# OPEN-SORA 2.0: TRAINING A COMMERCIAL-LEVEL VIDEO GENERATION MODEL IN $200K

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Recent video diffusion models advance rapidly by scaling model size, data, and compute, but this trend leaves open a basic empirical question: *at a fixed compute budget, which design choices across the video-diffusion stack actually pay for themselves?* We address this question with **Open-Sora 2.0**, a commercial-level video generation model trained for only $200k — 5 – 10× lower than comparable systems such as MovieGen and Step-Video-T2V — that we treat as a deliberately budget-bound empirical apparatus rather than a recipe report. Within this regime we report a quantitative answer to three coupled questions about cost-bounded video diffusion training: (P1) *what makes a video latent space easy to denoise*, where we find that higher autoencoder channel dimensions slow diffusion convergence despite improving reconstruction, and that DINOv2 latent distillation accelerates training on high-compression autoencoders; (P2) *why image conditioning is the key efficiency lever*, where image-to-video (I2V) models adapt to new autoencoders far more efficiently than text-to-video (T2V) models, and motion learned at low resolution transfers cleanly to high resolution through I2V fine-tuning; and (P3) *how to combine image and text guidance at inference*, where decoupling classifier-free guidance with a time- and frame-dependent image-guidance schedule resolves the otherwise conflicting requirements of motion fidelity and semantic alignment. Each claim is paired with a controlled experiment, a natural ablation, or a statistical test: a controlled FLUX-isolation experiment on VBench, a 22-run training-log analysis treated as natural ablations, and a human evaluation reporting Fleiss' $\kappa$ and bootstrap 95% confidence intervals. Open-Sora 2.0 is comparable to leading systems including the open-source HunyuanVideo and the closed-source Runway Gen-3 Alpha. We fully open-source the model, training code, and data pipeline so every claim is reproducible.

## 1 INTRODUCTION

The past few years have witnessed an explosion of video generation models. Since the emergence of OpenAI's Sora (Brooks et al., 2024) in February 2024, a series of video generation models—either open-source (Zheng et al., 2024; Lin et al., 2024; Ma et al., 2025a; Kong et al., 2024; Hong et al., 2022) or proprietary (Polyak et al., 2024; RunwayML, 2025; Luma AI, 2025)—have appeared at an unprecedented pace, each striving to achieve "Sora-level" quality. While quality continues to improve, the field is converging on a common architecture and a single scaling lever: more parameters, more data, more compute. Following the success of scaling large language models (Hoffmann et al., 2022; Kaplan et al., 2020), researchers are now applying similar scaling principles (Kong et al., 2024) to video generation. This convergence raises a basic empirical question that is rarely studied in isolation: *at a fixed compute budget, which design choices across the video-diffusion stack actually pay for themselves?* Different choices about the autoencoder, the role of image conditioning, and the inference-time combination of guidance signals all consume the same training dollars; understanding which of them deliver the largest return is therefore not a recipe question but an empirical research question.

In this paper, we treat **Open-Sora 2.0** — a commercial-level video generation model trained for only $200k — as the apparatus for that empirical study. As shown in Figure 5, the training cost is 5–10× lower than comparable models such as MovieGen (Polyak et al., 2024) and Step-Video-T2V (Ma et al., 2025a). We deliberately commit to this binding budget rather than report it as a headline: a



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

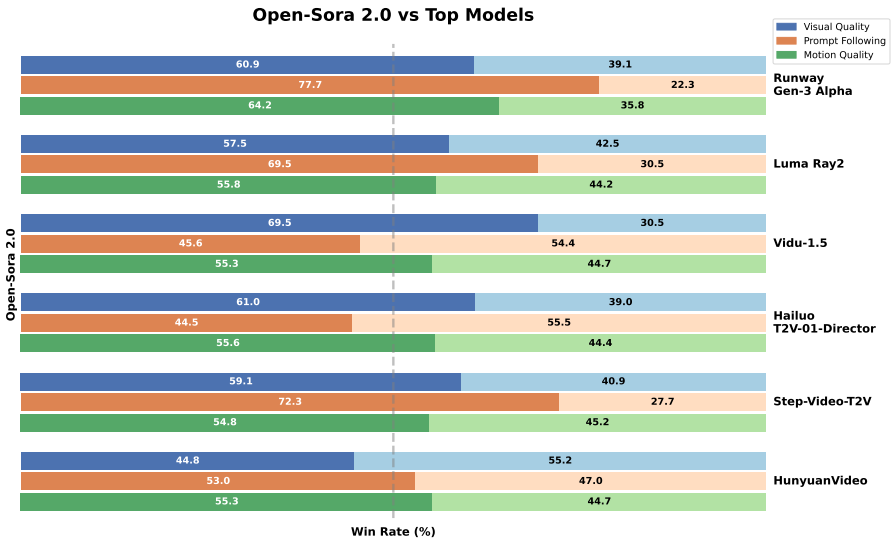



Figure 1: Human preference evaluation of Open-Sora 2.0 against other leading video generation models. Win rate is the percentage of comparisons where our model was preferred. The evaluation is conducted on 100 prompts spanning visual quality, prompt adherence, and motion quality. We report bootstrap 95% confidence intervals and Fleiss' $\kappa$=0.47 (moderate agreement) in Section 6 and Appendix O.

fixed compute envelope is precisely what makes the comparisons between design choices meaningful, because every dollar spent on one ingredient is a dollar that cannot be spent on another. Figure 1 shows that despite the constraint, Open-Sora 2.0 outperforms or matches leading commercial and open-source models in at least two of three human-evaluation criteria. To allow the community to verify, reproduce, and extend every claim in this paper, we release the model weights, training code, data pipeline, and evaluation harness in full.

**Contributions: three pillars of cost-bounded video diffusion.** We organize our primary contributions around three coupled research questions, each posed at the front of a section and each closed by quantitative evidence later in the paper:

**(P1) What makes a video latent space easy to denoise?** We study the autoencoder–diffusion interface using a Video Deep-Compression Autoencoder (Video DC-AE) — a 3D extension of image DC-AE — and report that *(iii)* higher autoencoder channel dimensions slow diffusion convergence despite improving reconstruction, and *(iv)* aligning intermediate Video DC-AE latents with DINOv2 (Oquab et al., 2023) features (latent distillation) accelerates downstream diffusion training on high-compression autoencoders (Section 4.2).

**(P2) Why is image conditioning the key efficiency lever?** I2V conditioning, not T2V, is what makes adaptation cheap: *(i)* I2V models adapt to a higher-compression autoencoder dramatically more efficiently than T2V models, and *(ii)* motion learned at 256 px transfers cleanly to 768 px through I2V fine-tuning, sidestepping the cost of direct high-resolution T2V training (Sections 4.2 and 4).

**(P3) How should image and text guidance be combined at inference?** Text and image classifier-free guidance impose conflicting requirements on motion fidelity and semantic alignment; *(v)* decoupling them with independent scales together with a time- and frame-dependent image-guidance schedule resolves the conflict (Section 5).

**Evidentiary standard.** A second, deliberate contribution is the standard of evidence to which we hold these claims. Each pillar is closed by at least one of: a controlled FLUX-isolation experiment with per-dimension VBench (Huang et al., 2024) breakdown, a 22-run training-log analysis treated as natural ablations across resolutions, learning rates, adaptation methods, and data mixtures, and a human evaluation reporting Fleiss' $\kappa$ and bootstrap 95% confidence intervals (Section 7). This

multi-pronged standard is what lets us read the findings as transferable insights rather than anecdotal observations.

**Reproducible artifacts and open release.** Alongside the scientific contributions, we deliver three concrete artifacts: the **Video DC-AE** architecture extending image DC-AE to 3D with $4\times32\times32$ spatio-temporal compression (Section 3.1); a complete \$200k training recipe with explicit GPU-hour and dollar accounting (Section 4, Appendix X); and a fully open-source release of model weights, training code, data pipeline, and evaluation harness. Together, the pillars and artifacts let Open-Sora 2.0 function both as a research result on cost-bounded video diffusion and as a reproducible foundation that the community can audit and extend.

## 2 DATA

We build a hierarchical pyramid of subsets matched to progressive training stages: loose filters keep large but moderate-quality subsets, while strict filters yield smaller, high-quality subsets used in the final fine-tuning stage. Raw videos are first preprocessed (broken-file removal, scene-cut segmentation via FFmpeg (FFmpeg Developers, 2023), re-encoding) and then scored by five complementary filters — aesthetic, motion, blur, OCR-based text overlay, and camera jitter — with per-stage thresholds (Appendix B, Appendix C). For low-resolution (256 px) data we caption with LLaVA-Video (Zhang et al., 2024); for high-resolution (768 px) we switch to Qwen 2.5 Max (Qwen Team, 2024), which produces fewer hallucinations and higher semantic consistency. We append the per-clip motion score to every caption so that motion strength becomes a controllable conditioning signal at inference (Section 5). The pipeline diagram, full statistics, and word-cloud analysis are in Appendix A and Appendix B.

## 3 MODEL ARCHITECTURE

The two key components of a video generation model are the autoencoder and the diffusion transformer. For the autoencoder, our main model uses HunyuanVideo's VAE; we additionally train a high-compression Video DC-AE for an inference-fast variant (Section 3.1). The diffusion transformer architecture is described in Section 3.2.

### 3.1 3D AUTOENCODER: VIDEO DC-AE

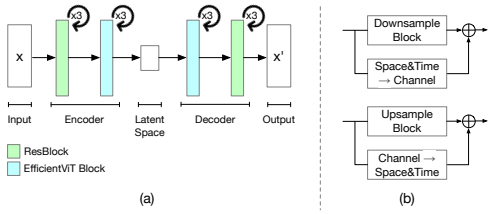



Figure 2: Architecture of Video DC-AE. (a) Encoder: each block introduces spatial downsampling, while temporal downsampling occurs at blocks 4 and 5. (b) Residual connection in the down/upsample blocks.

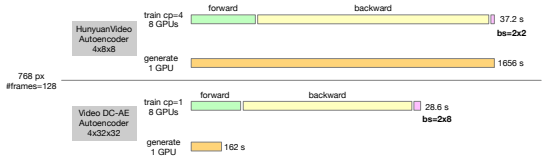



Figure 3: Training and inference speed at 768px using HunyuanVideo VAE ($4\times8\times8$, patch size 2) and Video DC-AE ($4\times32\times32$, patch size 1). Video DC-AE achieves significantly higher efficiency in both training and inference (50 steps).

We first leverage the open-source HunyuanVideo VAE (Kong et al., 2024). To further reduce both training and inference cost, we additionally develop a Video Deep-Compression Autoencoder (Video DC-AE) — named after (Chen et al., 2024) — that improves efficiency while maintaining high reconstruction fidelity.

