Meta

Movie Gen: A Cast of Media Foundation Models

The Movie Gen team @ Meta[1]
[1]A detailed contributor list can be found in the appendix of this paper.

We present Movie Gen, a cast of foundation models that generates high-quality, 1080p HD videos with different aspect ratios and synchronized audio. We also show additional capabilities such as precise instruction-based video editing and generation of personalized videos based on a user's image. Our models set a new state-of-the-art on multiple tasks: text-to-video synthesis, video personalization, video editing, video-to-audio generation, and text-to-audio generation. Our largest video generation model is a 30B parameter transformer trained with a maximum context length of 73K video tokens, corresponding to a generated video of 16 seconds at 16 frames-per-second. We show multiple technical innovations and simplifications on the architecture, latent spaces, training objectives and recipes, data curation, evaluation protocols, parallelization techniques, and inference optimizations that allow us to reap the benefits of scaling pre-training data, model size, and training compute for training large scale media generation models. We hope this paper helps the research community to accelerate progress and innovation in media generation models.

Date: October 4, 2024
Blogpost: https://ai.meta.com/blog/movie-gen-media-foundation-models-generative-ai-video/

1 Introduction

Imagine a blue emu swimming through the ocean. Humans have the astonishing ability to imagine such a fictional scene in great detail. Human imagination requires the ability to compose and predict various facets of the world. Simply imagining a scene requires composing different concepts while predicting realistic properties about motion, scene, physics, geometry, audio *etc*. Equipping AI systems with such generative, compositional, and prediction capabilities is a core scientific challenge with broad applications. While Large Language Models (LLMs) (Dubey et al., 2024; Touvron et al., 2023; Brown et al., 2020; Team Gemini, 2023) aim to learn such capabilities with a text output space, in this paper we focus on media – image, video, audio – as the output space. We present Movie Gen, a cast of media generation foundation models. Movie Gen models can natively generate high fidelity images, video, and audio while also possessing the abilities to edit and personalize the videos as we illustrate in Figure 1.

We find that scaling the training data, compute, and model parameters of a simple Transformer-based (Vaswani et al., 2017) model trained with Flow Matching (Lipman et al., 2023) yields high quality generative models for video or audio. Our models are pre-trained on internet scale image, video, and audio data. Our largest foundation text-to-video generation model, Movie Gen Video, consists of 30B parameters, while our largest foundation video-to-audio generation model, Movie Gen Audio, consists of 13B parameters. We further post-train the Movie Gen Video model to obtain Personalized Movie Gen Video that can generate personalized videos conditioned on a person's face. Finally, we show a novel post-training procedure to produce Movie Gen Edit that can precisely edit videos. In conjunction, these models can be used to create realistic personalized HD videos of up to 16 seconds (at 16 FPS) and 48kHz audio, and the ability to edit real or generated videos.

The Movie Gen cast of foundation models is state-of-the-art on multiple media generation tasks for video and audio. On text-to-video generation, we outperform prior state-of-the-art, including commercial systems

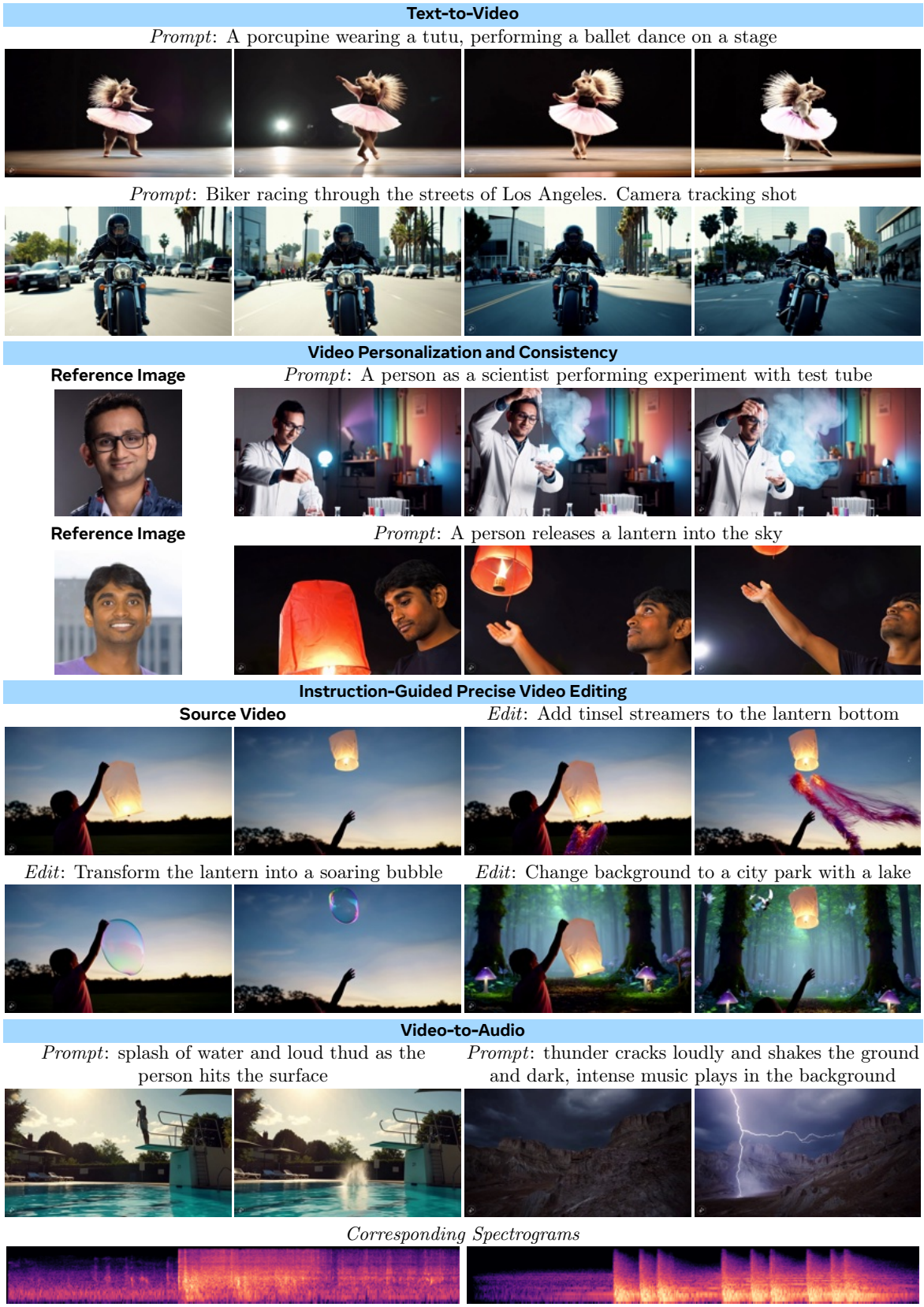


Figure 1 Examples of the different capabilities of Movie Gen. The Movie Gen cast of models generates videos from text prompts, supports the generation of videos consistent with characters in provided reference images, supports precise video editing given user provided instructions, and generates videos with synchronized audio. Videos in this Figure found at https://go.fb.me/MovieGen-Figure1.

such as Runway Gen3 (RunwayML, 2024), LumaLabs (LumaLabs, 2024), OpenAI Sora (OpenAI, 2024) on overall video quality as shown in Table 6. Moreover, with Personalized Movie Gen Video and Movie Gen Edit we enable new capabilities on video personalization and precise video editing respectively, and both these capabilities are missing from current commercial systems. On both these tasks too, we outperform all prior work (Table 15 and Table 17). Finally, Movie Gen Audio, outperforms prior state-of-the-art, including commercial systems such as PikaLabs (Pika Labs) and ElevenLabs (ElevenLabs) for sound-effect generation (Table 29), music generation (Table 30), and audio extension.

To enable future benchmarking, we hope to publicly release multiple benchmarks—Movie Gen Video Bench (Section 3.5.2), Movie Gen Edit Bench (Section 5.2.1), Movie Gen Audio Bench (Section 6.3.2). We also provide thorough details on model architectures, training, inference, and experimental settings which we hope will accelerate research in media generation models.

2 Overview

The Movie Gen cast of models generates videos with synchronized audio, personalized characters, and supports video editing as illustrated in Figure 1.

We achieve these wide capabilities using two foundation models:

- **Movie Gen Video**. A 30B parameter foundation model for joint text-to-image and text-to-video generation that generates high-quality HD videos of up to 16 seconds duration that follow the text prompt. The model naturally generates high-quality images and videos in multiple aspect ratios and variable resolutions and durations. The model is pre-trained jointly on $\mathcal{O}(100)$M videos and $\mathcal{O}(1)$B images and learns about the visual world by 'watching' videos. We find that the pre-trained model can reason about object motion, subject-object interactions, geometry, camera motion, and physics, and learns plausible motions for a wide variety of concepts. To improve the video generations, we perform supervised finetuning (SFT) on a small set of curated high-quality videos and text captions. We present the model architecture and training details in Section 3.
- **Movie Gen Audio**. A 13B parameter foundation model for video- and text-to-audio generation that can generate 48kHz high-quality cinematic sound effects and music synchronized with the video input, and follow an input text prompt. The model naturally handles variable length audio generation and can produce long-form coherent audio for videos up to several minutes long via audio extension techniques. We pre-train the model on $\mathcal{O}(1)$M hours of audio and observe that it learns not only the physical association, but also the psychological associations between the visual and the audio world. The model can generate diegetic ambient sounds matching the visual scene even when the source is unseen, and also diegetic sound effects synchronized with the visual actions. Moreover, it can generate non-diegetic music that supports the mood and aligns with the actions of the visual scene, and blend sound effects and background music professionally. We further perform SFT on a small set of curated higher quality (text, audio) and (video, text, audio) data which improves the overall audio quality and aims for cinematic styles. The model and training recipe are outlined in Section 6.

We add video personalization and video editing capabilities to our foundation Movie Gen Video model via post-training procedures:

- **Personalization** enables the video generation model to condition on text as well as an image of a person to generate a video featuring the chosen person. The generated personalized video maintains the identity of the person while following the text prompt. We use a subset of videos containing humans, and automatically construct pairs of (image, text) inputs and video outputs to train the model. We outline the post training strategy for personalization in Section 4.
- **Precise Editing** allows users to effortlessly perform precise and imaginative edits on both real and generated videos using a textual instruction. Since large-scale supervised video editing data is harder to obtain, we show a novel approach to train such a video editing model without supervised video editing data (Section 5). We provide examples of our model's video editing capabilities in https://go.fb.me/MovieGen-Figure24.

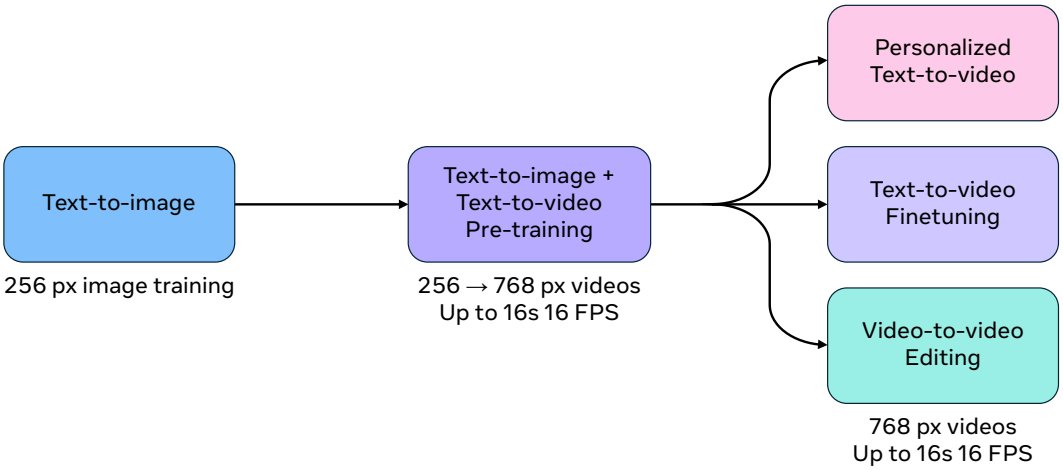


Figure 2 Training recipe for Movie Gen Video. We first pre-train our model for the text-to-image task followed by joint text-to-image and text-to-video pre-training at increasingly higher spatial resolutions (Section 3). We finetune the model on high aesthetic and motion quality videos to improve our video generations. We also add additional capabilities like personalization (Section 4) and video-to-video editing (Section 5).

3 Joint Image and Video Generation

We train a single joint foundation model, Movie Gen Video, for the text-to-image and the text-to-video tasks. Given a text prompt as input, our foundation model generates a video consisting of multiple RGB frames as output. We treat images as a single frame video, enabling us to use the same model to generate both images and videos. Compared to video data, paired image-text datasets are easier to scale with diverse concepts and styles (Ho et al., 2022a; Girdhar et al., 2024) and thus joint modeling of image and video leads to better generalization. Our training recipe is illustrated in Figure 2. We perform our training in multiple stages for training efficiency. We first pretrain our model only on low-resolution 256 px images followed by joint pre-training on low-resolution images and videos, and high-resolution joint training. We finetune the model on high quality videos to improve the generations. Additionally, we add capabilities such as personalization and editing by post-training.

For improved training and inference efficiency, we perform generation in a spatio-temporally compressed latent space. Towards this, we train a single temporal autoencoder model (TAE) to map both RGB images and videos into a spatio-temporally compressed latent space, and vice-versa. We encode the user-provided text prompt using pre-trained text-encoders to obtain text prompt embeddings, which are used as conditioning for our model. We use the Flow Matching training objective (Lipman et al., 2023) to train our generative model. Taking sampled noise and all provided conditioning as input, our generative model produces an output latent. This is passed through the TAE decoder to map it back to the pixel space and produce an output image or video. We illustrate the overview of the joint image and video generation pipeline in Figure 3.

We focus on simplicity when making design choices for all components in our foundation model, including the training objective, backbone architecture, and spatio-temporal compression using the TAE. These choices, which include using the LLaMa3 (Dubey et al., 2024) backbone architecture for the joint image-video generation model, allow us to confidently scale the model size while allowing for efficient training. Our largest 30B parameter model can directly generate video at different aspect ratios (*e.g.*, 1:1, 9:16, 16:9), of multiple lengths (4 – 16 seconds) at 768 × 768 px resolution (scaled appropriately based on the aspect ratio). Our Spatial Upsampler can further increase the spatial resolution to produce a video in full HD 1080p resolution.

Next, we describe the model architecture, pretraining and finetuning procedures for the foundation Movie Gen Video model.

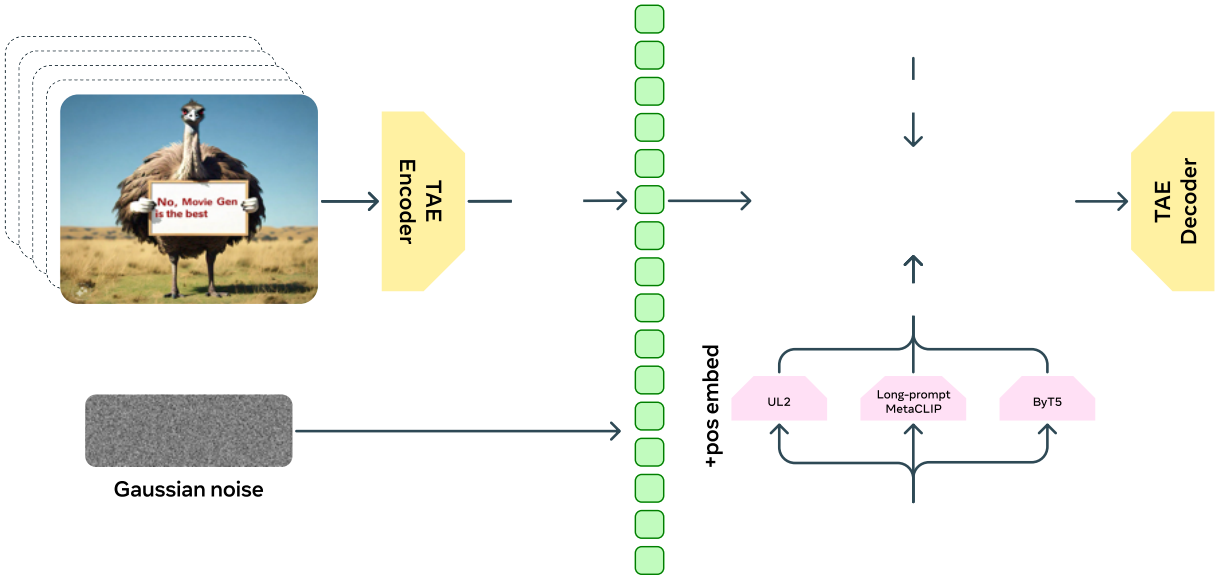


Figure 3 Overview of the joint image and video generation pipeline. We train our generative model on a spatio-temporally compressed latent sapce, which is learnt via a temporal autoencoder model (TAE). User-provided text prompts are encoded using pre-trained text-encoders, and used as conditioning. Our generative model takes sampled Gaussian noise and all provided conditioning as input, and generates an output latent, which is decoded to an output image or video using the TAE decoder.

3.1 Image and Video Foundation Model

We describe the key components of the Movie Gen Video model—the spatio-temporal autoencoder (TAE), the training objective for image and video generation, model architecture, and the model scaling techniques we use in our work.

3.1.1 Temporal Autoencoder (TAE)

For the purposes of efficiency, we encode the RGB pixel-space videos and images into a learned spatio-temporally compressed latent space using a Temporal Autoencoder (TAE), and learn to generate videos in this latent space. Our TAE is based on a variational autoencoder (Kingma, 2013) and compresses the input pixel space video $\mathbf{V}$ of shape $T' \times 3 \times H' \times W'$ to a continuous-valued latent $\mathbf{X}$ of shape $T \times C \times H \times W$, where $T < T'$, $H < H'$, $W < W'$. In our implementation, we compress the input $8\times$ across each of the spatio-temporal dimensions, *i.e.*, $T'/T = H'/H = W'/W = 8$. This compression reduces the overall sequence length of the input to the Transformer backbone, enabling the generation of long and high-resolution video at native frame rates. This choice also allows us to forego frame-interpolation models commonly used in prior work (Girdhar et al., 2024; Singer et al., 2023; Ho et al., 2022a), thereby simplifying our model.

TAE architecture. We adopt the architecture used for image autoencoders from (Rombach et al., 2022) and 'inflate' it by adding temporal parameters: a 1D temporal convolution after each 2D spatial convolution and a 1D temporal attention after each spatial attention. All temporal convolutions use symmetrical replicate padding. Temporal downsampling is performed via strided convolution with stride of 2, and upsampling by nearest-neighbour interpolation followed by convolution. Downsampling via strided convolution means that videos of any length are able to be encoded (notably including images, which are treated as single-frame videos) by discarding spurious output frames as shown in Figure 4. Similar to (Dai et al., 2023), we find that increasing the number of channels in the latent space $\mathbf{X}$ improves both the reconstruction and the generation performance. We use $C = 16$ in this work. We initialize the spatial parameters in the TAE using a pre-trained image autoencoder, and then add the temporal parameters to inflate the model as described above. After inflation, we jointly train the TAE on both images and videos, in a ratio of 1 batch of images to 3 batches of

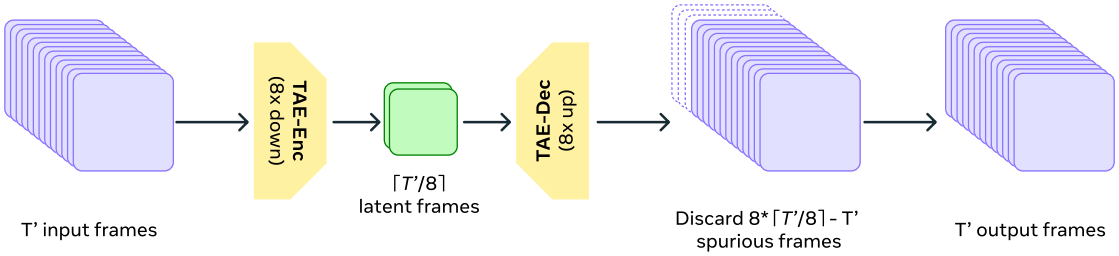


Figure 4 Variable length video encoding and decoding using the TAE. The TAE encoder encodes T' frames in the RGB pixel space to $\lceil T'/8 \rceil$ latent frames. The TAE decoder then decodes to $8 \times \lceil T'/8 \rceil$ real frames, and any spurious frames $(8 \times \lceil T'/8 \rceil - T')$ are discarded.

videos.

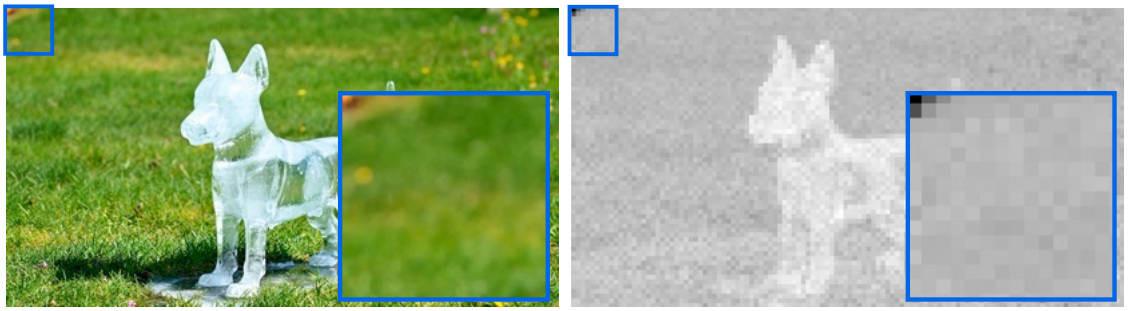
Improvements to the training objective. We find that the standard training objective used in (Rombach et al., 2022) leads to a 'spot' artifact in the decoded pixel-space videos, as shown in Figure 5. On further inspection, we found that the model produced latent codes with high norms ('latent dots') in certain spatial locations, which when decoded led to 'spots' in the pixel space. We hypothesize that this is a form of shortcut learning, where the model learns to store crucial global information in these high-norm latent dots. A similar phenomenon has been documented in (Darcet et al., 2023), where the authors discovered that vision Transformers can produce high-norm latent tokens, and also in (Karras et al., 2024), where they found that eliminating global operators such as group norms resolves the issue.

Rather than change the model architecture, we opt to add a term to the loss which penalizes the model for encoding latent values which are far from the mean. Concretely, given an input latent $\mathbf{X}$, our outlier penalty loss (OPL) is given by

$$\mathcal{L}_{\text{OPL}}(\mathbf{X}, r) = \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \max\left(\|\mathbf{X}_{i,j} - \text{Mean}(\mathbf{X})\| - r\,\|\text{Std}(\mathbf{X})\|, 0\right), \tag{1}$$

where r is a scaling factor which denotes how far outside of the standard deviation a latent value needs to be to be penalized. For images, equation (1) is used as-is; for videos, T is rolled into the batch dimension. Adding $\mathcal{L}_{\text{OPL}}$ to the typical variational autoencoder losses (reconstruction, discriminator, and perceptual) removes the dot artifacts. In practice, we set $r = 3$ and a large loss weight ($1e5$) for the outlier loss.

Efficient inference using temporal tiling. Encoding and decoding high resolution long videos, *e.g.*, up to 1024×1024 px and 256 frames naïvely is not feasible due to memory requirements. To facilitate inference with large videos, we divide both the input video and latent tensor into tiles along the temporal dimension, encode and/or decode each tile, and stitch the result together at the output. When tiling, it is possible to



(a) Spot artifact (b) Latent dot

Figure 5 Spot artifact and corresponding latent dot. (a) Frame from a generated video displaying a spot artifact in the top left corner, (b) Visualization of a TAE feature channel, where the corresponding latent dot is visible.

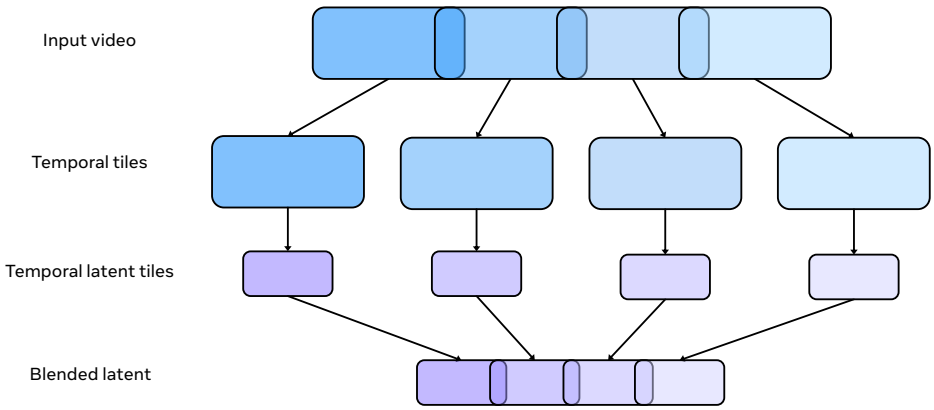


Figure 6 Tiled inference using the TAE. An input video is split across the time dimension into uniform tiles, with optional overlap. Each tile is sent through the model forward pass. If overlap was used, a linearly weighted blend is performed during reconstruction.

include some overlap between tiles, with an additional weighted blend between adjacent tiles when stitching tiles back together. Overlapping and blending can be applied to both the encoder and decoder, and has the effect of removing boundary artifacts at the cost of additional computation. In practice, we use a tile size of 32 raw frames (or 4 latent frames), tile without overlap in the encoder, and tile with overlap of 16 raw frames (or 2 latent frames) in the decoder. For blending, we use a linear combination between frames i and $i+1$, $x^j_{\text{blend}} = \sum_j^N \left[w^j x^j_i + \left(1 - w^j\right) x^j_{i+1} \right]$, where j is indexed over the N overlapping frames, and $w^j = j/N$. Figure 6 shows the basic flow of tiled inference.

3.1.2 Training Objective for Video and Image Generation

We use the Flow Matching (Lipman et al., 2023) framework to train our joint image and video generation model. Flow Matching generates a sample from the target data distribution by iteratively changing a sample from a prior distribution, *e.g.*, Gaussian. At training time, given a video sample in the latent space $\mathbf{X}_1$, we sample a time-step $t \in [0, 1]$, and a 'noise' sample $\mathbf{X}_0 \sim \mathcal{N}(0, 1)$, and use them to construct a training sample $\mathbf{X}_t$. The model is trained to predict the velocity $\mathbf{V}_t = \dfrac{d\mathbf{X}_t}{dt}$ which teaches it to 'move' the sample $\mathbf{X}_t$ in the direction of the video sample $\mathbf{X}_1$.

While there are numerous ways to construct $\mathbf{X}_t$, in our work, we use simple linear interpolation or the optimal transport path (Lipman et al., 2023), *i.e.*,

$$\mathbf{X}_t = t\ \mathbf{X}_1 + (1 - (1 - \sigma_{\text{min}})t)\ \mathbf{X}_0,$$

where $\sigma_{\text{min}} = 10^{-5}$. Thus, the ground truth velocity can be derived as

$$\begin{aligned} \mathbf{V}_t &= \frac{d\mathbf{X}_t}{dt} \\ &= \mathbf{X}_1 - (1 - \sigma_{\text{min}})\mathbf{X}_0. \end{aligned}$$

Denoting the model parameters by θ and text prompt embedding $\mathbf{P}$, we denote the predicted velocity as $u(\mathbf{X}_t, \mathbf{P}, t)$. The model is trained by minimizing the mean squared error between the ground truth velocity and model prediction,

$$\mathbb{E}_{t,\mathbf{X}_0,\mathbf{X}_1,\mathbf{P}} \| u(\mathbf{X}_t, \mathbf{P}, t; \theta) - \mathbf{V}_t \|^2. \tag{2}$$

As in prior work (Esser et al., 2024), we sample t from a logit-normal distribution where the underlying Gaussian distribution has zero mean and unit standard deviation.

Inference. At inference, we first sample $\mathbf{X}_0 \sim \mathcal{N}(0, 1)$ and then use an ordinary differential equation (ODE) solver to compute $\mathbf{X}_1$ using the model's estimated values for $\dfrac{d\mathbf{X}_t}{dt}$. In practice, there are multiple design

choices in the exact ODE solver configuration, *e.g.*, first or higher order solvers, step sizes, tolerance, *etc.* that affect the runtime and precision of the estimated $\mathbf{X}_1$. We use a simple first-order Euler ODE solver with a unique discrete set of N time-steps tailored to our model, as described in Section 3.4.2.

Signal-to-noise ratio. The time-step t controls the signal-to-noise (SNR) ratio, and our simple interpolation scheme for constructing $\mathbf{X}_t$ ensures zero SNR when $t = 0$. This ensures that, during training, the model receives pure Gaussian noise samples and is trained to predict the velocity for them. Thus, at inference, when the model receives pure Gaussian noise at $t = 0$ it can make a reasonable prediction.

Most video generation models (Ho et al., 2022a; Girdhar et al., 2024; Blattmann et al., 2023b; Singer et al., 2023) are trained using the diffusion formulation (Sohl-Dickstein et al., 2015; Ho et al., 2020, 2022a). Recent work (Girdhar et al., 2024; Lin et al., 2024) shows that choosing the right diffusion noise schedules with a zero terminal signal-to-noise ratio is particularly important for video generation. Standard diffusion noise schedules do not ensure a zero terminal SNR, and thus need to be modified for video generation purposes. As noted above, our Flow Matching implementation naturally ensures zero terminal SNR. Empirically, we found that Flow Matching was more robust to the exact choice of noise schedules and it outperforms diffusion losses (see Section 3.6.2). Thus, we adopt Flow Matching for its simplicity and high performance.

3.1.3 Joint Image and Video Generation Backbone Architecture

As discussed in Section 3.1.1, we perform generation in a learned latent space representation of the video. This latent code is of shape $T \times C \times H \times W$. To prepare inputs for the Transformer backbone, the video latent code is first 'patchified' using a 3D convolutional layer (Dosovitskiy et al., 2021) and then flattened to yield a 1D sequence. The 3D convolutional layer uses a kernel size of $k_t \times k_h \times k_w$ with a stride equal to the kernel size and projects it into the same dimensions as needed by the Transformer backbone. Thus, the total number of tokens input to the Transformer backbone is $THW/(k_t k_h k_w)$. We use $k_t = 1$ and $k_h = k_w = 2$, *i.e.*, we produce 2×2 spatial patches.

We use a factorized learnable positional embedding to enable arbitrary size, aspect ratio, and video length (Dehghani et al., 2024) inputs to the Transformer. Absolute embeddings of D dimensions can be denoted as a mapping $\phi(i) : [0, \text{maxLen}] \rightarrow \mathbb{R}^D$ where i denotes the absolute index of the patch. We convert the 'patchified' tokens, *i.e.*, output of the 3D convolutional layer, into separate embeddings ϕ_h, ϕ_w and ϕ_t of spatial h, w, and temporal t coordinates. We define H_{max}, W_{max}, and T_{max} as the maximum sequence length (maxLen) for each dimension, which correspond to the maximum spatial size and video length of the patchified inputs. We calculate the final positional embeddings by adding all the factorized positional embeddings together. Finally, we add the final positional embeddings to the input for all the Transformer layers. Compared with adding the positional embeddings to the first layer only, adding to all layers can effectively reduce the distortion and morphing artifacts, especially in the temporal dimension.

We build our Transformer backbone by closely following the Transformer block used in the LLaMa3 (Dubey et al., 2024) architecture. We use RMSNorm (Zhang and Sennrich, 2019) and SwiGLU (Shazeer, 2020) as in prior work. We make three changes to the LLaMa3 Transformer block for our use case of video generation using Flow Matching:

1. To incorporate text conditioning based on the text prompt embedding $\mathbf{P}$, we add a cross-attention module between the self-attention module and the feed forward network (FFN) to each Transformer block. We leverage multiple different text encoders due to their complementary strengths, as explained in the following section, and simply concatenate their embeddings in a single sequence to construct $\mathbf{P}$.
2. We add adaptive layer norm blocks to incorporate the time-step t to the Transformer, as used in prior work (Peebles and Xie, 2023).
3. We use full bi-directional attention instead of causal attention used in language modeling.

We intentionally keep the design of our backbone simple and similar to LLMs, specifically LLaMa3. This design choice allows us scale the model size and training, as discussed in Section 3.1.6, using similar techniques as used in LLMs. Empirically, we find that our architecture design performs on par or better than specialized blocks used in prior work (Balaji et al., 2022; Esser et al., 2024) while being more stable to train across a range of hyperparameters such as model size, learning rate, and batch size. We list the key hyperparameters for our

Layers	Model Dimension	FFN Dimension	Attention Heads	Activation Function	Normalization
48	6144	16384	48	SwiGLU	RMSNorm

Table 1 Architecture hyperparameters for the Movie Gen Video 30B parameter foundation model. Our model architecture is a Transformer (Vaswani et al., 2017) and we closely follow the LLaMa3 (Dubey et al., 2024) design space. Our model contains 30B parameters in the Transformer itself without including the the text embedding models, TAE, *etc.*

largest model in Table 1 and illustrate the Transformer block in Figure 8 with details of feature dimensions in a number of key places of our Transformer backbone.

3.1.4 Rich Text Embeddings and Visual-text Generation

We use pre-trained text encoders to convert the input text prompt p into a text embedding $\mathbf{P}$, which we use as conditioning input for the video generation backbone. We use a combination of UL2 (Tay et al., 2022), ByT5 (Xue et al., 2022), and Long-prompt MetaCLIP as text encoders to provide both semantic-level and character-level text understanding for the backbone. The Long-prompt MetaCLIP model is obtained by finetuning the MetaCLIP text encoder (Xu et al., 2023) on longer text captions to increase the length of input text tokens from 77 to 256. We concatenate the text embeddings from the three text encoders after adding separate linear projection and LayerNorm layers to project them into the same 6144 dimension space and normalize the embeddings. The UL2 and Long-prompt MetaCLIP text encoders provide prompt-level embeddings with different properties—UL2 is trained using massive text-only data and potentially provides strong text reasoning abilities in its features; Long-prompt MetaCLIP provides text representations that are aligned with visual representations that are beneficial for cross-modal generation. The character-level ByT5 encoder is only used to encode visual text, *i.e.*, the part of the text prompt that may explicitly ask for a character string to be generated in the output.

