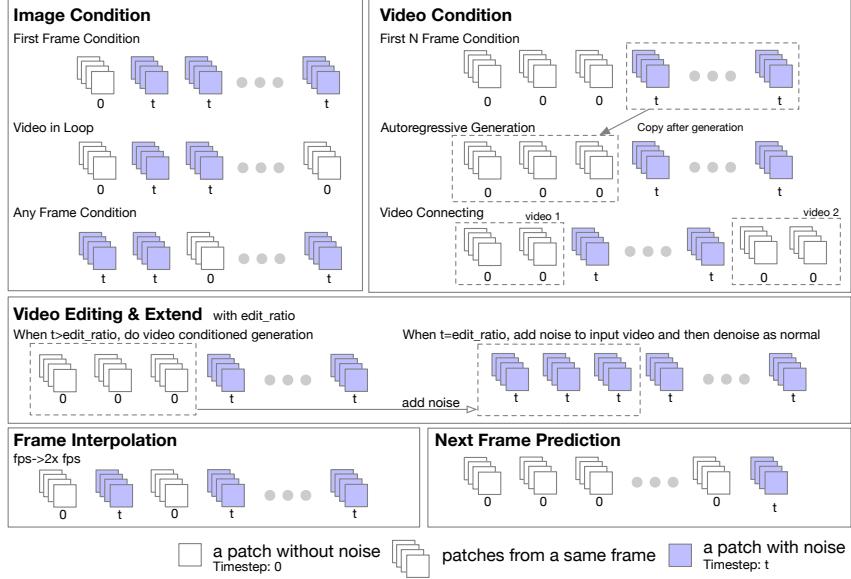


Figure 7: A general framework for image and video to video generation.

### 3.3 Conditioning

While text-to-video generation is highly versatile, certain applications require greater control. Transformers are inherently adaptable and can be extended to support tasks such as image-to-image and video-to-video generation. To enable such conditioning, we propose a masking strategy for image and video inputs, as illustrated in Figure 7.

In this strategy, frames designated for conditioning are unmasked. During the forward pass, these unmasked frames are assigned a timestep of 0, while other frames retain their diffusion timesteps. However, directly applying this strategy to a pre-trained model often yields suboptimal results, as the diffusion model has not been trained to manage mixed timesteps within a single sample.

Inspired by UL2 [30], we address this issue by introducing a random masking strategy during training. Specifically, frames are randomly unmasked with patterns such as the first frame, the first k frames, the last frame, the last k frames, a combination of the first and last k frames, or entirely random frames. Using Open-Sora 1.0 as a baseline, we experimented with masking applied to 50% of training samples and observed that the model effectively learned image and video conditioning capabilities after 10k steps, with minimal impact on text-to-video performance. A lower masking probability (e.g., 30%) resulted in reduced conditioning effectiveness. Consequently, we pre-trained the model from scratch with this masking strategy.

To further enhance the model’s control capabilities, we appended scores to captions, using them as additional conditioning inputs. These scores include aesthetic scores, motion scores, and camera motion descriptors. For instance, a caption for a video with an aesthetic score of 5.5, a motion score of 10, and detected camera motion of “pan left” would be formatted as: *[Original Caption] aesthetic score: 5.5, motion score: 10, camera motion: pan left*. During inference, these scores can also be adjusted to influence video generation. For camera motion conditioning, we manually labeled 13,000 high-confidence clips.

This approach provides the model with a nuanced understanding of conditioning inputs, improving its ability to generate high-quality, context-aware videos across a variety of tasks.

## 4 Training Strategy

Training video generation models demands substantial computational resources. To optimize efficiency, we adapt a pre-trained image generation model to a video generation model and employ a multi-stage training strategy. Specifically, we utilize flow matching [18] with a learning rate of  $5 \times 10^{-5}$ . The entire training process spans 68k steps and requires approximately 35,000 H100 GPU

hours. These approaches significantly reduces the training cost while achieving high-quality video generation performance.