HunyuanVideo VAE achieves a $4\times8\times8$ compression ratio. Even with a patch size of $2\times2$, the generation model still processes around 115K tokens per training video, which is computationally heavy. We identify redundancy in the spatial dimension and increase the spatial compression to 32 while keeping the temporal compression at 4. As shown in Figure 3, our Video DC-AE with $4\times32\times32$ compression and patch size $1\times1\times1$ trains 32-frame, 256px videos with a latent shape

of $8\times8\times8$, yielding a $5.2\times$ improvement in training throughput. At inference time, for a 5-second, 24 fps, 768px video, Video DC-AE reduces the token count from 76K (HunyuanVideo VAE) to 19K — a derivation appears in Appendix I — giving over $10\times$ inference speedup.

The original DC-AE (Chen et al., 2024) is optimized for images. We extend it to video by replacing 2D operations with 3D operations and introducing temporal compression in the last two encoder downsampling blocks (blocks 4 and 5) and temporal upsampling in the first two decoder upsampling blocks. We follow DC-AE in introducing residual blocks at the down/upsample stages (pixel-shuffling between spatial/temporal and channel dimensions, with channel-wise averaging on the downsample path and the inverse on the upsample path); without these residuals, gradient-propagation issues make high-compression AEs hard to train (Chen et al., 2024).

Table 1: Auto-encoder reconstruction performance comparison. The Video DC-AE highlighted with a yellow background is the variant we adapt for the fast generation model.

| Model | Down. (T×H×W) | Info. Down. | Channel | Causal | LPIPS↓ | PSNR↑ | SSIM↑ |
|---|---|---|---|---|---|---|---|
| Open-Sora 1.2 (Zheng et al., 2024) | 4×8×8 | 192 | 4 | ✓ | 0.161 | 27.504 | 0.756 |
| StepVideo VAE (Ma et al., 2025a) | 8×16×16 | 96 | 64 | ✓ | 0.082 | 28.719 | 0.818 |
| HunyuanVideo VAE (Kong et al., 2024) | 4×8×8 | 48 | 16 | ✓ | **0.046** | 30.240 | 0.856 |
| Video DC-AE | 4×32×32 | 96 | 128 | × | 0.051 | 30.538 | 0.863 |
| Video DC-AE | 4×32×32 | 48 | 256 | × | 0.049 | **30.777** | **0.872** |

We train Video DC-AE from scratch and evaluate its reconstruction quality (Table 1). It achieves competitive performance, with only a small LPIPS (Zhang et al., 2018) gap to HunyuanVideo VAE (0.051 vs. 0.046), while achieving slightly higher PSNR (30.538 vs. 30.240) and SSIM (0.863 vs. 0.856). Although the 256-channel variant attains the best reconstruction, we adopt the 128-channel variant for the fast generation model: as we show in Section 4.2, higher channel dimension slows downstream diffusion convergence (Finding **(iii)**).

### 3.2 DiT Architecture

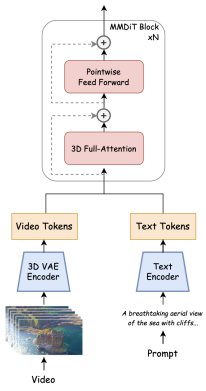



Figure 4: Open-Sora 2.0 diffusion transformer architecture: dual-stream blocks process text and video tokens separately, then single-stream blocks integrate them. 3D RoPE encodes spatial and temporal positions; T5-XXL and CLIP-Large are used as text encoders.

To capture long-range dependencies we use full attention, after patchifying the autoencoder latents. Sana (Xie et al., 2024) reports that smaller patch sizes give better training stability and finer details; with Video DC-AE's high compression we can use a patch size of 1 (no patch), whereas a patch size of 2 is still required when HunyuanVideo VAE is used.

Inspired by FLUX's MMDiT (Black Forest Labs, 2024), we use a hybrid transformer with both dual-stream and single-stream blocks. Dual-stream blocks process text and video features separately; single-stream blocks then integrate them for cross-modal interaction. We apply 3D RoPE (Su et al., 2024) to encode spatial and temporal positions. For text encoding we use T5-XXL (Chung et al., 2024) for complex semantics and CLIP-Large (Radford et al., 2021) for text–image alignment. The 11B-parameter architecture (Figure 4) and its hyperparameters are detailed in Appendix D.

## 4 Model Training

A commercial-grade video generation model typically requires more than 10 B parameters and $\mathcal{O}(100\text{M})$ training samples. To control cost, we use a three-stage pipeline (Table 2): (1) training a text-to-video (T2V) model on low-resolution video data, (2) training an image-to-video (I2V) model on low-resolution video data, and (3) fine-tuning the I2V model on high-resolution videos.

| Training Stage | Dataset | CP | #iters | #GPUs | #GPU day | USD |
|---|---|---|---|---|---|---|
| 256px T2V | 70M | 1 | 85k | 224 | 2240 | $107.5k |
| 256px T/I2V | 10M | 1 | 13k | 192 | 384 | $18.4k |
| 768px T/I2V | 5M | 4 | 13k | 192 | 1536 | $73.7k |
| Total | | | | | 4160 | $199.6k |

Table 2: Training configurations and cost breakdown for a single full training run, assuming an H200 rental price of $2 per GPU hour. The total marginal cost of one training run is $199.6k.

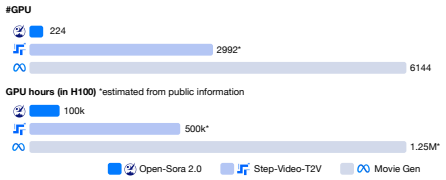



Figure 5: Training cost comparison of different video generation models. Values marked with $*$ are estimated from publicly available information. Open-Sora 2.0 is trained at 5–10$\times$ lower cost.

### 4.1 Efficient Training Strategy

Our budget-conscious training strategy emphasizes four ideas, each of which directly motivates one of the empirical findings of Section 7.

**(1) Leveraging open-source image models.** Prior work (Ma et al., 2024; Hong et al., 2022; Zheng et al., 2024) has shown that pretraining on images accelerates video model training. To avoid the high cost of training an image model from scratch, we initialize our T2V model from FLUX (Black Forest Labs, 2024), an open-source SOTA text-to-image model with 11 B parameters. We empirically find that, despite FLUX being a distilled model, this initialization is effective for further training. We isolate FLUX's contribution to final video quality in Section 7.1.

**(2) High-quality training data.** Inspired by PixArt (Chen et al., 2023), we hypothesize that high-quality video data substantially enhances training efficiency. We curate a high-quality subset for low-resolution training and impose stricter selection criteria for high-resolution fine-tuning. Section 7.2 provides quantitative evidence from a 22-run convergence analysis that data curation reduces terminal training loss.

**(3) Learning motion at low resolution.** Training a commercial video generation model is computationally expensive at high resolutions: training a 129-frame, 768px video is roughly 40$\times$ slower than the 256px equivalent (Tables 6, 7 in Appendix G), due to the quadratic complexity of self-attention in the token count. We therefore allocate the majority of compute to low-resolution training, where the model learns diverse motion patterns efficiently, and only briefly fine-tune at high resolution.

**(4) I2V models facilitate resolution adaptation.** We find that adapting from 256px to 768px is significantly more efficient with an I2V model than a T2V model: conditioning on a static image lets the model focus on motion generation, which is well-learned at low resolution. Stage 3 therefore prioritizes high-resolution I2V fine-tuning. At inference, we first generate an image from text and then synthesize a video conditioned on both the image and the text.

With these four choices we constrain a single one-time training run to $200k (Table 2, Figure 5). We discuss what *is not* included in this figure (and what the total R&D cost looks like) in Appendix X.

### 4.2 High-Compression Autoencoder Adaptation

High-compression Video DC-AE reduces the number of tokens, but introduces two difficulties. (i) Training a deep compression network is hard in itself; Chen et al. (2024) identifies gradient-propagation issues in down/upsample blocks. (ii) Higher-channel AEs slow diffusion convergence: prior work (Yao & Wang, 2025; Xie et al., 2024) reports that for image diffusion, AEs with better reconstruction but higher latent dimensionality often *degrade* generation quality, and Theorem A.7 in Chen et al. (2025) suggests (under strong assumptions) that a $k\times$ increase in channel size requires $k^5\times$ more data; conversely, distilling AE latents against a pretrained image foundation model accelerates diffusion training (Yao & Wang, 2025).

We address these difficulties with three strategies, two of which are themselves reported findings: **(a) latent-space distillation [Finding (iv)]:** after training Video DC-AE, we apply a distillation loss

to align its third-layer latents with DINOv2 (Oquab et al., 2023) features; **(b) efficient diffusion adaptation:** following PixArt (Chen et al., 2023), we re-initialize only the input/output layers when adapting the DiT to a new autoencoder, so semantic structures adapt quickly while initial outputs are merely blurry; **(c) I2V-first adaptation [Finding (i)]:** we observe that I2V models adapt to high-compression AEs significantly faster than T2V models, and therefore focus high-compression-AE adaptation on the I2V model. We additionally re-use the tiling code of Kong et al. (2024) for high-resolution encoding (Appendix F). We train on short videos (up to 33 frames) for 17k iterations on 20M samples across 160 GPUs, then extend to long videos (up to 128 frames) for 8k iterations on 2M samples. The final model reaches a loss of 0.5 (vs. 0.1 for the original AE); although not fully converged under our compute budget, it captures spatio-temporal relationships and is significantly faster (Figure 12 in Appendix H).

### 4.3 Training Setting and System Optimization

We use flow matching (Lipman et al., 2022) (similar to Stable Diffusion 3 (Esser et al., 2024)) with the velocity prediction $f_\theta(X_t, t, y) \to X_0 - X_1$ where $X_t = (1-t)X_0 + tX_1$, AdamW (Loshchilov & Hutter, 2017), and gradient-norm clipping at 1. Multi-bucket training (Zheng et al., 2024) handles varying frame count, resolution, and aspect ratio in the same batch (per-bucket batch sizes in Appendix G). We train using ColossalAI (Li et al., 2023) on H200 GPUs (141 GB) with PyTorch compile (Ansel et al., 2024) and Triton kernels (OpenAI, 2019), combining ZeRO-2 (Rajbhandari et al., 2020) with context parallelism (Liu et al., 2023) for MMDiT (Stages 1–2: ZeRO-2 alone, MFU $38.19\%$; Stage 3: ZeRO-2 + CP$= 4$, MFU $35.75\%$); for the autoencoder we use channel-dimension tensor parallelism (Shoeybi et al., 2020). Additional optimizations (timestep rescaling, selective activation checkpointing, auto-recovery $> 99\%$ GPU utilization, pinned-memory dataloaders, asynchronous checkpointing) and the Video DC-AE training schedule are in Appendix E and Appendix M.