Controlling the FPS. We use FPS conditioning to control the length of the generated videos by pre-appending the sampling FPS value of each training video to the input text prompt (*e.g.*, "FPS-16"). During pre-training, we sample video clips at their original FPS with minimum of 16 FPS. In finetuning, we sample clips at two fixed FPS values of 16 and 24.

3.1.5 Spatial Upsampling

We use a separate Spatial Upsampler model to convert our 768 px videos to full HD (1080p) resolution. This lowers the overall computational cost for high resolution generation, since the base text-to-video model processes fewer tokens.

As shown in Figure 7, we formulate spatial upsampling as a video-to-video generation task, that generates a HD output video conditioned on a lower-resolution input video. The low-resolution video is first spatially upsampled using bilinear interpolation in the pixel space to the desired output resolution. Next, the video is converted to the latent space using a VAE. We use a frame-wise VAE for the upsampler to improve pixel sharpness. Finally, a latent space model generates the latents of a HD video, conditioned on the latents of the corresponding low-resolution video. The resulting HD video latents are subsequently decoded into pixel space frame-wise using the VAE decoder.

Implementation details. Our Spatial Upsampler model architecture is a smaller variant (7B parameters) of the text-to-video Transformer initialized from a text-to-image model trained at 1024 px resolution, allowing for better utilization of high-resolution image data. The Spatial Upsampler is trained to predict the latents of a video which are then decoded frame-wise using the VAE's decoder. Similar to (Girdhar et al., 2024), the encoded video is concatenated channel-wise with the generation input and is fed to the Spatial Upsampler Transformer. The additional parameters at the input, due to concatenation, are zero initialized (Singer et al., 2023; Girdhar et al., 2024). We train our Spatial Upsampler on clips of 14 frames at 24 FPS on ∼400K HD videos. We apply a second-order degradation (Wang et al., 2021) process to simulate complex degradations in the input and train the model to produce HD output videos. At inference time, we will use our Spatial Upsampler on videos that have been decoded with the TAE. To minimize this potential train-test discrepancy, we randomly substitute the second-order degradation with artifacts produced by the TAE. Due to the strong

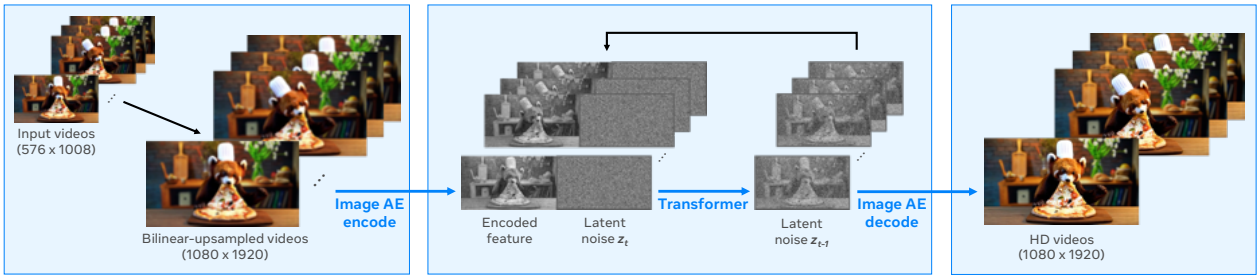


Figure 7 Overview of the Spatial Upsampler. Our upsampler is a conditional video-to-video model that upsamples the 768 px video to full HD 1080p. First, the input 768 px video is bilinearly upsampled to HD and then encoded to the latent space of an image encoder. The video latents are concatenated with noise, and denoised using a trained transfomer. Finally, the denoised latents are passed to the image decoder to produce the upsampled video.

input conditioning, *i.e.*, the low-resolution video, we observed that the model produces good outputs with as few as 20 inference steps. This simple architecture can be used for various multiples of super resolution; however, we train a 2× spatial super-resolution model for our case. Similar to TAE tiling (Section 3.1.1), we upsample videos using a sliding window approach with a window size of 14 and an overlap of 4 latent frames.

Improved temporal consistency with Multi-Diffusion. Memory constraints prohibit us from training the Spatial Upsampler on longer video durations. As a result, during inference, we upsample videos in a sliding window fashion resulting in noticeable inconsistencies at the boundaries. To prevent this, we leverage MultiDiffusion (Bar-Tal et al., 2023), a training-free optimization that ensures consistency across different generation processes bound by a common set of constraints. Specifically, we use a weighted average of the latents from overlapping frames in each denoising step, facilitating the exchange of information across consecutive windows to enhance temporal consistency in the output.

3.1.6 Model Scaling and Training Efficiency

We describe the key details that allow us to scale and efficiently train the Movie Gen Video 30B parameter foundation model. In the following section, we will (1) outline hardware and infrastructure details, (2) compare and contrast our training setup to state-of-the-art LLMs (Touvron et al., 2023; Dubey et al., 2024), and (3) discuss model parallelism methods used for Movie Gen Video.

Infrastructure. We trained the media generation models using up to 6,144 H100 GPUs, each running at 700W TDP and with 80GB HBM3, using Meta's Grand Teton AI server platform (Baumgartner and Bowman, 2022). Within a server there are eight GPUs which are uniformly connected via NVSwitches. Across servers GPUs are connected via 400Gbps RoCE RDMA NICs. Training jobs are scheduled using MAST (Choudhury et al., 2024), Meta's global-scale training scheduler.

Comparison with Large Language Models. LLMs use structured causal attention masks to enforce token casuality, unlike the full bi-directional attention used in Movie Gen Video. This causal masking can be leveraged to provide approximately a 2× speedup compared to attention without the causal mask while also reducing peak memory requirements (Dao, 2024).

Secondly, state-of-the-art LLMs such as LLaMa3 (Dubey et al., 2024) use Grouped-Query Attention (GQA) instead of Multi-head Attention (MHA), which reduces the number of K-, V-heads and thus the total dimension of the key and value projections. This results in a reduction in FLOPs and tensor memory size while also improving memory bandwidth utilization. Furthermore, autoregressive LLMs gain additional inference time benefits through the use of GQA due to a reduction in their K, V-cache size. In part due to the non-autoregressive design of Movie Gen Video, we do not explore this architectural design choice and leave it for future work.

Similar to current LLMs like LLaMa3, our training is divided into stages of varying context lengths, where our context length varies depending on the spatial resolution (256 px or 768 px). For 768 px training this results in a context length of ∼73K tokens (768×768 px video with 256 frames, compressed $8 \times 8 \times 8$ through the TAE, and $2 \times 2 \times 1$ through patchification). But unlike LLMs which are trained at shorter context lengths for

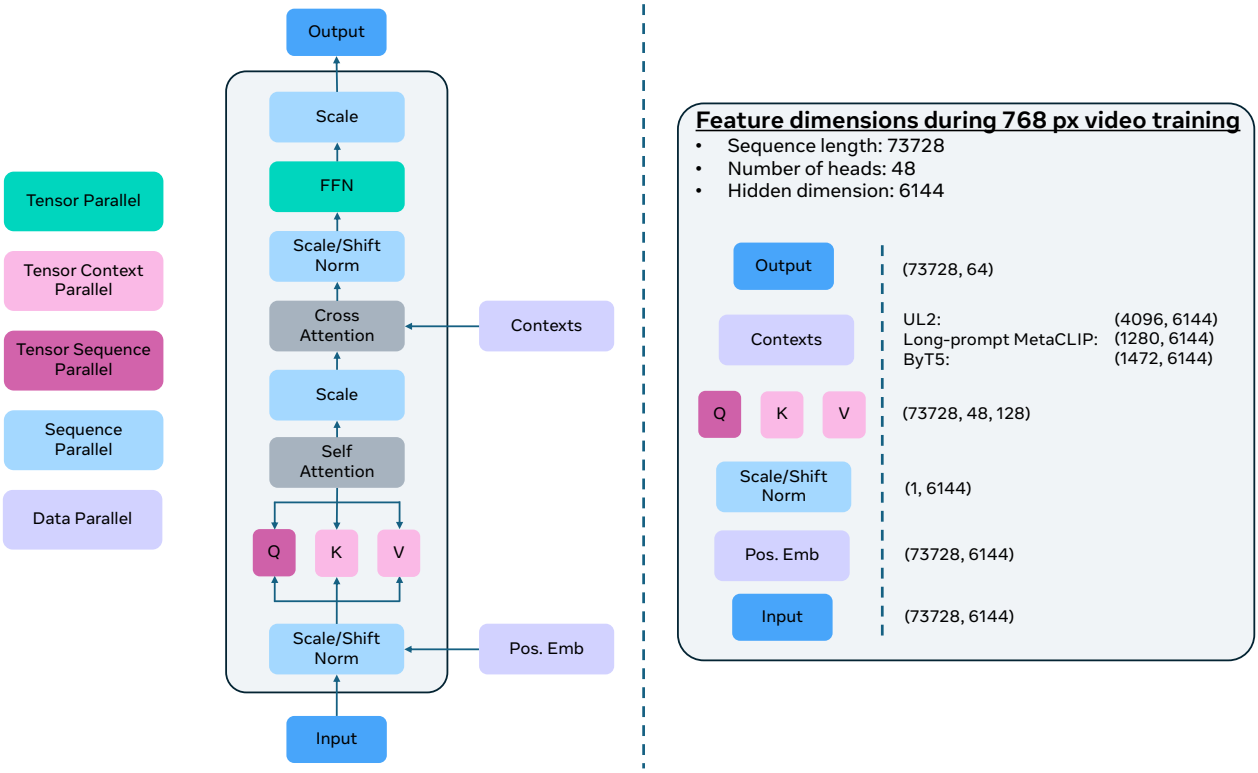


Figure 8 Movie Gen Video Transformer backbone and model parallelism applied. Left: we illustrate the Transformer backbone and color-code different model parallelizations used to shard our 30B model (described in Section 3.1.6). Right: Feature dimensions in a number of key steps during the most expensive stage of Movie Gen Video training, processing 768 px video inputs with a per-sample sequence length of 73K tokens.

the majority of the training budget, the majority of our training FLOPs are expended on long-context 768 px training (see Table 3). Due to the quadratic nature of self-attention, which is at the heart of a Transformer block, scaling to very large context lengths requires immense computation (FLOPs). This makes it even more important to optimize our training setup for long-context training.

Model Parallelism. Our large model size and extremely long-context lengths necessitate the use of multiple parallelisms for efficient training. We employ 3D parallelism to support model-level scaling across three axes: number of parameters, input tokens, and dataset size, while also allowing horizontal scale-out to more GPUs. We utilize a combination of fully sharded data parallelism (Rajbhandari et al., 2020; Ren et al., 2021; Zhao et al., 2023), tensor parallelism (Shoeybi et al., 2019; Narayanan et al., 2021), sequence parallelism (Li et al., 2021; Korthikanti et al., 2023), and context parallelism (Liu et al., 2023a; NVIDIA, 2024).

In the following, we describe different parallelisms and how they are utilized in different parts of our Transformer backbone (as illustrated in Figure 8).

- **Tensor-parallelism (TP)** shards the weights of linear layers either along columns or rows, and results in each GPU involved in the sharding performing *tp-size* less work (FLOPs) and generating *tp-size* less activations for column-parallel shards and consuming *tp-size* less activations for row-parallel shards. The cost of performing such a sharding is the addition of all-reduce communication overheads in both the forward (row-parallel) and backward (column-parallel) passes.
- **Sequence-parallelism (SP)** builds upon TP to also allow the sharding of the input over the sequence dimension *for layers which are replicated and in which each sequence element can be treated independently.* Such layers, *e.g.*, LayerNorm, would otherwise perform duplicate compute and generate identical (and thus replicated) activations across the TP-group.
- **Context-parallelism (CP)** enables a *partial* sharding over the sequence dimension for the *sequence-dependent*

softmax-attention operation. CP leverages the insight that for any given *(source (context), target (query))* sequences pair, *softmax-attention is only sequence-dependent over the context and not the query*. Therefore in the case of self-attention where the input source and target sequences are identical, CP allows the attention computation to be performed with only an all-gather for the K and V projections (instead of Q, K, and V) in the forward pass, and a reduce-scatter for their associated gradients in the backward.

Additionally, due to the separation of behavior between the Q and K, V projections, the performance of CP is variable not only in the context length, but also the size of the context dimension. A consequence of this is the differentiation of scaling performance and overhead characterstics for CP between Movie Gen Video and state-of-the-art LLMs, such as LLaMa3, which use GQA and thus generate smaller K, V tensors to be communicated (*e.g.*, $8\times$ smaller for LLaMa3 70B).

- **Fully Sharded Data Parallel (FSDP)** shards the model, optimizer and gradients across all data-parallel GPUs, synchronously gathering and scattering parameters and gradients throughout each training step.

Overlapping Communication and Computation. While parallelism techniques can enable training large sequence Transformer models by partitioning FLOP and memory demands across GPUs, their direct implementation can introduce overheads and inefficiencies. We build an analytical framework to model compute and communication times that allows us identify duplicated activations that require inter-GPU communication, and thus design a highly optimized model parallel solution. Our new custom implementation of model parallelization, written in PyTorch and compiled into CUDAGraphs, achieves strong activation memory scaling and minimizes exposed communication time. We provide more details on our optimized training setup in Appendix A.2.

3.2 Pre-training

3.2.1 Pre-training Data

Our pre-training dataset consists of $\mathcal{O}(100)$M video-text pairs and $\mathcal{O}(1)$B image-text pairs. We follow the pre-training data curation strategy similar to (Dai et al., 2023) for image-text data curation and focus on video data curation in this section.

Our original pool of data consists of videos that are 4 seconds to 2 minutes long, spanning concepts from different domains such as humans, nature, animals, and objects. Our data curation pipeline yields our final pre-training set of clip-prompt pairs, where each clip is 4s – 16s long, with single-shot camera and non-trivial motion. Our data curation pipeline is illustrated in Figure 9. It consists of three filtering stages: 1) visual filtering, 2) motion filtering, and 3) content filtering, and one captioning stage. The filtered clips are annotated with detailed generated captions containing 100 words on average. We describe each stage in detail below.

Visual filtering. We apply a series of 6 filters to remove videos with low visual quality. We remove videos of size smaller than a minimum width or height of 720 px. We filter on aspect ratio to achieve a mix of 60% landscape and 40% portrait videos. We prefer landscape videos over portrait videos due to their longer duration, better aesthetics, and stable motion. We use a video OCR model to remove videos with excessive text. We also perform scene boundary detection using FFmpeg (FFmpeg Developers) to extract 4 to 16 second long clips from these videos. We then train simple visual models to obtain prediction signals for filtering based on frame-level visual aesthetic, visual quality, large borders, and visual effects. Following Panda-70M (Chen

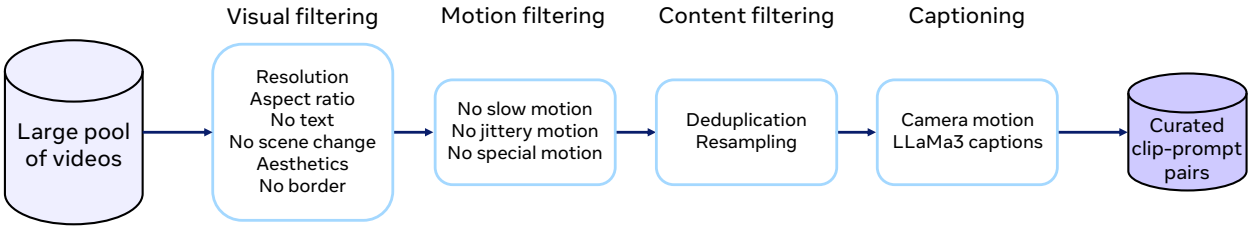


Figure 9 Movie Gen Video pre-training data curation pipeline. We filter a large pool of videos through multiple stages to produce a small set of high-quality training clips with associated prompts.

Duration	FPS	#video frames	#latent frames
10.67s	24	256	32
16s	16	256	32
12s - 16s	21 - 16	256	32
8s - 12s	24 - 16	192	24
4s - 8s	32 - 16	128	16

Table 2 Duration buckets for the T2V pre-training datasets. We split our training videos into five buckets based on their duration and FPS. Videos in each bucket have the same number of latent frames which allows for easy batching of training data. The buckets for the last three rows are based on the original duration of the video clips. The buckets for the first two rows are created based on middle clips sampled from videos of 10.67-12 seconds and $\geq$ 16 seconds respectively.

et al., 2024), we remove the first few seconds of clips whose start coincides with the beginning of a video, as the beginning of a video usually contains unstable camera movement or transition effects.

Motion filtering. We follow prior work (Girdhar et al., 2024) to automatically filter out low motion videos. First, we used an internal static video detection model to remove videos with no motion. Next, we identified videos with 'reasonable' motion based on their VMAF motion scores and motion vectors (FFmpeg Developers). To remove videos with frequent jittery camera movements, we used Shot Boundary Detection from the PySceneDetect (PySceneDetect Developers) library. Finally, we removed videos with special motion effects, *e.g.*, slideshow videos.

Content filtering. To ensure diversity in our pre-training set, we remove perceptually duplicate clips in our pre-training set using similarity in a copy-detection embedding (Pizzi et al., 2022) space. We also reduce the prevalence of dominant concepts by resampling to create our training set. We cluster semantic embeddings from a video-text joint embedding model to identify fine grained concept clusters. Next, we merge duplicate clusters and sample clips from each merged cluster according to the inverse square root of the cluster size (Mahajan et al., 2018).

Captioning. We create accurate and detailed text prompts for the video clips by using the LLaMa3-Video (Dubey et al., 2024) model. We finetune the 8B and 70B variants of the model for the video captioning task and use these models to caption the entire training set of video clips. Our training set consists of 70% 8B captions and 30% 70B captions. To enable cinematic camera motion control, we train a camera motion classifier which predicts one of 16 classes of camera motion *e.g.*, zoom-out, pan-left, *etc.* (see Appendix B.2 for more details). We prefix high-confidence camera motion predictions to the previously generated text captions. At inference, this allows a user to specify explicit camera control for video generation.

Multi-stage data curation. We curate 3 subsets of pre-training data with progressively stricter visual, motion, and content thresholds to meet the needs of different stages of pre-training. First, we curated a set of video clips with a minimum width and height of 720 px for low-resolution training. Next, we filtered the set to provide videos with a minimum width and height of 768 px for high-resolution training. Finally, we curated new videos augmenting our high-resolution training set. Our high resolution set has 80% landscape and 20% portrait videos, with at least 60% of them containing humans. During curation, we established a taxonomy of 600 human verbs and expressions and performed zero-shot text-to-video retrieval using the taxonomy to select videos with humans in them. We preserved the frequency of these human videos during content resampling. See Appendix B.1 for details on thresholds for curation of these videos.

Bucketization for variable duration and size. To accommodate diverse video lengths and aspect ratios, we bucketize the training data according to aspect ratio and length. The videos in each bucket lead to the exact same latent shape which allows for easy batching of training data. We use five aspect ratio buckets for both image and video datasets. Thus, our model can generate images and videos of different aspect ratios, *e.g.*, 1024×576 for landscape and 576×1024 for portrait. We define five duration buckets (4s – 16s) and adjust the number of latent frames according to video length (see Table 2). As described in Section 3.1.4, we introduce FPS control by adding an FPS token to the text caption, allowing us to sample videos at different frame rates (16 – 32 FPS).

Training stage	Dataset	TP	CP	bs/GPU	#GPUs	global bs	learning rate	#iters	#seen samples
256 px T2I	$\mathcal{O}$(1)B images	1	1	6	1536	9216	1e-4	210k	1.94B
256 px T2I/V	$\mathcal{O}$(100)M clips	4	1	2	3072	1536	6e-5	123k	173.6M
		4	1	2	6144	3072	6e-5	72k	221.2M
	Total							**185k**	**394.8M**
768 px T2I/V	$\mathcal{O}$(100)M clips	4	1	1	6144	1536	6e-5	19.6k	30.1M
		4	1	1	6144	1536	3e-5	11k	16.9M
		4	2	1	6144	768	2e-5	15.9k	12.2M
		4	2	1	4096	512	1e-5	28k	14.6M
	Total							**74.5k**	**73.8M**

Table 3 Progressive recipe and datasets for T2I/V pre-training. Note that: (1) besides the listed video datasets, the same image dataset used for T2I training is also used in T2I/V training with a ratio of 1:10 over video data, and (2) the video datasets of difference sources are sampled with ratios respecting to the volume of the dataset. (3) global bs = (bs/GPU * #GPUs) / (TP * CP)

3.2.2 Training

We describe the training details for our 30B parameter model. To improve training efficiency and model scalability, we employ a multi-stage training procedure, similar to (Girdhar et al., 2024). This procedure consists of three main steps:

- Initial training on the text-to-image (T2I) task, followed by joint training on both text-to-image and text-to-video (T2V) tasks;
- Progressive resolution scaling from low-resolution 256 px data to high-resolution 768 px data;
- Continuous training using improved datasets and optimized training recipes while working with compute and time restrictions.

The training recipe is summarized in Table 3. We maintain a validation set of unseen videos and monitored the validation loss throughout training. We observed that the validation loss for our model correlated well with visual quality judged by human evaluations.

Text-to-Image Warm-up Training. Jointly training T2I/V models is significantly slower and more memory-intensive than training T2I models alone, primarily due to the substantially longer latent token length (*e.g.*, 32×). Furthermore, we observed that directly training T2I/V models from scratch results in a slower convergence speed than initializing them from a T2I model. For instance, after training for the same number of GPU hours, we noticed significantly worse visual and temporal quality for both T2I and T2V tasks compared to our proposed multi-stage training approach. To address this, we begin with a T2I-only warm-up training stage. Rather than training at the target 768 px resolution, we train this stage at a lower resolution (256 px) which allows us to train with a larger batch size and on more training data for the same training compute.

T2I/V Joint Training. After the T2I warmup training, we train the model jointly for text-to-image and text-to-video. To enable the joint training, we double the spatial positional embedding (PE) layers to accommodate various aspect ratios, add new temporal PE layers to support up to 32 latent frames, and initialize spatial PE layers from the T2I model with 2× expansion. We first use 256 px resolution images and videos for the T2I/V joint training. For the 768 px stage, we expand the spatial PE layers by 3×. Table 3 summarizes the training recipe.

- **256 px T2I/V stage.** We use a large batch size of 1536 samples and a larger learning rate $6e^{-5}$ that results in stable training. After 123k iterations, we double the number of GPUs, yielding the 2× bigger global batch size and a significant drop in the validation loss. We stop the training at 185k iterations after 395M (4+ epochs) video samples.
- **768 px T2I/V stage.** We observe that the validation loss decreases quickly in the first 10k iterations and then fluctuates, see Figure 15. We reduce the learning rate by half at 19.6k iterations which further

reduces the loss. We continue to train the model and decrease the learning rate whenever the validation loss plateaus.

3.3 Finetuning

As in prior work (Dai et al., 2023; Girdhar et al., 2024), we improve the motion and aesthetic quality of the generated videos by finetuning the pre-trained model on a small finetuning set of selected videos. The finetuning set videos and captions are manually curated, and thus we term this stage as supervised finetuning. During this stage, we train multiple models and combine them to form the final model through a model averaging approach. While our model can generate high quality images, we find that post-training specifically for images results in a significant boost in quality. We describe the image-specific post-training recipe in Section 3.7 and describe the video-specific post-training recipe next.

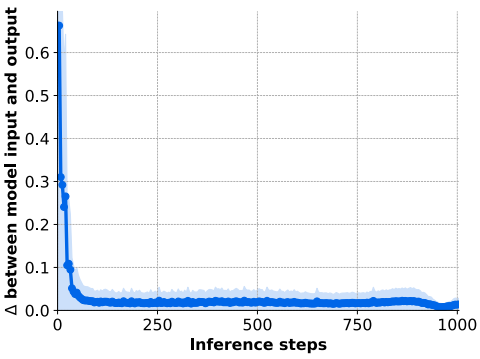
Finetuning Video Data. We aim to collect a finetuning set of high quality videos with good motion, realness, aesthetics, with a wide range of concepts, and with high quality captions. For finding such videos, we start with a large pool of videos and apply both automated and manual filtering steps (taking motivation from the curation recipe from (Dai et al., 2023)). There are four key stages that are run in sequence, each operating on the output of the previous stage: (1) Establishing a set of candidate videos. Here, we use automated filters that set strict thresholds on aesthetics, motion, scene change. Additionally, we remove videos with small subjects using an object detection model (Zhou et al., 2022) on frames. This stage results in a few million videos but with an unbalanced distribution of concepts. (2) Balancing the concepts in set of videos. The goal of this stage is to obtain a small enough subset of concept-balanced videos such that each can be manually filtered in the following steps. We used our taxonomy of human verbs and expressions, defined in Section 3.2.1, to perform text k-NN methods to retrieve videos for each concept from the candidate pool of videos. We manually picked a few visually appealing seed videos per concept and performed video k-NN to get a concept-balanced subset of videos. For k-NN, we used the video and text embeddings from a video-text joint embedding model. (3) Manually identifying cinematic videos. Many aspects to high quality finetuning data cannot be reliably captured by automated filters with high precision and recall. At this stage, we instead rely on manual filtering. Here we ensure that the remaining videos have angled (natural sunshine or studio) lighting, vivid (but not over-saturated) colors, no clutter, non-trivial motion, no camera shake, and no edited effects or overlay text. During this stage, annotators additionally clip videos to the desired duration that will be trained on by selecting the best, most compelling clip of the video. (4) Manually captioning the videos. In detail, human annotators refine LLaMa3-Video generated captions by fixing incorrect details and ensuring the inclusion of certain key video details. These include camera control, human expressions, subject and background info, detailed motion description and lighting information. At this stage humans annotate six additional camera motion and position types (see Appendix B.2). Our video finetuning data is set to have duration between 10.6s and 16s. In practice, 50% of videos are 16s long, while the rest of 50% videos are between 10.6s to 16s.

Supervised Finetuning Recipe. In video supervised finetuning (SFT), we use the same model architecture as the pre-training stage, and finetune the model with the pre-training checkpoints as initialization. Different from pre-training that uses large-scale data, large batch sizes and training resources, we instead use relatively a small batch size and 64 nodes (512 H100 GPUs) to train the model, and use a cosine learning rate scheduler (Loshchilov and Hutter, 2017). Similar to the pre-training stage, for videos that are at 16s, we train with 16 FPS, and for videos that are between 10.6s to 16s, we train with 24 FPS. As a result, our model is trained to best support the generation of videos in both 10s and 16s.

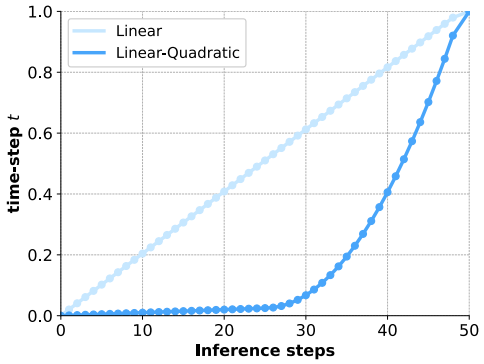
Model Averaging. Our experiments reveal that the choice of different sets of finetune data, hyperparameters as well as pre-train checkpoints significantly affects key aspects of the model's behavior, including motion, consistency, and camera control. To harness the diverse strengths of these models, we employ a model averaging approach. Similar to LLaMa3 (Dubey et al., 2024), we average models obtained from SFT experiments that use various versions of finetune data, hyperparameters and pre-train checkpoints.

3.4 Inference

In this section, we describe the different hyper-parameters and settings used for sampling from Movie Gen Video. For comparisons to prior work, we use a text classifier-free guidance scale of 7.5, and we use the



(a) Average change in model outputs between inference steps



(b) The linear-quadratic t-schedule

Figure 10 The linear-quadratic t-schedule. (a) Average change in transformer block input *vs*. output across inference steps, using a 1000-step linear t-schedule. (b) A 50-step linear and linear-quadratic t-schedule. Since the biggest changes in the model block inputs/outputs occur at early time-steps, our scheduler copies the first 25 low time-steps and bridges the remaining time-steps with 25 quadratic steps.

linear-quadratic sampler described in Section 3.4.2 with 50 steps (emulating 250 linear steps). We also use an inference prompt rewrite on the input text prompt, as described below.

3.4.1 Inference Prompt Rewrite

As mentioned in Section 3.2.1, we train the model with high quality video/image-text pairs, and these training captions are characterized by their dense details and consistent paragraph structure. However, the writing style and length of prompts in the inference stage vary widely. For instance, most users typically type less than 10 words, which is shorter than the average length of training captions. To bridge the distribution gap between training captions and inference prompts, we utilize LLaMa3 (Dubey et al., 2024) to transform the original input prompts into more detailed ones. The key details of the inference prompt rewrite model are:

- We employ a standardized information architecture to rephrase the prompts, ensuring consistency in the visual composition.
- We refine the rewritten prompts by replacing complex vocabulary with more accessible and straightforward terminology, thereby enhancing their clarity and comprehensibility.
- We observe that excessively elaborate descriptions of motion details can result in the introduction of artifacts in the generated videos, highlighting the importance of striking a balance between descriptive richness and visual fidelity.

Efficient Inference Rewrite Model. To improve the computation efficiency of the inference rewrite model, we developed a teacher-student distillation approach for this purpose. Initially, we built a prompt rewrite teacher model based on the LLaMa3 70B model, using detailed prompt instructions and in-context learning examples from the foundation model training set. We then gathered human-in-the-loop (HITL) finetuning data. This was achieved by using the LLaMa3 70B prompt rewrite model as the teacher to conduct inference on a large prompt pool, and selecting high-quality rewrite pairs through human evaluations following the quality guideline. Finally, we finetuned a 8B LLaMa3 model on the HITL prompt rewrite pairs to obtain the final prompt rewrite model to reduce the latency burden to the whole system.

3.4.2 Improving Inference Efficiency

To sample videos efficiently, we use the Euler sampler with a unique t-schedule tailored to our model. Empirically, we found that Euler outperforms higher-order solvers like midpoint (Atkinson, 1991) or adaptive solvers like Dopri5 (Dormand and Prince, 1980). We observed that reducing the number of inference steps for video generation is more challenging than for image generation due to the additional time dimension, *i.e.*, the

quality and prompt alignment of the generated motion are more sensitive to the number of inference steps compared to static images. For example, videos generated with 250, 500, or 1000 linear steps show noticeable differences in scene composition and motion quality. While techniques such as distillation (Salimans and Ho, 2022; Kohler et al., 2024) can be used to speed up the model inference, they require additional training. Next, we show a simple inference-only technique that can lead up to $\sim 20\times$ speed up with a few lines of code.

We found that we can closely approximate the quality of an N-step video generation process with merely 50 steps by implementing a linear-quadratic t-schedule. This approach follows the first 25 steps of an N-step linear schedule and then approximates the remaining $N - 25$ steps with 25 quadratically placed steps. For example, a video generated with 1000 linear steps can be precisely emulated by 25 linear steps followed by 25 quadratic steps, whereby the linear steps are identical to the first 25 linear steps of a 1000-step linear schedule. The linear-quadratic strategy is predicated on the observation that the first inference steps are pivotal in setting up the scene and motion of the video. This is visualized in Figure 10, where we plot the average change between the input and output of each transformer block at every inference step. Similar behavior is observed in the diffusion-based fast video model PAB (Zhao et al., 2024), where the average per-step difference of attention blocks follows a U-shaped pattern, compared to the L-shaped curve in Figure 10. Since most changes occur in the first solver steps, taking over the first linear steps of an N-step schedule followed by much bigger steps is enough to approximate the full N-step result. The quadratic spacing of the latter steps is crucial, as it emphasizes the importance of the early stages in the flow-matching sequence. In practice, we use a 50-step linear-quadratic schedule emulating $N = 250$ linear steps for optimal results.

3.5 Evaluation

In this section, we explain how we evaluate the text-to-video quality of Movie Gen Video and other models. Our goal is to establish clear and effective evaluation metrics that identify a model's weaknesses and provide reliable feedback. We explain the different text-to-video evaluation axes and their design motivations in Section 3.5.1. We introduce our new benchmark, Movie Gen Video Bench, in Section 3.5.2. Throughout this work, we use human evaluation to assess the quality of generated videos across various evaluation axes. When evaluating each axis, we conduct pairwise A/B tests where expert human evaluators assess two videos side-by-side. Evaluators are instructed to choose a winner based on the axis being measured, or to declare a tie in case of no clear winner. We include a discussion on the motivation and reliability of using human evaluations and on existing automated metrics in Section 3.5.3.

3.5.1 Evaluation Axes

Evaluating text-to-video generation presents unique challenges compared to text-to-image tasks, primarily due to the added complexity of the temporal dimension. For a video to be considered high quality, it must stay faithful to the provided text prompt, maintain a high visual quality across frames without noticeable flaws, and be visually appealing with a photorealistic style. To assess these factors, we evaluate the quality of generated videos across three main axes: (1) Text-alignment, (2) Visual quality, and (3) realness and aesthetics. Each axis, along with their fine-grained sub-axes, is described in details below and summarized in Table 4.

Text-alignment. This axis measures how well a generated video aligns with the provided prompt. An input prompt can include wide-ranging descriptions of subject appearance, motion, background, camera motion, lighting and style, visual text, *etc.* Human evaluators are asked to pay close attention to these specific aspects and select the video that aligns more closely with the prompt. To provide more nuanced feedback, evaluators

Text-alignment	Visual quality			Realness & Aesthetics	
Subject & Motion alignment	Overall	Frame consistency	Motion Completeness	Motion Naturalness	Realness	Aesthetics

Table 4 Evaluation axes for text-to-video generation. We evaluate video generations across 3 axes, each of which is composed of multiple fine-grained sub-axes. Text-alignment evaluates the 'alignment' between the input text prompt and the video. Visual quality, Realness & Aesthetics evaluate the quality of the generated video independent of the input text prompt.

are also asked to specify their reasoning based on two orthogonal sub-axes: **Subject match**: This measures alignment of subject appearance, background, lighting and style; and **Motion match**: This measures the alignment of motion-related descriptions.

Visual Quality. Compared to visual quality in text-to-image generation, much of the perceived quality in generated videos stems from the quality of motion – a video-specific dimension. Therefore, in text-to-video visual quality evaluation, we focus on measuring the model's ability to generate consistent, natural, and sufficient amounts of motion in the output videos. To capture these critical aspects, we propose the following four sub-axes, which we outline below.