#### 4.1 Multi-resolution and Multi-aspect-ratio

As highlighted in Sora’s report, training with the original resolution, aspect ratio, and length of videos improves sampling flexibility and enhances framing and composition. To achieve this goal, we evaluated three approaches:

- NaViT [9]: This method supports dynamic sizes within the same batch through masking with minimal efficiency loss. However, its implementation is complex and may not fully leverage optimized kernels like Flash Attention [8].
- Padding (FiT [19]): This approach supports dynamic sizes within the same batch by padding smaller resolutions to match the largest one. While simple, padding results in inefficient memory usage for varying resolutions.
- Bucket (SDXL [23], PixArt [4]): This method supports dynamic sizes across different batches by grouping samples into pre-defined “buckets.” Within each batch, the resolution, frame count, and aspect ratio are fixed. Bucketing avoids the complexities of masking or padding and benefits from optimized operations on uniform-sized inputs. However, it limits flexibility to a fixed set of sizes.

For simplicity and efficiency, we adopt the bucket-based approach. We pre-define a set of fixed resolutions, aspect ratios, and frame lengths and allocate samples to buckets accordingly. Each bucket is defined as a triplet of (resolution, number of frames, aspect ratio) that covers most common video formats. Before each training epoch, the dataset is shuffled, and samples are assigned to the largest bucket that fits their resolution and frame length.

To further optimize computational resources, we introduce two additional attributes for each bucket: probability of keeping in the bucket and batch size. According to the probability, high-resolution videos are downsampled to a lower resolution, effectively reducing computational cost. Batch sizes are adjusted per bucket to balance GPU load, ensuring efficient resource utilization. By fine-tuning these parameters, we achieve a balanced distribution of samples across buckets and improve overall training efficiency while maintaining high-quality video generation.

This bucket-based strategy provides a practical trade-off between implementation simplicity and computational efficiency, enabling flexible training with diverse video resolutions and aspect ratios.

#### 4.2 Model Adaptation

We begin with the PixArt- $\Sigma$  2K checkpoint [5], a model trained with DDPM [13] and SDXL VAE at much higher resolutions. The model is efficiently adapted to video generation tasks by finetuning on a smaller dataset. The adaptation process comprises several sequential stages, all conducted on 8 H100 GPUs.

1. Multi-resolution image generation: Training the model to handle resolutions from 144p to 2K over 20k steps.
2. QK-normalization integration: Adding QK-norm for stability, 18k training steps.
3. Transition to rectified flow: Shifting from discrete-time DDPM to continuous-time rectified flow, with 10k training steps.
4. Enhanced rectified flow training: Incorporating logit-norm sampling and resolution-aware timestep sampling, training for 33k steps.
5. Smaller AdamW epsilon: Following SD3, using a reduced epsilon ( $1 \times 10^{-15}$ ) with QK-norm for stability, trained for 8k steps.
6. New VAE and FPS conditioning: Replacing the original VAE with Open-Sora’s, adding FPS conditioning to timestep conditioning, and training for 25k steps. Normalizing each channel proved crucial for rectified flow training.
7. Temporal attention blocks: Adding zero-initialized temporal attention blocks, initially trained on images for 3k steps.

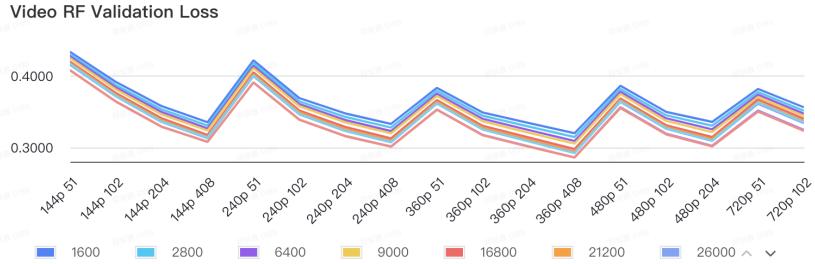


Figure 8: Validation loss for varying lengths and different resolutions.