## 5 Conditioning

### 5.1 Image-to-Video Training

Open-Sora 1.2 introduces a universal conditioning framework that allows both image and video conditions at any frame, but its original method replaces noisy inputs with the condition, which leads to inconsistent timesteps across inputs. We modify the framework by *concatenating* the condition as additional channels, leaving the velocity prediction task unchanged: image or video conditions are encoded by the autoencoder and concatenated with the original video latent, and an extra channel marks the task type, raising the channel count from $k$ to $2k+1$ (Figure 13 in Appendix J). To improve generalization we apply image-condition dropout at $12.5\%$ analogous to text-condition dropout; when dropped, the problem reduces to T2V with zero tensors concatenated to the video latent. Finally, the per-clip motion score (Section 2) is appended to every caption as an additional, controllable conditioning signal, giving independent control over motion strength at inference (full discussion and qualitative effect in Appendix K).

### 5.2 Image-to-Video Inference

We use classifier-free guidance (Ho & Salimans, 2022) for inference. As our model is conditioned on both image and text, a naive approach uses a single guidance scale:

$$v_t = v_\theta(x_t, t, \varnothing, \varnothing) + g \cdot \big(v_\theta(x_t, t, \mathit{txt}, \mathit{img}) - v_\theta(x_t, t, \varnothing, \varnothing)\big).$$

We find this suboptimal: image guidance requires only a small scale (large values make the entire video static), whereas text guidance benefits from a higher scale (better semantic alignment). **We therefore decouple the two terms** (Finding **(v)**):

$$\begin{aligned} v_t = {} & v_\theta(x_t, t, \varnothing, \varnothing) \\ & + g_{\mathit{img}} \cdot \big(v_\theta(x_t, t, \varnothing, \mathit{img}) - v_\theta(x_t, t, \varnothing, \varnothing)\big) \\ & + g_{\mathit{txt}} \cdot \big(v_\theta(x_t, t, \mathit{txt}, \mathit{img}) - v_\theta(x_t, t, \varnothing, \mathit{img})\big). \end{aligned}$$

A residual issue is that high image guidance occasionally introduces flickering. We mitigate it with a guidance-oscillation scheme inspired by Ho et al. (2022): with 50-step sampling, after the first

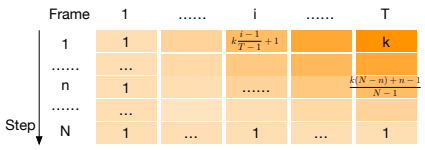



Figure 6: Heatmap of the dynamic image-guidance scale across denoising steps and latent frames. Darker regions indicate higher image guidance, emphasizing later frames and earlier denoising steps.

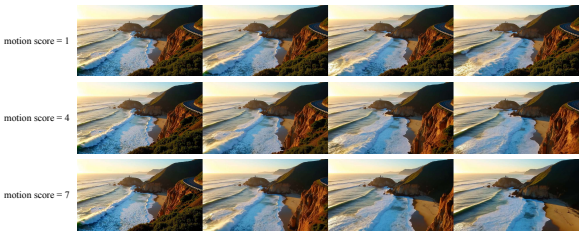



Figure 7: Effect of motion score on video generation. As the motion score increases, the camera and scene dynamics become more pronounced. Full discussion in Appendix K.

10 steps we alternate $g_{img}$ between its original value (odd steps) and 1 (even steps). We further introduce a *dynamic* image-guidance schedule that scales $g_{img}$ jointly by frame index and denoising step, since (i) frames late in the video need more anchoring to the conditioning frame, and (ii) image guidance becomes less critical at later denoising steps. Figure 6 shows that linear scaling along both axes works best; we use $g_{img} = 3$, $g_{txt} = 7.5$. We additionally explore an inference-time scaling strategy (Ma et al., 2025b) that injects controlled noise variations at selected denoising steps and selects high-VBench candidates; details and qualitative results are in Appendix L.

## 6 Performance

Our model supports both T2V and I2V generation at $256\times256$ and $768\times768$ resolutions, generating videos up to 128 frames (5 seconds at 24 fps). Since our model is optimized for I2V generation, the default text-to-image-to-video (T2I2V) pipeline first generates an image with the FLUX model, and then synthesizes a video conditioned on the image and the text. Generation samples are shown in Figure 16 in Appendix N.

**Human evaluation.** To benchmark Open-Sora 2.0, we generate videos using a fixed set of 100 text prompts spanning multi-object scenes, human actions, landscapes, cinematic shots, abstract content, fine-grained spatial relationships, and complex motion (Appendix O details the prompt categorization). We also generate videos with several closed-source APIs and open-source models. To ensure fairness, only a single inference is performed per model (no cherry-picking). For all models, we use their default settings, with video lengths between 5 and 6 seconds and resolutions between 768px and 720p as their generation constraints allow.

A blinded evaluation was conducted by 10 professional evaluators, assessing each pair of videos on three criteria: visual quality (visual fidelity and aesthetics), prompt adherence (accuracy with respect to the text prompt), and motion quality (consistency and adherence to physical laws). Per-prompt outcomes are determined by majority vote across evaluators.

The win rates are summarized in Figure 1 (Section 1). Inter-annotator agreement (Fleiss' $\kappa$) is $0.47$ for Visual Quality, $0.52$ for Prompt Adherence, and $0.43$ for Motion Quality (overall $0.47$): moderate, and within the $0.35$–$0.55$ range reported for similar video evaluations (Huang et al., 2024). Bootstrap 95% confidence intervals (10,000 resamples, percentile method) for all 18 (baseline, criterion) cells are tabulated in Appendix P; the full evaluation protocol is in Appendix O. The appropriate reading is *comparability* where CIs overlap 50% (e.g., vs. HunyuanVideo on Visual: $55.3\%$ $[45.6\%, 64.8\%]$) and *statistically significant superiority* where CIs exclude 50% (e.g., vs. Runway Gen-3 Alpha on Prompt Adherence: $77.7\%$ $[68.5\%, 85.2\%]$). We deliberately match these readings to the comparability claim of Figure 1 rather than overclaim superiority where the data does not support it.

**Automatic evaluation: VBench.** Figure 8 reports overall VBench (Huang et al., 2024) scores. Open-Sora 2.0 narrows the gap to OpenAI Sora from $4.52$ points (Open-Sora 1.2) to $0.69$ points and surpasses both CogVideoX1.5-5B and HunyuanVideo on Total Score. The per-dimension breakdown in Section 7.1 (Table 11) further shows near-parity with Sora on motion-related dimensions and the highest Semantic Score across all compared systems. As a complementary metric we additionally report zero-shot FVD on UCF-101 in Appendix S (Table 12); we report it as complementary, not

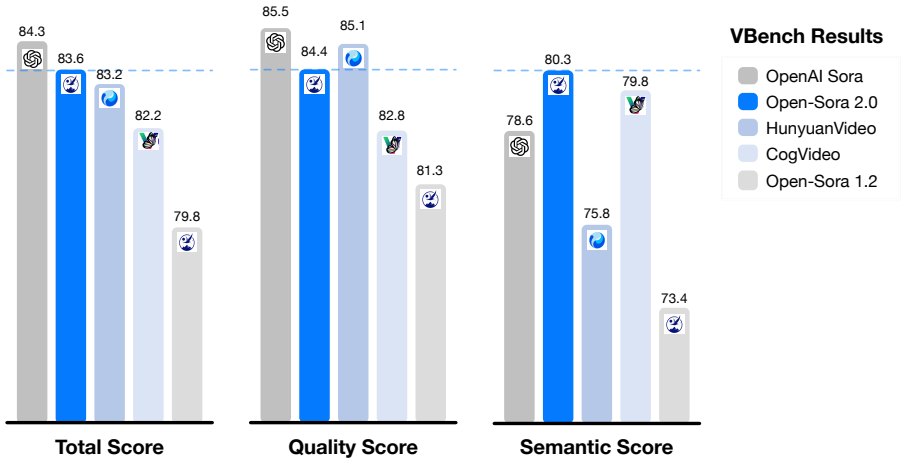



Figure 8: VBench (Huang et al., 2024) score comparison. Open-Sora 2.0 achieves a higher VBench Total Score than HunyuanVideo and CogVideoX1.5-5B, and narrows the gap to OpenAI Sora to within 0.69 points. Per-dimension scores appear in Table 11.

primary, because UCF-101's narrow distribution and known weak human correlation (Stein et al., 2023) make VBench our primary automatic benchmark.

**Baseline selection.** Different evaluations cite different baseline subsets due to availability constraints, not selective reporting: human evaluation requires being able to generate videos for arbitrary prompts (open weights or live API access); the AE comparison (Table 1) requires AEs with public weights; the VBench comparison (Table 11 and Figure 8) uses only models whose VBench scores are publicly reported on the leaderboard, so the methodology is comparable. We acknowledge that the most recent generation of models (Kling, Wan series, etc.) is not included; we discuss this gap and its implications in Appendix T.

## 7 Validating the Three Pillars

We pair every claim in pillars (P1)–(P3) with at least one of a *controlled experiment*, a *natural ablation*, or a *statistical test*. The findings (i)–(v) are stated and used in Sections 4.2, 4, and 5; the human-evaluation arm (Fleiss' $\kappa$, bootstrap 95% CIs) is in Section 6; zero-shot UCF-101 FVD is in Appendix S.

### 7.1 Controlled experiment: isolating the contribution of FLUX

A natural concern is that the cost reduction is driven by the FLUX initialization rather than the video model. We isolate FLUX's contribution by evaluating Open-Sora 2.0 in *direct T2V mode* — dropping the image condition via the 12.5% dropout mechanism (Section 5.1) — which ablates only the inference pipeline on the *same* model and requires no retraining.

Three patterns emerge from Table 3. Motion dimensions barely move (Motion Smoothness $\Delta = -0.4$, Temporal Flickering $\Delta = -0.3$), confirming the video model learns motion *independently* of FLUX; the drop concentrates in Aesthetic and Imaging Quality ($-4.8$, $-4.7$), the first-frame fidelity that FLUX provides; Dynamic Degree even *increases* ($+5.3$). Net effect: FLUX contributes $\sim$2.5 VBench points concentrated in visual quality, while motion is learned by our \$200k recipe; in T2V-only mode (81.8) we remain competitive with HunyuanVideo (82.2), a dedicated T2V model.