- **Frame consistency**: This metric assesses the temporal consistency of generated content. Violations of frame consistency can manifest as morphing-like artifacts, blurred or distorted objects, or content that abruptly appears or disappears. We consider frame consistency a crucial measure of the model's ability to understand object framing and relationships in motion, as inconsistencies or distortions often arise when the model fails to accurately represent interactions between objects or their environment. Additionally, frame consistency reflects the model's capacity to handle challenging tasks, such as prompts that require fast-moving content, *e.g.*, in sports scenarios, where maintaining consistent appearance is especially difficult; or reasoning about occlusions, *e.g.*, objects re-appearing after being occluded.
- **Motion completeness**: This measures whether the output video contains enough motion. A lack of motion completeness may occur when the prompt involves out-of-distribution or unusual subjects (*e.g.*, monsters, ghosts) or real-world objects performing unusual activities (*e.g.*, people flying, pandas playing piano). Due to limited training data for such scenarios, the model may struggle to generate enough amount of motion, resulting in either static videos or those with only camera movement. Motion completeness evaluates the magnitude of motion in the video. A win on this axis indicates a greater amount of motion, even if it includes distortion, fast motion, or appears unnatural.
- **Motion naturalness**: This metric assesses the model's ability to generate natural and realistic motion, demonstrating a solid understanding of real-world physics. It covers aspects such as natural limb movements, facial expressions, and adherence to physical laws. Motion that appears unnatural or uncanny will be penalized.
- **Overall quality**: For a given pair of videos being compared, the above three metrics might not result in the same winner. To resolve this, we introduced the overall quality sub-axis, where human evaluators are asked to pick the winning video that has better "overall" quality given the previous three sub-axes. This is a holistic metric that asks the human annotators to use their perception and to balance the previous signals to capture overall how good a generated video is.

Realness & Aesthetics. Realness and aesthetics evaluate the model's ability to generate photorealistic videos with aesthetically pleasing content, lighting, color, style, *etc*. We ask human evaluators to evaluate along two sub-axes:

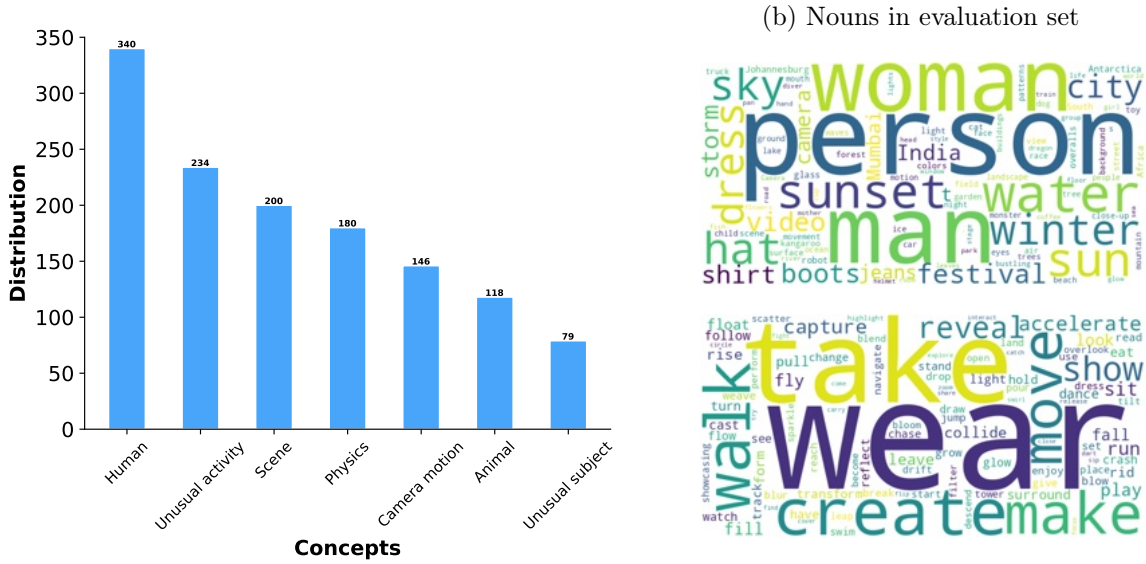
- **Realness**: This measures which of the videos being compared most closely resembles a real video. For *fantastical* prompts that are out of the training set distribution (*e.g.*, depicting fantasy creatures or surreal scenes), we define realness as mimicking a clip from a movie following a realistic art-style. We additionally ask the evaluators to select a reason behind their choice *i.e.*, "subject appearance being more realistic" or "motion being more realistic".
- **Aesthetics**: This measures which of the generated videos has more interesting and compelling content, lighting, color, and camera effects. Again, we ask the evaluators to provide details justifying their choice from "content being more appealing/interesting", and "lighting/color/style being more pleasing".

3.5.2 Evaluation benchmark

In order to thoroughly evaluate video generations, we propose and hope to release a benchmark, Movie Gen Video Bench, which consists of 1000 prompts that cover all the different testing aspects summarized above. Our benchmark is more than 3× larger than the prompt sets used in prior work (Singer et al., 2023; Girdhar et al., 2024) for evaluation. We specifically include prompts capturing the following concepts of interest: 1) human activity (limb and mouth motion, emotions, *etc*.), 2) animals, 3) nature and scenery, 4) physics (fluid

Prompt	Concept	Motion level
A marathon runner crossing the finish line after a grueling race.	human activity	high
A curious cat peering out from a cozy hiding spot.	animals	medium
A peaceful countryside with rolling hills and a setting sun.	nature & scenery	low
A high-speed video of champagne being poured into a glass, with bubbles rising rapidly.	physics – fluid dynamics	high
A cute golden dragon walking like a model on stage, as the audience claps for him.	unusual subjects	low

Table 5 Sample evaluation prompts from our proposed Movie Gen Video Bench benchmark. We sample prompts that cover a wide range of concepts, with varying motion levels.



(a) Prompt concept distribution

Figure 11 Summary visualization of evaluation prompts in Movie Gen Video Bench. (a) Since each prompt may encompass multiple testing concepts, we assign one or more concept tags to each prompt. As a result, the total count in the distribution exceeds 1000. (b), (c) Visualization of the common nouns and verbs in the evaluation set, respectively.

dynamics, gravity, acceleration, collisions, explosions, *etc.*), 5) unusual subjects and unusual activities. To test the generation quality at different motion levels, we also tag each prompt with high/medium/low motion. We show examples of evaluation prompts used in Table 5 and show the distribution of evaluation prompts across concepts in Figure 11.

We evaluate the model quality on the entire evaluation prompt set as well as break down the quality by individual testing metrics. Prompts involving unusual subjects and unusual motion help test model's ability to generalize to out-of-distribution content.

3.5.3 Evaluation Discussion

Here, we motivate our decision for using human evaluation as opposed to automated metrics.

Necessity of human evaluations for video generation. The motivation for choosing human evaluation stems from the complexity of evaluating video generation. For Text-alignment, evaluating the alignment of motion over time requires understanding how actions unfold and evolve in relation to the prompt. Humans are particularly skilled at recognizing temporal coherence and handling ambiguities when the context is abstract or complex, whereas automated methods may only capture static frame-level correspondences. When evaluating visual quality, such as motion naturalness or detecting inconsistencies in object appearance across frames, humans excel due to their innate understanding of real-world physics and object behavior. Similarly, assessing realness and aesthetics heavily depends on human perception and preference. Across all three tasks, we find that existing automated metrics struggle to provide reliable results, reinforcing the need for human evaluation.

Reliability. An important aspect concerning reliability in evaluations is the randomness introduced both on the modeling side due to the probabilistic nature of the generative models, and the human evaluation side due to the annotation variance. Defining objective criteria to measure generations remains challenging and humans can still be influenced by other factors such as personal biases or preferences. We describe our efforts to reduce evaluation variance and increase the reliability of the human evaluations. We take four key steps towards minimizing evaluation variance: (1) We provide human evaluators with detailed evaluation guidelines and video examples, narrowing the definitions of evaluation axes and sub-axes to minimize subjectivity. Additionally, inspired by the JUICE metric (Girdhar et al., 2024), we found that asking evaluators to indicate the reason of their choices helps reduce annotation variance and improve agreement among evaluators. (2) We evaluate the models over a large set of prompts (*e.g.*, 1000 for Movie Gen Video Bench, 3× larger than (Singer et al., 2023; Girdhar et al., 2024)) covering a wide variety of concepts. (3) We use a majority vote system, with a majority vote from three annotations for each Text-alignment and Visual quality question, and a majority vote from six annotations for realness and aesthetic questions, as these are more subjective. (4) We conduct thorough and frequent audits of human annotations to resolve edge cases and correct mislabelings.

Automated Metrics for text-to-video evaluation. Prior works in text-to-video generation have relied upon automated metrics for assessing video quality. Similar to recent studies (Dai et al., 2023; Podell et al., 2023; Singer et al., 2023; Girdhar et al., 2024; Ho et al., 2022a; Barratt and Sharma, 2018; Chong and Forsyth, 2020; Ge et al., 2024; Huang et al., 2024) we find that automated metrics such as FVD (Unterthiner et al., 2019) and IS (Salimans et al., 2016) do not correlate with human evaluation scores for video quality, and hence do not provide useful signal for model development or comparison. Some prior works utilize discriminative models for generated media evaluation axes, including text faithfulness with CLIP (Radford et al., 2021). One key limitation of such automated metrics is that they are inherently limited by the performance of the underlying discriminative model (Rambhatla and Misra, 2023). A key challenge in using discriminative models for evaluating text-to-video generation is the inavailability of sufficiently effective and expressive video-text discriminative models. We note that other interesting automated metrics for generated video evaluation exist, such as those based on structure-from-motion (Li et al., 2024), that we did not explore the use of here.

Enabling fair comparison to Movie Gen Video. To enable fair and easy comparison to Movie Gen Video for future works, we hope to publicly release our non cherry picked generated videos for the Movie Gen Video Bench prompt set.

3.6 Results

In this section, we describe the experiments and results for Movie Gen Video. We first include comparisons to prior work for text-to-video generation in Section 3.6.1. We ablate key design decisions for Movie Gen Video in Section 3.6.2. We include key results and ablations for the TAE in Section 3.6.3, and an evaluation of the Spatial Upsampler in Section 3.6.5. We include comparisons to prior work for text-to-image generation in Section 3.7.

3.6.1 Comparisons to Prior Work

Where possible, we obtain non-cherry picked generated videos from the Movie Gen Video Bench prompt set for the prior work methods, and compare to these using non-cherry picked videos from Movie Gen Video for the same prompts. This includes the black-box commercial models that offer API access through their website: Runway Gen3 (RunwayML, 2024), LumaLabs (LumaLabs, 2024), Kling1.5 (KlingAI, 2024). We also compare to closed source text-to-video methods (OpenAI Sora), where our only option is to compare to them using the prompts and videos from their publicly released examples. Note that the publicly released videos for closed source methods are likely to be 'best' representative samples obtained through cherry picking. Hence for fair comparison, we compare to OpenAI Sora by methodically manually choosing one video from five generated options from Movie Gen Video for each prompt. One additional challenge when comparing to prior work is that each method generates videos at different resolutions and aspect-ratios. We reduce annotator bias (Girdhar et al., 2024) by downsampling Movie Gen Video's videos for each comparison such that they match in these aspects. Full details on this postprocessing and the OpenAI Sora comparison can be found in Appendix C.2.

We compare Movie Gen Video to prior work for text-to-video generation on different evaluation axes

	Movie Gen Video net win rate *vs.* prior work				
	Runway Gen3	LumaLabs	OpenAI Sora	Kling1.5	σ
Overall Quality	35.02	60.58	8.23	3.87	±5.07
Consistency	33.1	42.14	8.22	13.5	±4.08
Motion Naturalness	19.27	29.33	4.43	0.52	±3.98
Motion Completeness	-1.72	23.59	8.86	-10.04	±1.68
Text-alignment	10.45	12.23	17.72	-1.99	±3.74
Realness	48.49	61.83	11.62	37.09	±2.52
Aesthetics	38.55	48.19	6.45	26.88	±4.84

Table 6 Movie Gen Video vs. prior work. The comparison uses either generated videos from the Movie Gen Video Bench prompt set (Runway Gen3, LumaLabs, Kling1.5) or prompts from publicly released videos on their website (OpenAI Sora). A detailed summary of information from prior work is shown in Table 40. We measure the net win rate (win% - loss% of our model) which has a range of $[-100\%, 100\%]$. To assess statistical significance, we perform an annotation variance analysis (Appendix C.1), with the net win standard deviation, σ, indicated in the table above. A significant win/loss is identified when the net win rate is beyond 2σ (95% CI), a moderate win/loss within 1–2 σ (68% CI), and performance is considered on par within 1σ.

described in Section 3.5. The results are shown in Table 6. We report the net win rate of our model, which can lie in the range $[-100, 100]$. On overall quality, Movie Gen Video strongly outperforms Runway Gen3 (35.02%) and LumaLabs with the net win rate beyond 2σ. Our generations moderately net win over OpenAI Sora (8.23%) (net win rate within 1-2 σ) and are on par with Kling1.5 (3.87%). Against Runway Gen3, LumaLabs and OpenAI Sora, we see that Movie Gen Video either outperforms or is on par across all quality breakdown axes, including large net wins against Runway Gen3 on motion naturalness (19.27%) and frame consistency (33.1%), against Sora on frame consistency (8.22%) and motion completeness (8.86%). These significant net wins demonstrate Movie Gen Video's ability to simulate the real world with generated videos that respect physics, with motion that is both reasonable in magnitude but consistent and without distortion. Against Kling1.5, we see that Movie Gen Video significantly net wins in frame consistency (13.5%) but loses on motion completeness (-10.04%). We note that this large motion completeness paired with poor frame consistency shows Kling1.5's tendency to occasionally generate unnaturally large motion with distortion. As indicated in Section 3.5.1, motion completeness only evaluates the magnitude of motion in the video, regardless of distortion, fast motion, or being unnatural.

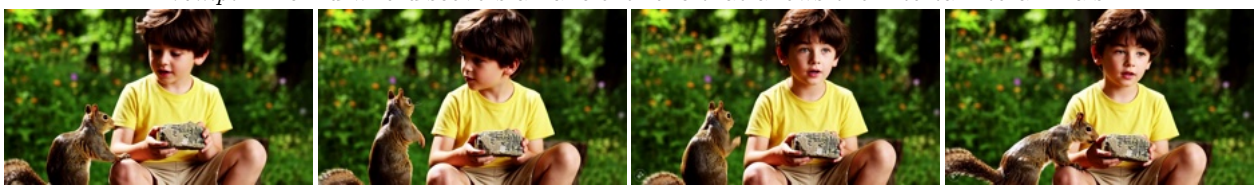
On realness and aesthetics, Movie Gen Video significantly outperforms Runway Gen3, LumaLabs and Kling1.5 on both metrics, with 48.49%, 61.83% and 37.09% net win rates on realness, respectively. Compared to OpenAI Sora, Movie Gen Video has a significant win on realness with 11.62% net win rate beyond 2σ and a moderate win over OpenAI Sora on aesthetics with 6.45% net win rate within 1–2 σ.

This demonstrates Movie Gen Video's ability to generate photorealistic and visually compelling content. For text faithfulness, Movie Gen Video outperforms OpenAI Sora, Runway Gen3, LumaLabs and is on par with Kling1.5.

Several generated videos from Movie Gen Video are shown in Figure 12. Movie Gen Video is able to generate high quality videos for both natural prompts (see Figure 12) and out-of-distribution prompts describing fantastical scenes from outside of the training set distribution (see Figure 1). The generated videos contain complex motion, depicting detailed content over the video's duration *e.g.*, a firefighter running into and then out of a burning forest or a puppy searching for, finding its owner, and continuing its quest (see Figure 12).

Qualitative comparisons between Movie Gen Video and prior work are shown in Figure 13 and Figure 14. As shown, Movie Gen Video generates realistic and high quality videos with natural-looking motion that is well aligned to the text prompt. Movie Gen Video generates objects and identities that are consistent over the entire duration of the video, and that obey the laws of physics. Differently, the prior work can struggle to generate videos that are simultaneously high quality and with good text-alignemnt.

Prompt: A child who discovers an ancient relic that allows them to talk to animals



Prompt: Firefighter running through a burning forest

Prompt: A lost puppy that leads its finder on an epic quest

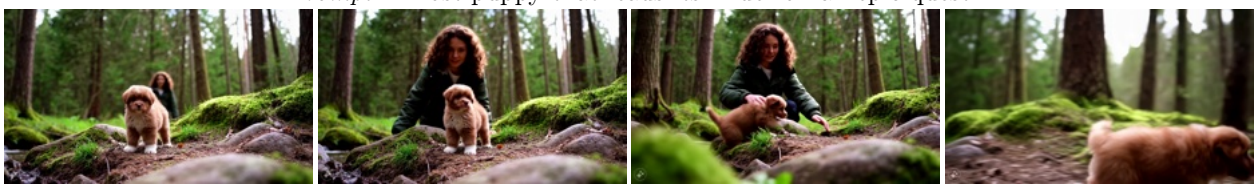
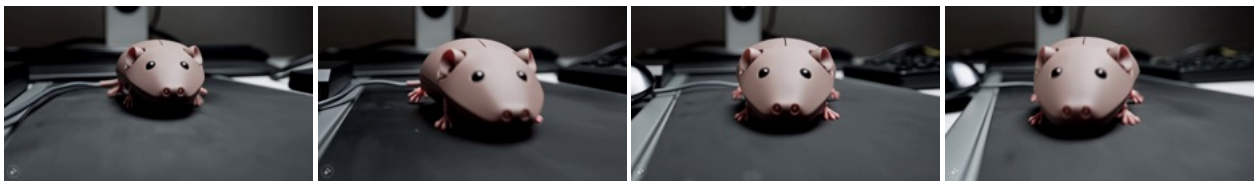


Figure 12 Generated videos from Movie Gen Video. Movie Gen Video generates high quality, visually compelling videos that are aligned to complex text prompts. Videos in this Figure found at https://go.fb.me/MovieGen-Figure12.

Prompt: A computer mouse with legs running on a treadmill

Movie Gen Video



Runway Gen3

LumaLabs

Kling1.5

Figure 13 Qualitative comparisons between Movie Gen Video and prior work. Here, we show generated videos for the same prompt for Movie Gen Video, Runway Gen3, LumaLabs, and Kling1.5. Unlike prior work, Movie Gen Video generates videos that are high quality, with natural, realistic motion, and that are aligned to the text prompt. Videos in this Figure found at https://go.fb.me/MovieGen-Figure13.

Prompt: a kangaroo in purple overalls and boots walking in Johannesburg during sunset

Movie Gen Video

OpenAI Sora

Prompt: a toy robot in a green dress and sun hat walking in Antarctica during a storm

Movie Gen Video

OpenAI Sora

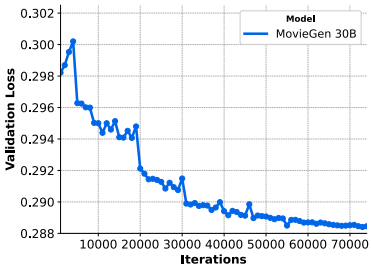
Figure 14 Qualitative comparisons between Movie Gen Video and prior work. Here, we show two generated videos for the same prompt for Movie Gen Video and OpenAI Sora. Movie Gen Video generates natural-looking videos with realistic motion, even for the out-of-training-set-distribution prompts shown here. As shown here, for such prompts OpenAI Sora can tend to generate less realistic videos (*e.g.*, the *cartoonish* kangaroo in the second row), that can be missing the motion details described in the text prompt (*e.g.*, the non-walking robot in the bottom row). Videos in this Figure found at https://go.fb.me/MovieGen-Figure14.

Correlation between validation loss and human evaluation. In Figure 15, we show the validation loss for Movie Gen Video as a function of pretraining steps and observe that it decreases smoothly. We take pre-trained checkpoints after every few thousand iterations and evaluate them in a pairwise comparison. We observe that the validation loss is well correlated with human evaluation results as the later checkpoints with lower validation loss perform better in the evaluations. This suggests that the Flow Matching validation loss can serve as a useful proxy for human evaluations during model development.

Effect of finetuning. We leverage supervised finetuning, described in Section 3.3 to further improve the video generation quality. In Table 7, we compare the evaluation metrics between pre-trained model and finetuned models at 24 FPS with 10.6s video duration. We find that finetuning leads to a significant improvement on both the Visual quality and Text-alignment metrics.

3.6.2 Ablations

Here, we ablate the critical design decisions for Movie Gen Video. For all ablations described in this section, we use a simpler, smaller baseline training and model setup than used for the main results. We analyze the effect of each design decision quantitatively via text-to-video human evaluation on a subset of Movie Gen Video Bench containing 381 prompts, termed Movie Gen Video Bench-Mini, and report results on text



Model A iterations	Model B iterations	Model A net win rate *vs.* model B: Overall Quality	Frame Consistency	Motion Completeness	Motion Naturalness	Text-alignment
15.6k	10.2k	16.0	9.0	4.67	7.66	5.67
19.2k	15.6k	13.66	-0.67	5.34	5.66	3.34
37k	26.4k	0.39	-4.3	0.78	-1.95	3.13
59k	54k	6.33	-1.33	8.34	3.0	8.0

Figure 15 Validation loss correlates with human evaluations. Left: We plot the validation loss for the 768 px stage T2I/V pre-training. We decrease the learning rate whenever the validation loss plateaus. Right: We observe that the validation loss is well correlated with human evaluation results of the corresponding checkpoints, especially for the text alignment and overall quality.

Finetune net win rate *vs.* pre-trained: Overall Quality	Consistency	Motion Completeness	Motion Naturalness	Text Alignment
34.65	8.14	18.38	10.5	9.97

Table 7 Comparison of finetuned and pre-trained models. We conduct A/B comparison between finetuned and pre-trained models, where the scores are winning rates of finetune model minus the winning rate of the pretrained model, which shows that finetuning significantly improves over the pretrained model. The videos are evaluated at 24 FPS with 10.6s video duration.

Method	Q	A
FM *vs.* Diffusion	16.53	7.08

(a)

Method	Q	A
Video *vs.* Image caption	−0.80	10.80

(b)

Method	Q	A
LLaMa3-like *vs.* DiT	18.63	12.60

(c)

Table 8 Key design decisions in Movie Gen Video. Each table shows the net win rate, in terms of the overall Quality (Q) and Text-alignment (A), on adopting a design decision *vs.* a model that does not have it. See Section 3.6.2 for details.

faithfulness and overall quality (see Section 3.5). For each ablation, every aspect of the model except for the design decision being tested is held constant for fair comparison. Next, we describe the simpler baseline setup followed by each ablation result. Unless described otherwise here, all other settings for the ablation experiments follow our 30B model including text encoders, flow matching objective, image training set, *etc.*

Baseline model setup for ablations. We use a 5B parameter version of Movie Gen Video of trained to produce 352×192 videos of $4 - 8$s. We use the TAE described in Section 3.1.1, which does $8\times$ compression across every spatio-temporal dimension to produce latents of shape $16 \times 24 \times 44$. This smaller Movie Gen Video model has 32 layers in the transformer with 3072 embedding dimension and 24 heads.

Baseline training setup for ablations. We use a two-stage training pipeline: (1) text-to-image pretraining; (2) text-to-image and text-to-video joint training. For simplicity, we used a smaller dataset of 21M videos, captioned with LLaMa3-Video 8B, that have a constant landscape aspect ratio for video training. First, we train the model on the image dataset with a learning rate of 0.0003, a global batch size of 9216 on 512 GPUs for 96K iterations. Next, we perform joint text-to-image and text-to-video training with an iteration ratio of $0.02 : 1$ where the global batch size is 4096 for images and 256 for videos. We use a learning rate of 5e-5 and train for 100K iterations.

Ablation Result - Training objective. We compare the Flow Matching training objective to the diffusion training objective. Following (Girdhar et al., 2024), we use the v-pred and zero terminal-SNR formulation of diffusion for training which is effective for video generation. As the human evaluation results in Table 8a show, Flow Matching leads to better generations both in terms of overall quality and text alignment while controlling for all other factors. Empirically, we also found this result to also hold across a range of model sizes and thus use Flow Matching to train our models.

	Normalization	Normalization eps	Normalization Affine	Activation Function	Bias in FC layers
Movie Gen	RMSNorm	1e-5	True	SwiGLU	No
DiT	LayerNorm	1e-6	False	SiLU	Yes

Table 9 Architecture differences between Movie Gen's LLaMa3 architecture and DiT. All other hyperparameters remain equal for the architecture ablation.

Ablation Result - Effect of video captions. As described in 3.2.1, our video generation model is trained using clips from real videos and LLaMa3-Video generated video clip captions. To assess the importance of video captions, we compare our LLaMa3-Video 8B video captioning model to an image-based captioning scheme also based on LLaMa. This image-based captioning model captions the first, middle, and last frame of the video clip and then uses LLaMa to rewrite these three image-based captions into a single video caption. We refer to this model as LLaMa3-FramesRewrite. We first compare the quality of the two captioning schemes with human evaluations based A/B testing. Human raters are asked to pick between two given captions for the same clip. LLaMa3-Video generated captions are preferred 67% of the time while LLaMa3-FramesRewrite captions are only preferred 15% of the time. We visually observe that the video captioning model is able to accurately describe more fine-grained details regarding movements in the video. These fine-grained details provide a stronger supervision signal for training the video generation model, significantly improving overall prompt alignment by 10.8% (Table 8), with most of the increase coming from motion alignment (+10.7%) particularly on prompts that require ask for a high degree of motion in the output video (+16.1%).

Ablation Result - Model architecture. In our work, we choose a transformer architecture based on LLaMa3 (see Section 3.1.3). We compare this to a Diffusion Transformer (Peebles and Xie, 2023) based model, which is commonly used in the media generation literature (Peebles and Xie, 2023; OpenAI, 2024; Ma et al., 2024a). The architecture differences between these two models can be seen in Table 9. As shown in Table 8c, we find that our LLaMa3 based architecture significantly outperforms the Diffusion Transformer on both quality (18.6%) and text-alignment (12.6%). This significant result shows that the LLaMa3 architecture has an advantage over the commonly used DiT for media generation. Our goal with Movie Gen is to scale to large model sizes, and to the best of our knowledge we find no detailed examples in the literature of scaling the Diffusion Transformer to very large scales. This result demonstrates that we can confidently transition from the commonly used Diffusion Transformer to architectures more commonly used in LLMs such as LLaMa3, the scaling behavior of which has been well documented (Touvron et al., 2023; Dubey et al., 2024).

3.6.3 TAE Results

We present here results and ablations from important design decisions for the temporal autoencoder (TAE). For evaluation, we report the reconstructed peak signal-to-noise ratio (PSNR), structural similarity (SSIM) (Wang et al., 2004), and Fréchet Inception distance (FID) (Heusel et al., 2017) of video clips split from the training set, with 2s, 4s, 6s, and 8s duration, each with 200 examples. We also measure the same metrics on a validation split of the image training set. For video reconstruction evaluation, metrics are averaged over video frames.

Qualitative Results. We show sample reconstructions from our TAE in Figure 16 with frames from the original video and the reconstruction after the TAE encoder and decoder. We observe that the TAE can reconstruct the video frames while preserving visual detail. The TAE reconstruction quality decreases for high frequency spatial details in images and video frames, and fast motion in videos. When both high frequency spatial details and large motion are present in a video, this can lead to a loss in detail, as can be seen in the examples in Figure 16, where fine details are smoothed out in the reconstruction.

Quantitative metrics. Table 10 compares our TAE against a baseline frame-wise autoencoder that does not perform any temporal compression. Our baseline also produces a 8 channel latent space as is standard for autoencoders used for frame-wise encoding in prior work (Blattmann et al., 2023a; Girdhar et al., 2024). On video data, we observe that the TAE achieves a competitive performance to the frame-wise encoder while achieving a 8× higher temporal compression. On images, the TAE outperforms the frame-wise model, an improvement that can be attributed to the increased channel size of the latent (8 *vs.* 16) (Dai et al., 2023).

Figure 16 Real (left) and TAE reconstructed (right) videos. The TAE compresses the video by a factor of 8× across each of the three spatiotemporal dimensions. We observe that the reconstructions from the TAE maintain visual detail present in the original videos.

	Video (512 px)			Image (512 px)		
	SSIM ↑	PSNR ↑	FID ↓	SSIM ↑	PSNR ↑	FID ↓
Frame-wise AE	0.9348	34.11	0.9352	0.8877	30.83	1.7588
TAE	0.9093	32.25	1.4872	0.9231	32.16	0.2873

Table 10 TAE reconstruction metrics comparison between an frame-wise autoencoder and our TAE model. We observe that the TAE achieves comparable performance to a frame-wise autoencoder for video reconstruction while achieving a 8× higher compression.

3.6.4 TAE Ablations

We now perform a series of ablation experiments for the design choices in training our TAE model.

Baseline setting for ablations. For simplicity, we use a TAE model with a smaller 4× compression ratio that produces a 8-channel latent space.

2.5D vs. 3D attention & convolutions. We compare using 2.5D, *i.e.*, 2D spatial attention/convolutions followed by 1D temporal attention/convolutions to using 3D spatiotemporal attention/convolutions in the TAE. In Table 11, we observe that the 3D spatiotemporal attention leads to slightly better reconstruction metrics. However, we found that this improvement was not large enough to justify the larger memory and compute costs associated with a fully 3D model compared to a 2.5D model. Thus, we use 2.5D for our TAE.

	Video (256 px)		
	SSIM ↑	PSNR ↑	FID ↓
TAE 2.5D	0.845	28.51	3.678
TAE 3D	0.852	28.80	4.715

Table 11 Comparing TAE models with 2.5D vs. 3D convolutions and attentions. We observe that while the 3D models perform slightly better than the 2.5D models, the gap in performance between the two models is small.

Effect of outlier penalty loss. We ablate the effect of adding the outlier penalty loss (OPL) from Section 3.1.1. The addition of this loss removes the artifacts from generated and reconstructed videos as seen in Figure 5, and improves reconstruction performance. We first train a baseline model without OPL for 50K iterations. We then finetune this model with OPL for 10K iterations and compare it to a baseline finetuned without OPL for 20K iterations. The results, summarized in Table 12, suggest that OPL finetuning improves the reconstruction for both images and videos.

3.6.5 Spatial Upsampler Results

Here, we include some results from the spatial upsampler described in Section 3.1.5. A visual comparison of the upsampling process is presented in Figure 17, which shows the 200 px and 400 px crops before and after

	Video (512 px)			Image (512 px)		
	SSIM ↑	PSNR ↑	FID ↓	SSIM ↑	PSNR ↑	FID ↓
TAE w/o OPL finetune	0.897	31.11	1.389	0.845	28.25	0.568
TAE w/ OPL finetune	0.910	31.93	1.241	0.862	29.26	0.614

Table 12 Effect of OPL on Temporal Autoencoder reconstructions. We evaluate the effect of OPL finetuning on the reconstruction quality and observe that OPL improves both image and video metrics.

Figure 17 Qualitative visualization of Spatial Upsampler: A visual comparison of the 200 px\400 px crops before\after upsampling. The Spatial Upsampler improves high frequency details in the output.

upsampling. The results demonstrate that the upsampler effectively sharpens and enhances visual details, producing a more refined and detailed output.

3.7 Text-to-Image Generation

The Movie Gen model is trained jointly on videos and images, and hence is able to generate both videos and images. To further validate the model's image generation capabilities, we continued training it with an image autoencoder and compared its performance to prior work in image generation. The following sections provide detailed experimental settings and evaluation results.

3.7.1 Method

For the Text-to-Image model, our goal is to generate realistic images. We utilize the Movie Gen model as an initialization and replace the TAE with an image autoencoder. We then train the model on the text-to-image generation task, allowing it to generate images based on text descriptions. The final resolution is 1024 px. For post-training, we curated a total of $\mathcal{O}(1000)$ images created by in-house artists for quality-tuning, following the approach outlined in (Dai et al., 2023). We finetuned the model for 6k steps with a learning rate of 0.00001 and a batch size of 64. We used a constant learning rate scheduler with 2000 warm-up steps.

3.7.2 Results

To measure the quality of our Text-to-Image generation results, we use human evaluators to evaluate the following axes: (a) text faithfulness, and (b) visual quality. For evaluating text faithfulness, we use a pairwise A/B comparison set up where evaluators select which image aligns better with a given generation prompt. Evaluators are asked to choose which of the choices A or B, is better, or equal, in terms of text alignment. For visual quality, we use a a similar pairwise A/B comparison set up and ask raters to help select the image that looks more realistic. Evaluators are asked to look for flaws in the generation, such as errors in the number of fingers or arms, or visual text spelling errors, before making their decision. For creating the benchmarking prompts, we analyzed typical text-to-image user prompts and generated categories and distributions, and leverage LLMs to produce user prompts that mimic real users prompts.

We compare with the best contemporary Text-to-Image models including Flux.1 (Black Forest Labs, 2024), OpenAI Dall-E 3 (OpenAI, 2024), Midjourney V6.1 (Midjourney, 2024), and Ideogram V2 (Ideogram, 2024) available at time of benchmark These are however black-box commercial solutions, which makes fair comparison

a challenge. Similar to Text-to-Video evaluation, we obtain non-cherry picked generated images from the benchmark prompts for the prior work methods, and compare to them using non-cherry picked images from Movie Gen for the same prompts. To ensure consistent comparison across all models and evaluation axes, we utilize the ELO rating system to establish rankings based on battle records converted from raw human evaluation results. For A/B comparison evaluations, the "win/tie/lose" on a given prompt between two models were directly interpreted as one battle record. This approach allowed us to combine ratings on all evaluation axes to generate an overall performance. The comparison results are summarized in Figure 18, where we see that our model achieves the highest ELO rating compared to all recent state-of-the-art text-to-image methods available at the time of benchmarking. In Figure 19, we show some qualitative results of our generations.