8. Mask strategy for temporal blocks: Focusing temporal attention blocks exclusively on videos using a masking strategy, trained for 38k steps.

After completing this adaptation, the model retains its ability to generate high-quality images while gaining multiple benefits for video generation.

1. Accelerated training and inference: Rectified flow reduces the number of sampling steps from 100 to 30 for videos, significantly decreasing inference time.
2. Enhanced stability: QK-norm enables more aggressive optimization, improving training efficiency.
3. Efficient temporal compression: The new VAE compresses the temporal dimension by a factor of 4, reducing the computational cost.
4. Resolution flexibility: The model can generate videos at multiple resolutions, from 144p to 2K, supporting diverse use cases.

This comprehensive adaptation not only enhances the model’s video generation capabilities but also ensures efficient and scalable training, setting a new standard for open-source diffusion-based video generation.

### 4.3 Multi-stage Training

To optimize performance within a limited computational budget, we carefully organized the training data by quality and split the training process into three stages. The model was trained on a  $12 \times 8$  GPU setup over approximately two weeks, completing around 70k steps.

In the first stage, the model was trained on the Webvid-10M dataset (40k hours of video) for 30k steps (2 epochs). This dataset primarily contains videos with resolutions below 360p and watermarks, making it ideal for initial training. We focused on videos with resolutions of 240p and 360p, spanning lengths of 2 to 16 seconds. The original dataset captions were used for training.

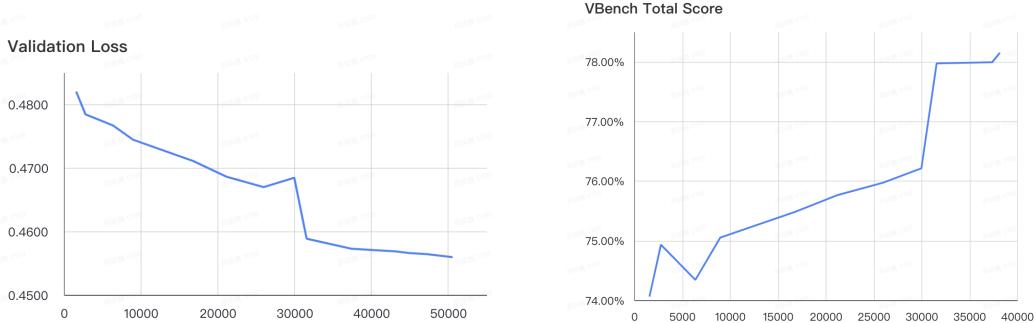


Figure 9: Validation loss and VBench score during training.

Table 2: VBench score comparison of different open-source models.

Model	Total Score (%)	Quality Score (%)	Semantic Score (%)
CogVideo [14]	67.01	72.06	46.83
Latte [20]	77.29	79.72	67.58
LaVie [33]	77.08	78.78	70.31
Show-1 [38]	78.93	80.42	72.98
OpenSoraPlan V1.1 [17]	78.00	80.91	66.38
OpenSoraPlan V1.2 [17]	75.98	<b>81.51</b>	53.88
OpenSoraPlan V1.3 [17]	77.23	80.14	65.62
Open-Sora 1.0	75.91	78.82	64.28
Open-Sora 1.1	75.66	77.74	67.36
Open-Sora 1.2	<b>79.76</b>	81.35	<b>73.39</b>

In the second stage, we trained the model on the Panda-70M dataset. Given the variable quality of this dataset, we used the official 30M subset, filtered to retain only videos with aesthetic scores above 4.5, resulting in a 20M subset (41k hours). Training primarily focused on resolutions of 360p and 480p for 23k steps, equivalent to 0.5 epochs. Although this training stage was not fully completed, it provided sufficient improvements to prepare the model for broader use.