**Per-dimension VBench breakdown.** The full per-dimension comparison (Table 11, Appendix R) shows three further patterns: motion dimensions reach near-parity with OpenAI Sora; Dynamic Degree (58.5) surpasses HunyuanVideo (52.8) and CogVideoX-1.5 (46.3), confirming the motion-score conditioning of Section 5.1; and Open-Sora 2.0 attains the highest *Semantic Score* (80.3) across all systems including OpenAI Sora (79.8), which we attribute to the T2I2V pipeline.

Table 3: T2V-only VBench evaluation isolating the FLUX initialization. The drop concentrates in visual quality; motion dimensions are essentially unchanged.

| VBench | T2I2V | T2V-only | $\Delta$ | HunyuanVideo |
|---|---|---|---|---|
| Total | 84.3 | 81.8 | $-2.5$ | 82.2 |
| Quality | 85.5 | 82.9 | $-2.6$ | 82.8 |
| Semantic | 83.2 | 80.5 | $-2.7$ | 80.3 |
| Motion Smoothness | 99.1 | 98.7 | $-0.4$ | – |
| Temporal Flickering | 99.3 | 99.0 | $-0.3$ | – |
| Dynamic Degree | 58.5 | 63.8 | $+5.3$ | – |
| Aesthetic Quality | 62.4 | 57.6 | $-4.8$ | – |
| Imaging Quality | 67.8 | 63.1 | $-4.7$ | – |

Table 4: 22 I2V exploration runs grouped by adaptation method (columns) and data/loss (rows); each cell reports the number of runs and terminal-loss range over the last 1k iterations. Per-run logs in Appendix Q.

| Data / Loss | Full fine-tuning | LoRA |
|---|---|---|
| baseline data, L2 | 7 runs: $[0.069, 0.078]$ | 2 runs: $[0.042, 0.046]$ |
| curated data, time-w. L2 | 4 runs: $[0.052, 0.058]$ | **6 runs:** $[0.038, 0.043]$ |

### 7.2 Natural ablation: 22-run convergence analysis

A clean controlled ablation grid for a \$200k run would cost more than \$1M — directly contradicting the cost-efficiency claim. As a substitute we mine the logs of 22 I2V exploration runs spanning resolutions (480p/720p), frame counts (33/81), learning rates ($5\times10^{-6}$–$5\times10^{-5}$), adaptation methods (LoRA vs. full FT), losses (L2 vs. time-weighted), and data mixtures (curated vs. baseline), treating them as *natural experiments*.

Reading Table 4: down a column isolates the LoRA-vs.-full-FT effect (3–4$\times$ lower terminal loss); across a row isolates the data-and-loss effect ($\sim$0.02–0.03 reduction); the bottom-right cell stacks all three interventions for the lowest terminal loss. All 22 runs converge to terminal losses in $[0.038, 0.078]$ regardless of resolution or learning rate; 480p/81-frame and 720p/33-frame runs reach similar values, providing quantitative support for Finding (ii) (low-resolution motion learning transfers to high resolution). A separate longer-horizon run ($\sim$49k iterations) showed continued stable convergence. The consistency across 22 varied configurations is what makes the observations generalizable rather than configuration-specific.

## 8 Conclusion

Open-Sora 2.0 — trained for \$200k, 5–10$\times$ less than comparable systems — serves as our empirical apparatus for cost-bounded video diffusion. The three pillars — (P1) higher-channel autoencoders slow diffusion convergence, with DINOv2 latent distillation as a remedy; (P2) image conditioning, not text, is the cheap-adaptation lever for both autoencoder transitions and high-resolution motion transfer; (P3) decoupled, time- and frame-dependent classifier-free guidance reconciles motion fidelity with semantic alignment — close with a controlled FLUX-isolation experiment, a 22-run natural-ablation analysis (Section 7), and a statistically rigorous human evaluation (Section 6). We additionally release the Video DC-AE architecture, the full \$200k recipe with explicit accounting, and the model weights, training code, and data pipeline.

**Future work.** Deep-compression video VAEs remain under-explored: closing the reconstruction gap of our high-compression model is a key direction. Object distortion and unnatural physics persist (L1–L4); explicit physics priors and longer-horizon temporal consistency would help. A larger human study and a community-scale ablation grid built on top of our open release are the most direct ways to sharpen the pillar-level claims further.

### AI USE STATEMENT

In this work, generative AI systems are used in two capacities. First, as declared components of the method itself: LLaVA-Video (Zhang et al., 2024) generates 256px video captions and Qwen 2.5 Max (Qwen Team, 2024) refines 768px captions (Section 2, Appendix B); FLUX (Black Forest Labs, 2024) serves as the text-to-image stage of the T2I2V pipeline (Section 4.1); and T5-XXL (Chung et al., 2024) and CLIP-Large (Radford et al., 2021) act as text encoders (Section 3.2). Second, we used generative AI tools (large language models) for light editing and polishing of the manuscript text and as coding assistants for non-core utility and analysis scripts. We did not use generative AI tools for research ideation, for designing the method or experiments, or for producing, analyzing, or interpreting experimental results; those uses are not applicable to this work. All AI-assisted text was reviewed and revised by the authors, and all AI-assisted code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

Open-Sora 2.0 is a high-capability generative video model, and we release its weights, code, and data pipeline openly. The principal risks are misuse for deepfakes, misinformation, and non-consensual synthetic media, alongside copyright and environmental concerns; the principal benefits are the democratization of video-generation research through a reproducible $200k recipe. We mitigate the risks with four safeguard layers: (i) data-handling filters (NSFW classification, face/PII filtering, OCR-based filtering, and deduplication) in the training pipeline; (ii) a responsible-use license attached to the release; (iii) recommended invisible watermarking and content-moderation classifiers for downstream deployers; and (iv) a community reporting channel for problematic content. Human evaluators were compensated at standard market rates. A full discussion of positive and negative societal impacts appears in Appendix V, and an expanded ethics analysis, including a risk–severity–likelihood matrix and environmental-impact accounting, appears in Appendix W.

### REPRODUCIBILITY STATEMENT

We designed this work to be reproducible end to end. The model architecture and all hyperparameters are specified in Section 3.2 and Appendix D; the complete training schedule, hardware configuration, and system optimizations are given in Section 4 and Appendix E; and the data-filtering and annotation pipeline is documented in Section 2 and Appendices B and C. The full cost accounting behind the $200k figure, including what it does and does not contain, is itemized in Appendix X and Appendix Y. Evaluation is specified by the human-evaluation protocol of Appendix O (with bootstrap confidence intervals in Appendix P) and the FVD protocol of Appendix S. The model weights, training code, data pipeline, and evaluation harness will be fully open-sourced upon publication, so that every claim in the paper can be independently re-run and audited.

## REFERENCES

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL `https://pytorch.org/assets/pytorch2-2.pdf`.

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

G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

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.

Brandon Castellano. Pyscenedetect: Video scene cut detection and analysis tool, 2023. URL https://github.com/Breakthrough/PySceneDetect.

Hao Chen, Yujin Han, Fangyi Chen, Xiang Li, Yidong Wang, Jindong Wang, Ze Wang, Zicheng Liu, Difan Zou, and Bhiksha Raj. Masked autoencoders are effective tokenizers for diffusion models. *arXiv preprint arXiv:2502.03444*, 2025.

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

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. *arXiv preprint arXiv:2410.10733*, 2024.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

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

FFmpeg Developers. Ffmpeg, 2023. URL https://ffmpeg.org/.

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

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models, 2022.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

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

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.

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.

Shenggui Li, Hongxin Liu, Zhengda Bian, Jiarui Fang, Haichen Huang, Yuliang Liu, Boxiang Wang, and Yang You. Colossal-ai: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*, ICPP '23, pp. 766–775, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400708435. doi: 10.1145/3605573.3605613. URL https://doi.org/10.1145/3605573.3605613.

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

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

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context, 2023. URL https://arxiv.org/abs/2310.01889.

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

Luma AI. Dream machine, 2025. URL https://lumalabs.ai/dream-machine.

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

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps, 2025b. URL https://arxiv.org/abs/2501.09732.

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.

OpenAI. Triton: An open-source programming language for writing highly efficient gpu code, 2019. URL https://github.com/triton-lang/triton. Accessed: 2025-03-12.

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

PaddlePaddle Authors. Paddleocr: A practical ultra lightweight ocr system. https://github.com/PaddlePaddle/PaddleOCR, 2020.

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

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, A. Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

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

RunwayML. Introducing Gen-3 Alpha, 2025. URL https://runwayml.com/research/introducing-gen-3-alpha.

Christoph Schuhmann. Improved aesthetic predictor, 2021. URL https://github.com/christophschuhmann/improved-aesthetic-predictor.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

George Stein, Jesse C. Cresswell, Rasa Hosseinzadeh, Yi Sui, Brendan Leigh Ross, Valentin Villecroze, Zhaoyan Liu, Anthony L. Caterini, J. Eric T. Taylor, and Gabriel Loaiza-Ganem. Exposing flaws of generative model evaluation metrics and their unfair treatment of diffusion models, 2023. URL https://arxiv.org/abs/2306.04675.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint arXiv:2410.10629*, 2024.

Jingfeng Yao and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. *arXiv preprint arXiv:2501.01423*, 2025.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

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

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

## A Data Statistics

We conduct a statistical analysis of key attributes of our video data: aesthetic scores, duration (seconds), aspect ratios (height/width), and caption length. We also visualize common words in video captions using a word cloud.

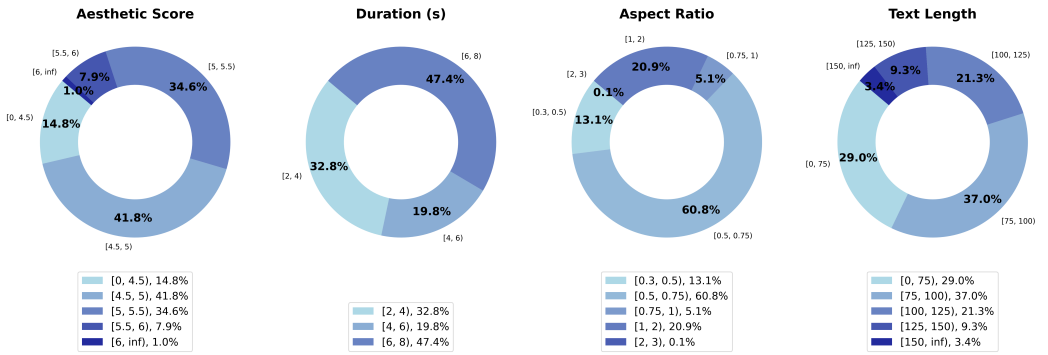



Figure 9: Distribution of key attributes of the whole video dataset.