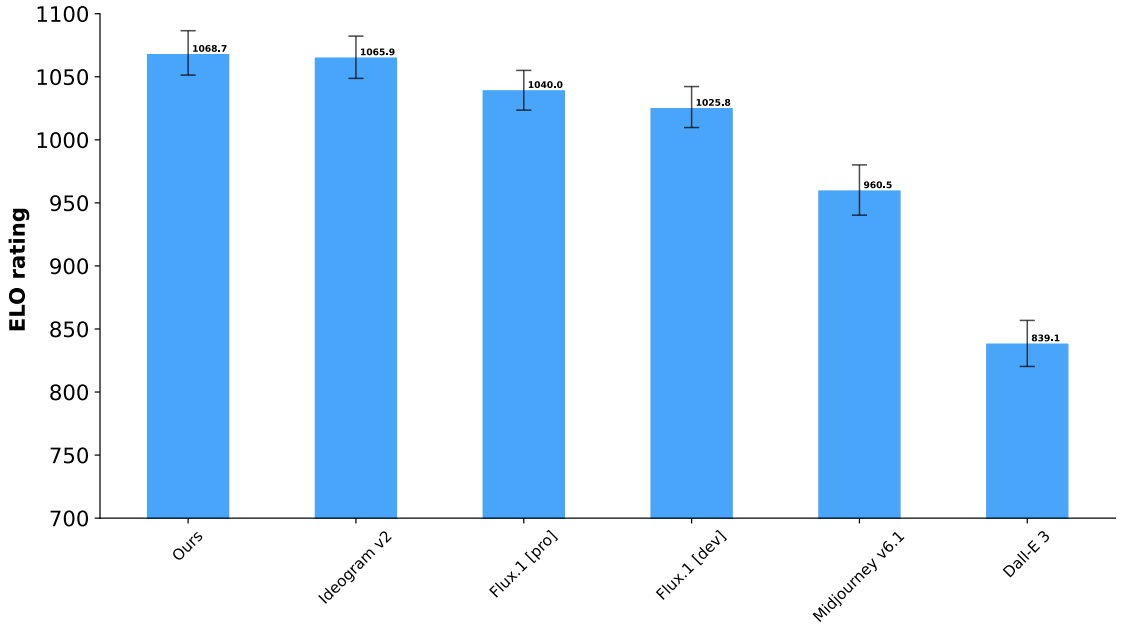


Figure 18 ELO rating comparison of text-to-image methods. We compare our image generation model to state-of-the-art text-to-image models and observe that it performs competitively with recent approaches.

Figure 19 Example T2I Images from Movie Gen. Generation prompts from left to right: (1) A focused writer, with a trusty pencil by their side, sits in a modern, minimalist home office with a large window providing ample natural light. (2) A hiker wearing a backpack walks along a trail with a rugby ball in hand, looking up at a soaring airplane overhead, the sun shining down on a beautiful day. (3) A styled hair with a headband inspired a graphic designer using a pen tablet and stylus to create vibrant digital art in a cubism style. (4) A vivid landscape painting on an easel stands in a bright, natural light-filled studio, showcasing its vibrant colors and textures, while a painter works

4 Video Personalization

Generating personalized high quality videos that accurately capture an individual's identity is an important research area with significant practical applications. We integrate personalization into video generation, yielding state-of-the-art outcomes as detailed in this section. We describe our novel model architecture in Section 4.1 followed by the training recipe in Section 4.2.1 and Section 4.3. We explain the evaluation criteria for personalization in Section 4.4 and show quantitative results in Section 4.5.

4.1 Model

We extend our 30B Movie Gen Video model for Personalized Text-to-Video generation, PT2V, by conditioning the model on the identity information extracted from an input reference image in addition to the text prompt. Figure 20 illustrates the architecture of our PT2V model initialized from the T2V Movie Gen Video weights. We use vision token concatenation in the condition, enabling integration into a unified framework, which allows to scale up the model size. Similar to (He et al., 2024b), we extract identity features from a masked face image using a trainable Long-prompt MetaCLIP vision encoder (Xu et al., 2023), followed by a projection layer to align them with the text feature dimension. Our training strategy includes a PT2V pre-training phase followed by PT2V high quality finetuning.

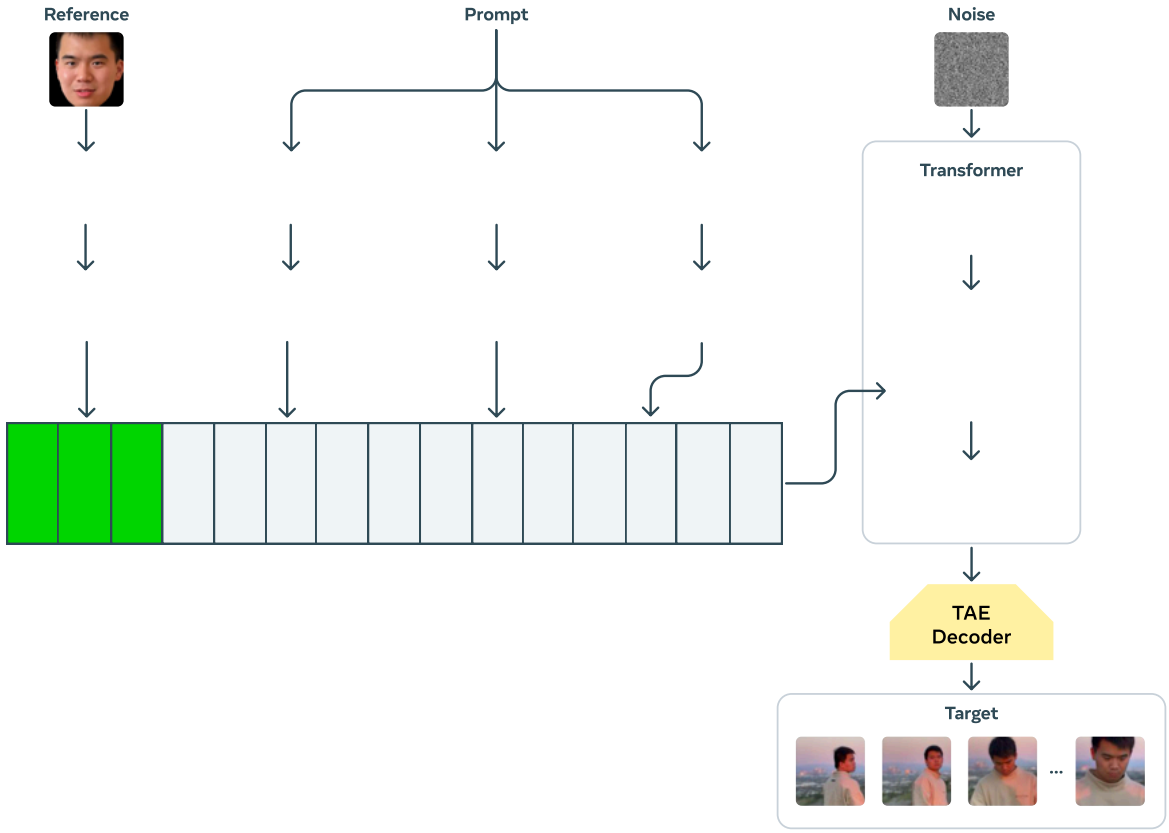


Figure 20 Architecture and inference pipeline of the Personalized Movie Gen Video (PT2V) model. We initialize the model from Movie Gen Video's weights and add additional learnable parameters to enable conditioning on a reference image. We encode the reference image using a trainable vision encoder initialized from Long-prompt MetaCLIP (Xu et al., 2023), and concatenate the embedding with the text prompt embedding. Frozen layers are illustrated in yellow while trainable ones in green.

4.2 Pre-training

4.2.1 Pre-training Data

For the PT2V training sets, our focus is exclusively on videos where the same person appears across all frames. We curate this training set from the Movie Gen Video pre-training datasets described in Section 3.2.1. To achieve this, we first filter the raw T2V videos based on captions by selecting those with human-related concepts. We extract frames at one-second intervals and apply a face detector to keep videos that contain a single face and where the ArcFace cosine similarity score (Deng et al., 2019) between consecutive frames exceeds 0.5. This processing provides us with $\mathcal{O}(1)$M text-video pairs where a single person appears, with durations from 4s to 16s. Based on the source reference face, our PT2V training dataset can be categorized into "paired" and "cross-paired" data. We define "paired" data as cases where the reference image is taken from the same video clip, while "cross-paired" data refers to cases where the reference image originates from a different video but features the same subject.

Paired Data. For each selected text-video pair, we uniformly sample 5 frames from the video clip, yielding $\mathcal{O}(10)$M paired training samples. For each frame, we crop the face area and segment the face region to prevent the model attending to non-critical areas such as the background.

Cross-Paired Data. We observed that training solely on the above paired data makes the model easily learn a copy-paste shortcut solution, *i.e.*, the generated video always follows the expression or the head pose from the reference face. To address this issue, we collect training pairs where the reference image comes from a different video of the same person.

We collected both real and synthetic cross-paired data samples. $\mathcal{O}(10)$K real cross-pairs from a subset of our pre-training data that contains different camera views of the same scene. For the synthetic cross-paired data, we use a pre-trained personalization image generation model (He et al., 2024b) to create synthetic reference images. Specifically, we apply the model to the first frame of each video from the paired data, generating images with diverse prompts to vary expressions, head poses, and lighting conditions, *etc*. To maintain identity consistency, we discard any generated images with an ArcFace similarity score below 0.7 compared to the reference image. In total, this process yields $\mathcal{O}(1)$M synthetic cross-paired data samples.

4.2.2 Pre-training recipe

There are three goals in PT2V pre-training: 1) train the model to condition on a reference image and preserve the identity, 2) generate long personalized videos, and 3) improve generated human expressions and motion naturalness. We found that directly training the model on long videos is inefficient and often leads to slow identity injection to the personalized model since (1) training speed is nearly proportional to the square of the number of latent frames (tokens), and (2) the weak reference image-to-video correspondence in long videos makes the task more challenging. More details on the pre-training recipe is shared in Figure 21.

Stage-I: Identity injection. In the first stage of PT2V pre-training, we simplify the problem by conditioning the model on the reference image and training on short videos. Specifically, we truncate the TAE embedding to 8 latent frames (corresponding to 64 RGB video frames) to accelerate identity injection using the paired training samples. We freeze the vision encoder and only train the transformer backbone. We observe that the model can quickly learn to follow the reference image during this stage, as measured by the average ArcFace similarity score in Figure 21.

Stage-II: Long video generation. To recover the model's capability to generate long videos, we continue training the PT2V model from Stage-I with a larger number of latent frames, similar to the pre-trained T2V model in Table 2. This stage substantially enhances the consistency of long video generation, particularly in terms of background and motion coherence.

Stage-III: Improve naturalness. Since the model in stage-I and stage-II has been trained on the paired image-video samples, it often demonstrates a strong copy-paste effect. For instance, in the generated video frames, the person tends to gaze directly at the camera, resulting in an unnatural-looking facial expression. We improve video naturalness and facial expression in stage-III by training on the cross-paired samples where the reference image is not from the corresponding target video. We leverage both real cross-paired data and

Training stage	TP	CP	bs/Node	#GPUs	global bs	learning rate	frame length	#iters	#seen videos
Stage-I	4	2	1	4096	512	2e-5	64	18k	9.21M
Stage-II	4	2	1	4096	512	2e-5	128/192/256	8k	4.1M
Stage-III	4	2	1	4096	512	2e-5 Transformer 1e-7 Vision Encoder	128/192/256	4k	2.05M
Total								**30k**	**15.36M**

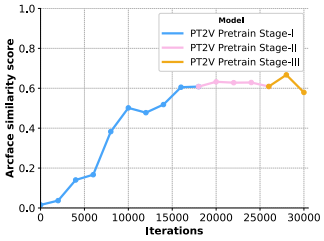


Figure 21 Personalized Movie Gen Video (PT2V) pre-training. Left: PT2V pre-training recipe. Our training recipe consists of three stages as described in Section 4.2.2. Right: ArcFace similarity score of the PT2V Stage I-III pre-training. The model gradually follows the identity as the training going on in Stage-I. Stage-II maintains the identity similarity of Stage-I. In Stage-III, identity similarity tends to fluctuate due to training with cross-paired data.

synthetic cross-paired data in this stage as discussed in Section 4.2.1. We also finetune the vision encoder to extract more detailed identity features from the reference image.

4.3 Supervised Finetuning

Similar to T2V, we further improve the video aesthetics in a high-quality finetuning stage by leveraging high quality aesthetic data.

4.3.1 Finetuning Dataset

The large scale pre-training data enables the model to generate videos following the identity from the reference face image. Similar to the post-training of Movie Gen Video (see Section 3.3), we collect a small set of high-quality finetuning data, with the goal of generating highly aesthetic videos with good quality motion. To match the visual quality and aesthetics of Movie Gen Video, we started from the T2V finetuning set and collected videos with a single person. Subsequently, we manually selected videos with diverse human actions, ensuring that the dataset captured a variety of movements and behaviors. In total, our final finetuning set contains $\mathcal{O}(1000)$ high-quality videos with both paired and real cross-paired reference images used with a 1:1 ratio.

4.4 Evaluation

We evaluate the quality of our PT2V models across three axes: identity preservation, video quality, and video-text alignment. The latter two axes are similar to T2V A/B evaluations in Section 3.5, where video quality can be further broken down into overall quality, frame consistency, motion completeness, and motion naturalness. To measure identity preservation, given an identity reference image and the generated video clip, the annotators are asked to rate on how well the generated character's face captures the reference person likeness in both the best and the worst frame (identity score), as well as how visually consistent the faces are among the generated frames containing the reference person (face consistency score). These two scores are measured in an absolute sense with ratings as "really similar", "somewhat similar", and "not similar" for the identity question and "really consistent", "somewhat consistent", and "not consistent" for the face consistency question. Annotators were trained to follow specific guidelines for the labeling on these axes and are constantly audited for quality.

Evaluation Dataset. We selected 50 subjects who were not seen during training as the reference face in the evaluation data. These reference face images include both frontal and side views. For each image, we pair it with 5-7 unique prompts, and curate 330 image-prompt pairs for evaluation. Similar to the T2V evaluation datasets, these prompts cover different human activities and facial expressions. We follow the same prompt rewrite as Section 3.4.1 to bridge the gap between our training and inference captions.

Method	Identity_{best} (↑)	Identity_{worst} (↑)	Face Consistency (↑)
ID-Animator	3.69%	3.08%	79.69%
PT2V Pre-train	71.91%	66.36%	97.53%
PT2V Finetune	65.52%	60.19%	95.61%

Table 13 Personalized Movie Gen Video (PT2V) evaluation. We compare our model after the pre-training and supervised high-quality finetuning stages against ID-Animator (He et al., 2024a) on Identity score on the best similar frame, the worst similar frame, and face consistency across frames.

Method	Identity_{best} (↑)	Identity_{worst} (↑)	Face Consistency (↑)
Frozen Vision Encoder	73.93%	66.67%	83.50%
Trainable Vision Encoder	90.10%	87.22%	99.69%
Cross-Paired Training	71.79%	66.36%	97.53%

Table 14 Ablation study for Personalized Movie Gen Video in terms of identity preservation metrics. We observed that the trainable vision encoder better preserves identity than the frozen vision encoder. Note that although cross-paired training decrease the identity similarity, it leads to more diverse head poses and more natural expressions.

	PT2V-Finetune net win rate *vs.* ID-Animator
Overall Quality	64.74
Consistency	22.18
Motion Naturalness	37.38
Motion Completeness	5.17
Text Alignment	53.20

(a)

	PT2V-Pretrain net win rate *vs.* T2V-Pretrain
Overall Quality	3.95
Consistency	10.33
Motion Naturalness	-1.82
Motion Completeness	-5.16
Text Alignment	-11.25

(b)

Table 15 Personalized Movie Gen Video (PT2V) evaluation on video quality and text alignment. (a) Net win rate (win% - loss%) of our PT2V after supervised finetuning *vs.* SOTA (ID-Animator (He et al., 2024a)). PT2V significantly outperforms ID-Animator in all metrics. (b) PT2V *vs.* Movie Gen Video (T2V) without the visual conditioning. We observed that PT2V wins consistency and performs on par in overal quality accounting for statistical significance, but loses in motion completeness and prompt alignment due to the narrow concept distribution (activities, objects, *etc.*) of PT2V.

4.5 Results

In Table 13 and Table 15a, we compare our Personalized Movie Gen Video after supervised finetuning with ID-Animator (He et al., 2024a). For the Identity score, we aggregate the "really similar" and "somewhat similar" scores in the best frame, and for the consistency score, we aggregate the "really consistent" and "somewhat consistent" scores. As evident, our method significantly outperforms the baseline by a large margin in all axes of identity preservation, video quality, and text alignment. We also compare it with Movie Gen Video without the visual conditioning in terms of video quality and text alignment in Table 15b.

We present generated videos from Personalized Movie Gen Video in Figure 23. The first four videos are generated with the same prompt but different identities, and the latter four are generated with the same identity but different prompts. The generated videos follow the identity with diverse motion and camera views. Qualitative comparisons between Personalized Movie Gen Video and ID-Animator (He et al., 2024a) are shown in Figure 22. Personalized Movie Gen Video consistently outperforms ID-Animator in terms of identity consistency and video quality.

4.5.1 Ablations

We ablate the impact of key design choices in our 30B Personalized Text-to-Video training pipeline.

Effect of training visual conditioning embedding. Our models use an embedding from a visual encoder of the

face as the visual embedding to condition the generation. We study whether training this embedding jointly during the video generation task improves performance. We re-train the third stage of our model with either a fixed or trainable vision encoder model and report the evaluation results in Tables 14 and 16. We observe that using a fixed vision encoder compromises identity preservation significantly, −16% as seen in Table 14.

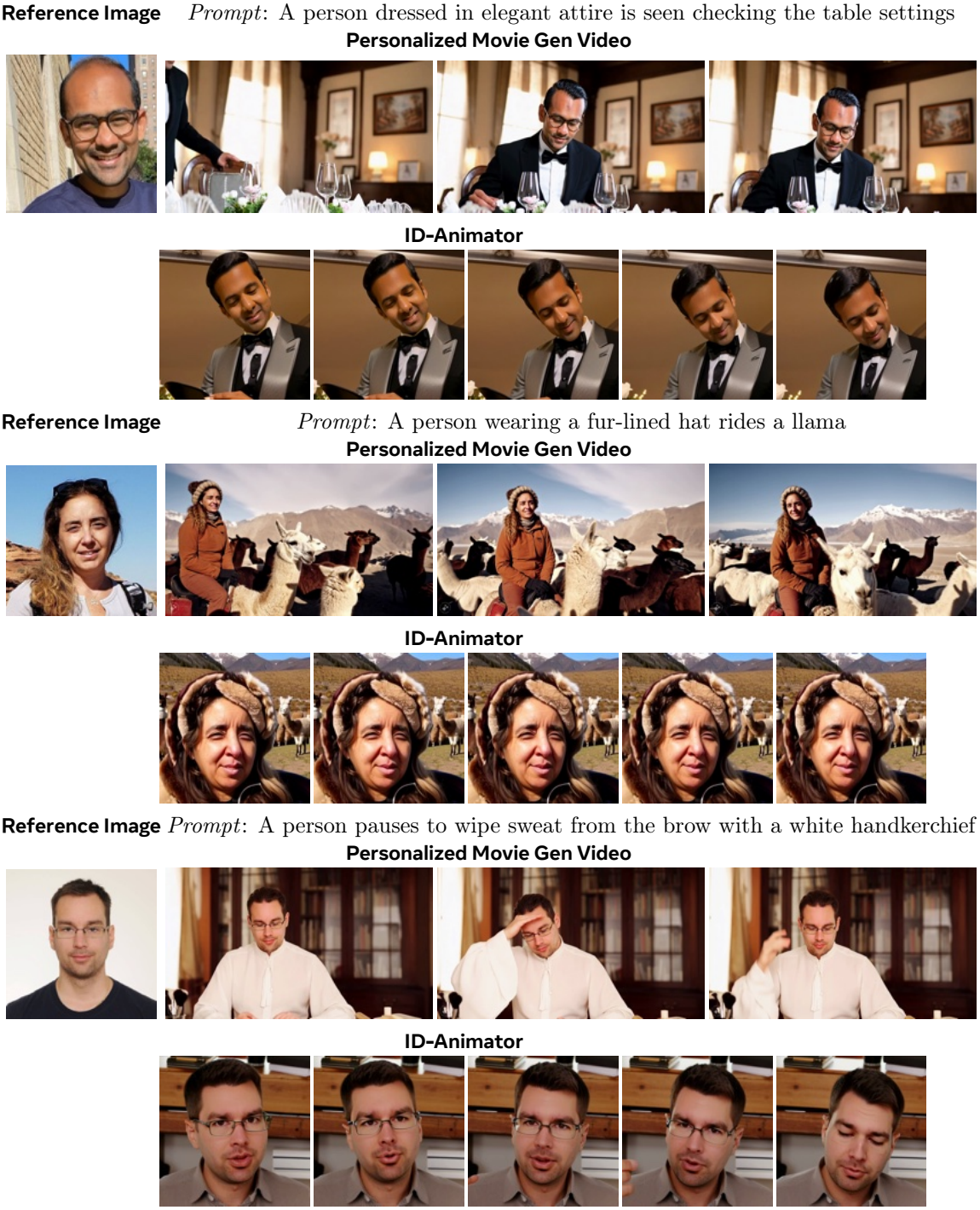


Figure 22 Qualitative comparisons between Personalized Movie Gen Video and prior work. Here, we show two generated videos for the same prompt, showcasing the output of Personalized Movie Gen Video (first row), and ID-Animator (second row) for comparison. Videos in this Figure found at https://go.fb.me/MovieGen-Figure22.

Figure 23 Generated videos from Personalized Movie Gen Video. Personalized Movie Gen Video generates high quality videos that follows the reference identity. Videos in this Figure found at https://go.fb.me/MovieGen-Figure23.

	Trainable *vs*. Frozen Vision Encoder	Cross-Paired *vs*. Paired	Finetune *vs*. Pretrain
Overall Quality	0.91	13.68	26.53
Consistency	6.38	-5.16	9.15
Motion Naturalness	-2.43	26.14	9.45
Motion Completeness	-2.72	7.6	5.49
Text Alignment	-4.56	27.36	-1.82

Table 16 Ablation study for Personalized Movie Gen Video in terms of Text-to-Video evaluation metrics. Each column shows the net win rate on adopting a design decision *vs*. a model that does not have it. Using a trainable vision encoder is comparable to the frozen one in terms of quality and alignment metrics while boosting the identity preservation significantly as in Table 14. The results also confirm the importance of cross-paired training data and supervised finetuning for video naturalness, higher quality and text alignment.

Effect of cross-paired data. Our training pipeline uses cross-paired data, *i.e.*, where the image of the face used to condition the generation comes from a different video clip than the video clip to be generated. We observe in Table 14 that cross-paired training leads to a decrease in identity metrics, however, it is crucial in improving facial expressions and natural movement in the generated videos. Human annotation in Table 16 reveals that cross-pair trained model improves text alignment by 27.36% and overall quality by 13.68%, especially 26.14% in motion naturalness.

Effect of high quality finetuning. We show the impact of a final high-quality finetuning stage on all axes of video quality and text alignment in Table 16 and on identity preservation in Table 13. Similarly, since our high-quality finetuning set includes cross-paired data, identity drops slightly while video quality and naturalness is improved significantly.

5 Instruction-Guided Precise Video Editing

As video content continues to dominate across various platforms, the demand for accessible, controllable, and precise video editing tools is rapidly increasing. In particular, there is a growing interest in developing text-guided video editing models. This interest arises from the limitations of more traditional software, which is inaccessible to most users and time-consuming for expert users. In contrast, text-guided video editing models aim to enable *any* user to edit a video (whether real or generated) easily, quickly, and precisely through natural language. However, the development of high-performing video editing models is hindered by the scarcity of supervised video editing data. In this section we introduce Movie Gen Edit, a model that achieves state-of-the-art results in video editing, and outline our approach for training it without any supervised video editing data.[1]

Our approach for training Movie Gen Edit is guided by two main assumptions. The first is that explicitly training the model for video editing offers significantly greater potential compared to training-free methods (Meng et al., 2022; Geyer et al., 2023). Moreover, to fully control all aspects of the input video, we must train the model to process the entire video input rather than limited proxy features of the input video (*e.g.*, depth maps) (Esser et al., 2023; Liang et al., 2023; Yan et al., 2023). Second, unlike tasks where abundant supervised data can be collected (*e.g.*, text-to-video), it is far less practical to gather supervised video editing data. Consequently, any large-scale training for video editing is expected to suffer from discrepancies between training and test-time scenarios. Therefore, the second assumption is that minimizing train-test discrepancies is crucial to unlocking the model's full potential. Accordingly, our approach involves several training stages that aim to gradually reduce such train-test discrepancies.

When considering the text-to-video model from Section 3, a clear discrepancy is that it was never trained to alter a media input based on an editing instruction. Therefore, in the first stage we train the text-to-video model with a multi-tasking objective that alternates between image editing, which we treat as *single-frame video editing*, and video generation (Section 5.1.2). While the model demonstrates some generalization to video editing after this stage, it often produces blurry videos. We attribute these artifacts to the distribution

[1]We provide examples of our model's video editing capabilities in https://go.fb.me/MovieGen-Figure24.

shift between training the model on single-frame video editing and testing it on multi-frame video editing. Thus, in the second stage we introduce two new synthetic tasks that more closely resemble *multi-frame video editing* and finetune the model on them (Section 5.1.3). The first task creates a synthetic video editing example by animating image editing examples using random affine augmentations. The second task casts video segmentation as a video editing task, by requiring the model to mark a specific object in the video using a specific color. After this stage, the main observed artifacts are lack of natural motion and oversaturation of newly generated elements. To address these issues, in the third and final stage, we introduce an adaptation of backtranslation for video editing, enabling us to train the model on multi-frame, high-quality output videos. We demonstrate that human annotators prefer Movie Gen Edit more than 74% of the time when compared to the previous state-of-the-art (Singer et al., 2024) on the TGVE+ benchmark (Wu et al., 2023c; Singer et al., 2024).

Finally, to facilitate the proper evaluation of the next generation of video editing models we collect a new comprehensive video editing benchmark, which we call Movie Gen Edit Bench (Section 5.2). This benchmark spans six different video editing tasks, each containing diverse editing instructions and corresponding videos. Unlike previous benchmarks, which assume models are limited to square, short, low resolution, and low FPS videos, Movie Gen Edit Bench includes videos with varied aspect ratios, resolutions, FPS, and more.

5.1 Model

Given the scarcity of supervised video editing data, methods for training models to perform video editing are prone to train-test discrepancies, resulting in suboptimal quality. To address this challenge, we introduce a multi-stage approach that progressively minimizes these discrepancies. We explain below the architecture modifications made to support video editing and then detail each step of our approach. The process is visualized in Figure 24.

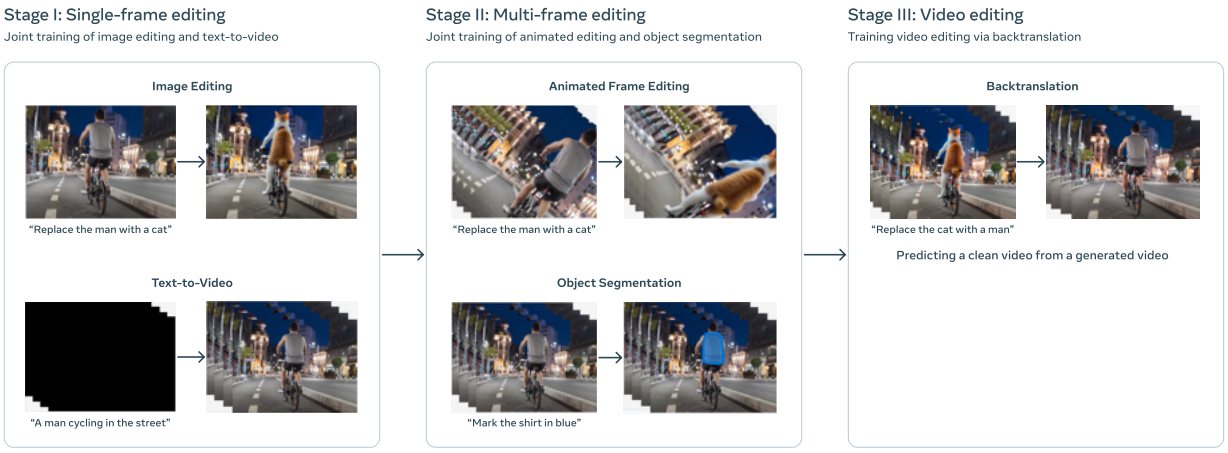


Figure 24 Extending the text-to-video model to video editing. We add video editing capabilities to the text-to-video model using three training stages: single-frame editing (Section 5.1.2), multi-frame editing (Section 5.1.3), and video editing via backtranslation (Section 5.1.4). We provide examples of our model's video editing capabilities in https://go.fb.me/MovieGen-Figure24.

5.1.1 Model Architecture and Initialization

To support video editing, we introduce several adaptations to the architecture described in Section 3. First, we enable input video conditioning by adding additional input channels to the patch embedder. This allows us to concatenate the latent video input with the noisy output latent video along the channels dimension, and provide the concatenated latent videos to the model. Additionally, following Emu Edit (Sheynin et al., 2024), we incorporate support for conditioning the model on specific editing tasks (*e.g.*, adding an object, changing the background, *etc.*). Specifically, our model has a learned task embedding vector for each task. For a given task, the model applies a linear transformation on the corresponding task embedding, producing four

embeddings that are concatenated to the text encoders' hidden representations. We also apply a second linear transformation to the task embedding, and add the resulting vector to the time-step embedding. Crucially, to fully preserve the model's video generation capabilities, we set all newly added weights to zero and initialize the remaining weights from the pre-trained text-to-video model.

Formally, the video editing architecture is conditioned on the following triplet $\mathbf{c} = (\text{TAE}(\mathbf{c}_{vid}), \mathbf{c}_{instruct}, j)$, where $\mathbf{c}_{vid}$ is the input video, TAE is the temporal auto-encoder, $\mathbf{c}_{instruct}$ is the editing instruction prompt, and j is the task-id of the relevant editing operation. We update the flow step in Eq. 2 to be $u(\mathbf{X}_t, \mathbf{c}, t; \theta)$, where $\mathbf{X}_t$ are the latents of the output video x_{vid} at step flow t, and θ are the model parameters. For brevity, we omit the task-id, j, and the activation of the temporal autoencoder, TAE, in the rest of the section.

5.1.2 Stage I: Single-frame Video Editing

We begin by training the model to utilize an editing instruction and a video input during the denoising process of the output video. However, since we lack supervised video editing data, we leverage an image editing dataset, treating image editing as single-frame video editing. Concretely, the image editing dataset is composed of triplets of $\mathbf{c}_{img-edit} = (\mathbf{c}_{img}, \mathbf{c}_{instruct}, x_{img})$, where $\mathbf{c}_{img}, x_{img}$ are the input and output images which we treat as single-frame videos. Clearly, high-quality *video* editing demands more than just precise editing of individual frames. For example, it is essential to ensure that the output video maintains temporal consistency and that any newly generated elements appear natural. Therefore, we aim to preserve the temporal consistency and generation quality of our model by simultaneously training it on both image editing and text-to-video generation.

As the new model architecture expects a video input as an additional condition, we condition the model on a black video during video generation training. Formally, given a text-to-video dataset with pairs $(\mathbf{c}_{txt}, x_{vid})$ of caption and target video, we create the following triplet, $\mathbf{c}_{text-to-video} = (\mathbf{c}_{\emptyset}, \mathbf{c}_{instruct}, x_{vid})$, where $\mathbf{c}_{\emptyset}$ is a black video, and $\mathbf{c}_{instruct}$ is the video output caption with $\mathbf{c}_{txt}$ rephrased as an instruction.

Due to difference in the sequence length between image editing and video generation, an image editing step requires significantly fewer operations than a video generation step. Therefore, we accelerate training by alternating between image editing and video generation batches, instead of mixing both tasks within each batch. Additionally, because our model is already trained on text-to-video generation, we further accelerate training by sampling image editing batches five times more frequently than video generation batches. Hence, we update Eq. 2 as follows:

$$\mathbb{E}_{t,\mathbf{X}_0,\mathbf{X}_1,\mathbf{c}} \|u(\mathbf{X}_t, \mathbf{c}, t; \theta) - \mathbf{V}_t\|^2, \quad \text{where } \mathbf{c} \sim \text{Categorical}\left(\left\{\mathbf{c}_{text-to-video} : \frac{1}{6}, \mathbf{c}_{img-edit} : \frac{5}{6}\right\}\right). \tag{3}$$

Interestingly, in preliminary experiments we found that naïvely using the first frame's positional embedding during image editing training leads to completely distorted outputs when testing the model on video editing. We resolve this issue by instead using a randomly sampled temporal positional embedding as the positional embedding for the image We train the model using this objective for thirty thousand steps.

5.1.3 Stage II: Multi-frame Video Editing

The trained model from Stage I (5.1.2) is capable of both precisely editing images and generating high-quality videos from text. However, it produces very blurry edited videos when tasked with video editing. We hypothesize that these artifacts are due to train-test discrepancies between Stage I training and video editing. The most significant discrepancy that we identify is that the model is not conditioned on multi-frame video inputs during Stage I training. We try to mitigate the blurriness artifacts by creating two complementary datasets that do include multi-frame videos inputs and outputs. We describe each of these datasets below and discuss the model's performance after training on them. Additionally, we visualize the two tasks in Figure 25.