The final stage involved a curated collection of approximately 2M high-quality video clips from diverse sources, totaling 5k hours. While videos from MiraData and Vript included captions generated by GPT, other sources were captioned using PLLaVA. This stage focused on higher resolutions (720p and 1080p) to enhance the model’s capability to handle larger resolutions. A 25% masking ratio was applied during training, which spanned 15k steps (around 2 epochs).

For validation, we sampled 1k videos from Pixabay to evaluate the model’s performance. The evaluation loss was calculated for images and videos of varying lengths (2s, 4s, 8s, 16s) across different resolutions (144p, 240p, 360p, 480p, 720p). Losses were averaged over 10 equidistant timesteps for each configuration.

We also tracked VBench scores during training. VBench [15] is an automated benchmark for evaluating short video generation. Scores were computed using 240p 2-second videos, providing additional validation of the model’s progress. Both evaluation loss and VBench scores confirm consistent improvements in the model’s performance throughout the training process. The VBench score and validation loss during training are shown in Figure 9.

More samples of video generation are available at <https://hpcatech.github.io/Open-Sora/>. Table 2 presents the VBench scores of various models, demonstrating that Open-Sora achieves state-of-the-art performance in video generation among open-source models.

## 5 Conclusion

Open-Sora represents a significant step forward in open-source video generation, providing a comprehensive framework that includes data processing, training code, and model weights. By successfully reproducing key techniques from the Sora report and enabling the generation of high-quality videos up to 16 seconds long, with resolutions up to 720p and controllable motion dynamics, Open-Sora democratizes access to advanced video generation technology. This initiative not only fosters community collaboration but also sets a foundation for future advancements in the field.

## References

- [1] M. Bain, A. Nagrani, G. Varol, and A. Zisserman, “Frozen in time: A joint video and image encoder for end-to-end retrieval,” in *IEEE International Conference on Computer Vision*, 2021.
- [2] A. Blattmann *et al.*, “Stable video diffusion: Scaling latent video diffusion models to large datasets,” *arXiv preprint arXiv:2311.15127*, 2023.
- [3] T. Brooks *et al.*, “Video generation models as world simulators,” 2024. [Online]. Available: <https://openai.com/research/video-generation-models-as-world-simulators>.
- [4] J. Chen *et al.*, “Pixart-alpha: Fast training of diffusion transformer for photorealistic text-to-image synthesis,” *arXiv preprint arXiv:2310.00426*, 2023.
- [5] J. Chen *et al.*, “Pixart-sigma: Weak-to-strong training of diffusion transformer for 4k text-to-image generation,” in *European Conference on Computer Vision*, Springer, 2025, pp. 74–91.
- [6] T.-S. Chen *et al.*, “Panda-70m: Captioning 70m videos with multiple cross-modality teachers,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 13 320–13 331.
- [7] P. Contributors, *Video cut detection and analysis tool*, 2024. [Online]. Available: <https://github.com/Breakthrough/PySceneDetect>.
- [8] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, “Flashattention: Fast and memory-efficient exact attention with io-awareness,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 344–16 359, 2022.
- [9] M. Dehghani *et al.*, “Patch n’pack: Navit, a vision transformer for any aspect ratio and resolution,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [10] P. Esser *et al.*, “Scaling rectified flow transformers for high-resolution image synthesis,” in *Forty-first International Conference on Machine Learning*, 2024.
- [11] Y. Guo *et al.*, “Animatediff: Animate your personalized text-to-image diffusion models without specific tuning,” *arXiv preprint arXiv:2307.04725*, 2023.
- [12] A. Gupta *et al.*, “Photorealistic video generation with diffusion models,” in *European Conference on Computer Vision*, Springer, 2025, pp. 393–411.
- [13] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.
- [14] W. Hong, M. Ding, W. Zheng, X. Liu, and J. Tang, “Cogvideo: Large-scale pretraining for text-to-video generation via transformers,” *arXiv preprint arXiv:2205.15868*, 2022.
- [15] Z. Huang *et al.*, “Vbench: Comprehensive benchmark suite for video generative models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 807–21 818.
- [16] M. Liao, Z. Zou, Z. Wan, C. Yao, and X. Bai, “Real-time scene text detection with differentiable binarization and adaptive scale fusion,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 1, pp. 919–931, 2022.
- [17] B. Lin *et al.*, “Open-sora plan: Open-source large video generation model,” *arXiv preprint arXiv:2412.00131*, 2024.
- [18] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, “Flow matching for generative modeling,” *arXiv preprint arXiv:2210.02747*, 2022.
- [19] Z. Lu *et al.*, “Fit: Flexible vision transformer for diffusion model,” *arXiv preprint arXiv:2402.12376*, 2024.
- [20] X. Ma *et al.*, “Latte: Latent diffusion transformer for video generation,” *arXiv preprint arXiv:2401.03048*, 2024.
- [21] I. Molybog *et al.*, “A theory on adam instability in large-scale machine learning,” *arXiv preprint arXiv:2304.09871*, 2023.
- [22] W. Peebles and S. Xie, “Scalable diffusion models with transformers,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4195–4205.
- [23] D. Podell *et al.*, “Sdxl: Improving latent diffusion models for high-resolution image synthesis,” *arXiv preprint arXiv:2307.01952*, 2023.
- [24] C. Raffel *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