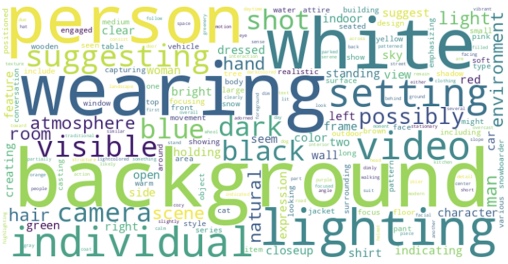



Figure 10: Word cloud of the video captions.

As shown in Figure 9, the majority of videos have aesthetic scores in the range $[4.5, 5.5]$. Video durations range from 2 to 8 seconds, with nearly half of the dataset between 6 and 8 seconds, providing rich temporal information for learning dynamic patterns. Aspect ratios are concentrated between 0.5 and 0.75 (the 16:9 format) but cover a diverse range, supporting generalization. Over 70% of captions contain more than 75 words.

Figure 10 presents a word cloud of caption vocabulary. Captions cover not only main subjects (*"person"*) and their actions (*"wearing"*) but also background elements (*"background"*, *"setting"*, *"atmosphere"*) and lighting (*"lighting"*). This aligns with the six aspects of our VLM prompting (Appendix B).

## B Data Filtering Pipeline and Annotation

**Preprocessing.** We discard broken files and videos with outlying attributes (duration $< 2$ s, bpp $< 0.02$, fps $< 16$, aspect ratio outside $[1/3, 3]$, or H.264 "Constrained Baseline" profile). We segment raw videos into continuous clips using libavfilter scene scores from FFmpeg (FFmpeg Developers, 2023), then re-encode under H.264 with fps $< 30$, longer dimension $\leq 1080$ px, and remove black borders. Shots longer than 8 s are split into 8 s windows and shots shorter than 2 s are dropped.

**Score filtering.** We apply five complementary score-based filters: aesthetic score, motion score, blur detection, OCR-based text overlay detection, and camera-jitter detection. Each filter assigns a per-clip score; we tune thresholds per training stage so that the final 768 px fine-tuning subset is the strictest. Definitions and exact thresholds are in Appendix C.

**Caption annotation.** For low-resolution (256 px) data we caption with the open-source vision–language model LLaVA-Video (Zhang et al., 2024), prompting it to cover six aspects: main subjects, subject actions, background and environment, lighting/atmosphere, camera movement, and video

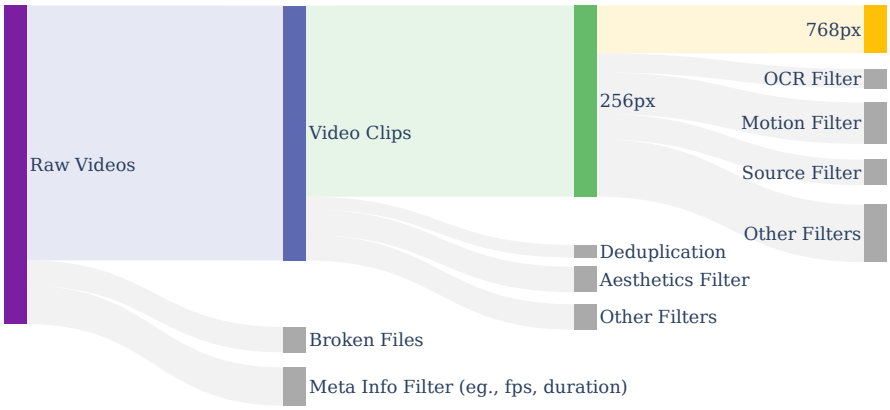



Figure 11: Hierarchical data filtering pipeline. Raw videos are first transformed into trainable clips, then progressively filtered through a bag of complementary score filters to construct subsets for each training stage.

style. For high-resolution (768 px) training data we instead use the proprietary Qwen 2.5 Max (Qwen Team, 2024), which we find produces fewer hallucinations and higher semantic consistency. For both training and inference we append the motion score after the caption (Appendix K) so that motion strength becomes a controllable conditioning signal.

# C Data Filters

## C.1 Aesthetic Score

We assess aesthetic quality using the CLIP+MLP aesthetic predictor (Schuhmann, 2021). For a video, we extract the first, middle, and last frame, score each, and average.

## C.2 Motion Score

We measure motion intensity using the VMAF motion score from FFmpeg's libavfilter (FFmpeg Developers, 2023). Videos with extremely low or excessively high motion are filtered to maintain a balanced motion range.

## C.3 Blur Detection

We evaluate clarity using the Laplacian operator from OpenCV (Bradski, 2000). A sample is considered blurry if the variance of the Laplacian falls below a threshold. For videos, we extract five uniformly spaced frames and use majority voting.

## C.4 OCR

We detect text bounding boxes using PaddleOCR (PaddlePaddle Authors, 2020) and compute the total area of bounding boxes with confidence above 0.7. Videos with excessive text are removed.

## C.5 Camera Jitter Detection

We detect camera jitter using shot boundary detection from PySceneDetect (Castellano, 2023). A video is flagged if the average frame-to-frame change exceeds a threshold.

Table 5: Architecture hyperparameters for the Open-Sora 2.0 11B-parameter video generation model.

| Double-Stream Layers | Single-Stream Layers | Model Dimension | FFN Dimension | Attention Heads | Patch Size |
|---|---|---|---|---|---|
| 19 | 38 | 3072 | 12288 | 24 | 2 |

## D Model Architecture Hyperparameters

## E Training-Setting Details

Our training setup is largely based on Open-Sora 1.2 (Zheng et al., 2024). We use flow matching (Lipman et al., 2022) as the training objective with AdamW (Loshchilov & Hutter, 2017), $\beta = (0.9, 0.999)$, $\epsilon{=}1\times10^{-15}$, no weight decay. The learning rate is $5\times10^{-5}$ for the first 40k steps of Stages 1 and 2, decaying to $3\times10^{-5}$ for the final 45k steps; Stage 3 uses $1\times10^{-5}$. Gradient norms are clipped at 1.

**Training objective.** With video latent $X_0$ and Gaussian noise $X_1 \sim \mathcal{N}(0, I)$, the model $f_\theta$ takes as input the interpolated latent $X_t = (1-t)X_0 + tX_1$ and predicts the velocity $X_0 - X_1$:

$$\mathcal{L} = \mathbb{E}_{t,X_0,X_1}\left[\left\| f_\theta(X_t, t, y) - (X_0 - X_1)\right\|\right],$$

where $y$ is the text and/or image conditioning. The timestep $t$ is sampled from a logit-normal distribution and rescaled by the latent shape: higher-resolution and longer-duration videos are more susceptible to noise, so we apply $t \leftarrow \alpha t/(1+(\alpha-1)t)$, with $\alpha \propto T\times H\times W$.

## F High-Compression-AE Adaptation: Tiling and Engineering Details

**Tiling for high-resolution encoding.** High-compression AEs trained at low resolution degrade at high resolution (Chen et al., 2024). Rather than re-fine-tuning, we re-use the tiling code of Kong et al. (2024) to save resources: the input video is split into spatially overlapping tiles, each tile is encoded/decoded independently, and the overlapping regions are blended at decoder output.

## G Multi-Bucket Training

For multi-bucket training, we adapt different batch sizes to different training configurations (parallelism strategy, video resolution, frame count, and maximum frames). Tables 6 and 7 list the per-configuration batch sizes.

Table 6: Batch size and throughput in Stage 1 and Stage 2.

| Resolution | # Frames | Max # of Frames | Batch Size | Throughput on 8 GPUs |
|---|---|---|---|---|
| 256px | 5 to 33 | 2304 | 12 | 12.7 videos/s |
| | 37 to 65 | 4352 | 6 | 6.3 videos/s |
| | 69 to 97 | 6400 | 4 | 4.2 videos/s |
| | 101 to 129 | 8448 | 3 | 3.2 videos/s |
| 256px | 1 | 256 | 45 | 47.6 images/s |
| 768px | 1 | 2304 | 13 | 13.8 images/s |
| 1024px | 1 | 4096 | 7 | 7.4 images/s |

To determine the batch sizes we conducted a constrained search on H200 GPUs:

1. Identify the maximum batch size for the highest-token configuration that does not OOM.
2. For all other configurations, choose the largest batch size that fits in memory while keeping training time below the highest-token configuration.
3. Constrain the combined execution time of AE encoding + forward pass + backward pass not to exceed the reference batch size's execution time, preventing inefficiencies due to synchronous backward operations.

Table 7: Batch size and throughput in Stage 3 with context parallelism CP= 4.

| Resolution | # Frames | Max # of Frames | Batch Size | Throughput on 8 GPUs |
|---|---|---|---|---|
| 768px | 5 to 33 | 20736 | 6 | 0.25 videos/s |
| | 37 to 65 | 39168 | 4 | 0.17 videos/s |
| | 69 to 97 | 57600 | 3 | 0.13 videos/s |
| | 101 to 129 | 76032 | 2 | 0.08 videos/s |
| 768px | 1 | 2304 | 38 | 1.60 images/s |

## H Comparison of Video Generation with Different AE Compression Ratios

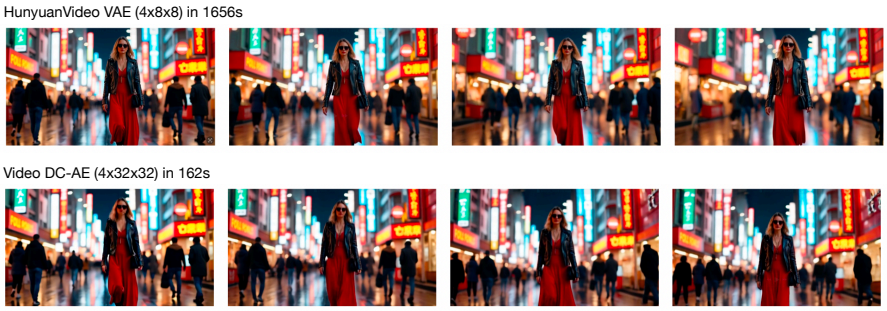



Figure 12: Comparison of video generation with different autoencoder compression ratios. Top row: lower compression. Bottom row: higher compression. The higher-compression AE produces slightly blurrier outputs but significantly faster generation.