Animated Frame Editing. We create an animated frame editing dataset by leveraging a video-caption pair dataset $(\mathbf{c}_{txt}, x_{vid})$. The process begins by prompting a language model (*e.g.*, LLaMa3) with the caption $\mathbf{c}_{txt}$ (*e.g.*, "A person walking down the street") to generate an editing instruction $\mathbf{c}_{instruct}$ (*e.g.*, "Put the person at the beach") and an output caption for the desired edited image $\hat{\mathbf{c}}_{txt}$ (*e.g.*, "A person walking at the beach"). Next, a random frame x_{frame} is selected from x_{vid}, and we apply a single-frame editing model,

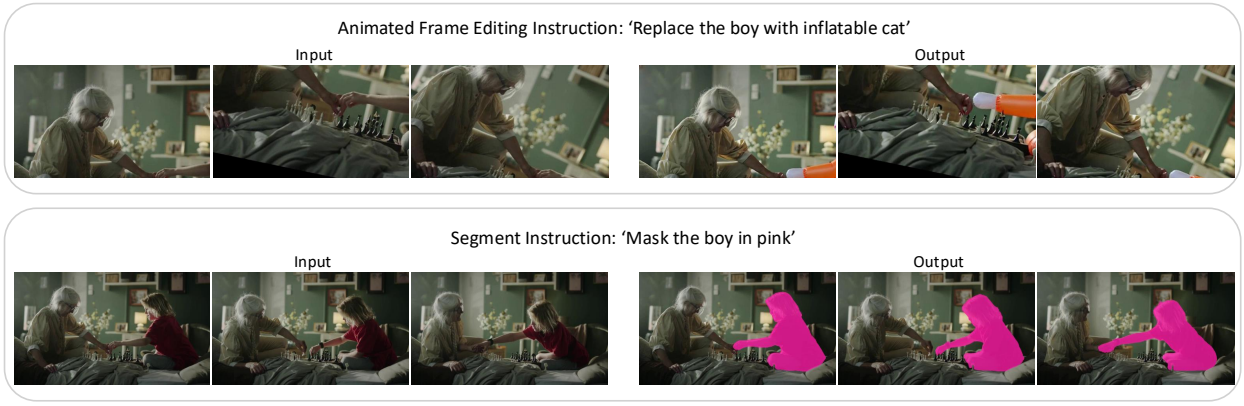


Figure 25 Multi-Frame Editing Stage. The model is trained on two synthetic multi-frame editing tasks: Animated Frame Editing and Generative Instruction-Guided Video Segmentation.

p_θ (introduced in Stage I, Section 5.1.2) to generate an edited frame $\hat{x}_{frame} \sim p_\theta(x_{frame}, \mathbf{c}_{instruct})$. We filter the resulting data points, $(\mathbf{c}_{txt}, \mathbf{c}_{instruct}, \hat{\mathbf{c}}_{txt}, x_{frame}, \hat{x}_{frame})$ using automated image editing metrics in a similar method to the one described in (Sheynin et al., 2024). To animate both the input and edited frames, we use an iterative process. In each iteration $i < n$, a random affine transformation $\mathcal{F}_i$ is applied to both the input frame $x^{(i-1)}_{frame}$ and the edited frame $\hat{x}^{(i-1)}_{frame}$, producing the next frames $x^{(i)}_{frame}$ and $\hat{x}^{(i)}_{frame}$. This process results in an animated sequence of input frames $\hat{\mathbf{c}}_{vid} = \{x^{(i)}_{frame}\}^n_{i=0}$ and edited frames $\hat{x}_{vid} = \{\hat{x}^{(i)}_{frame}\}^n_{i=0}$. Finally, combining the animated frames with the editing instruction forms a multi-frame editing example $\mathbf{c}_{animated} = (\hat{\mathbf{c}}_{vid}, \mathbf{c}_{instruct}, \hat{x}_{vid})$. The full process is outlined in Algorithm 1, and a visual example of animated frame editing is provided in Figure 25.

Algorithm 1 Animated Frame Editing Dataset Creation

Require: Video-caption dataset (c_{txt}, x_{vid}), editing model p_θ
Input: Video x_{vid}, Caption c_{txt}
Output: Animated frames $\hat{c}_{vid}$, Editing instruction $c_{instruct}$, Animated edited frames $\hat{x}_{vid}$
Generate editing instruction: $c_{instruct} \sim \text{LLaMa3}(c_{txt})$
Sample random frame: $x_{frame} \sim x_{vid}$
Generate edited frame: $\hat{x}_{frame} \sim p_\theta(x_{frame}, c_{instruct})$
Initialize animated sequences: $\hat{c}_{vid} \leftarrow \emptyset, \hat{x}_{vid} \leftarrow \emptyset$
Set initial frames: $x^{(0)}_{frame} \leftarrow x_{frame}, \hat{x}^{(0)}_{frame} \leftarrow \hat{x}_{frame}$
for $i = 1$ to n **do**
    Sample random affine augmentation: $\mathcal{F}_i$
    Apply augmentation: $x^{(i)}_{frame} \leftarrow \mathcal{F}_i(x^{(i-1)}_{frame})$, $\hat{x}^{(i)}_{frame} \leftarrow \mathcal{F}_i(\hat{x}^{(i-1)}_{frame})$
    Append frames to sequences: $\hat{c}_{vid} \leftarrow \hat{c}_{vid} \cup \{x^{(i)}_{frame}\}$, $\hat{x}_{vid} \leftarrow \hat{x}_{vid} \cup \{\hat{x}^{(i)}_{frame}\}$
end for
Return: $(\hat{c}_{vid}, c_{instruct}, \hat{x}_{vid})$

Generative Instruction-Guided Video Segmentation. The lack of natural motion in animated frame editing examples poses a clear discrepancy between animated frame editing and video editing. To address this, we complement the animated frame editing task with the task of generative instruction-guided video segmentation, which extends the Segment task from Emu Edit (Sheynin et al., 2024) from images to videos. In this task, the model is required to edit a video by marking a specific object in a particular color based on the given instruction.

We begin by collecting editing instructions using a procedure similar to the one employed while collecting animated frame editing examples. However, we prompt the language model to generate an instruction, $\mathbf{c}_{instruct}$, to *mark* a particular subject or object in the video in a specific color, and to output the name of the

edited object (*e.g.*, "apple"). We then use DINO (Liu et al., 2023c) and SAM 2 (Ravi et al., 2024) to extract the segmentation mask for the object in the video. Finally, we create the target video, $\hat{x}_{vid}$, by marking the object in the relevant color using the extracted segmentation mask. Following the notation described above, the paired data, $\mathbf{c}_{segmentation} = (\mathbf{c}_{vid}, \mathbf{c}_{instruct}, \hat{x}_{vid})$, then consists of a real input video, $\mathbf{c}_{vid}$, an instruction to mark a specific object in a certain color, $\mathbf{c}_{instruct}$, and a corresponding edited video, $\hat{x}_{vid}$.

Training. We finetune the model from Stage I on these datasets, alongside text-to-video generation using multi-task training for one thousands steps. During training we sample animated frame editing examples three times more frequently than generative instruction-guided video segmentation and text-to-video generation. To put it formally, we update the sampling in Eq. 3 as follows

$$\mathbf{c} \sim \text{Categorical}\left(\left\{\mathbf{c}_{text-to-video} : \frac{1}{5}, \mathbf{c}_{animated} : \frac{3}{5}, \mathbf{c}_{segmentation} : \frac{1}{5}\right\}\right). \tag{4}$$

We observe that this stage mitigates the blurriness artifacts from Stage I; however, newly generated elements in the edited video exhibit less motion than desired, and at times appear oversaturated.

5.1.4 Stage III: Video Editing via Backtranslation

Figure 26 Backtranslation Stage. The model is trained to denoise the clean input video while conditioning on the edited generated video.

While Stage II training (Section 5.1.3) mitigates most of the artifacts observed in Stage I (Section 5.1.2), we notice that newly generated elements often lack motion and sometimes appear oversaturated. These artifacts are likely due to the output videos in the animated frame editing dataset, which lack natural motion and are model-generated. Therefore, in Stage III, we create video editing data with *real* output videos. Similarly to Stage II (Section 5.1.3), we assume access to a dataset of videos x_{vid} and corresponding captions $\mathbf{c}_{txt}$ (*e.g.*, "*Apples on a table*"). We employ LLaMa3 to create an editing instruction $\mathbf{c}_{instruct}$ (*e.g.*, "*Put the apples in a small basket*"), and an output caption $\hat{\mathbf{c}}_{txt}$ (*e.g.*, "*Apples in a small basket on the table*"). Then, we use the model from Stage II to generate an edited video $\hat{x}_{vid} \sim p_\theta(x_{vid}, \mathbf{c}_{instruct})$ based on the input video x_{vid} and editing instruction $\mathbf{c}_{instruct}$. Afterward, we utilize $\mathbf{c}_{txt}, \hat{\mathbf{c}}_{txt}, x_{vid}, \hat{x}_{vid}$ to filter the generated examples based on automatic ViCLIP scores, following a similar filtering process as in Stage II.

A naïve approach would be to tune the model on the resulting dataset, $(x_{vid}, \mathbf{c}_{instruct}, \hat{x}_{vid})$, teaching the model to predict its own generations. However, in this case, the output videos are likely to contain the very same artifacts we aim to mitigate. Therefore, we adapt the backtranslation technique from natural language processing (Edunov et al., 2018) to video editing. Specifically, we prompt LLaMa3 using $(\mathbf{c}_{txt}, \hat{\mathbf{c}}_{txt}, \mathbf{c}_{instruct})$ to generate an editing instruction, $\mathbf{c}_{instruct-bwd}$, that should alter the generated video $\hat{x}_{vid}$ into the original video x_{vid} (*e.g.*, "*Remove the small basket and put the apples on the table*"). Then, we build a synthetic paired dataset, $\mathbf{c}_{backstranslation} = (\hat{x}_{vid}, \mathbf{c}_{instruct-bwd}, x_{vid})$, and use it to train the model to denoise the clean video x_{vid} while conditioning on the potentially noisy video $\hat{x}_{vid}$ and editing instruction $\mathbf{c}_{instruct-bwd}$. In this manner, we construct a weakly-supervised video editing dataset, with real output videos.

5.2 Evaluation

We evaluate the capabilities of our model against two main video editing benchmarks. The first benchmark, TGVE+ (Singer et al., 2024), is a recently proposed extension of the TGVE benchmark (Wu et al., 2023c). While this benchmark is comprehensive, it features low-resolution, low-FPS, short, and square videos. This is in contrast to state-of-the-art video generation models and most media content, which typically feature higher

resolution, longer videos with higher FPS, and varied aspect ratios. Therefore, to enable proper evaluation of next-generation video editing models with more relevant video inputs, we introduce a new benchmark, called Movie Gen Edit Bench. This benchmark consists of videos with varying resolutions, FPS, lengths, and aspect ratios. We compare our approach against several baselines and measure its effectiveness across multiple axes, including fidelity to the user instructions and input video, and overall visual quality.

5.2.1 Video Editing Benchmarks

The TGVE+ benchmark (Singer et al., 2024; Wu et al., 2023c) consists of seventy-six videos, each accompanied by seven editing instructions for the following tasks: (i) local object modification, (ii) style change, (iii) background change, (iv) simultaneous execution of multiple editing tasks, (v) object removal, (vi) object addition, and (vii) texture modification. While the benchmark offers a comprehensive evaluation across a diverse set of editing tasks, the videos in the benchmark are of 480 $\times$ 480 px resolution, and 3.20 seconds length at 10 FPS, or 8.00 seconds at 16 FPS. In contrast, real user videos are expected to have a higher resolution, higher FPS, and may contain various aspect ratios. Hence, it is unclear whether evaluation against TGVE+ will accurately reflect video editing performance on real user videos. Moreover, current foundational video generation models (OpenAI, 2024; RunwayML, 2023, 2024) can operate at high resolution (*e.g.*, 768p or 1080p), 16 or more FPS, multiple aspect ratios, and can process much longer videos than those from TGVE.

Thus, to enable the evaluation of video editing using more practical videos, we collect a new benchmark, Movie Gen Edit Bench, that aims to evaluate the video editing capabilities of the next generation of video editing models. To build Movie Gen Edit Bench, we rely on videos from the publicly released Segment-Anything-V2 (Ravi et al., 2024) dataset. For each video out of the 51,000 videos found in the dataset, we generate a caption using a similar approach to the one in Section 3.2.1, calculate its motion score (Farnebäck, 2003; Bradski, 2000), and calculate its aesthetics score (Schuhmann et al., 2022). We then filter all videos with an aesthetics score lower than the median score of the dataset. For each category, we bin the videos based on their motion score and sample videos uniformly from the bins for each category. Overall, the benchmark validation set has 64 videos, whereas the test set has 128 videos.

To facilitate a realistic benchmark with editing instructions written by humans, we employ crowd workers. For each video and for each editing operation, we assign crowd workers the task of writing down a creative editing instruction. Finally, to support the use of CLIP-based image editing evaluation metrics (similar to those used in (Sheynin et al., 2024)), we additionally collect an input caption and output caption. Thus, the benchmark has altogether 1,152 examples, and spans six different editing tasks.

5.2.2 Video Editing Measures

Our experiments evaluate the ability of video editing models to modify an input video while accurately following the provided instructions and preserving the structure and elements that should remain unchanged. We assess the video editing performance of our model and the baselines using both Human evaluation and automated metrics. For automated evaluation, we use the main automatic metrics reported by (Singer et al., 2024), which account for both temporal and spatial coherence. Specifically, we measure (i) ViCLIP text-image direction similarity (ViCLIP$_{dir}$), which evaluates the alignment between changes in captions and corresponding changes in the videos, and (ii) ViCLIP output similarity (ViCLIP$_{out}$), which measures the similarity between the edited video and the output caption.

For Human evaluation, we follow the standard evaluation protocol of TGVE+ (Singer et al., 2024; Wu et al., 2023c). Human annotators are presented with an input video, the editing instruction, and a pair of edited videos. We then ask the raters to respond to the following questions: (i) Text Alignment: Which edited video more accurately reflects the given caption, (ii) Structure: which edited video better maintains the structural integrity of the original input, and (iii) Quality: which edited video is visually more appealing and aesthetically superior. Additionally, we extend this protocol with a fourth question: (iv) Overall: considering quality, structure, and text alignment, which edited video is better.

5.3 Results

In this section, we compare our model with leading video editing baselines. We then analyze the importance and impact of the main design and implementation choices in our approach (Section 5.3.2).

5.3.1 Comparisons to Prior Work

We evaluate our model against different video editing baselines, including both training-free methods, and methods that require prior training such as our method. A common training-free method for video editing is Stochastic Differential Editing (SDEdit) (Meng et al., 2022) which performs image editing by adding noise to the input video and then denoising it while conditioning the model on a descriptive caption. Recent video foundation models (Bar-Tal et al., 2024; Brooks et al., 2024), have used SDEdit for video editing, and demonstrated its ability to maintain the overall structure of the input video. However, this approach can lead to the loss of important details, such as subject identity and texture, making it less effective for precise editing. In our experiments, we utilize SDEdit with the base T2V model, Movie Gen Video, and perform the denoising process for 60% of the total iterations. Another prominent approach for video editing is to inject information about the input or generated video from key frames via cross-attention interactions (Wu et al., 2023a; Yatim et al., 2023).

On the other hand, the current top performing methods for video editing utilize prior training while overcoming the lack of supervised datasets for video editing. For example, InsV2V (Cheng et al., 2024) extends the general approach of InstructPix2Pix (Brooks et al., 2023) to video editing, enabling the creation and training of a video editing model using synthetic data. EVE (Singer et al., 2024), relies on unsupervised training by employing knowledege distillation from two expert models, one for image editing and the other for text-to-video generation (see Section 7.4 for more details). Finally, we compare to Tune-A-Video (TAV) (Wu et al., 2023b) which served as the baseline in the TGVE contest. TAV tunes a text-to-image model to a specific video, followed by inverting the input video and using the inverted noise to generate the output video. We compare with all of the baselines described above on the TGVE+ benchmark.

In addition, we evaluate our method versus SDEdit and Runway Gen3 Video-to-Video[2] (Runway Gen3 V2V) (RunwayML, 2024) on Movie Gen Edit Bench (Sec. 5.2.1). We compare to Runway Gen3 V2V in two settings. The first, employs Runway Gen3 V2V on all tasks comprising the benchmark – (i) local object modification, (ii) style change, (iii) background change, (iv) object removal, (v) object addition, and (vi) texture modification. However, as we observe that Runway Gen3 V2V struggles to preserve fine details in the input video (in contrast to general structure), in the second setting we focus on the style editing task, denoted as Runway Gen3 V2V Style. We omit comparison with other baselines, as they are mostly limited to operating on short videos with 32 frames and do not fully utilize Movie Gen Edit Bench's videos duration, resolution, or varied aspect ratios.

Results of our evaluation versus all baselines are presented in Table 17. Throughout this section we report 'win rates', which can lie in the range $[0, 100]$, where 50 indicates a tie between two models. Human raters prefer Movie Gen Edit over all baselines by a significant margin on both benchmarks. On the TGVE+ benchmark, our model is preferred 74% more often than the current state-of-the-art EVE in the overall Human evaluation criterion. In terms of automated metrics, Movie Gen Edit presents state-of-the-art results on the ViCLIP$_{dir}$ metric. On the ViCLIP$_{out}$ metric, Movie Gen Edit performance is comparable to EVE. However, unlike Movie Gen Edit, EVE has access to the video output caption which is used for calculating the ViCLIP$_{out}$ score.

On the Movie Gen Edit Bench, our method is preferred over the Runway Gen3 V2V and Runway Gen3 V2V Style settings. Interestingly, when compared to Runway Gen3 V2V Style, Human evaluation metrics highlight our advantage in maintaining the structure of the input videos. Compared to SDEdit, Movie Gen Edit is preferred by human raters in Human evaluation criterions by a significant margin, despite a lower ViCLIP$_{out}$ score. Similarly to EVE, SDEdit has an advantage in the ViCLIP$_{out}$ automatic metric as it has access to the same output caption that ViCLIP$_{out}$ uses.

[2]Runway Gen3 Video-to-Video videos were collected on September 24th, 2024.

Dataset	Method	Human Evaluation				Automated	
		Text	Struct.	Quality	Overall	ViCLIP$_{dir}$ ↑	ViCLIP$_{out}$ ↑
TGVE+	TAV (Wu et al., 2023b)	85.00	81.94	91.57	89.70	0.131	0.242
	STDF (Yatim et al., 2023)	84.43	61.60	73.21	74.43	0.093	0.227
	Fairy (Wu et al., 2023a)	84.15	77.52	84.20	84.91	0.140	0.197
	InsV2V (Cheng et al., 2024)	73.75	66.60	70.73	70.85	0.174	0.236
	SDEdit (Meng et al., 2022)	85.51	90.07	76.19	80.59	0.131	0.241
	EVE (Singer et al., 2024)	69.48	70.05	75.18	74.38	0.198	0.251
	Movie Gen Edit (Ours)	–	–	–	–	0.225	0.248
Movie Gen Edit Bench	Runway Gen3 V2V (RunwayML, 2024)	88.14	98.33	83.14	93.33	0.068	0.188
	Runway Gen3 V2V Style (RunwayML, 2024)	55.55	73.61	58.33	59.72	0.124	0.214
	SDEdit (Meng et al., 2022)	94.37	86.34	85.14	91.96	0.124	0.239
	Movie Gen Edit (Ours)	–	–	–	–	0.209	0.224

Table 17 Comparison with video editing baselines on the TGVE+ and Movie Gen Edit Bench benchmarks. We report ViCLIP metrics and human ratings. Human evaluation shows the win rate of Movie Gen Edit (Ours) against the baselines. Runway Gen3 videos were collected on September 24th, 2024. For the human evaluations we report 'win rates', which can lie in the range $[0, 100]$, where 50 indicates a tie between two models.

5.3.2 Ablations

In this section, we aim to assess and quantify the importance and impact of the main design and implementation choices in our approach. Unless stated otherwise, ablations are conducted on the validation set of Movie Gen Edit Bench (Section 5.2.1) using the Human evaluation metrics described in Section 5.2.2.

Stage I: Multi-tasking versus Adapter. As mentioned in Section 5.1.2, the first stage of our approach involves training the model using a multi-tasking objective that alternates between image editing and video generation. However, an alternative approach would be to train an image editing adapter on top of the text-to-video generation model. The advantage of this approach is that by freezing the weights of the Text-to-Video model, one can ensure that the model's video generation capabilities are preserved. However, this approach is more memory demanding and typically requires providing the model with two text inputs for video editing: (i) a video caption for the frozen text-to-video model, and (ii) an editing instruction for the trained adapter.

To ablate this design choice, we implement a variant of a ControlNet adapter (Zhang et al., 2023a) that aligns with the model described in Section 5.1.2. Specifically, we freeze the original text-to-video model and clone a trainable copy of it. We apply the same adaptations as described in Section 5.1.1 to the trainable model. Finally, we follow (Zhang et al., 2023a) by introducing a zero-initialized convolutional layer after each layer of the trainable model. During the forward pass, the frozen model gets a caption that describes the output video, and the trainable model gets similar inputs as described in Section 5.1.2. After each layer we add the hidden states of the trainable model to the hidden states of the frozen model.

We train the adapter on image editing using the same image editing data as used in Stage I (Section 5.1.2) for 10K iterations and compare it to the Stage I model trained for the same number of iterations.

We evaluate the models' image editing capabilities on the Emu Edit benchmark (Sheynin et al., 2024) and measure performance using the L_1 distance between the input and output images, distance between DINO (Liu et al., 2023c) features of the input and output images, and several CLIP-based image editing evaluation metrics: CLIP$_{im}$ estimates whether the model preserved elements from the input image by measuring the CLIP-space distance between the input image and the edited image. CLIP$_{out}$ estimates whether the model followed the editing instruction by measuring the CLIP-space distance between a caption describing the desired edited image and edited image itself. CLIP$_{dir}$ estimates if the elements that were supposed to change were edited correctly, while ensuring that elements intended to remain unchanged were preserved. Furthermore, we conduct a human evaluation in which human raters assess text alignment and image faithfulness. During this assessment the human raters see the original image and instruction alongside two modified images and are asked: (i) which edited image better preserves the required elements from the input image, and (ii) which edited image best follows the editing instruction.

As can be seen, the Stage I variant achieves comparable results to the ControlNet variant on CLIP$_{im}$, CLIP$_{out}$, and DINO metrics. However, it achieves a significantly better performance on both the CLIP$_{dir}$ and L1 metrics. Furthermore, Human evaluation indicates that full model training results in edits that are better

aligned with the editing instructions and more faithful to the input images. This indicates that full model training can better support high quality editing than a ControlNet adapter.

Method	Human Evaluation		Automated				
	Text align.	Image faith.	$\text{CLIP}_{dir}\uparrow$	$\text{CLIP}_{im}\uparrow$	$\text{CLIP}_{out}\uparrow$	L1$\downarrow$	DINO$\uparrow$
ControlNet adapter	69.325	66.235	0.115	0.843	0.235	0.099	0.794
Stage I (Section 5.1.2)	–	–	0.128	0.840	0.238	0.088	0.789

Table 18 Ablation on the best way to incorporate image editing information to a text-to-video model. We compare two variants: (i) training a ControlNet on a frozen text-to-video model, and (ii), multitask learning on text-to-video and image editing. Human evaluation shows the win rate of the Stage I model against the ControlNet adapter.

Method	Text	Structure	Quality	Overall
Animated Image Editing	70.67	53.4	61.31	61.16

Table 19 Contribution of training the second stage using animated frames compared to animated images. Human evaluation shows the win rate of the Stage II model (Section 5.1.3) versus its animated *image* editing counterpart.

Method	Text	Structure	Quality	Overall
Standard	44.66	72.56	73.61	70.23

Table 20 Contribution of training with backtranslation rather than standard fine-tuning. Human evaluation shows the win rate of Movie Gen Edit versus performing Stage III (Section 5.1.4) with standard fine-tuning rather than backtranslation.

Method	Text	Structure	Quality	Overall
Stage II (vs Stage I)	61.65	90.86	91.9	89.29
Movie Gen Edit (vs Stage II)	49.36	59.78	61.86	60.82

Table 21 Contribution of each stage in our approach. In each row, we show the win rate of the model from a certain stage when compared to its previous stage counterpart.

Stage II: Animated Frame/Image Editing As mentioned in Section 5.1.3, the second stage of our approach involves finetuning the model from Stage I (Section 5.1.2) on an animated *frame* editing dataset. However, a more straightforward alternative would have been to animate the *image* editing dataset from Stage I, thereby avoiding the need to collect a new single-frame editing dataset. To explore this choice, we train the model from Stage I using a similar approach to the one described in Section 5.1.3, but animate the image editing dataset from Stage I rather than the frame-editing dataset. As shown in Table 19, human raters consistently prefer the outputs of the model from Stage II over its animated image editing counterpart. Specifically, they find the model to be more text faithful in over 70% of the time, and rate its quality higher over 61% of the time.

Stage III: Backtranslation versus Standard Fine-tuning. During Stage III (Section 5.1.4), we generate edited videos using the model from Stage II, apply filtering, and then perform backtranslation training. In this ablation, we assess whether backtranslation is necessary or if standard fine-tuning on model generated outputs suffices. We follow the same training protocol as in Stage III, but instead of backtranslation, we train the model to predict the generated video, $\hat{x}_{vid}$ from the input video x_{vid} and original editing instruction $\mathbf{c}_{instruct}$. As shown in Table 20, while training with backtranslation slightly degrades text faithfulness when compared to standard finetuning, it provides very significant improvements in structure, quality, and crucially, overall preference.

Evaluating the Contribution of Each Stage. To assess the contribution of each training stage, we compare the model from each stage with the model from the previous stage. As shown in Table 21, Stage II (Section 5.1.3)

demonstrates significant improvements over the Stage I model, with human evaluators preferring it more than 89% of the time. The benefits of Stage III (Section 5.1.4) are more subtle, with human evaluators preferring Movie Gen Edit over the Stage II model in more than 60% of cases. Importantly, most of the contributions from Stage III are reflected in the improved quality of the edited videos, with only a very minor trade-off in text faithfulness.

6 Joint Sound Effect and Music Generation

Our goal with Movie Gen Audio is to generate soundtracks for both video clips and short films (Holman, 2012), which may range from a few seconds to a few minutes. The soundtrack considered in this work includes ambient sound, sound effects (Foley), and instrumental music, but does not include speech or music with vocals. In particular, the ambient sound should match the visual environment, the sound effects should be temporally aligned with the actions and plausible with respect to the visual objects, music should express the mood and sentiment of the video, blend properly with sound effects and ambient, and align with scenes as what one would expect when watching a movie.

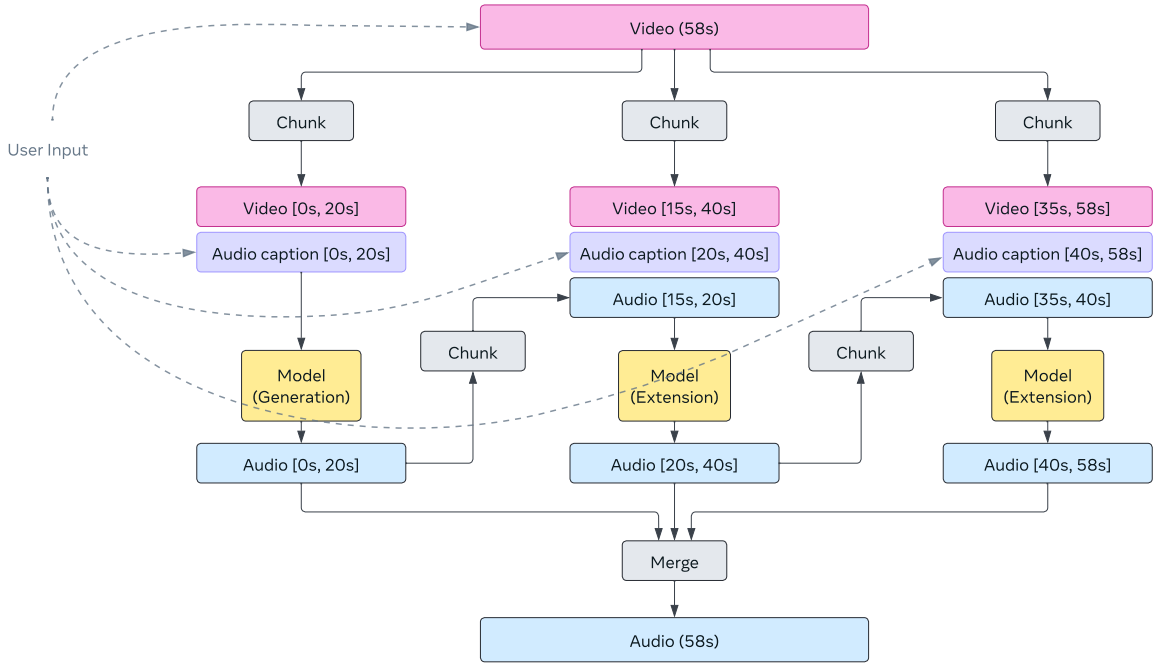


Figure 27 Movie Gen Audio extension diagram. A user provides a video (*e.g.*, 58s), and audio caption for each video chunk (*e.g.*, 20s). Starting from the second chunk, the model takes not only the video chunk and the caption, but also a segment from the previously generated audio (*e.g.*, the last 5s) in order to generate a new chunk that is coherent with the previous one.

In order to generate soundtracks for variable duration of videos, we build a single model that can perform both *audio generation* given a video, and *audio extension* given a video with partially generated audio. We aim to generate up to 30 seconds of audio in a single shot, and allow the model to utilize extension to generate audio of arbitrary lengths. Figure 27 illustrates the process for long-form video generation.

We enable audio extension by training the model to perform masked audio prediction, where the model predicts the audio target given the whole video and its surrounding audio. The surrounding audio can be empty (*i.e.*, audio generation), before or after the target audio (*i.e.*, audio extension in either direction), or around the target (*i.e.*, audio infilling). Audio infilling is useful for fixing small segments that contains artifacts or unwanted sound effects.

Lastly, for sound design purposes, users would often want to specify what and how acoustic events should be added to the video, such as deciding what on-screen sounds to emphasize, what off-screen sounds to add, whether there is background music, and what style to generate for the music. To provide users more control, we enable text prompting.

6.1 Model

We adopt the flow-matching (Lipman et al., 2023) based generative models and the diffusion transformer (DiT) (Peebles and Xie, 2023) model architecture. Additional conditioning modules are added to provide control. Figure 28 illustrates the model architecture.

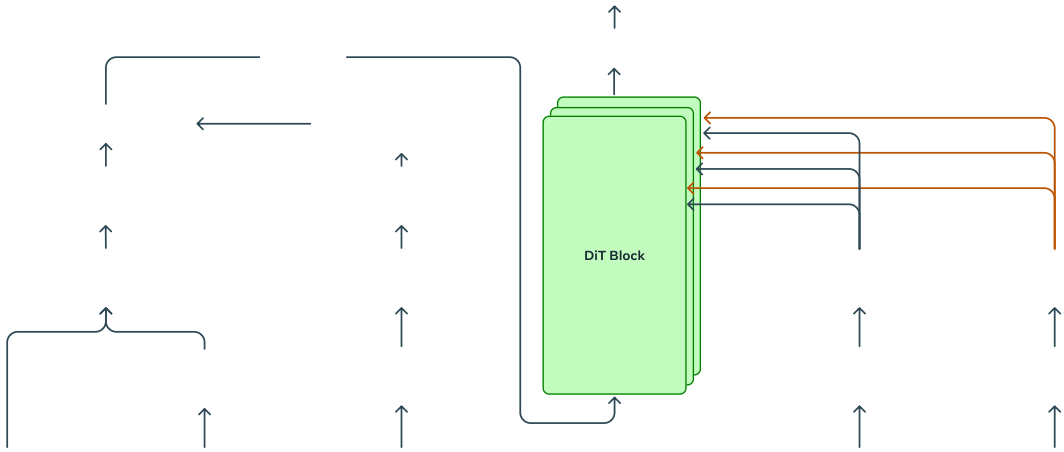


Figure 28 Movie Gen Audio model diagram. Yellow blocks denote input, blue blocks denote pre-trained and frozen modules, gray blocks denote operations without learnable parameters, green blocks denote learnable modules, and the pink block shows the output velocity $u(\mathbf{X}_t, \mathbf{c}, t; \theta)$. Conditioning input $\mathbf{c}$ includes masked audio context, video, and text. $\mathbf{X}_t$ is a sample from p_t, and t is the flow time step. For audio context, we replace the masked frames with zeros for DAC-VAE output.

6.1.1 Flow-matching

We also use the Flow Matching (Lipman et al., 2023) objective as described in Section 3.1.2 to train the Movie Gen Audio model. The same optimal transport path is used for constructing $\mathbf{X}_t$ which is now an audio sample in the latent space that we will describe in Section 6.1.3, and the same logit-normal distribution is used for sampling flow-step t. Instead of conditioning only on text prompt embedding $\mathbf{P}$, Movie Gen Audio is conditioned on multimodal prompt $\mathbf{c}$ which will be described in Section 6.1.3.

We choose diffusion-style models (Ho et al., 2020; Song et al., 2020) over discrete token-based language models (Kreuk et al., 2022) because (1) it shows strong empirical performance in sound, music, and speech generation (Liu et al., 2023b; Ghosal et al., 2023; Majumder et al., 2024; Shen et al., 2023; Huang et al., 2023), (2) its non-autoregressive nature permits flexible generation direction, and can be used for both infilling or out-filling in both directions, (3) modeling audio in continuous space enables applications of techniques such as SDEdit (Meng et al., 2022) for editing and multi-diffusion (Bar-Tal et al., 2023) for infinite-length audio generation, (4) it enables users to flexibly trade-off quality for runtime through configuring ODE parameters, and enjoys recent advancement in distillation or consistency training techniques that boost quality significantly at a much lower runtime. On the other hand, we choose flow-matching over diffusion because we found it achieves better training efficiency, inference efficiency, and performance compared to diffusion models as shown in recent works (Lan et al., 2024; Le et al., 2023; Vyas et al., 2023; Prajwal et al., 2024; Mehta et al., 2024; Esser et al., 2024).

6.1.2 Diffusion Transformer

Movie Gen Audio adopts the diffusion transformer (DiT) architecture (Peebles and Xie, 2023), which modulates the outputs of normalization layers with scale and bias, and outputs of self-attention and feed-forward layers with scale in each transformer block (Vaswani et al., 2017). A multi-layer perceptron (MLP) takes the flow time embedding as input and predicts the six modulation parameters (four scales and two biases). The MLP is shared across all layers, different from the original DiT, and only layer-dependent biases are added to the MLP outputs. This saves parameters without sacrificing performance. The next section describes how other inputs are conditioned.