- [25] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-resolution image synthesis with latent diffusion models*, 2021. arXiv: [2112.10752 \[cs.CV\]](https://arxiv.org/abs/2112.10752).
- [26] C. Schuhmann *et al.*, ‘‘Laion-5b: An open large-scale dataset for training next generation image-text models,’’ *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 278–25 294, 2022.
- [27] U. Singer *et al.*, ‘‘Make-a-video: Text-to-video generation without text-video data,’’ *arXiv preprint arXiv:2209.14792*, 2022.
- [28] A. Stergiou and R. Poppe, ‘‘Adapool: Exponential adaptive pooling for information-retaining downsampling,’’ 2021.
- [29] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. L. Roformer, ‘‘Enhanced transformer with rotary position embedding., 2021,’’ DOI: <https://doi.org/10.1016/j.neucom>, 2023.
- [30] Y. Tay *et al.*, ‘‘UI2: Unifying language learning paradigms,’’ *arXiv preprint arXiv:2205.05131*, 2022.
- [31] Unsplash, *The unsplash dataset*, 2024. [Online]. Available: <https://github.com/unsplash/datasets>.
- [32] W. Wang *et al.*, ‘‘Videofactory: Swap attention in spatiotemporal diffusions for text-to-video generation,’’ 2023.
- [33] Y. Wang *et al.*, ‘‘Lavie: High-quality video generation with cascaded latent diffusion models,’’ *International Journal of Computer Vision*, pp. 1–20, 2024.
- [34] H. Xu *et al.*, ‘‘Unifying flow, stereo and depth estimation,’’ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.
- [35] L. Xu, Y. Zhao, D. Zhou, Z. Lin, S. K. Ng, and J. Feng, ‘‘Pllava: Parameter-free llava extension from images to videos for video dense captioning,’’ *arXiv preprint arXiv:2404.16994*, 2024.
- [36] D. Yang *et al.*, *Vript: A video is worth thousands of words*, 2024. arXiv: [2406.06040 \[cs.CV\]](https://arxiv.org/abs/2406.06040).
- [37] L. Yu *et al.*, ‘‘Language model beats diffusion–tokenizer is key to visual generation,’’ *arXiv preprint arXiv:2310.05737*, 2023.
- [38] D. J. Zhang *et al.*, ‘‘Show-1: Marrying pixel and latent diffusion models for text-to-video generation,’’ *International Journal of Computer Vision*, pp. 1–15, 2024.