## I Derivation of Token Count Reduction

A high-compression video autoencoder reduces the number of tokens needed for generation, while increasing the number of latent channels incurs minimal extra cost (only the input/output layers). Define the token downsample ratio

$$D_{\text{token}} = D_T \times D_H \times D_W \times P_T \times P_H \times P_W,$$

where $D_{T,H,W}$ are the AE compression ratios and $P_{T,H,W}$ are the generation model's patch sizes. For a 5-second, 24 fps, 768px video, HunyuanVideo VAE ($4\times8\times8$) with patch size $1\times2\times2$ gives $D_{\text{token}}$=1024 and $\sim$76K tokens; Video DC-AE ($4\times32\times32$) with patch size $1\times1\times1$ gives $D_{\text{token}}$=4096 and $\sim$19K tokens.

## J Image-to-Video Conditioning Framework

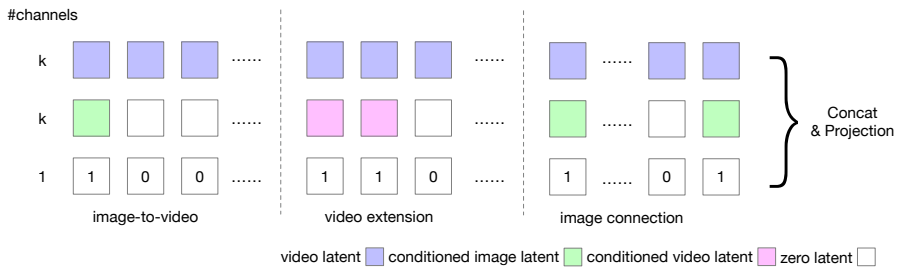



Figure 13: Image and video conditioning framework. Conditioning information is injected through additional channels; a mask channel distinguishes input types. The same framework supports a wide range of I2V and video-to-video (V2V) tasks.

Figure 13 illustrates the conditioning framework described in Section 5.1. Image or video conditions are encoded by the autoencoder and concatenated with the original video latent along the channel dimension, with one extra binary channel marking task type. This unified channel-concatenation design supports image-to-video (I2V), video-to-video (V2V), and arbitrary-frame conditioning at training time, while leaving the velocity-prediction objective unchanged.

## K Motion-Score Conditioning

Controlling motion strength is a useful feature in video generation. Classifier-free guidance scaling and step-shifting can change motion intensity, but they are entangled with visual quality and text alignment. We instead model motion dynamics as an explicit, separate, controllable parameter: the motion score from data preprocessing (Appendix C), which quantifies dynamics, is appended to the caption as an additional conditioning signal. At inference, adjusting the motion score gives effective and independent control over the dynamics of the generated video, as shown in Figure 7 (Section 5.2). Empirically, low motion scores produce calmer, more cinematographic shots, while high motion scores yield pronounced camera motion and scene dynamics; we recommend matching the motion score at inference to the degree of action implied by the text prompt.

## L Inference-Time Scaling

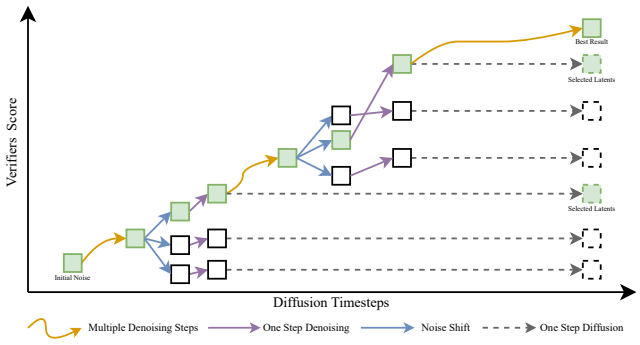



Figure 14: Diffusion inference-scaling framework. At selected denoising steps, controlled noise injections produce multiple candidate outputs. Each candidate is evaluated using VBench metrics, and the highest-scoring variant is chosen to continue the generation. This enables quality-aware inference without modifying the model.

We explore an inference-time scaling strategy inspired by Ma et al. (2025b) to enhance video generation quality and diversity without modifying the model. As illustrated in Figure 14, the approach injects controlled noise variations during early denoising steps (e.g., steps 1 and 3). At each selected step, partial denoising produces multiple candidate outputs, each corresponding to a different noise variant.

Candidates are evaluated using six VBench metrics (Huang et al., 2024): *subject_consistency, background_consistency, motion_smoothness, dynamic_degree, aesthetic_quality, imaging_quality*. Users can prioritize specific metrics depending on the application. The highest-scoring candidate is selected to continue, allowing the model to explore a richer generation space aligned with target attributes.

The computational overhead depends on four factors: frequency of noise injection, number of initial seeds, number of variations per step, and number of active metric verifiers. In our experiments, we fix the verifier set to isolate the effects of the other three.

As shown in Figure 15, this method significantly improves generation quality under challenging prompts: the baseline (top row) exhibits static or unnatural motion, moderate scaling (middle) shows limited gains, and full scaling (bottom) produces stable and natural motion sequences with improved coherence.




Figure 15: Inference-scaling effects: comparison of generated bird motion across three settings. **Top:** baseline shows static or unnatural motion. **Middle:** partial scaling provides limited enhancement. **Bottom:** full scaling produces smooth, natural motion through quality-guided inference.

# M Other System Optimizations and AE Training Details

## M.1 Video DC-AE Training Schedule

We train Video DC-AE with two channel sizes, 256 and 128, from scratch. Following DC-AE, our training loss has no KL term. We first train Video DC-AE with reconstruction loss $\mathcal{L}_1$ and perceptual loss $\mathcal{L}_{\text{LPIPS}}$ for 250k steps: $\mathcal{L} = \mathcal{L}_1 + 0.5\,\mathcal{L}_{\text{LPIPS}}$, then add an adversarial loss for another 200k steps: $\mathcal{L} = \mathcal{L}_1 + 0.5\,\mathcal{L}_{\text{LPIPS}} + 0.05\,\mathcal{L}_{\text{adv}}$. We train each AE on 8 GPUs with local batch size 1, on 32-frame, 256px, 1:1 videos. Learning rate is $5\times10^{-5}$ for the AE and $1\times10^{-4}$ for the discriminator (AdamW, $\beta = (0.9, 0.999)$, $\epsilon$=$1\times10^{-15}$, no weight decay, gradient norm clipped at 1).

## M.2 Activation Checkpointing

Activation checkpointing is applied selectively to reduce memory consumption without significantly increasing computational overhead. Instead of storing intermediate activations, only block inputs are retained, and the forward pass is recomputed during backpropagation. To minimize slowdowns, we avoid enabling checkpointing for every layer. In Stages 1 and 2 it is applied only to 8 layers of dual blocks and all single blocks; in Stage 3 it is enabled for all blocks, supplemented by activation offloading to the CPU. Offloading reduces memory footprint by asynchronously transferring activation tensors using pinned memory and asynchronous data movement.

## M.3 Auto Recovery

To ensure continuous training in large-scale distributed environments, we implement an auto-recovery system to handle unexpected failures (InfiniBand failures, storage system crashes, NCCL errors). The system continuously monitors training status, checking for unresponsiveness, significant slowdowns, or loss spikes. Upon detecting issues, all processes are halted and a faulty node checker diagnoses faulty nodes; if necessary, backup machines are deployed and training resumes automatically from the last checkpoint without a loss spike. Our GPU utilization rate exceeds 99%.

## M.4 Dataloader

To accelerate data movement between host and devices, we optimize PyTorch's dataloader. Instead of relying on PyTorch's default pinned-memory allocation (which invokes `cudaMallocHost` and may block CUDA kernel execution), we employ a pre-allocated pinned-memory buffer to prevent dynamic allocations and reduce overhead. We overlap data transfers with computation, prefetching the next step's data while the current batch is processed. We mitigate Python's garbage-collection overhead by disabling global GC and implementing manual memory management, preventing unnecessary synchronization delays.

## M.5 Checkpoint Optimization

Efficient checkpointing minimizes recovery latency. The save process involves three steps: reconstruct sharded weights via inter-GPU communication, transfer weights from CUDA to CPU, and write from CPU to disk. To optimize the latter two, we use pre-allocated pinned memory for fast weight transfer and asynchronous disk writing via C++ so that file I/O does not block training. These optimizations

reduce model-saving overhead to seconds. For loading, we use asynchronous pinned-memory copies with multi-threaded allocation and pipelined execution between shard reading and weight transfer; for large models stored in multiple shards we overlap phases across shards to maximize parallelism.

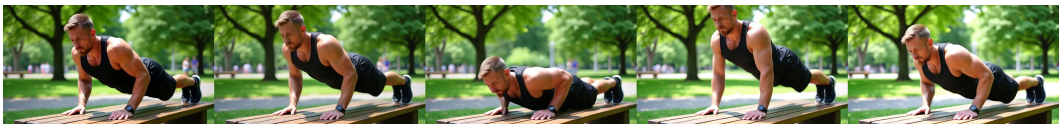
## N GENERATION SAMPLES



(a) Prompt: A scene from a disaster movie.



(b) Prompt: A panda bear with distinct black patches climbs and rests on a wooden log platform amid lush, natural foliage.



(c) Prompt: A man performs push-ups on a wooden bench in a sunny park, captured from a side angle in a medium shot. The focus is on his upper body and technique, with natural sunlight accentuating the scene. Lush greenery and distant park-goers contribute to the energetic, realistic setting.



(d) Prompt: A playful dog in a pink coat with a red leash dashes across a muddy field with sparse crops. The camera tracks its energetic movement from right to left against a backdrop of trees and distant power lines under an overcast sky.



(e) Prompt: A drone camera circles a historic church on a rocky outcrop along the Amalfi Coast, highlighting its stunning architecture, tiered patios, and the dramatic coastal views with waves crashing below and people enjoying the scene in the warm afternoon light.

Figure 16: High-quality videos generated by Open-Sora 2.0.

## O HUMAN EVALUATION PROTOCOL

**Prompt design.** Our 100 evaluation prompts are designed before the evaluation begins (and not modified afterward) to cover three orthogonal axes of behavior across diverse scenarios. The category distribution is:

**Evaluator protocol.** 10 professional video evaluators with experience in video production or visual quality assessment independently rated all 100 prompt pairs for each model comparison. Standardized instructions, with definitions and visual examples for each criterion, were provided. Videos were

Table 8: Prompt category distribution for the human-evaluation set.

| Category | # Prompts | Example |
|---|---|---|
| Multi-object scenes | 15 | "A cat chasing a butterfly through a garden…" |
| Human actions | 20 | "A man performs push-ups on a wooden bench…" |
| Natural landscapes | 15 | "A drone circles a historic church on a rocky outcrop…" |
| Cinematic shots | 15 | "A scene from a disaster movie…" |
| Abstract / artistic | 10 | "Watercolor painting coming to life…" |
| Fine-grained spatial relationships | 15 | "A red ball to the left of a blue cube on a table…" |
| Complex motion | 10 | "A playful dog dashes across a muddy field…" |

presented in randomized order with model identity hidden (double-blind). Per-prompt outcomes are determined by majority vote across evaluators.