6.1.3 Audio Representation and Conditioning Modules

Audio. The latent diffusion framework (Rombach et al., 2022) is adopted, where data (48kHz) is represented as compact 1D latent features of shape $T \times C$ at a much lower frame rate (25Hz) and $C = 128$ extracted from a separately trained DAC-VAE (Descript Audio Codec (Kumar et al., 2024) with variational autoencoder (Kingma, 2013) formulation) model. Compared to the commonly used Encodec (Défossez et al., 2022) features (75Hz, 128-d) for 24kHz audio in audio diffusion models (Shen et al., 2023; Vyas et al., 2023), our DAC-VAE offers a lower frame rate (75Hz→25Hz), a higher audio sampling rate (24kHz→48kHz), and much higher audio reconstruction quality. Specifically, to outperform Encodec under a similar bitrate, DAC adopts the multi-scale STFT discriminators to reduce the periodicity artifacts and adds the Snake (Ziyin et al., 2020) activation function to introduce periodic inductive biases inspired by the BigVGAN (Lee et al., 2022b) architecture. Although the code factorization technique of DAC also greatly reduces the quantization errors for much better reconstruction, discrete tokens are not necessary for diffusion-style models. Therefore, we remove the residual vector quantizer (RVQ) (van den Oord et al., 2017; Gray, 1984) from the DAC and trained with the variational autoencoder (VAE) (Kingma, 2013) objective (which adds a KL-regularization to encourage latents to be normally distributed). This significantly boosts the reconstruction performance especially at more compressed frame rates (25Hz).

Video. Long-prompt MetaCLIP fine-tuned from MetaCLIP (Xu et al., 2023) is used to extract a 1024-dimension embedding for each frame in a video. Since the frame rate of the video might not match that of the audio, we take the nearest visual frame for each audio frame. The resampled sequence is then projected to the DiT model dimension with a gated linear projection layer and added to the audio features frame by frame. Adding visual and audio features frame by frame improves video-audio alignment compared to concatenating features along the time dimension, because the former provides direct supervision of video-audio frame alignment. We have also explored reconstruction-based features extracted from a video-autoencoder for conditioning, which is expected to preserve more video details compared to contrastive features. However, the results were significantly worse and also slows down training due to the large feature dimension per frame. We concluded that the Long-prompt MetaCLIP features trained with a contrastive objective (Oord et al., 2018; Radford et al., 2021) encodes higher level semantic information that eases learning while keeping sufficient low-level details to capture the timing of each motion for the model to produce motion-aligned sound effects.

Audio context. We follow the Voicebox (Le et al., 2023) and Audiobox (Vyas et al., 2023) frameworks and condition on partially masked target audio, which we coined *audio context*. This enables the model to infill (or out-fill, depending on where the mask is) audio that is coherent with the context. Without conditioning on context, the audio would sound incoherent and change abruptly when stitching together audio segments generated independently given only the video, especially when audio contains heavy ambient sound or music. Audio context is also represented as a DAC-VAE feature sequence, and is concatenated with the noised audio latent along the channel dimension frame by frame. For masked frames, we replace it with zero-vectors. To perform audio generation without any audio context, we simply input a zero-vector sequence for audio context.

Text. We use text to provide additional guidance on target audio quality, sound events, and music style if music is present, which we collectively refer to as the *audio caption*, details of which is described in Section 6.2.4. T5-base (Raffel et al., 2020) is used to encode an audio caption into a sequence of 756-dimensional features, where sequence length is capped at 512 tokens. We insert a cross-attention layer right after the self-attention layer and before the feed-forward layer in each DiT transformer block for conditioning.

Putting it together, we denote an N_{aud} frame-long sample at flow-step t as $\mathbf{X}_t \in \mathbb{R}^{N_{aud} \times 128}$, and conditioning input as $\mathbf{c} = \{\mathbf{c}_{vid}, \mathbf{c}_{ctx}, \mathbf{c}_{txt}\}$, where $\mathbf{c}_{vid} \in \mathbb{R}^{N_{aud} \times 1024}$ is the resampled Long-prompt MetaCLIP features, $\mathbf{c}_{ctx} \in \mathbb{R}^{N_{aud} \times 128}$ is the masked DAC-VAE features that serves as audio context, and $\mathbf{c}_{txt} \in \mathbb{R}^{N_{txt} \times 756}$ is the T5 text feature sequence of N_{txt} tokens long. At each step, the model predicts a velocity $u(\mathbf{X}_t, \mathbf{c}, t; \theta) \in \mathbb{R}^{N_{aud} \times 128}$ that can be used to estimate $x_{t+\Delta t}$ during inference. We use an additional subscript to index a subsequence when necessary. For example, $\mathbf{c}_{vid,15:25}$ denotes a 10-frame segment starting on the 15th frame (0-based).

6.1.4 Inference: One-shot Generation

During training, each conditioning input (video, audio context, text) is dropped-out independently with some probabilities. This enables the model to perform (1) video-to-audio (V2A) generation (dropping out text and audio context), (2) text-instructed video-to-audio (TV2A) generation (dropping out audio context), (3) video-to-audio infilling or extension (dropping out text), and (4) text-instructed video-to-audio infilling or extension, with a single model by simply changing the conditioned inputs.

6.1.5 Inference: Audio Extension

Due to memory constraint and training efficiency considerations, training data is capped at a predetermined length. To generate high quality and coherent long-form audio for videos whose lengths are beyond the cap, we consider two algorithms: *segment-level autoregressive generation* and *multi-diffusion* (Bar-Tal et al., 2023).

Given a long video, we first split the video into *overlapping* segments, and assume text captions are available for all segments. At a high level, both algorithms run inference on individual segments first, and then consolidate the prediction. Information can propagate across segments through overlapping frames. Without loss of generality, we assume each segment is n_{win} frames long, and the end times of consecutive segments differ by n_{hop} frames. Two consecutive segments overlap by $n_{ctx} = n_{win} - n_{hop}$ frames. For a video $\mathbf{c}_{vid}$ of N frames, there will be $J = \lceil N/n_{hop} \rceil$ segments, and the j-th segment spans $[n^{(j)}_{start}, n^{(j)}_{end}] = [max(0, (j-1)n_{hop} - n_{ctx}), min(N, jn_{hop})]$ where $j \in [J]$. We denote the text caption for j-th segment as $\mathbf{c}^{(j)}_{txt}$, and the consolidated prediction at flow step t_i as $\mathbf{X}^{(j)}_{t_i}$, assuming the same T-step flow time schedule $\{t_i\}_{i=1}^T$ are used for all segments ($t_1 = 0$, $t_T = 1$, and $t_i < t_{i+1}\ \forall i$). Note that $\mathbf{X}^{(j)}_0$ is the noise drawn from the prior p_0 and $\mathbf{X}^{(j)}_1$ is the predicted audio for the j-th segment.

We introduce two extension methods below: segment-level autoregressive generation and multi-diffusion. Multi-diffusion achieves slightly better results empirically, which we use as the default method.

Segment-level autoregressive generation. This algorithm emulates autoregressive generation of language models but at the segment level. It generates one segment at a time conditioned on the information from the last n_{ctx} frames of the previous segment. Given the trajectory $\mathbf{X}^{(j)}_{t_i}$ from the j-th segment, we consider passing information through two routes when generating $\mathbf{X}^{(j+1)}_{t_i}$: *context conditioning* and *trajectory regularization*.

The first route, context conditioning, is to simply update the audio context $\mathbf{c}^{(j+1)}_{ctx}$ using the prediction from the previous segment. Specifically, we set $\mathbf{c}^{(j+1)}_{ctx} = [\mathbf{X}^{(j)}_{1,-n_{ctx}:}; \mathbf{0}]$, where $\mathbf{X}^{(j)}_{1,-n_{ctx}:}$ denotes the last n_{ctx} frames of $\mathbf{X}^{(j)}_1$, and $\mathbf{0}$ is a zero matrix of shape $n_{hop} \times C$.

The second route, trajectory regularization, is to pass information through the flow trajectory $\mathbf{X}^{(j+1)}_{t_i}$. We *blend* the trajectory from the previous segment with the current one at each flow step t_i with a weighting function $w \in \mathbb{R}^{n_{ctx}}_{\geq 0}$, such that the trajectory of the current segment does not diverge too much from the previous one for the overlapping segment. Algorithmically, let $\hat{\mathbf{X}}^{(j+1)}_{t_{i+1}} = \text{ODE-Solve}(u, \mathbf{X}^{(j+1)}_{t_i}, \mathbf{c}^{(j+1)}, t_i, t_{i+1})$ be the solution from the ODE solver at t_{i+1} taking the initial condition $\mathbf{X}^{(j+1)}_{t_i}$ at t_i with conditioning input $\mathbf{c}^{(j+1)} = \{\mathbf{c}^{(j+1)}_{vid}, \mathbf{c}^{(j+1)}_{ctx}, \mathbf{c}^{(j+1)}_{txt}\}$ and flow model u. We update the state to $\mathbf{X}^{(j+1)}_{t_{i+1},0:n_{ctx}} = w \odot \hat{\mathbf{X}}^{(j+1)}_{t_{i+1},0:n_{ctx}} + (1-w) \odot \mathbf{X}^{(j)}_{t_{i+1},-n_{ctx}:}$ for the overlapping frames and keep other frames intact $\mathbf{X}^{(j+1)}_{t_{i+1},n_{ctx}:n_{win}} = \hat{\mathbf{X}}^{(j+1)}_{t_{i+1},n_{ctx}:n_{win}}$, where $\odot$ denotes point-wise product. In the end, we also update the audio of the overlapping frames with $\mathbf{X}^{(j+1)}_{1,0:n_{ctx}}$, the consolidated prediction from the later segment. The blending function w dictates how trajectories are mixed for each frame. To encourage a smooth transition, we consider a linear ramping function

$w_n = n/n_{ctx}$ for $n \in [n_{ctx}]$, such that weights are higher on the previous segment for frames closer to it and vice versa.

To further boost the performance of segment-level autoregressive generation, we also explore *segment-level beam search*. At each segment, multiple candidates are generated and the resulting partial generations are ranked and pruned by a scoring model. Those top candidates are used as prefixes to generate multiple candidates for the next segment.

Multi-diffusion. Inspired by its success on leveraging a diffusion model trained on 512×512 images to generate panorama images that are 9 times wider (512×4608) and application to video upsampling described in Section 3.1.5, we explore multi-diffusion for audio extension, which is virtually the audio version of the panorama generation problem. At a high level, multi-diffusion solves the ODE for each step (*e.g.*, from t_i to t_{i+1}) in parallel for all segments, consolidate the prediction, and then continues with the next ODE step. This is in contrast to segment-level autoregressive generation, which solves the ODE for one segment completely (*i.e.*, from $t = 0$ to $t = 1$, potentially with multiple steps) before solving it for the next segment.

We next describe the algorithm in detail. To initialize, $\mathbf{X}_0$ is first drawn from a prior distribution p_0. At each step t_i, $\mathbf{X}_{t_i}$ is first split into chunks $\{\mathbf{X}_{t_i}^{(j)}\}_j$ based on the predefined segmentation. $\hat{\mathbf{X}}_{t_{i+1}}^{(j)}$ is computed as ODE-Solve$(u, \mathbf{X}_{t_i}^{(j)}, \mathbf{c}^{(j)}, t_i, t_{i+1})$ for all j. Then we merge the prediction as $\mathbf{X}_{t_{i+1}} = \sum_j \text{zero-pad}(m^{(j)} \odot \hat{\mathbf{X}}_{t_{i+1}}^{(j)}, j)$. The function zero-pad$(\mathbf{X}^{(j)}, j)$ maps a segment of shape $n_{win} \times C$ back to a sequence of shape $N \times C$ (shape of the $\mathbf{X}$), by padding zeros based on the segment's position (*i.e.*, the j-th segment spans from $n_{start}^{(j)}$ to $n_{end}^{(j)}$; hence $n_{start}^{(j)}$ zeros are padded at the beginning and $N - n_{end}^{(j)}$ zeros at the end). The soft-masking function $m^{(j)} \in \mathbb{R}_{\geq 0}^{n_{win}}$ defines how much the j-th segment contributes to each frame it spans. We note that $\sum_j \text{zero-pad}(w^{(j)}, j) = 1$, where for each frame the contribution over all segments sums to 1. This completes one step of the flow. The same process repeats until we reach $t_T = 1$. Note that the weights w used in segment-level autoregressive generation is applied to the overlapping frames (n_{ctx} frames), while the soft-masking functions $\{m^{(j)}\}_j$ are defined for all the frames each segment spans.

Figure 29 shows two variants of soft-masking functions (zero-padded) for two segments on the right column. The soft-masking functions m are derived from window functions $\hat{m} \in \mathbb{R}_{\geq 0}^{n_{win}}$ (the left column of the same figure, also zero-padded). Specifically, we have $\text{zero-pad}(m^{(j)}, j) = \left(\text{zero-pad}(\hat{m}^{(j)}, j) / \sum_{j'} \text{zero-pad}(\hat{m}^{(j')}, j')\right)$ given window functions $\{\hat{m}^{(j)}\}_j$. The uniform window function ($\hat{m}^{(j)} = \mathbf{1}$) in the top row is equivalent to the method proposed in Bar-Tal et al. (2023). However, we observe that this sometimes results in abrupt transition at the boundaries (especially for smaller models), because the prediction from two neighboring segments might still be considerably different on overlapping frames at $t = 1$. Hence the transition from taking 100% of the first segment to 50-50 from both segments leads to discontinuity, as shown on Figure 29 top row. Using the triangular window function (*i.e.*, Barlette window, $\hat{m}_n^{(j)} = \frac{2}{n_{win} - 1}\left(\frac{n_{win} - 1}{2} - \left|n - \frac{n_{win} - 1}{2}\right|\right)$) from the bottom row results in smoother transition, similar to what we observed in chunk-level autoregressive generation with the linear ramping function.

6.1.6 Model, Training, and Inference Configurations

The DiT model has 36 layers and attention/feed-forward dimension of 4,608/18,432, which has 13B parameters in total (excluding Long-prompt MetaCLIP, T5, and DAC-VAE). In pre-training, videos are capped at 30 seconds (750 frames) and randomly chunked if lengths exceed. In finetuning, we randomly sample 10s and 30s segments from the video. Fully-sharded data parallelism is used to fit the model size.

The model is trained in two stages: *pre-training* and *fine-tuning*, using the same objective, but different data (described in Section 6.2.2 and Section 6.2.3) and optimization configurations. For pre-training, the effective batch size is 1,536 sequences, and each sequence is capped at 30 seconds (750 tokens). The model is pre-trained for 500K updates, taking 14 days on 384 GPUs, using a constant learning rate of 1e-4 with a linear ramp-up for the first 5K steps.

For fine-tuning, the effective batch size is 256 sequences, also capped at 30 seconds. The model fine-tunes the pre-trained checkpoint for 50K updates on 64 GPUs, which takes one day. The learning rate linearly ramps up

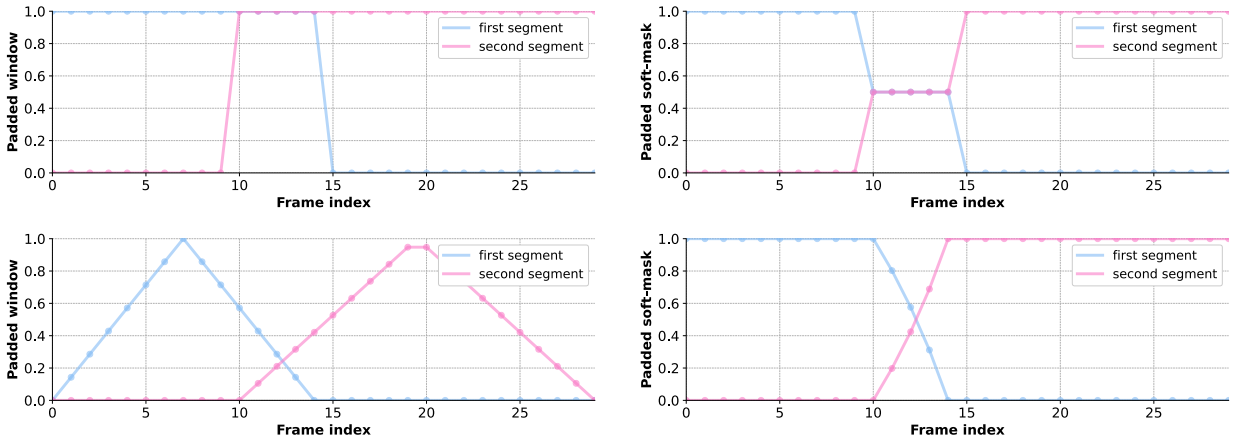


Figure 29 Comparing uniform weighting (top) and triangle window based weighting (bottom) for multi-diffusion merging with an example of $l_{win} = 20$ and $l_{ctx} = 5$. The left column shows the unnormalized weights contributed from the first segment ([0, 15]) and the second ([10, 30]), and the right column shows the normalized weights. With triangle window, the transition in the overlapping region ([10, 15]) is much smoother.

to 1e-4 for the first 5K steps, and then linearly decay to 1e-8 for the remaining steps. An exponential moving average (EMA) checkpoint with a decay of 0.999 is accumulated during finetuning and used for inference. AdamW optimizer with a weight decay of 0.1 and bf16 precision are used for both pre-training and fine-tuning.

To leverage classifier-free guidance (CFG) for inference, during training we drop conditioning inputs altogether (video, text, audio context) with a probability of 0.2. To enable both audio generation and audio extension, the masked audio is dropped (*i.e.*, completely masked) with a probability of 0.5, and otherwise masked between 75% to 100%. To reduce the reliance on either modality, text and video input are dropped independently with a probability of 0.1 for each.

For inference, the midpoint solver with 64 steps is used. We did not find using adaptive dopri5 solver or increasing the number of steps to boost the performance. We use CFG with a weight of 3 on unconditional vector fields and further conduct reranking with 8 candidates per sample. Quality score of 7.0 is used for sound effect generation, and 8.0 for joint sound effect and music generation. For audio extension, we use dynamic guidance (Wang et al., 2024c) with a weight of 3 and multi-diffusion with a triangle window of $n_{win} = 40$, $n_{hop} = 30$ and $n_{ctx} = 10$ by default.

6.2 Data

6.2.1 Audio-Visual Data Categorization

The relationship between audio and visual components are complex. For example, some sound effects like footsteps correlate with the low-level motion of objects in the scene, while others like canned laughter in sitcoms correlates with high-level semantics (*e.g.*, when something funny happens). Similarly, music in the video can either be played by someone in the scene, or be added during post production to enhance the storytelling power.

We classify audio by two axes and show an overview in Table 22. The first axis is audio type, which is divided into voice (speech and singing), non-vocal music, and general sound. An audio event detection (AED) model (Gemmeke et al., 2017) is used for automatic classification, where each sample may contain multiple types of audio. This axis solely considers the audio.

The second axis is diegetic/non-diegetic (Dykhoff, 2012; Stilwell, 2007). Diegetic audio components refer to those that *can be heard at the scene* (crowd talking, newscasters speaking, music of a live band performing, birds chirping) and have a causal relationship with the video, while non-diegetic, such as narrations in documentaries, background music in movies, or canned laughter in sitcoms, do not. Note that diegetic sounds can be either on-screen or off-screen (birds chirping in the forest is diegetic even when birds are not seen),

real or created in post-production (*i.e.*, Foley sound). A video can also contain both diegetic and non-diegetic sounds, which is especially common in professionally-produced videos like movies (Tan et al., 2017). We leverage a contrastive audio-video-text pre-training model (CAVTP) to determine how likely an audio sample is diegetic given the corresponding video. Because this model is trained on data that contain mostly diegetic sounds, the audio and the video embeddings are closer and have higher cosine similarity if the audio is diegetic and matches the video content.

	Voice	Non-vocal music	General sound
Diegetic	on-screen or off-screen people speaking	on-screen or off-screen live music performance	on-screen wave splashing, off-screen barking
Non-diegetic	narration	background music	canned laughter, riser

Table 22 Audio data classification and examples from each category. We focus primarily on diegetic general sounds, non-diegetic sound effects, and instrumental music.

To generate each class of sounds correctly, a model would learn different levels of relationships between audio and conditioning input.

1. Diegetic on-screen sounds have very strong correspondence between video and audio, where what sound should be heard when is deterministic. This demands stronger video understanding and dense action recognition capabilities from a model. The difficulty depends on how dense and structured the events are, where general sounds are overall easier than music and speech (*e.g.*, generating golf club hitting the ball is easier than generating a person playing a guitar matching the chords one presses).
2. Generating diegetic off-screen audio requires understanding what sounds may occur in what environments (*e.g.*, birds chirping are possible in a forest scene) and logical orders between events (*e.g.*, crowd cheering is likely to occur after, rather than before one performs a difficult trick). Hence, compared to on-screen sounds, it demands stronger reasoning capabilities.
3. Non-diegetic audio is correlated with the video at the semantic level. For example, background music needs to match the mood, and risers are often used to create a sense of tension or anticipation. This demands the deepest level of understanding beyond understanding the physics of the world and requires reasoning and modeling human emotions.

In this work, we focus primarily on *diegetic general sounds, non-diegetic sound effects, and instrumental music*. Generating diegetic speech is challenging when transcripts are not provided and when there are artifacts from the generated video. We also omit non-diegetic speech as it can be created with text-to-speech synthesis systems if scripts are given. As there are not only correlations between video and audio, but also between different classes of audio, we choose to build a model that generates *all classes of audio jointly*, instead of having separate models for diegetic/non-diegetic vocal/music/sound effects.

6.2.2 Pre-training Data

Pre-training aims to learn the structure of audio and alignment between audio and video/text from large quantities of data, including both low quality and high quality audio samples. Below we describe filtering criteria based on AED tags and CAVTP scores for pre-training data selection.

We start by sourcing data from a large volume and using the AED model to tag audio events for each sample based on the Audioset (Gemmeke et al., 2017) ontology that has 527 classes. We then drop any videos where silence is the dominant class.

Next, we map the remaining events to one of three categories: speech, non-vocal music (music), or general sound (sound). An event is mapped to "voice" if any of the "speech" and "singing" subclasses in the AudioSet ontology is tagged. Similarly, an event is mapped to "music" if any of the music subclasses in the ontology is detected. If any other subclass not mentioned above is deteced, the event is mapped to "sound". This means that an utterance can contain any combination of these three classes. After grouping samples by audio types, we use CAVTP score, which is the cosine similarity between audio and video emeddings from CAVTP, to

categorize an utterance into one of the buckets as described in Table 23. The thresholds are determined based on manual inspection.

Category	AED detected	CAVTP cosine similarity threshold
Diegetic	sound voice voice + sound	> 0.2
	music music + sound voice + music + sound	> 0.3
Non-diegetic	music	< 0.1
	sound + music	> 0.1 and < 0.25
Mixed	sound + voice + music	> 0.1 and < 0.25

Table 23 Pre-training data categorization. We show the categories, AED tags, and CAVTP thresholds used for grouping.

Table 24 shows the statistics for each category used for pre-training. We consider the diegetic-or-mixed audio (filtered by CAVTP score) along with a small proportion of non-diegetic background music. We prioritize general sound (filtered by AED tags), as learning low-level physics is challenging and the errors of which are most noticeable.

To reduce noises from the visual modality, we applied a series of quality filters to remove videos that contain text with OCR (Optical Character Recognition) (Liao et al., 2020), are static or are of low resolution(<480 px). The length of the videos have been constrained to be between 4s and 120s. Additionally, we leveraged copy detection embeddings (Pizzi et al., 2022) for visual deduplication.

Type	#samples (M)	#hours (K)
Sound	$\mathcal{O}(100)$	$\mathcal{O}(1{,}000)$
Music	$\mathcal{O}(10)$	$\mathcal{O}(100)$
Sound+Music	$\mathcal{O}(10)$	$\mathcal{O}(100)$
Sound+Voice	$\mathcal{O}(10)$	$\mathcal{O}(100)$
Sound+Music+Voice	$\mathcal{O}(10)$	$\mathcal{O}(100)$
Total	$\mathcal{O}(100)$	$\mathcal{O}(1{,}000)$

Table 24 Pre-training data for Movie Gen Audio by audio type. Only diegetic and mixed diegetic/non-diegetic videos are used in pre-training.

6.2.3 Finetuning Data

Once pre-training learns the foundational knowledge of audio structure and cross-modal alignment, finetuning aims to align the model output with what we expect in *cinematic soundtracks for videos*, which is very different from general recordings like those directly dumped from low-end devices (*e.g.*, cellphones or security cameras). Concretely, cinematic soundtracks are expected to be recorded with professional microphones and undergo post-production like mixing and mastering, in order to reduce unwanted noises (*e.g.*, pop noise, wind blowing to microphone) and balance the presence of various audio events as well as the level of background music (*e.g.*, suppressing ambient noise and irrelevant off-screen sounds, enhancing audio events like explosion or conversation relevant to storytelling, and mixing ambient music with fade-in/fade-out).

Broadly speaking, cinematic soundtracks differ from low-quality recordings in two aspects: audio quality (how it sounds) and sound design (what sounds to include). To bridge the gap, we include two sources of finetuning data (summarized in Table 25). First is the *cinematic split*, which includes clips that are professionally produced that often contains both diegetic and non-diegetic sounds (ambient and theme music). Clips with

vocals are excluded. An audio-visual cinematic classifier and an AED model are used for automatic data filtering, followed by human annotation for selection. The second is the *high quality audio split*, which includes high quality music (O(10)K hours) and sound effects (O(10)K hours) without videos. Such data are available in larger quantities compared to the first split, and can be used to boost the audio quality. During fine-tuning, cinematic videos and high-quality audio are mixed with a 10 batches to 1 batch ratio.

Split	#samples (K)	#hours (K)
Cinematic video (video+audio)	$\mathcal{O}$(100)	$\mathcal{O}$(1)
High-quality audio (audio-only)	$\mathcal{O}$(1,000)	$\mathcal{O}$(10)
Total	$\mathcal{O}$(1,000)	$\mathcal{O}$(10)

Table 25 Finetuning data for Movie Gen Audio. We show the split and statistics of finetuning data used to make the audio generations closer to cinematic soundtracks.

6.2.4 Caption Structure and Synthetic Caption

The caption is composed of four parts: audio quality, voice and music presence, sound caption (Kim et al., 2019), and music style caption (Manco et al., 2021). We use several models to create synthetic captions for all training data. Table 26 shows two examples.

Given the scale, we leverage several models to build synthetic captions for all training samples. Audio quality is a real-value number between 1 and 10 labeled by an audio quality prediction model (annotations are collected in a similar way to LAION aesthetic (Schuhmann et al., 2022), where 10 means the highest quality and 1 means the lowest). Voice and music presence are determined by the previously described AED model, where the former takes the binary output using a predetermined posterior threshold, and the latter is represented with AED posterior probability given the ambiguity (certain cinematic sound effects like risers may also be considered music). Sound caption is derived from a general audio caption model that provides free-form description about the sound. To boost the controllability on music, we further deploy a music caption model to add more details such as mood and genre. Note that music caption is appeneded regardless of whether the audio includes music or not. We find using both music probability from AED and music caption from music caption model provide the best control, because music caption model is trained on mainly music samples and tend to hallucinate even when music is absent.

Each sample is split into both 10-second chunks and 30-second chunks, and then captioned. Note that they are still segments of different lengths, which are from the last chunk of a sequence. During training, the 10-second and 30-second chunks are sampled with a 5 batches to 1 batch ratio.

Example captions for Movie Gen Audio
This audio has quality: 8.0. This audio does not contain speech. This audio does not contain vocal singing. This audio has a description: "gentle waves lapping against the shore, and music plays in the background.". This audio contains music with a 0.90 likelihood. This audio has a music description (if applicable): "A beautiful, romantic, and sentimental jazz piano solo.".
This audio has quality: 7.0. This audio does not contain speech. This audio does not contain vocal singing. This audio has a description: "fireworks exploding with loud booms and crackles.". This audio contains music with a 0.01 likelihood. This audio has a music description (if applicable): "A grand, majestic, and thrilling orchestral piece featuring a massive symphony orchestra with a soaring melody and pounding percussion, evoking a sense of awe and wonder.".

Table 26 Two example captions for Movie Gen Audio. The blue part describes audio quality, the orange part controls presence of speech, vocal, and music, the teal part controls general sounds, and the pink part provides fine-grained control of music styles.

6.3 Evaluation

We evaluate soundtrack generation mainly on audio quality, audio-video alignment, and audio-text alignment. We prioritize alignment to video over alignment to text, because text is used as a supplement and may not capture all the details in the video. Moreover, text input is not presented to the viewers in the final output. Table 27 summarizes the metrics. Correlations between subjective and objective metrics are studied in Appendix C.3.

	Subjective	Objective
Audio quality	Overall quality (Ovr.) Naturalness (Nat.) Professionalness (Pro.)	Audio quality score (AQual)
Video alignment	Diegetic sound correctness (Corr.) Diegetic sound synchronization (Sync.) Non-diegetic music mood alignment (Mood) Non-diegetic music motion/scene alignment (Action)	ImageBind score (IB)
Text alignment	Precision Recall	CLAP score

Table 27 Movie Gen Audio evaluation metrics. Audio quality measures the standalone quality of the generated audio, while video and text alignment measure how well the generated audio aligns with the input video and text respectively.

6.3.1 Metrics

Audio quality. We aim to evaluate how *natural* (free of artifacts) and *professional* (*e.g.*, volume balance, crispness) an audio sample sounds. For subjective tests, a pairwise protocol is adopted which asks raters to choose which audio has better overall quality and on those two axes, where the pair of audio samples are generated conditioned on the same video and text prompts when applicable. We report the Net Win Rate (NWT), defined as "win% - lose%" for pairwise comparisons. NWT ranges from −100% to 100%. Details are described at the end of the section.

For objective metric, the audio quality score (AQual) predicted by the model described in Section 6.2.4 is used as an automatic metric. We note that the model tends to assign higher scores to samples with music; hence the metric should not be used when comparing samples with music and those without music.

Note that we do not adopt Frechét audio distance (FAD) (Kilgour et al., 2018) or KL-divergence (KLD) metrics that are often reported in text-to-audio generation (Liu et al., 2023b; Vyas et al., 2023) because these metrics are not applicable to generated videos that do not have corresponding audio, which we mainly evaluate on in this paper.

Video alignment. This measures how well the audio is aligned with the video. For diegetic sound, we measure *correctness* and *synchronization*. Correctness reflects whether the *right type* of audio is generated with respect to the scene and the objects in the video (*e.g.*, dog barking versus cat meowing). Synchronicity on the other hand focuses on whether the audio is generated at the *right time* matching the motions in the video. For non-diegetic background music, we measure how well it supports the *mood* of the scene and how well the *score synchronizes* with the on-screen actions and scene changes (*i.e.*, action scoring). Similarly, a pairwise protocol is adopted for subjective tests and net win rate is reported.

For automatic metrics, the ImageBind (IB) score is used for measuring the alignment between video and diegetic sounds, as used in Mei et al. (2023). As mentioned earlier, when non-diegetic music is present in the video, the score usually decreases regardless whether the mood and the score matches because the model is trained on mostly diegetic data without non-diegetic music.

Text alignment. Finally, we measure *precision* (percentage of generated audio events that are in the text caption) and *recall* (percentage of audio events from the caption that are generated in the audio). We note that we focus more on recall than on precision, as the caption might not include all the acoustic events that

are supposed to be heard in a video. In terms of the subjective tests, raters are asked to rate on a scale from 1 to 5 for precision and recall. We adopt the standalone protocol instead of a pairwise one because text alignment is more objective and is easier to rate in absolute scale. For objective test, we measure this with CLAP score (Wu et al., 2023d), which is commonly used for text-to-audio generation and does not distinguish hallucination and missing errors.

Computing net win rate The reported subjective preference metric is Net Win Rate. For a given model pair, NWT is computed as follows: each item is evaluated by three raters. For each item evaluated, we take the mean of the preference between model A and model B (+1 if the model A is preferred, 0 if a tie and -1 if the model B is preferred). These are the *consensus* scores for each item. We then average these consensus scores across all items to obtain a net win rate of A (this is the expected fraction of items where model A is preferred minus the fraction of items where model B is preferred). To obtain 95% confidence intervals around Net Win Rate, we bootstrap resample the item-level consensus scores 1,000 times, compute Net Win Rate for each, and take difference between the 2.5%-ile and 97.5%-ile of the Net Win Rate as the 95% confidence interval. The net win rate of A *vs*. B ranges from -100% to 100%.

6.3.2 Audio Generation Benchmarks

To thoroughly evaluate audio generation, we consider multiple existing video sources including both real and generated videos, and propose and hope to release a benchmark Movie Gen Audio Bench which contains high quality videos generated by Movie Gen Video that cover a wide spectrum of audio events, and human reviewed sound and music captions for those videos.

We group videos into two categories: *single-shot* and *multi-shot*. Single-shot videos are available in larger quantities and cover a wider spectrum of sound effects, which are suitable for testing robustness and generalization. Multi-shot videos, extracted from short films, contain scene transitions and have stronger sentiment and deeper narratives than single-shot videos. Hence, they are suitable for evaluating video-music alignment and sound design perspectives, such as when music enters, how music evolves with the story and aligns with the cuts, and whether music and sound effects are mixed harmonically. We describe the composition of single-shot and multi-shot benchmarks next and Table 28 provides a summary.

Benchmark	Video sources	Type	Shot	Num. of samples	Max length
SReal	VGGSound	Real	Mostly single	51	10s
SGen	Runway Gen3, OpenAI Sora	Gen	Single	151	10s
Movie Gen Audio Bench	Movie Gen Video	Gen	Single	538	10s
MGen	Movie Gen Video, OpenAI Sora	Gen	Multi	107	15s

Table 28 Movie Gen Audio evaluation datasets. We use "SFX" or "SFX+music" suffix to indicate what version of text captions is used.

Single-shot. This includes VGGSound (Chen et al., 2020), OpenAI Sora (OpenAI, 2024), Runway Gen3 (RunwayML, 2024), and our proposed Movie Gen Audio Bench.

- VGGSound contains real videos and is widely used for training and evaluating video-to-audio generation models (Mei et al., 2023; Luo et al., 2024; Xing et al., 2024). However, we discovered that there are many duplicates or near duplicates (*e.g.*, with added static watermark or text) between the training and evaluation split, and some testing videos are static. We perform deduplication based on video embeddings, and manually review test sets to select 51 samples that are not static, do not contain diegetic speech, and mostly have motion synchronized sounds.
- OpenAI Sora has been used to demonstrate video-to-audio generation for generated videos (Xing et al., 2024; Mei et al., 2023), but the number of available videos is small and of limited domain. Hence, it is not suitable for being used alone as a benchmark. We review those used in text-to-video comparison (Appendix C.2) and selected 43 samples for audio generation evaluation.
- Movie Gen Audio Bench is the new benchmark dataset we create using Movie Gen Video. It includes 538 videos and is designed to cover various ambient environmens (*e.g.*, indoor, urban, nature, transportation) and sound effects (*e.g.*, human, animal, objects). This is the first large scale synthetic

benchmark for evaluating video-to-audio generation. To create this benchmark, we first define an ontology with 36 audio categories and audio concepts for each category (*e.g.*, expression → {cry, laugh, yell}), with a total of 434 concepts. Llama3 is next used to propose video prompts for each audio concept, and Movie Gen Video is used to generate videos given these prompts. We next review the generated videos to exclude those with artifacts that would severely impact the judgement of whether an audio fits the video, resulting in the final 538 videos.