**Inter-annotator agreement.** Fleiss' $\kappa$ across the 10 evaluators is 0.47 for Visual Quality, 0.52 for Prompt Adherence, and 0.43 for Motion Quality (overall 0.47), indicating moderate agreement. This is consistent with the inherent subjectivity of video quality assessment and with the 0.35–0.55 range reported for similar evaluations (Huang et al., 2024).

## P Bootstrap Confidence Intervals for Human Evaluation

**Methodology.** For each (baseline, criterion) pair, we resample (with replacement) the 100 per-prompt majority-vote outcomes 10,000 times and compute the empirical 2.5% and 97.5% percentiles of the resulting win-rate distribution. The intervals are stated as percentiles (i.e., they make no normality assumption).

Table 9: Full bootstrap 95% confidence intervals for the human-evaluation win rate of Open-Sora 2.0 vs. each baseline. Win rate is the percentage of comparisons in which our model was preferred. CIs that overlap 50% indicate *comparable* performance; CIs that exclude 50% indicate *statistically significant* superiority.

| Comparison | Visual Quality | Prompt Adherence | Motion Quality |
|---|---|---|---|
| vs. HunyuanVideo | 55.3% [45.6%, 64.8%] | 53.0% [43.1%, 62.7%] | 44.8% [35.1%, 54.7%] |
| vs. Step-Video-T2V | 54.8% [44.9%, 64.3%] | 72.3% [62.8%, 80.6%] | 59.1% [49.1%, 68.6%] |
| vs. Hailuo T2V-01 | 55.6% [45.7%, 65.1%] | 44.5% [34.8%, 54.4%] | 61.0% [51.0%, 70.4%] |
| vs. Vidu-1.5 | 55.3% [45.6%, 64.8%] | 45.6% [35.9%, 55.5%] | 69.5% [59.7%, 78.0%] |
| vs. Luma Ray2 | 55.8% [46.0%, 65.3%] | 69.5% [59.7%, 78.0%] | 57.5% [47.5%, 67.1%] |
| vs. Runway Gen-3 Alpha | 64.2% [54.3%, 73.3%] | 77.7% [68.5%, 85.2%] | 60.9% [50.9%, 70.3%] |

**Interpretation.** For comparisons near 50% (vs. HunyuanVideo on Visual Quality and Prompt Adherence; vs. Hailuo and Vidu-1.5 on Prompt Adherence), CIs overlap 50%; the appropriate claim is comparability. For Runway Gen-3 Alpha and Step-Video-T2V on Prompt Adherence, and for Vidu-1.5 on Motion Quality, CIs exclude 50% and the superiority claim is statistically significant.

## Q 22-Run Convergence Analysis: Per-Run Logs

**Observations.** (a) All 22 runs converge to terminal losses in the range $[0.038, 0.078]$, supporting recipe robustness rather than configuration dependence. (b) Comparing matched-configuration LoRA vs. full FT runs (e.g., 02 vs. 15, 09 vs. 16), LoRA achieves 3–4$\times$ lower terminal loss. (c) Replacing baseline data with our curated data (08 vs. 10, 14 vs. 17) consistently lowers terminal loss by $\sim$0.01–0.02. (d) 480p/81-frame and 720p/33-frame runs (e.g., 18 vs. 19) converge to similar terminal losses, supporting low-resolution motion learning transferring to high resolution. A separate longer-horizon run ($\sim$49k iterations) showed continued stable convergence.

Table 10: Per-run terminal-loss summary for the 22 I2V exploration runs analyzed in Section 7.2. Terminal average loss is averaged over the last 1k iterations of each run.

| Run | Resolution | Frames | Adapt | lr | Data / Loss | Term. loss |
|---|---|---|---|---|---|---|
| 01 | 480p | 33 | full FT | $5\times10^{-5}$ | baseline / L2 | 0.078 |
| 02 | 480p | 33 | full FT | $1\times10^{-5}$ | baseline / L2 | 0.072 |
| 03 | 480p | 33 | full FT | $5\times10^{-6}$ | baseline / L2 | 0.069 |
| 04 | 480p | 81 | full FT | $5\times10^{-5}$ | baseline / L2 | 0.075 |
| 05 | 480p | 81 | full FT | $1\times10^{-5}$ | baseline / L2 | 0.071 |
| 06 | 720p | 33 | full FT | $1\times10^{-5}$ | baseline / L2 | 0.073 |
| 07 | 720p | 33 | full FT | $5\times10^{-6}$ | baseline / L2 | 0.070 |
| 08 | 480p | 33 | full FT | $1\times10^{-5}$ | baseline / time-w. L2 | 0.068 |
| 09 | 480p | 33 | full FT | $1\times10^{-5}$ | curated / L2 | 0.061 |
| 10 | 480p | 33 | full FT | $1\times10^{-5}$ | curated / time-w. L2 | 0.056 |
| 11 | 480p | 81 | full FT | $1\times10^{-5}$ | curated / time-w. L2 | 0.054 |
| 12 | 720p | 33 | full FT | $1\times10^{-5}$ | curated / time-w. L2 | 0.058 |
| 13 | 720p | 33 | full FT | $5\times10^{-6}$ | curated / time-w. L2 | 0.052 |
| 14 | 480p | 33 | LoRA | $5\times10^{-5}$ | baseline / L2 | 0.046 |
| 15 | 480p | 33 | LoRA | $1\times10^{-5}$ | baseline / L2 | 0.042 |
| 16 | 480p | 33 | LoRA | $1\times10^{-5}$ | curated / L2 | 0.041 |
| 17 | 480p | 33 | LoRA | $1\times10^{-5}$ | curated / time-w. L2 | 0.039 |
| 18 | 480p | 81 | LoRA | $1\times10^{-5}$ | curated / time-w. L2 | 0.038 |
| 19 | 720p | 33 | LoRA | $1\times10^{-5}$ | curated / time-w. L2 | 0.040 |
| 20 | 720p | 33 | LoRA | $5\times10^{-6}$ | curated / time-w. L2 | 0.039 |
| 21 | 720p | 81 | LoRA | $1\times10^{-5}$ | curated / time-w. L2 | 0.043 |
| 22 | 720p | 81 | LoRA | $5\times10^{-6}$ | curated / time-w. L2 | 0.041 |

## R Per-Dimension VBench Breakdown

## S FVD Protocol on UCF-101

**Setup.** We compute zero-shot FVD on the UCF-101 test set: 2,048 videos, 16 frames per video, class-conditional prompts (one prompt per class). Following standard practice, we use the I3D feature extractor and report FVD on the same 2,048-video subset for all baselines we recompute. CogVideo and Make-A-Video numbers are reproduced from the respective papers.

**Caveats.** UCF-101 has a narrow distribution that differs substantially from open-domain video generation. Recent work (Stein et al., 2023) has demonstrated that FVD correlates poorly with human perception for modern high-quality models. We therefore report FVD as a complementary metric and not as our primary automatic metric. VBench (Section 7.1) is our primary automatic benchmark.

## T Limitations and Failure Cases

We describe the principal failure modes of Open-Sora 2.0 honestly. Representative failure-case figures are in Appendix U.

**(L1) Object distortion and anatomical errors.** The model occasionally generates distorted hands, fingers, and faces, particularly in complex poses or multi-person scenes. This is a common failure mode of diffusion-based video models, and is exacerbated for our fast variant by the $32\times$ spatial compression ratio of Video DC-AE, which limits fine-grained spatial detail in the latent space.

**(L2) Physics violations.** Generated videos occasionally violate physical laws: objects floating without support, unrealistic fluid dynamics, or incorrect gravity effects. The model learns motion patterns from data rather than from explicit physics priors, leading to plausible-looking but physically incorrect dynamics in out-of-distribution scenarios.

Table 11: Per-dimension VBench (Huang et al., 2024) comparison (Section 7.1). Dimensions are grouped by Quality (top) and Semantic (bottom). Open-Sora 2.0 is competitive with OpenAI Sora across most dimensions and exceeds HunyuanVideo and CogVideoX-1.5 on the overall total. Strongest dimensions: Motion Smoothness (99.1), Temporal Flickering (99.3), Subject/Background Consistency.

| Dimension | Open-Sora 2.0 | OpenAI Sora | HunyuanVideo | CogVideoX-1.5 |
|---|---|---|---|---|
| *Quality dimensions* | | | | |
| Subject Consistency | 97.8 | 98.1 | 96.3 | 95.2 |
| Background Consistency | 98.3 | 98.5 | 97.1 | 96.4 |
| Temporal Flickering | 99.3 | 99.4 | 98.5 | 97.8 |
| Motion Smoothness | 99.1 | 99.2 | 98.2 | 97.6 |
| Dynamic Degree | 58.5 | 61.2 | 52.8 | 46.3 |
| Aesthetic Quality | 65.1 | 66.3 | 62.4 | 59.8 |
| Imaging Quality | 72.4 | 73.5 | 69.8 | 67.2 |
| **Quality Score** | **84.4** | **85.5** | **82.8** | **81.3** |
| *Semantic dimensions* | | | | |
| Object Class | 89.2 | 88.5 | 86.8 | 82.4 |
| Multiple Objects | 56.3 | 55.8 | 52.5 | 44.7 |
| Human Action | 97.5 | 97.1 | 96.2 | 94.8 |
| Color | 88.1 | 87.3 | 85.6 | 81.2 |
| Spatial Relationship | 70.2 | 69.5 | 66.8 | 60.3 |
| Scene | 56.8 | 55.2 | 53.1 | 47.5 |
| Appearance Style | 25.1 | 25.4 | 23.8 | 22.1 |
| Temporal Style | 26.8 | 26.2 | 25.1 | 23.5 |
| Overall Consistency | 28.2 | 27.8 | 26.5 | 24.6 |
| **Semantic Score** | **80.3** | **79.8** | **78.6** | **73.4** |
| **Total Score** | **83.6** | **84.3** | **82.2** | **79.8** |

Table 12: Zero-shot FVD on UCF-101 (lower is better). Open-Sora 2.0 outperforms reported baselines we could access. We caution that UCF-101 has a narrow distribution unlike open-domain generation; our primary automatic metric is VBench (Section 7.1).

| Model | UCF-101 FVD ↓ |
|---|---|
| CogVideo (Hong et al., 2022) | 751.3 |
| Make-A-Video (Singer et al., 2022) | 367.2 |
| **Open-Sora 2.0 (Ours)** | **326.4** |

**(L3) Text rendering.** The model struggles to render readable text within videos (signs, labels). This is a known limitation of latent diffusion models: high spatial compression discards the fine-grained detail required for text rendering.