- Runway Gen3 contains 108 synthetic videos. It is created with a similar process as Movie Gen Audio Bench but with a subset of prompts. The goal is to include synthetic videos from different models that may contain different types of artifacts and artistic styles, in order to test the robustness of video-to-audio generation models.

We group them into three sets based on video properties: "SReal" is the real single-shot videos which includes VGGSound; "SGen" is the generated single-shot videos from prior video generation models which include OpenAI Sora and Runway Gen3; "Movie Gen Audio Bench" is the new generated single-shot benchmark we create, which we hope will facilitate future work for thorough text and video-to-audio generation benchmarking.

Multi-shot. We source 26 short films generated by OpenAI Sora and by Movie Gen Video to create this set. These videos are 30 second to 2 minute long, and are composed of multiple related shots. To compare our model with baseline methods, many of which have length limitations and do not support audio extension, we chunk these videos into 15-second segments and discard last segments shorter than 10 seconds, which results in 107 segments in total. The combined set is referred to as MGen.

Text prompt creation for SFX and SFX+music generation. For all the video samples, we use Llama3 (Dubey et al., 2024) to propose 5 sound and music captions for each video given its video caption, and manually select the best sound and music caption for each video. To generate non-diegetic music and sound effects jointly, we set the prompt to "This audio contains music with a 0.90 likelihood." for the music presence part of the text caption (see Table 26). In contrast, the likelihood is set to 0.01 if music is undesired. For baseline models that take text prompt as input, we use sound caption as input when generating sound effects only, and concatenate sound and music caption when generating sound effect and music jointly.

6.4 Results

6.4.1 Comparisons to Prior Work

We present qualitative and quantitative results of sound effect generation, joint sound effect and music generation, and long-form generation with audio extension in this section.

Sound effect generation. We first compare Movie Gen Audio with 4 open-sourced models (Diff-Foley (Luo et al., 2024), FoleyCraft (Zhang et al., 2024), VTA-LDM (Xu et al., 2024a), Seeing&Hearing (Xing et al., 2024)) and 2 blackbox commercial models (PikaLabs (Pika Labs), ElevenLabs (ElevenLabs)) on sound effect generation for single shot videos. Among these, Seeing&Hearing and PikaLabs support video and optional text input (denoted as TV2A when text is used and V2A otherwise), ElevenLabs supports text input (T2A), and others support video input (V2A). More details about the baselines can be found in Section 7.5.

Figure 30 shows 5 samples on Movie Gen Audio Bench. Table 29 presents the pairwise subjective evaluation results on audio quality and video alignment. Additional metrics can be found in Appendix D.1.1. At a high level, Movie Gen Audio outperforms all baselines on all metrics by a large margin: with 33.8% to 70.4% on synchronization, 27.5% to 82.2% on correctness for all videos, and 31.3% to 82.1% on overall quality for generated videos. Compared to the commercial baselines, Movie Gen Audio wins even more on generated videos, demonstrating its robustness. In terms of audio quality, we highlight that Movie Gen Audio wins more on *professionalness* than on *naturalness*, indicating that Movie Gen Audio learns to not only generate realistic sounds but also professionally produced sounds, which leads to higher overall quality.

Among the baselines, commercial models generally outperforms open-sourced models in audio quality, while remaining similar in audio-video alignment. Surprisingly text-based ElevenLabs model achieves similar performance to other baselines on video synchronization. This is likely because text-based model can still perform well for videos that contain mostly ambient sounds, and for challenging videos with dense actions, none of the baseline can generate well-aligned audio. Additionally, we observe that models using text prompts

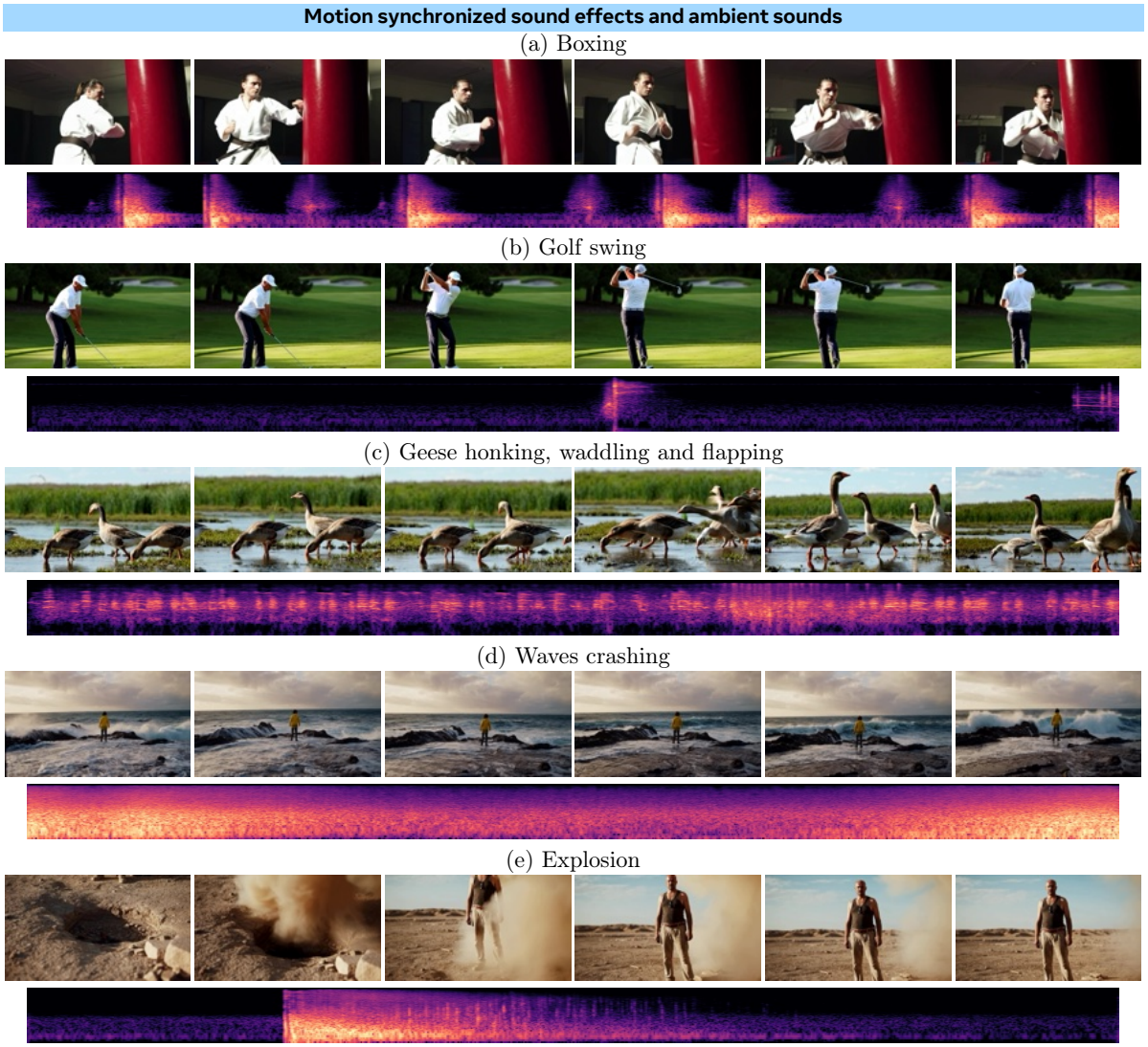


Figure 30 Video to SFX generation samples of Movie Gen Audio. It generates sounds that are tightly synchronized with the motions. Videos are from Movie Gen Audio Bench. Videos in this Figure found at https://go.fb.me/MovieGen-Figure30.

generally achieve better performance compared to their original forms. This shows text captions provides complimentary information for guiding generation.

Sound effect and music generation. We next showcase Movie Gen Audio's ability to generate cinematic soundtracks for short films that also include non-diegetic music supporting the mood and synchronized with the visual actions. We evaluate Movie Gen Audio on MGen SFX+music, where music likelihood is set to 0.90 in text prompts. As for the baselines, we need models that support text input to prompt them for joint SFX and music generation, since V2A models only produce diegetic SFX most of the time. Seeing&Hearing (S&H) (Xing et al., 2024) and PikaLabs (Pika Labs) are the only two options, while ElevenLabs is another option with only text input. From preliminary testing, we find neither Pika nor ElevenLabs can generate SFX and music jointly, so we do not consider them in this experiment.

In addition to joint generation, we include baselines that mix separately generated sound effects and music with an SNR sampled uniformly from [-5, 5]dB. For sound effect generation, we include the open-sourced baselines used in the previous section. For music generation, we consider the open-sourced S&H using both video and text input, and an external text-to-music generation API that accepts only text input. Sound captions (quoted text of the orange part in Table 26) are used for sound effect generation if text prompt is

Dataset	Baseline	Type	Movie Gen Audio net win rate *vs.* baseline				
			Quality			Video-SFX Alignment	
			Ovr.	Nat.	Pro.	Corr.	Sync.
SReal SFX	Diff-Foley (Luo et al., 2024)	V2A	$76.6_{\pm 12.6}$	$48.1_{\pm 15.6}$	$79.5_{\pm 11.1}$	$61.6_{\pm 13.0}$	$46.1_{\pm 14.3}$
	FoleyCraft (Zhang et al., 2024)	V2A	$69.2_{\pm 14.1}$	$57.2_{\pm 16.3}$	$69.2_{\pm 14.1}$	$50.4_{\pm 13.4}$	$49.7_{\pm 17.0}$
	VTA-LDM (Xu et al., 2024a)	V2A	$32.9_{\pm 18.5}$	$31.5_{\pm 18.5}$	$38.2_{\pm 18.9}$	$47.4_{\pm 16.7}$	$50.4_{\pm 16.3}$
	Seeing&Hearing (Xing et al., 2024)	V2A	$85.8_{\pm 9.3}$	$83.6_{\pm 11.1}$	$85.8_{\pm 9.3}$	$63.6_{\pm 14.8}$	$63.7_{\pm 14.1}$
	Seeing&Hearing (Xing et al., 2024)	TV2A	$76.8_{\pm 11.1}$	$67.9_{\pm 15.2}$	$76.8_{\pm 11.1}$	$56.1_{\pm 17.4}$	$51.3_{\pm 18.7}$
	PikaLabs (Pika Labs)	V2A	$58.6_{\pm 15.2}$	$49.7_{\pm 16.3}$	$60.0_{\pm 14.1}$	$56.9_{\pm 14.1}$	$48.8_{\pm 18.1}$
	PikaLabs (Pika Labs)	TV2A	$41.9_{\pm 20.4}$	$31.9_{\pm 23.0}$	$41.9_{\pm 20.4}$	$35.8_{\pm 18.5}$	$34.2_{\pm 18.4}$
	ElevenLabs (ElevenLabs)	T2A	$13.2_{\pm 21.5}$	$8.7_{\pm 21.5}$	$13.2_{\pm 21.5}$	$27.5_{\pm 18.9}$	$35.0_{\pm 19.3}$
SGen SFX	Diff-Foley (Luo et al., 2024)	V2A	$78.7_{\pm 6.8}$	$76.2_{\pm 6.6}$	$78.5_{\pm 6.6}$	$82.2_{\pm 5.4}$	$70.4_{\pm 8.7}$
	FoleyCraft (Zhang et al., 2024)	V2A	$65.0_{\pm 8.7}$	$59.5_{\pm 8.5}$	$65.0_{\pm 8.6}$	$57.2_{\pm 7.7}$	$49.6_{\pm 10.0}$
	VTA-LDM (Xu et al., 2024a)	V2A	$77.7_{\pm 7.0}$	$63.8_{\pm 7.7}$	$76.8_{\pm 7.1}$	$61.7_{\pm 8.2}$	$58.2_{\pm 9.0}$
	Seeing&Hearing (Xing et al., 2024)	V2A	$82.1_{\pm 7.4}$	$76.9_{\pm 8.0}$	$82.6_{\pm 7.3}$	$63.6_{\pm 8.6}$	$33.8_{\pm 10.1}$
	Seeing&Hearing (Xing et al., 2024)	TV2A	$76.2_{\pm 7.1}$	$75.4_{\pm 7.1}$	$76.1_{\pm 7.3}$	$64.1_{\pm 7.9}$	$33.8_{\pm 10.1}$
	PikaLabs (Pika Labs)	V2A	$61.2_{\pm 10.6}$	$55.5_{\pm 10.7}$	$62.6_{\pm 9.6}$	$56.2_{\pm 12.5}$	$52.1_{\pm 12.7}$
	PikaLabs (Pika Labs)	TV2A	$53.6_{\pm 11.6}$	$46.0_{\pm 11.6}$	$54.5_{\pm 11.4}$	$44.6_{\pm 12.9}$	$39.4_{\pm 11.7}$
	ElevenLabs (ElevenLabs)	T2A	$49.7_{\pm 9.8}$	$45.3_{\pm 9.9}$	$47.3_{\pm 9.8}$	$31.8_{\pm 8.1}$	$35.5_{\pm 9.5}$
Movie Gen Audio Bench SFX	FoleyCraft (Zhang et al., 2024)	V2A	$71.4_{\pm 4.0}$	$60.7_{\pm 4.2}$	$71.9_{\pm 4.0}$	$57.4_{\pm 4.3}$	$53.3_{\pm 5.1}$
	Seeing&Hearing (Xing et al., 2024)	TV2A	$71.5_{\pm 4.0}$	$70.0_{\pm 3.9}$	$71.4_{\pm 3.9}$	$59.4_{\pm 4.4}$	$51.4_{\pm 5.3}$
	ElevenLabs (ElevenLabs)	T2A	$31.3_{\pm 5.6}$	$27.4_{\pm 5.4}$	$31.1_{\pm 5.5}$	$38.3_{\pm 5.1}$	$36.0_{\pm 6.0}$

Table 29 Sound effect generation pairwise subject evaluation. This table compares Movie Gen Audio with prior work on audio quality and video alignment. We report net win rate, which has a range [-100%, 100%], and its 95% confidence intervals. Positive values indicate Movie Gen Audio outperforms the baseline on the metric.

Dataset	Baseline method		Movie Gen Audio net win rate *vs.* baseline						
			Quality			Video-SFX Alignment		Video-Music Alignment	
	SFX Model	Music Model	Ovr.	Nat.	Pro.	Corr.	Sync.	Mood	Action
MGen SFX+music	*joint sound effect and music generation*								
	S&H TV2A		$89.9_{\pm 5.0}$	$82.4_{\pm 5.9}$	$89.9_{\pm 5.0}$	$76.1_{\pm 6.7}$	$81.3_{\pm 6.5}$	$67.5_{\pm 8.8}$	$71.8_{\pm 8.3}$
	separate sound effect and music generation								
	S&H TV2A	S&H TV2A	$67.7_{\pm 8.6}$	$69.3_{\pm 8.4}$	$66.4_{\pm 8.7}$	$48.6_{\pm 9.4}$	$42.3_{\pm 7.6}$	$59.3_{\pm 8.4}$	$65.1_{\pm 9.6}$
	S&H TV2A	External API T2A	$12.5_{\pm 11.8}$	$11.3_{\pm 11.4}$	$11.3_{\pm 11.9}$	$55.2_{\pm 9.8}$	$57.4_{\pm 9.0}$	$32.5_{\pm 12.3}$	$26.8_{\pm 9.3}$
	Diff-Foley V2A	External API T2A	$27.4_{\pm 11.1}$	$20.6_{\pm 11.1}$	$28.0_{\pm 10.9}$	$25.2_{\pm 11.1}$	$22.7_{\pm 8.4}$	$49.2_{\pm 9.3}$	$42.2_{\pm 11.0}$
	FoleyCraft V2A	External API T2A	$38.9_{\pm 10.0}$	$39.8_{\pm 10.3}$	$32.6_{\pm 10.6}$	$60.8_{\pm 8.7}$	$60.9_{\pm 10.2}$	$18.0_{\pm 11.2}$	$23.9_{\pm 10.7}$
	VTA-LDM V2A	External API T2A	$38.7_{\pm 11.5}$	$28.5_{\pm 12.8}$	$35.3_{\pm 11.4}$	$80.4_{\pm 6.1}$	$77.0_{\pm 7.9}$	$52.9_{\pm 9.2}$	$45.2_{\pm 7.2}$

Table 30 Sound effect and music generation pairwise subject evaluation. This table compares Movie Gen Audio with prior work on audio quality and video alignment. We report net win rate, which has a range [-100%, 100%], and its 95% confidence intervals. Positive values indicate Movie Gen Audio outperforms the baseline on the metric.

supported, and music captions (quoted text of the pink part in Table 26) are used for both music generation models. We show qualitative samples on both single- and multi-shot videos in Figure 31.

Table 30 presents the pairwise subjective evaluation results on audio quality and video alignment for both sound effects and music. Similar to the single-shot scenario, we outperform all baselines significantly across all aspects of alignment and quality. Notably, the margin by which we surpass the joint generation baseline S&H TV2A is even larger than in the sound effect-only case. Separately generating sound effects and music with S&H improves from joint generation, but it stills falls behind Movie Gen Audio. This highlights the limitations of existing public V2A models in cinematic content creation.

Although incorporating the high quality music generated by the external API greatly improves music quality, this approach still falls short compared to our proposed model, especially on the alignment metrics. There are two main reasons. Since music and sound effects are generated separately, the correlation between them (*e.g.*, music volume should be lowered when there are prominent sound effects) cannot be modeled. Moreover, because the external API is a text-to-music model that is entirely unaware of video, it cannot generate music capturing the scene and mood changes in the video.

Audio extension. Lastly, we evaluate Movie Gen Audio's ability to generate long-form audio using the audio

Figure 31 Video to SFX and music generation samples of Movie Gen Audio. Movie Gen Audio can generate cinematic music that supports the mood, aligns with visual scene transitions, and blends harmonically with sound effects. Videos in this Figure found at https://go.fb.me/MovieGen-Figure31.

extension methods described in Section 6.1.5. Figure 32 shows three long videos with cinematic soundtracks generated by Movie Gen Audio with audio extension.

For quantitative comparison, since there is no prior work on audio extension for video-to-audio generation, we compare against a simple stitching method where audio is generated independently for each segment and then stitched together. We use Movie Gen Audio as the base model (denoted as "Movie Gen Audio stitch"). Ideally, audio extension should be evaluated on long-form generated videos. However, there are limited sources (26 full videos from MGen). We also found raters having trouble staying focused comparing long videos. Both factors contribute to large variance on subjective tests. As a workaround, we probe whether audio extension leads to *smooth transitions* across segment boundaries, using shorter videos and generating both sound effect and music (SGen SFX+music). We set the segment size to 5.5 seconds, where each video from SGen is split into two segments. Single-shot generated video evaluation also enables us to use Seeing&Hearing as an additional baseline without stitching. On this evaluation set, we set $n_{hop} = 5.5$ seconds and $n_{ctx} = 5.5$ seconds. Comparison of different extension methods and configurations are presented in ablation studies in Section 6.4.2.

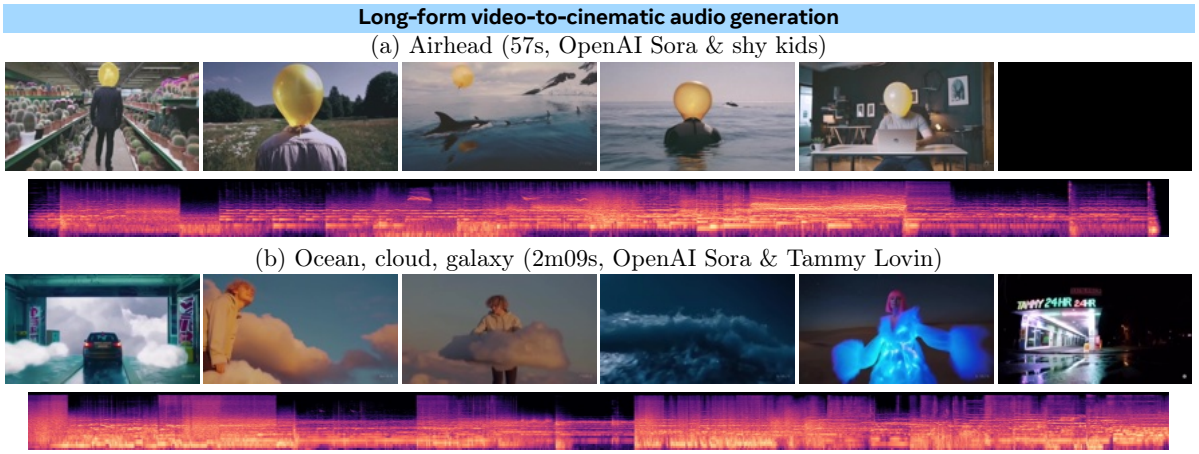


Figure 32 Examples of sound track generation for long videos with Movie Gen Audio. Movie Gen Audio can generate long-form coherent and cinematic soundtracks for videos up to a few minutes long. It learns to compose music that progresses with the story. For example, in Airhead, verse 1 starts at 0:10 that aligns with the protagonist walking, verse 2 starts at 0:22 when the ballon starts floating around the world and the music intensity builds up, and finally music calms down and fades out in the last scene at 0:42. Videos in this Figure found at https://go.fb.me/MovieGen-Figure32.

Table 31 presents the pairwise subjective evaluation results on audio quality and video alignment. Movie Gen Audio with extension outperforms both baselines as expected. In particular, we note that Movie Gen Audio-extension and Movie Gen Audio-stitch use the same base model, and hence they should have similar quality and alignment *within* each segment. However, we can observe from the table that the audio quality and the video-music alignment metrics are significantly worse for Movie Gen Audio-stitch, because stitching independently generated audio would lead to abrupt transition and incoherent music.

Dataset	Baseline method	Subjective: Movie Gen Audio extension net win rate *vs.* baseline						
		Quality			Video-SFX Alignment		Video-Music Alignment	
		Ovr.	Nat.	Pro.	Corr.	Sync.	Mood	Action
SGen SFX+music	Movie Gen Audio stitch	$34.5_{\pm 11.4}$	$33.7_{\pm 11.1}$	$34.5_{\pm 11.6}$	$19.6_{\pm 10.0}$	$5.6_{\pm 11.9}$	$20.3_{\pm 10.8}$	$26.5_{\pm 9.9}$
	Seeing&Hearing	$85.1_{\pm 5.6}$	$79.5_{\pm 6.5}$	$85.1_{\pm 5.6}$	$66.2_{\pm 7.9}$	$61.0_{\pm 9.8}$	$54.6_{\pm 10.5}$	$26.2_{\pm 8.8}$

Table 31 Audio extension vs. simple stitching. This table compares Movie Gen Audio with extension to simple stitch-based baseline and Seeing&Hearing on audio quality and video alignment. We report net win rate, which has a range [-100%, 100%], and its 95% confidence intervals. Positive values indicate Movie Gen Audio with extension outperforms the baseline on the metric.

6.4.2 Ablations

We ablate critical design decisions for Movie Gen Audio in this section, including text prompts, scaling, data, and extension methods. Unless otherwise described, we use the 13B parameter model and evaluate on SGen SFX for sound effect generation.

Text prompt: audio quality control. We vary the audio quality specified in the text prompt (blue part in Table 26) and demonstrate it can effectively control the audio quality. We evaluate both SFX and joint SFX+music generation, and present object and subjective metrics on both audio quality and video-SFX alignment in Figure 33 and Table 32, respectively. Qualitative samples are shown in Appendix D.1.2.

As we increase the conditioned audio quality, the predicted audio quality scores consistently improve on both datasets, and aligns with the subjective tests mostly up to 6.5. Human raters show similar preference to 6.5 and 7.0 for SFX generation, where quality is harder to differentiate at that level. These results validate the correlation between the subjective quality metric and the objective proxy metric (AQual), and demonstrate the effectiveness of quality control. In terms of the impact on video-SFX alignment, the impact is not significant on SFX generation (IB score is between 0.33 and 0.35, and subjective preference is not significant for any pair). In contrast, we observe significant improvement on SFX+music generation from 5.0 to 6, where IB

Dataset	Prompted audio qual.		A net win rate *vs.* B				
			Quality			Video-SFX Alignment	
	Model A	Model B	Ovr.	Nat.	Pro.	Corr.	Sync.
SFX	5.5	5.0	$54.6_{\pm 20.0}$	$50.0_{\pm 21.7}$	$54.6_{\pm 20.0}$	$3.3_{\pm 16.7}$	$5.1_{\pm 22.2}$
	6.0	5.5	$32.4_{\pm 21.5}$	$37.7_{\pm 17.6}$	$32.3_{\pm 21.7}$	$5.9_{\pm 15.0}$	$3.1_{\pm 19.4}$
	6.5	6.0	$29.8_{\pm 23.4}$	$27.9_{\pm 22.8}$	$30.8_{\pm 22.8}$	$-9.8_{\pm 18.6}$	$-15.9_{\pm 23.5}$
	7.0	6.5	$-6.6_{\pm 22.8}$	$-4.1_{\pm 22.8}$	$-8.8_{\pm 21.1}$	$11.5_{\pm 13.8}$	$-3.1_{\pm 19.2}$
SFX+music	5.5	5.0	$58.6_{\pm 19.0}$	$49.9_{\pm 19.1}$	$58.6_{\pm 19.0}$	$16.1_{\pm 16.1}$	$20.9_{\pm 15.0}$
	6.0	5.5	$27.0_{\pm 22.8}$	$33.9_{\pm 20.6}$	$28.2_{\pm 22.8}$	$23.1_{\pm 19.5}$	$21.9_{\pm 21.4}$
	6.5	6.0	$24.5_{\pm 20.6}$	$13.5_{\pm 21.7}$	$25.6_{\pm 20.6}$	$6.6_{\pm 15.6}$	$15.1_{\pm 17.5}$
	7.0	6.5	$16.2_{\pm 20.6}$	$22.9_{\pm 21.7}$	$15.0_{\pm 22.2}$	$-2.2_{\pm 12.8}$	$-11.1_{\pm 13.3}$

Table 32 Movie Gen Audio audio quality control ablations with subjective tests

score improves from 0.28 to 0.32 and raters show significant preference to higher conditioned quality. More details on metric correlation can be found in Appendix C.3.

(a) Audio quality score *vs.* prompted audio quality. **(b)** ImageBind score *vs.* prompted audio quality.

Figure 33 Movie Gen Audio audio quality control ablations with objective tests

Text prompt: control SFX and music styles Figure 34 and Figure 42 in the appendix present examples of audio event and music style control through text prompts. We observe in Figure 34 that text is particularly useful for dictating what *unseen* sound events should be generated. On the other hand, Figure 42 shows that text prompts can effectively control the music style, rendering different emotions for the same video.

Text prompt: with vs. without prompts. We study the impact of using text prompts during training and generation. Here we pre-train and fine-tune two 1B parameter models, one with text input and other without, denoted as TV2A and V2A, respectively. Text dropout is used for training for TV2A, so we can generate samples using only video input with that model as well. We denote model trained with caption and generating without caption as "TV2A → V2A", and similarly for other setups.

Results are presented in Table 33. We first note that on subjective tests, using text captions slightly improves quality, and significantly improves most alignment metrics. Second, when running inference without text prompt, the model trained with text prompts (TV2A → V2A) outperforms the one without (V2A → V2A) especially on the subjective alignment metrics, showing that text can facilitate learning audio-visual correspondence. Third, using text prompts can effectively guide model to generate the desired sound effects as shown by the higher CLAP score (0.37 *vs.* 0.23), since there is still a high level of ambiguity on what sound events should present given a video.

Model: scaling. We study the benefit of scaling and compares models of four different sizes: 300M, 3B, 9B, and 13B parameters. The 300M model adopts the same architecture and configuration as Vyas et al. (2023), while the remaining ones use the DiT architecture described in Section 6. Performance generally improves across all metrics as the model scales up, as shown in Table 34.

Data: effectiveness of fine-tuning. We compare performance before and after fine-tuning in Table 35. Fine-tuning significantly enhances both audio quality and video alignment. Qualitatively, the generated videos exhibit a much more cinematic feel after fine-tuning, which highlights the importance of high-quality data

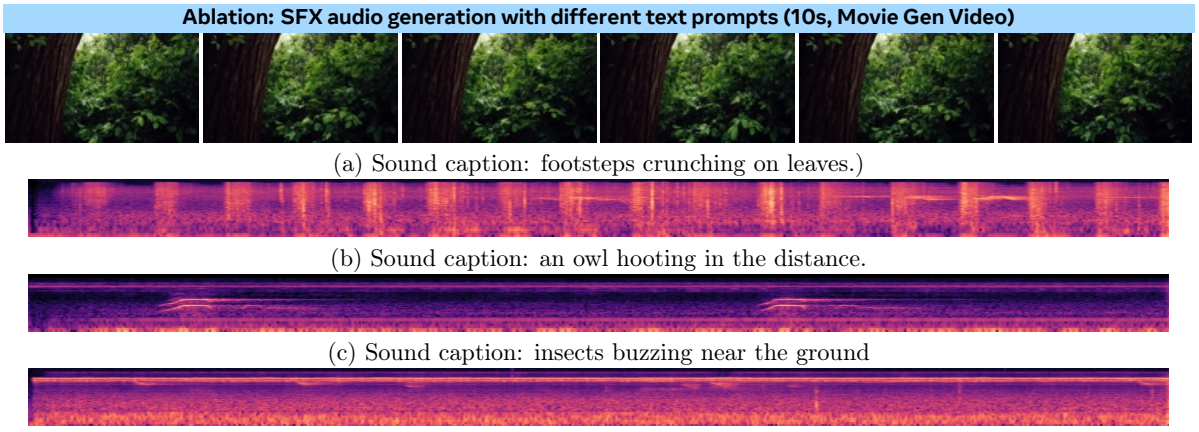


Figure 34 Examples of generated audio with different audio description in the text prompt with Movie Gen Audio. Videos in this Figure found at https://go.fb.me/MovieGen-Figure34.

Model and inference setup		A net win rate *vs.* B				
		Quality			Video-SFX Alignment	
Model A	Model B	Ovr.	Nat.	Pro.	Corr.	Sync.
TV2A → TV2A	TV2A → V2A	$12.5_{\pm 20.0}$	$12.5_{\pm 20.0}$	$11.1_{\pm 20.7}$	$20.4_{\pm 20.3}$	$17.5_{\pm 21.6}$
TV2A → TV2A	V2A → V2A	$15.9_{\pm 17.4}$	$10.3_{\pm 17.1}$	$18.8_{\pm 18.0}$	$26.7_{\pm 18.2}$	$32.1_{\pm 19.7}$

(a) Subjective metrics.

Model	CLAP
TV2A → TV2A	0.37
TV2A → V2A	0.23*
V2A → V2A	0.21*

(b) Objective metrics.

Table 33 Movie Gen Audio text prompt ablations. TV2A and V2A are models trained with and without text caption, "w/ cap" and "w/o cap" indicate whether text caption is used or not during inference. We put "*" on CLAP results when text is not used at inference.

curation for the fine-tuning process.

Data: effectiveness of high-quality audio-only data for fine-tuning. During fine-tuning, we supplement the cinematic audio-video data (Cin-AV), with an additional high-quality audio data (HQ-A) including both music and sound effects. We show in Table 36 that inclusion of high-quality audios yields significant improvement on quality and even slightly improves video alignment for SFX-only generation. For joint sound effect and music generation, it leads to significant improvement on video-music alignment. The inclusion of large-scale text-sound effect and text-music pairs enables the model to effectively disentangle different audio types. The alignment between audio and video thus also improves, along with the overall quality of the generated sound.

Extension: autoregressive vs. multi-diffusion. We compare the default extension method used in the main results (multi-diffusion, MD, with triangle window) with the other method (segment-level autoregressive generation, AR) and other configurations (windowing function for MD, use of beam search and conditioning methods for AR) described in Section 6.1.5. A one-shot generation topline that generates audio for the entire video without extension is also included. We evaluate on the SGen SFX+music set, following the setup described in Section 6.4.1, and study models in two scale: 3B and 13B. Results are shown in Table 37. We show qualitative samples for extension methods from the 13B model in Figure 35.

We observed (1) most methods are statistically similar on video-SFX alignment metrics, (2) multi-diffusion outperforms most alternative extension methods on quality significantly on 3B, but the gap disappears after scaling to 13B, (3) multi-diffusion is on par with the one-shot generation topline at 3B and even marginally better at 13B, (4) the proposed triangle window leads to smoother transitions compared to the uniform window proposed in (Bar-Tal et al., 2023) and results in higher audio quality (*vs.* "MD w/ uni. win") at 3B, but the gain again disappears at the larger scale. (5) beam search improves autoregressive generation ("AR w/ traj. reg. & tri. win." *vs.* "AR w/ traj. reg. & tri. win. & beam") at 3B, likely because the sample quality varies more for different seeds at smaller scales.

Model parameter		A net win rate *vs.* B				
		Quality			Video-SFX Alignment	
Model A	Model B	Ovr.	Nat.	Pro.	Corr.	Sync.
3B	300M	$29.9_{\pm 19.0}$	$25.1_{\pm 18.7}$	$20.2_{\pm 19.1}$	$35.2_{\pm 14.3}$	$34.9_{\pm 19.5}$
9B	3B	$34.6_{\pm 18.8}$	$36.7_{\pm 18.8}$	$36.7_{\pm 18.4}$	$19.4_{\pm 13.4}$	$20.1_{\pm 17.3}$
13B	9B	$11.0_{\pm 21.3}$	$10.3_{\pm 21.3}$	$11.7_{\pm 21.3}$	$18.8_{\pm 19.3}$	$23.1_{\pm 16.8}$

(a) Subjective metrics.

Model	CLAP
300M	0.23
3B	0.35
9B	0.35
13B	0.38

(b) Objective metrics.

Table 34 Scaling the Movie Gen Audio model size. We observe that scaling the model size generally improves performance across all metrics.