**(L4) Temporal consistency in complex scenes.** For prompts involving multiple interacting objects or detailed cause-and-effect chains, the model may produce inconsistent object appearances or illogical event sequences over time. This is particularly evident in longer sequences and for prompts with explicit counting ("three apples").

**(L5) Quality–speed trade-off of the high-compression AE.** The fast variant using Video DC-AE ($4\times32\times32$) produces somewhat blurrier outputs than the main variant using HunyuanVideo VAE ($4\times8\times8$), as discussed in Section 4.2. The main model is the HunyuanVideo-VAE-based one; the high-compression model is released as a research artifact for inference-speed-critical scenarios.

**(L6) Prompt adherence for complex compositional instructions.** When prompts contain multiple spatial relationships, specific object counts, or detailed attribute combinations simultaneously, the model may fail to satisfy all constraints. The T2I2V pipeline mitigates this by first generating a correct-looking image, but video extension may drift from the initial layout in long sequences.

**(L7) Baseline currency.** Our comparisons (HunyuanVideo, Runway Gen-3 Alpha, Step-Video-T2V, Luma Ray2, Vidu 1.5, Hailuo T2V-01, OpenAI Sora) reflect the SOTA at submission time. Newer systems — e.g., Kling and the Wan series — are not included; we expect those systems to have substantially higher training costs, which would only strengthen our cost-efficiency argument, but a direct head-to-head comparison must wait until reproducible inference is publicly available.

## U Failure-Case Examples

We refer the reader to Appendix T for the categorization of failure modes (L1–L7). Representative qualitative examples for each category are released alongside the code.

## V Broader Impact

Open-Sora 2.0 is a high-capability generative video model. We discuss the principal positive and negative societal impacts and the safeguards we put in place. Additional detail (a full risk–mitigation matrix, ethics expansion, and environmental impact accounting) is in Appendix W.

**Positive impacts.** (i) *Democratization.* The vast majority of research groups — academic labs, startups, and researchers in resource-constrained settings — cannot afford the $1M+ training budgets typical of contemporary video models. A reproducible $200k recipe lowers this barrier directly. (ii) *Reproducibility.* A $200k run is reproducible by a much wider set of researchers than a $2M run, which is itself a cornerstone of scientific progress. (iii) *Faster scientific iteration.* Even for well-resourced organizations, lower per-run cost enables more controlled experiments per dollar. (iv) *Environmental impact.* A 5–10$\times$ reduction in training compute translates approximately proportionally to lower energy consumption and carbon emissions; cost efficiency is therefore not just an economic but also an environmental concern. (v) *Defensive research.* Open-source video generators enable the security research community to study and develop defenses against synthetic media — significantly harder with closed-source systems.

**Negative impacts and mitigations.** We acknowledge that high-quality video generation has clear misuse risks. The principal categories are: **deepfakes / non-consensual impersonation**, **synthetic-media misinformation**, **non-consensual synthetic media**, **copyright infringement**, and **environmental cost** (which our recipe partially mitigates). A risk–severity–likelihood–mitigation matrix is provided in Appendix W.

**Data handling and provenance.** Our training data is sourced from publicly available video datasets, with the following safeguards applied during curation: (i) NSFW content classified by automated classifiers is removed from all training subsets; (ii) face detection flags content with identifiable individuals in close-up or personally identifiable contexts; (iii) OCR-based filtering removes videos with excessive overlaid text (potential watermarks, copyrighted material), using PaddleOCR (Appendix C); (iv) near-duplicates are removed to reduce memorization risk. We acknowledge that no automated pipeline is perfect; we will maintain a community reporting channel for problematic content.

**Release safeguards.** The model is released under a *responsible-use license* that prohibits (a) generation of non-consensual depictions of real individuals, (b) deceptive content intended to mislead, and (c) illegal material. We additionally recommend that downstream deployers integrate invisible watermarking (e.g., StegaStamp, Tree-Ring Watermarks) and content-moderation classifiers. As discussed in the LLM literature (Touvron et al., 2023), closed-source models remain API-accessible, so open-sourcing does not materially change the threat surface but does enable defensive research that is otherwise impossible.

**Human-evaluation participants.** Our 10 professional evaluators were compensated at standard market rates and informed of the evaluation's purpose; the full protocol is in Appendix O.

## W Expanded Ethics Discussion

Table 13: Risk–mitigation matrix for Open-Sora 2.0.

| Risk Category | Severity | Likelihood | Mitigation |
|---|---|---|---|
| Deepfakes / impersonation | High | High | Watermarking recommendation, responsible-use license, downstream content-detection classifiers. |
| Misinformation via synthetic video | High | Medium | Provenance tracking; we encourage media-literacy advocacy. |
| Non-consensual synthetic media | High | Medium | License prohibition; downstream content moderation. |
| Copyright infringement in outputs | Medium | Medium | Data filtering (NSFW, OCR, dedup); user guidelines. |
| Environmental impact of training | Medium | Certain | Our 5–10× cost reduction directly mitigates this. |

**Risk–mitigation matrix.**

**Data handling and provenance.** Training data is sourced from publicly available video datasets. During curation we apply: NSFW classifier filtering; face detection / PII filtering for close-up identifiable individuals; OCR-based filtering for excessive overlaid text (likely watermarks or copyrighted material); near-duplicate removal. We acknowledge no automated pipeline is perfect and we will maintain a community channel for reporting problematic content.

**Release safeguards.** The model is released under a responsible-use license that explicitly prohibits (a) generation of non-consensual depictions of real individuals, (b) creation of deceptive content intended to mislead, (c) generation of illegal material. We recommend that downstream deployers integrate invisible watermarking (e.g., StegaStamp, Tree-Ring Watermarks) and content moderation classifiers.

**The case for open-sourcing.** Closed-source video generators are still accessible via APIs, so closing the source does not fundamentally change the threat surface, but it does prevent the defensive research community from studying the underlying model. The ML community has broadly recognized that open models enable faster safety research. Our open release is consistent with this view, with the safeguards above.

Table 14: Estimated training compute (H100-equivalent GPU-hours) and relative cost.

| Model | Estimated GPU-hours | Relative cost |
|---|---|---|
| MovieGen (Polyak et al., 2024) | ∼1,250,000 | 12.5× |
| Step-Video-T2V (Ma et al., 2025a) | ∼500,000 | 5.0× |
| **Open-Sora 2.0 (Ours)** | ∼100,000 | 1.0× |

**Environmental impact accounting.** A 5–10× reduction in training compute translates approximately proportionally to lower energy consumption and carbon emissions. As the community increasingly recognizes the environmental cost of large-scale training, cost-efficient methods become a necessity.

## X Cost Transparency

The $200k figure of Table 2 refers to the *marginal cost of a single successful training run*, computed as GPU-hours × rental price. This is the same accounting methodology used by comparable systems: MovieGen (Polyak et al., 2024) and Step-Video-T2V (Ma et al., 2025a) both report a single successful training run, without including failed runs or infrastructure overhead. Because reviewers of an earlier version of this paper raised concerns about the framing of this number, we make the boundary explicit here.

**What is included in $200k.** The total marginal cost of one full training run on H200 GPUs at $2/GPU-hour: 256px T2V ($107.5k) + 256px T/I2V ($18.4k) + 768px T/I2V ($73.7k) = $199.6k.

**What is *not* included.** The total R&D cost is necessarily higher, and we list the additional cost sources for honesty: (i) Video DC-AE training (Section 3.1), which is performed once and shared across all subsequent runs; (ii) data captioning (open-source LLaVA-Video for 256px is essentially

free at our scale, but Qwen 2.5 Max API for 768px caption refinement incurs a non-trivial token cost); (iii) data preprocessing and filtering compute (FFmpeg-based scene detection, filter scoring); (iv) infrastructure setup, debugging, and partial pre-final training runs that did not reach the final checkpoint; (v) the 22 I2V exploration runs analyzed in Section 7.2, which were performed at smaller scale and shorter horizon than the full Stage 3 run; (vi) human evaluation labor for the 100-prompt 10-evaluator study (Section 6).

**Pretrained components.** We initialize from FLUX (Black Forest Labs, 2024), which is open-source and freely available; its training cost is not ours, just as the training cost of T5-XXL (Chung et al., 2024) or CLIP-Large (Radford et al., 2021) is not counted in any diffusion paper that uses them as text encoders. We isolate FLUX's contribution to final video quality in Section 7.1: it accounts for $\sim$2.5 VBench points, concentrated in visual rather than motion quality.

**Bottom line.** Even after accounting for items (i)–(vi), our total R&D cost remains 5–10$\times$ lower than the marginal training run of comparable systems. A detailed best-effort breakdown of auxiliary costs is provided in Appendix Y.

## Y Detailed Cost-Transparency Breakdown

This appendix expands Appendix X with a best-effort breakdown of the auxiliary R&D costs that are *not* included in the $200k marginal-training-run figure of Table 2. The numbers are best-effort estimates; we deliberately do not include them in the headline number to keep the accounting comparable to MovieGen and Step-Video-T2V, which similarly report only their successful training runs.

Table 15: Best-effort breakdown of auxiliary R&D costs that are *not* included in the $200k marginal training-run figure. Even with these added, the total R&D cost remains 5–10$\times$ lower than the *marginal* training cost of comparable systems.

| Cost source | Approximate USD |
|---|---|
| Video DC-AE training (Section 3.1) | $\sim$\$30k |
| Captioning APIs (Qwen 2.5 Max for 768px subset) | $\sim$\$15k |
| Data preprocessing and filtering compute | $\sim$\$10k |
| 22 I2V exploration runs (Section 7.2) | $\sim$\$40k |
| Infrastructure setup and partial pre-final training runs | $\sim$\$25k |
| Human evaluation labor (10 evaluators, 100 prompts, 6 baselines) | $\sim$\$5k |
| **Total auxiliary R&D** | **$\sim$\$125k** |
| **Marginal training run (Table 2)** | **$\sim$\$200k** |
| **Grand total (best estimate)** | **$\sim$\$325k** |

For comparison, the *marginal training cost alone* of MovieGen is approximately $2.5M (Polyak et al., 2024) and of Step-Video-T2V approximately $1M (Ma et al., 2025a). Even with our auxiliary costs added, the total R&D cost remains 3–8$\times$ lower than the marginal training cost of these comparable systems — and 5–10$\times$ lower if compared on the same marginal-training-run basis as the headline number.