		A net win rate *vs.* B				
		Quality			Video-SFX Alignment	
Model A	Model B	Ovr.	Nat.	Pro.	Corr.	Sync.
FT	PT	$41.7_{\pm 15.3}$	$37.8_{\pm 16.3}$	$43.0_{\pm 14.7}$	$31.0_{\pm 16.0}$	$24.9_{\pm 17.7}$

Table 35 Effectiveness of fine-tuning Movie Gen Audio. When comparing the pre-trained (PT) model to the finetuned (FT) model, we observe that generations from the finetuned model are significantly better.

7 Related work

7.1 Text-to-Image Generation

Diffusion models have revolutionized the field of text-to-image generation. While a comprehensive review of all text-to-image models is beyond the scope of this paper, we will focus on the most relevant ones that have been published, productionized or open-sourced, and thus widely used by a large user base.

The seminal work of latent diffusion models (Rombach et al., 2022) proposes compressing the original image space to latent space using a variational autoencoder, which improves training and inference efficiency, thus popularizing latent-space-based diffusion models. Dalle-3 (OpenAI, 2024) proposes using GPT to rewrite image captions, reducing noise in curated internet-scale text-image pairs for more effective training. Emu (Dai et al., 2023) proposes using a higher latent dimension and fine-tuning a pre-trained model with a small, high-quality dataset to exclusively generate high-quality, professional-looking images. Stable Diffusion 3 (Esser et al., 2024) proposes using rectified flow transformers with a multimodal diffusion backbone to improve generation quality.

In MovieGen, we also use a 16-channel variational autoencoder and flow transformers with prompt rewrite to achieve both high visual quality and text alignment.

7.2 Text-to-Video Generation

The swift progress in text-to-image generation has led to substantial improvements in temporally coherent high quality video generation. After the success of diffusion models for image generation (Dhariwal and Nichol, 2021; Ramesh et al., 2022), they have been vastly used to improve video synthesis (Ho et al., 2022b). Several works introduce zero-shot video generation by enriching the pre-trained text-to-image generation models with motion dynamics (Khachatryan et al., 2023; Wu et al., 2023b). DirecT2V (Hong et al., 2023) leverages instruction-tuned large language models for zero-shot video creation by dividing user inputs into separate prompts for each frame.

Several other papers propose a cascaded or factorized approach for text-to-video generation. Imagen-Video (Ho et al., 2022a) and Make-A-Video (Singer et al., 2023) trained a deep cascade of spatial and temporal layers via pixel diffusion modeling while many other works focus more on applying diffusion to the latent space of an auto-encoder for more efficiency (Blattmann et al., 2023b; An et al., 2023; Wang et al., 2023d,c,a). AnimateDiff (Guo et al., 2023) introduces a pre-trained motion module into a pre-trained T2I model. Emu-Video (Girdhar et al., 2024), Stable Video Diffusion (Blattmann et al., 2023a), I2VGen-XL (Zhang et al.,

Dataset	FT data: Model A	FT data: Model B	A net win rate *vs.* B: Quality: Ovr.	Quality: Nat.	Quality: Pro.	Video-SFX Alignment: Corr.	Video-SFX Alignment: Sync.	Video-Music Alignment: Mood	Video-Music Alignment: Action
SFX	Cin-AV + HQ-A	Cin-AV	$21.5_{\pm 18.7}$	$24.3_{\pm 17.3}$	$22.8_{\pm 18.7}$	$11.7_{\pm 17.4}$	$9.6_{\pm 18.1}$	n/a	n/a
SFX+music	Cin-AV + HQ-A	Cin-AV	$1.7_{\pm 18.0}$	$3.0_{\pm 18.0}$	$0.4_{\pm 18.3}$	$0.0_{\pm 14.7}$	$8.6_{\pm 16.0}$	$16.4_{\pm 12.8}$	$14.9_{\pm 11.8}$

Table 36 Using high quality audio-only data in fine-tuning Movie Gen Audio. Including high quality audio-only data significantly improves audio quality and even video alignment for SFX-only generation.

Model	Extension method B	A net win rate *vs.* B: Quality: Ovr.	Quality: Nat.	Quality: Pro.	Video-SFX Alignment: Corr.	Video-SFX Alignment: Sync.	Video-Music Alignment: Mood	Video-Music Alignment: Action
	Extension method A: MD with triangle window							
3B	AR w/ context cond. & beam	$26.7_{\pm 12.5}$	$24.7_{\pm 11.1}$	$28.6_{\pm 13.1}$	$8.2_{\pm 11.4}$	$7.9_{\pm 11.3}$	$7.2_{\pm 10.2}$	$5.4_{\pm 8.4}$
	AR w/ traj. reg. & tri. win.	$18.5_{\pm 10.2}$	$16.5_{\pm 10.3}$	$18.4_{\pm 10.0}$	$4.1_{\pm 10.5}$	$10.6_{\pm 11.8}$	$2.4_{\pm 10.4}$	$-4.0_{\pm 8.3}$
	AR w/ traj. reg. & tri. win. & beam	$10.6_{\pm 11.0}$	$11.7_{\pm 9.7}$	$10.6_{\pm 11.0}$	$-6.4_{\pm 10.8}$	$-1.7_{\pm 11.6}$	$10.8_{\pm 9.7}$	$10.8_{\pm 9.4}$
	MD w/ uni. win.	$11.5_{\pm 7.9}$	$12.4_{\pm 7.8}$	$10.7_{\pm 8.2}$	$1.1_{\pm 8.5}$	$2.3_{\pm 9.0}$	$5.2_{\pm 8.5}$	$7.2_{\pm 8.7}$
	One-shot generation (topline)	$4.9_{\pm 12.7}$	$9.6_{\pm 11.7}$	$4.6_{\pm 12.8}$	$2.4_{\pm 8.8}$	$-4.6_{\pm 10.4}$	$2.7_{\pm 9.6}$	$3.7_{\pm 7.7}$
	Extension method A: MD with triangle window							
13B	AR w/ context cond. & beam	$3.4_{\pm 11.1}$	$4.1_{\pm 11.4}$	$4.0_{\pm 11.0}$	$10.0_{\pm 11.1}$	$12.5_{\pm 11.3}$	$9.8_{\pm 10.1}$	$13.9_{\pm 8.6}$
	AR w/ traj. reg. & tri. win.	$3.6_{\pm 11.4}$	$0.7_{\pm 10.3}$	$3.0_{\pm 11.4}$	$7.6_{\pm 10.0}$	$3.1_{\pm 11.6}$	$16.5_{\pm 11.4}$	$3.7_{\pm 8.7}$
	AR w/ traj. reg. & tri. win. & beam	$1.5_{\pm 10.2}$	$6.0_{\pm 10.3}$	$3.6_{\pm 10.3}$	$-2.0_{\pm 9.0}$	$3.0_{\pm 10.8}$	$13.9_{\pm 9.5}$	$15.4_{\pm 10.1}$
	MD w/ uni. win.	$0.7_{\pm 11.1}$	$-3.2_{\pm 10.2}$	$0.7_{\pm 11.1}$	$-1.6_{\pm 9.6}$	$-2.5_{\pm 10.7}$	$-1.5_{\pm 10.5}$	$0.6_{\pm 8.6}$
	One-shot generation (topline)	$11.7_{\pm 11.6}$	$9.3_{\pm 10.6}$	$12.3_{\pm 11.7}$	$0.3_{\pm 9.9}$	$-1.6_{\pm 11.3}$	$-0.5_{\pm 10.3}$	$-3.0_{\pm 8.9}$

Table 37 Ablation study on audio extension methods and configurations. We compare different audio extension methods across two model scales and evaluate the generations.

2023b), Dynamicrafter (Xing et al., 2023), VideoGen (Li et al., 2023a), and VideoCrafter1 (Chen et al., 2023a) add an image as an extra conditioning to the T2V model. Lumiere (Bar-Tal et al., 2024) uses a Space-Time U-Net to generate the full temporal duration of the video at once. SEINE (Chen et al., 2023c) facilitates the smooth integration of shots from diverse scenes and generates videos of various lengths through auto-regressive prediction.

A few papers have studied the role of noise scheduling for more coherent (Ge et al., 2023; Qiu et al., 2023; Luo et al., 2023) and longer (Kim et al., 2024) video generation. While most of the above text-to-video generation models use a U-Net based architecture, Snap-Video (Menapace et al., 2024) and OpenAI Sora (OpenAI, 2024) show the scalability and out-performance of transformer architectures for diffusion-based video generation. Latte (Ma et al., 2024a) also uses a DiT instead of the U-Net backbone for text-to-video generation. On the other hand, a couple of works have been focused on transformer models within an auto-regressive framework (Yan et al., 2021; Kondratyuk et al., 2023; Hong et al., 2022; Wu et al., 2022; Villegas et al., 2023; Ge et al., 2022). RIVER (Davtyan et al., 2023) uses flow matching for efficient video prediction by conditioning on a small set of past frames in the latent space of a pre-trained VQGAN. In this work, we leverage a Llama3 transformer architecture (Dubey et al., 2024) and train a text-to-video generation model within a flow matching framework (Lipman et al., 2023).

Encoding videos into a latent space. Since their inception (Rombach et al., 2022; Esser et al., 2021), encoder/decoder models have been a core part of latent generative architectures, and serve to compress raw media (images, video, audio) into a lower-dimensional latent space. Latent diffusion models (Rombach et al., 2022) typically use either a normal variational autoencoder (VAE) (Kingma, 2013), a quantized VAE such as a VQVAE (van den Oord et al., 2017) or VQGAN (Esser et al., 2021) and its variants (Lee et al., 2022a), which adds a GAN discriminator loss (Goodfellow et al., 2014) to achieve improved reconstruction quality with greater compression. Most image and video generation models use convolutional autoencoders, though transformer-based encoding models such as Efficient-VQGAN (Cao et al., 2023), ViT-VQGAN (Yu et al., 2021), and TiTok (Yu et al., 2024) show promising results using vision transformers. For the Movie Gen autoencoder, we found best results using a continuous convolutional VAE with discriminator loss.

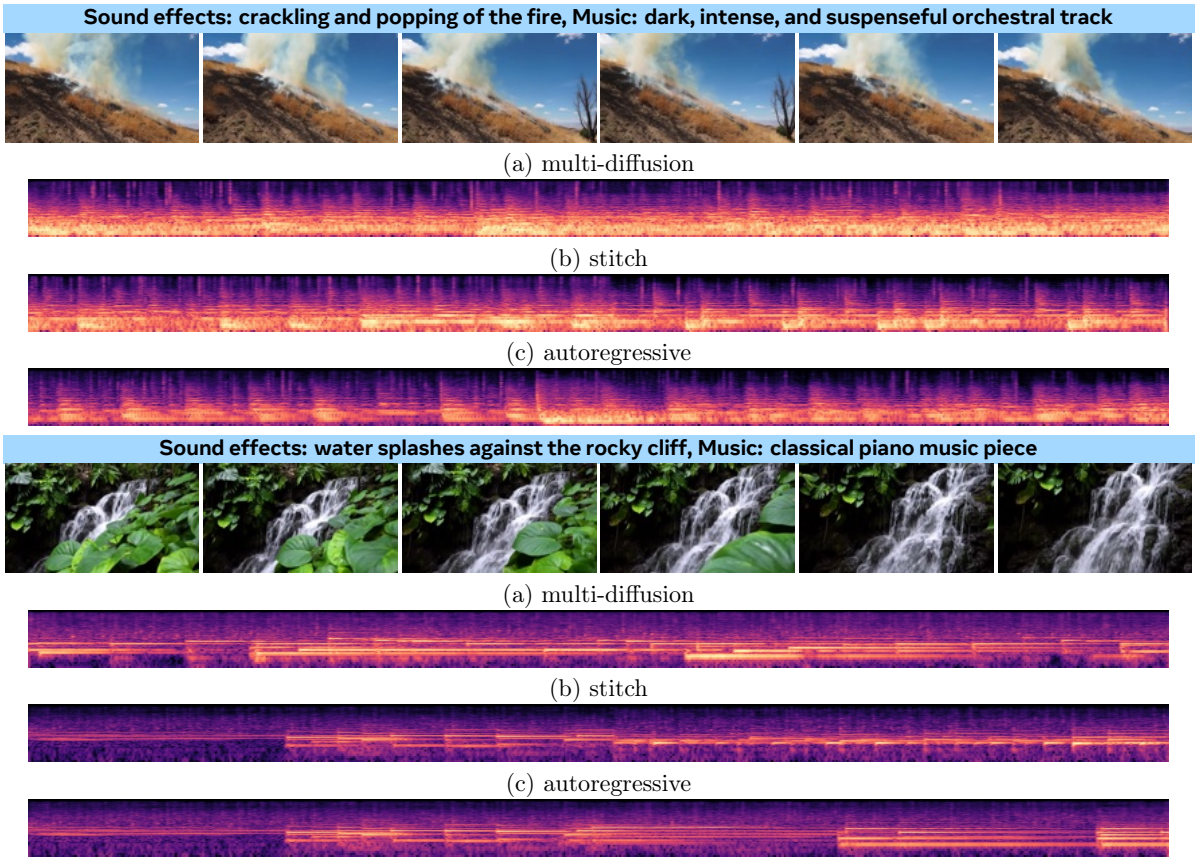


Figure 35 Examples of sound track generation using different audio extension methods with Movie Gen Audio. Each video comprises 2 segments of about 5 seconds each. Both multi-diffusion and autoregressive generate coherent samples with smooth transitions at the boundary. Stitching results in noticible music artifacts at the boundary. Videos in this Figure found at https://go.fb.me/MovieGen-Figure35.

Of the methods using convolutional autoencoders, the models have been split between image (2D) and video (3D) models. First are those which use an image VAE (with or without quantization) and encode frame-by-frame — for example Stable Video Diffusion (Blattmann et al., 2023a), Latent Shift (An et al., 2023), VideoLDM (Blattmann et al., 2023b), Emu-Video (Girdhar et al., 2024), and CogVideo (Hong et al., 2022). Models which use image encoders are typically unable to directly generate long, high FPS videos due to lack of temporal compression. A more recent alternative has been to use 3D or mixed 2D-3D models. For example, MAGViT (Yu et al., 2023a) uses a 3D VQGAN with both 3D and 2D downsampling layers, with average pooling for downsampling, and W.A.L.T. (Gupta et al., 2023) and MAGViT-V2 (Yu et al., 2023b) use fully 3D convolutional encoders with strided downsampling. In our work, we chose to use an interleaved 2D-1D (*e.g.*, 2.5D) convolutional encoder, where we trade off a slight improvement to reconstruction quality from a fully 3D model for lower memory and computational costs.

A feature of W.A.L.T., MAGViT-V2 and also some ViT-based encoders such as C-ViViT (Villegas et al., 2023) is the inclusion of causality, which is typically implemented for convolutions through an asymmetrical padding. Causality is usually implemented because the first frame is always encoded independently, allowing images to be explicitly encoded for joint image and video generation. However, we have found that causal encoding is not necessary to encode images and videos jointly — symmetrical padding functions for joint image and video generation, and encoded images with symmetrically-padded convolutions are able to be used as conditioning for image-to-video models. Symmetrical padding works for different video lengths as long as replicate padding is used; a similar result is reported by TATS (Ge et al., 2022).

7.3 Image and Video Personalization

Personalized Image Generation. Prior work in personalized image generation has primarily focused on two technical directions: 1) identity-specific tuning and 2) tuning-free methods. Identity-specific tuning trains a text-to-image model to incorporate the identity by finetuning on a specific identity. Textual Inversion (Gal et al., 2022) finetunes special text tokens for the new identity. DreamBooth (Ruiz et al., 2023a) selects a few images from the same identity as reference as well as a special text token to represent the identity concept. LoRA techniques (Hu et al., 2021) have been explored to tune a light-weight low-rank adapter to accelerate the training process. HyperDreamBooth (Ruiz et al., 2023b) further reduces the training latency by directly predicting the initial weights of LoRA from the reference images. A major drawback of identity-specific tuning personalization methods is that the final model has parameters that are trained for and associated with a specific identity, and this process does not scale well to multiple users.

To overcome the limitations of the identity-specific tuning methods, another line of research extracts vision embeddings from the reference image and directly injects it into the diffusion process. This direction is more scalable as all users can share the same base model. ELITE (Wei et al., 2023) extracts vision features from the reference image and converts it to the text-embedding space through a local and a global mapping. PhotoMaker (Li et al., 2023c) merges the vision and text tokens and replaces the original text tokens for cross-attention. PhotoVerse (Chen et al., 2023b) incorporates an image adapter and a text adapter to merge the vision and language tokens respectively. IP-Adapter-FaceID-Plus (Ye et al., 2023) leverages face embedding and clip vision encoder for identity preservation. InstantID (Wang et al., 2024a) is a control-based method that adds ControlNet (Zhang et al., 2023a) to further control the pose and facial expression. MoA (Ostashev et al., 2024) proposes a mixture-of-attention architecture to better fuse the vision reference and the text prompts. Imagine Yourself (He et al., 2024b) proposed a full parallel model architecture, a multi-stage finetuning strategy, and a novel synthetic paired data generation mechanism for better identity preservation, prompt alignment, and visual quality.

Personalized Video Generation. While the aforementioned works have shown promising results in personalized image generation, adding personalization capability to video generation remains a challenging and unsolved problem. There are a few novel challenges in the area of personalized video generation: 1) compared to personalized image generation, personalized video generation needs to support more diverse and complex modifications on the reference image, *e.g.*, turning the head, changing poses, and camera motion movements, 2) personalized video generation increases the expected quality threshold on expression and motion naturalness due to its temporal nature, and 3) finetuning a video model is much more costly than finetuning an image model, given the larger model and input sizes.

One direction of personalized video generation utilizes pose to control the video generation. Both GAN-based methods (Chan et al., 2019; Yoon et al., 2021) and diffusion-based method (Wang et al., 2024b,b; Hu et al., 2024; Xu et al., 2024b) have been proposed to generate videos following reference poses. These models are designed to animate the reference image towards the target motion, and are good for motions like singing and dancing. However, these models require a pose sequence as reference, usually extracted from a real video, limiting its usage to a broader scope of scenearios. Also, these models often introduce occlusion and unnatural motion due to the non-ideal pose extraction and control.

On the other hand, preliminary works have been proposed to turn a personalized image model to a personalized video model. Magic Me (Ma et al., 2024b) uses identity-specific finetuning to inject identity into a video generation model. ID-Animator (He et al., 2024a) leverages a face adapter to extract identity information from single reference facial image for personalized video generation. DreamVideo (Wei et al., 2024) and Still-Moving (Chefer et al., 2024) combine an identity adapter and a motion adapter for flexible video customization. CustomCrafter (Wu et al., 2024) proposes a plug-and-play module for subject concept injection. Our work also focuses on using identity extraction and control methodology, as it is more generalizable to different users.

7.4 Instruction-Guided Video Editing

In the task of text-based video editing[3] the user provides the model with a video (either real or generated) along with an editing instruction text that specifies how they would like to alter the video. The model is then expected to precisely modify the input video according to the given instruction, changing only the specified elements while preserving those that should remain intact. The main challenge in developing a high-performing video editing model arises from the difficulty in collecting supervised data for this task.

As a result of this challenge, most prior work relies on training-free approaches (Meng et al., 2022; Geyer et al., 2023; Wang et al., 2023b; Khachatryan et al., 2023; Li et al., 2023b; Ceylan et al., 2023; Kara et al., 2023; Yang et al., 2023), which can be applied to any text-to-video model without requiring additional training. In contrast to training-free methods, some approaches train models to generate videos by providing additional features of the video as input (*e.g.*, depth or segmentation maps) (Esser et al., 2023; Liang et al., 2023; Yan et al., 2023). However, these methods are inherently limited, as they cannot control features that were not incorporated during training. For example, preserving the identity of an object or subject is impossible when conditioning only on the depth maps of a video. Overall, both training-free methods and feature-based approaches tend to be imprecise and have been shown to perform worse than methods that explicitly adapt model parameters to process the entire video input during training (Singer et al., 2024; Qin et al., 2023).

The current state-of-the-art approach for video editing, Emu Video Edit (EVE) (Singer et al., 2024), employs two training stages to develop a video editing model. First, it trains dedicated adapters for text-image-to-video generation and image editing on top of a shared text-to-image model. Next, it performs Factorized Diffusion Distillation (FDD) to align the adapters towards video editing. In each training step of FDD, the model first generates an edited video through multiple diffusion steps. Then, the edited video is given as input to two adversarial losses and two knowledge distillation losses, which provide supervision for the quality of the edited video. Finally, this supervision is backpropagated through both the different losses and the entire generation chain.

We identify several key differences when comparing our approach to EVE. First, we initialize training from a text-to-video model and perform full model training (Section 5.1.2), rather than training adapters for text-to-video and image editing on top of a shared text-to-image model. Additionally, EVE's FDD backpropagates supervision through multiple forward passes with the model during generation, and an additional forward pass using each of the models it uses to provide supervision. This makes FDD an order of magnitude more memory demanding than our approach, which limits the scalability of FDD. By applying our approach to the Movie Gen Video (see Section 3, we demonstrate that we can significantly surpass the reported results by EVE and set new state-of-the-art results in video editing.

7.5 Audio Generation

Video-to-audio generation. There are many recent studies exploring video-to-audio generation. Most of them are based on latent diffusion models similar to ours (Luo et al., 2024; Xu et al., 2024a; Xing et al., 2024; Zhang et al., 2024) with a few exceptions being token-based language models (Kondratyuk et al., 2023; Mei et al., 2023). Diff-Foley (Luo et al., 2024) and VTA-LDM (Xu et al., 2024a) are the standard latent diffusion models with U-Net architectures conditioned on video features, extracted from a pre-trained contrastive audio-video encoder (CAVP (Luo et al., 2024)) and video-text encoder (CLIP4CLIP (Luo et al., 2022)), respectively. Seeing-and-hearing (S&H) (Xing et al., 2024) proposes a training-free method that uses ImageBind as a classifier guidance to guide a pre-trained diffusion-based text-to-audio (TTA) model (AudioLDM (Liu et al., 2023b)) to generate video aligned audio. FoleyCrafter (Zhang et al., 2024) uses the adaptor-based approach (Zhang et al., 2023a) to finetune a pre-trained TTA model to add video control. To enhance the temporal alignment between audio and video, it further conditions on timestamps to inform the model which segments are sounded/silence, which are predicted from the video during inference.

Most of these models are directly trained on, or built on TTA models trained on solely in-the-wild datasets, such as VGGSound (550 hours) (Chen et al., 2020) or AudioSet (5K hours) (Gemmeke et al., 2017). These datasets have several limitations. In terms of quality, many of the videos are recorded with non-professional devices like smartphones or low-end cameras, and hence both the video and the audio quality are subpar. In

[3]For brevity, we refer to text or instruction-based video editing simply as video editing.

terms of sound design, most of the videos are uploaded by amateur creators who had done none or minimal post-processing and post-production, which contain only diegetic sounds. Compared to professionally created films, such videos contain an abundance of distracting sounds (*e.g.*, irrelevant off-screen speech, wind noise, high ambient noise), do not emphasize main sound events (*e.g.*, exaggerated breathing sound, Foley sounds for footsteps and object clatters), and lack carefully designed non-diegetic sounds that are critical for a cinematic feeling (*e.g.*, use of riser, braam, underscore music, action scoring). Training on such datasets inevitably prevents the resulting model from generating cinematic soundtracks including both music and sound effects.

On the other hand, the size of prior video-to-audio generation models are relatively small, typically ranging from 300M to 1.3B parameters (Xing et al., 2024; Luo et al., 2024; Mei et al., 2023). Combined with the data size, the scale also limits the performance of these models, as we demonstrated in the ablation study (see Section 6.4.2) that scaling to 13B significantly improves both quality and video alignment. Compared with the prior works that also offer text control, we additionally provide quality control and fine-grained music control, which alleviates the quality issue when training on mixed-quality data, and improves soundtrack design flexibility.

There are a few products offering video-to-audio capabilities, including PikaLabs[4] and ElevenLabs.[5], but neither can really generate motion-aligned sound effects or cinematic soundtracks with both music and sound effects. PikaLabs supports sound effect generation with video and optionally text prompts; however it will generate audio longer than the video where a user needs to select an audio segment to use. This implies under the hood it may be an audio generation model conditioned on a fixed number of key image frames. The maximum audio length is capped at 15 seconds without joint music generation and audio extension capabilities, preventing its application to soundtrack creation for long-form videos. ElevenLabs leverages GPT-4o to create a sound prompt given four image frames extracted from the video (one second apart), and then generates audio using a TTA model with that prompt. Lastly, Google released a research blog[6] describing their video-to-audio generation models that also provide text control. Based on the video samples, the model is capable of sound effects, speech, and music generation. However, the details (model size, training data characterization) about the model and the number of samples (13 samples with 11 distinct videos) are very limited, and no API is provided. It is difficult to conclude further details other than the model is diffusion-based and that the maximum audio length may be limited as the longest sample showcased is less than 15 seconds.

Video to music generation. Many studies often focus on symbolic music (MIDI) generation (Di et al., 2021; Zhuo et al., 2022; Kang et al., 2024) for piano or other instruments, as MIDI is easier to predict compared to raw audio. Compared to end-to-end modeling, such a paradigm imposes many restrictions. First, MIDI is a form of music transcription which cannot capture all the details from the original music. Hence, the generated music from such systems tend to sound more monotonic. Second, it requires MIDI annotation or a high quality music transcription model, which limits the sources of training data one can consider. Lastly, such models cannot learn the relationship between music and other audio components like sound effects and speech, which are not trivial in cinematic films.

Most of these works, along those directly predicting audio (Zhu et al., 2022), extract low-level music-related features from videos, such as human motion, scene change timing, and tempo, for conditioning. These features along with the training data (*e.g.*, dancing videos) are rather domain specific, which cannot generalize to general videos.

Our work is most related to Su et al. (2023) and Tian et al. (2024). Both prior works extract general video features (CLIP, flow, image tokens) and predict general audio representations (EnCodec (Défossez et al., 2022), Soundstream (Zeghidour et al., 2022), w2vBERT (Chung et al., 2021)). In addition to our novel scaling, the main differences in our work are twofold: first we aim for joint non-diegetic music and sound effect generation while these studies only focus on non-diegetic music; second we adopt diffusion modeling which is free of tokenization information loss and hence enjoys the other benefits described in previous sections, while Tian et al. (2024) points out explicitly that their quality is suffer from the audio codec limitation.

[4] https://pika.art/

[5] https://github.com/elevenlabs/elevenlabs-examples/tree/main/examples/sound-effects/video-to-sfx

[6] https://deepmind.google/discover/blog/generating-audio-for-video/

8 Conclusion

The Movie Gen cast of foundation models represents a significant improvement in text-to-video generation, video personalization, video editing, as well as sound effect and music generation. We show that training such models by scaling data, training compute, and model size together leads to such significant improvements. We focus on curating high quality large scale data for pre-training and relatively smaller scale but even higher quality data for finetuning. This general recipe works well for improving the quality of image, video, and audio generation. We introduce a new approach for equipping strong video foundation models with state-of-the-art video editing capabilities without relying on supervised video editing data. This is achieved through multi-task training on image editing and video generation, followed by two short fine-tuning stages: one on synthetic multi-frame editing data, and another on video editing via backtranslation.

Despite these improvements, we observe that video generation models still suffer from issues – artifacts in generated or edited videos around complex geometry, manipulation of objects, object physics, state transformations *etc*. Generated audio is sometimes out of synchronization when motions are dense (*e.g.*, tap dance), visually small or occluded (*e.g.*, footsteps), or when it requires finer-grained visual understanding (*e.g.*, recognizing the guitar chords). It currently does not support voice generation either due to our design choices. Reliable benchmarking of media generation models is important for identifying such shortcomings and for future research. Having access to a few cherry picked generations or black box systems without clear details on model or data makes reliable comparisons hard. Along with details on models, data, and inference, we additionally release multiple non cherry picked generations and prompt sets to enable easy and reliable comparisons for future work. We also note that defining objective criteria evaluating model generations using human evaluations remains challenging and thus human evaluations can be influenced by a number of other factors such as personal biases, backgrounds *etc*. While our models are trained separately for video and audio generation, developing models that can generate these modalities jointly is an important area of research.

Safety considerations. The Movie Gen cast of foundation models were developed for research purposes and need multiple improvements before deploying them. We consider a few risks from a safety viewpoint. Any real world usage of these models requires considering such aspects. Our models learn to associate text and optionally additional inputs like video to output modalities like image, video and audio. It is also likely that our models can learn unintentional associations between these spaces. Moreover, generative models can also learn biases present in individual modalities, *e.g.*, visual biases present in the video training data or the language used in the text prompts. Our study in this paper is limited to text inputs in the English language. Finally, when we do deploy these models, we will incorporate safety models that can reject input prompts or generations that violate our policies to prevent misuse.

Contributors and Acknowledgements

A large number of people at Meta worked to create Movie Gen. We list **core contributors** (people who worked on Movie Gen for at least 2/3rd of the runtime of the project), and **contributors** (people who worked on Movie Gen for at least 1/3rd of the runtime of the project). We list all contributors in alphabetical order of the first name.

Core Contributors

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, 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, Tao Xu, 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

Contributors

Albert Pumarola, Alejandro Ruiz, Ali Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, DeShawn Wallace, Dimitry Vengertsev, Edgar Schönfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Joseph Kim, Keren Lonstein, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Matthew Yu, Rashel Moritz, Ryan Henkel, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Simran Motwani, Steve Fine, Tara Fowler, Tianhe Li, Vladan Petrovic, Yuming Du

Acknowledgements

Ahmad Al-Dahle, Ahuva Goldstand, Ajay Ladsaria, Akash Jaiswal, Akio Kodaira, Andrew Treadway, Andrés Alvarado, Antoine Toisoul, Baishan Guo, Bernie Huang, Boris Araya, Brandon Wu, Brian Ellis, Chao Zhou, Chen Fan, Chen Kovacs, Ching-Feng Yeh, Chris Moghbel, Connor Hayes, Daniel Ho, Daniel Lee, Daniel Li, Danny Trinh, David Kant, David Novotny, Delia David, Dong Li, Ellen Tan, Emory Lin, Fraylie Nord, Gabriella Schwarz, Gael Le Lan, Jeff Wang, Jiabo Hu, Jianyu Huang, Jiecao Yu, Jiemin Zhang, Jinho Hwang, Joelle Pineau, Jongsoo Park, Junjiao Tian, Karthik Sivakumar, Kathryn Stadler, Lindsey Kishline, Manohar Paluri, Matt Setzler, Max Raphael, Mengyi Shan, Nick Zacharov, Pasan Hapuarachchi, Peter Carras, Philip Woods, Prash Jain, Prashant Ratanchandani, Ragavan Srinivasan, Rebecca Kogen, Ricky T. Q. Chen, Robbie Adkins, Rod Duenes, Roman Shapovalov, Ruihan Shan, Russ Maschmeyer, Shankar Regunathan, Shaun Lindsay, Sreeram R Chakrovorthy, Thai Quach, Tiantu Xu, Tom Monnier, Ty Toledano, Uriel Singer, Vlad Shubin, Wei Jiang, Will Seyfer, Xide Xia, Xinyue Zhang, Yael Yungster, Yang Liu, Yang Shu, Yangyang Shi, Yaron Lipman, Yash Mehta, Ye Jia, Zhaoheng Ni

References

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

Kendall Atkinson. *An introduction to numerical analysis*. John Wiley & Sons, 1991.

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. eDiff-I: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. MultiDiffusion: Fusing diffusion paths for controlled image generation. In *ICML*, 2023.

Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Yuanzhen Li, Tomer Michaeli, Oliver Wang, Deqing Sun, Tali Dekel, and Inbar Mosseri. Lumiere: A space-time diffusion model for video generation. *arXiv preprint arXiv:2401.12945*, 2024.

Shane Barratt and Rishi Sharma. A note on the Inception score. *arXiv preprint arXiv:1801.01973*, 2018.

Jeremy Baumgartner and Matt Bowman. Meta Open Compute Project, Grand Teton AI platform. https://engineering.fb.com/2022/10/18/open-source/ocp-summit-2022-grand-teton/, 2022.

Black Forest Labs. FLUX, 2024. https://blackforestlabs.ai/.

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

Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023b.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 1952.

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

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to follow image editing instructions. In *CVPR*, 2023.

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. https://openai.com/research/video-generation-models-as-world-simulators.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Shiyue Cao, Yueqin Yin, Lianghua Huang, Yu Liu, Xin Zhao, Deli Zhao, and Kaigi Huang. Efficient-VQGAN: Towards high-resolution image generation with efficient vision transformers. In *ICCV*, 2023.

Duygu Ceylan, Chun-Hao Paul Huang, and Niloy Jyoti Mitra. Pix2Video: Video editing using image diffusion. In *ICCV*, 2023.

Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *ICCV*, 2019.

Hila Chefer, Shiran Zada, Roni Paiss, Ariel Ephrat, Omer Tov, Michael Rubinstein, Lior Wolf, Tali Dekel, Tomer Michaeli, and Inbar Mosseri. Still-moving: Customized video generation without customized video data. *arXiv preprint arXiv:2407.08674*, 2024.

Haoxin Chen, Menghan Xia, Yingqing He, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Jinbo Xing, Yaofang Liu, Qifeng Chen, Xintao Wang, Chao Weng, and Ying Shan. VideoCrafter1: Open diffusion models for high-quality video generation. *arXiv:2310.19512*, 2023a.

Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. VGGSound: A large-scale audio-visual dataset. In *ICASSP*, 2020.

Li Chen, Mengyi Zhao, Yiheng Liu, Mingxu Ding, Yangyang Song, Shizun Wang, Xu Wang, Hao Yang, Jing Liu, Kang Du, et al. PhotoVerse: Tuning-free image customization with text-to-image diffusion models. *arXiv preprint arXiv:2309.05793*, 2023b.

Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang-wei Chao, Byung Eun Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, and Sergey Tulyakov. Panda-70M: Captioning 70M videos with multiple cross-modality teachers. *arXiv preprint arXiv:2402.19479*, 2024.

Xinyuan Chen, Yaohui Wang, Lingjun Zhang, Shaobin Zhuang, Xin Ma, Jiashuo Yu, Yali Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. SEINE: Short-to-long video diffusion model for generative transition and prediction. In *ICLR*, 2023c.

Jiaxin Cheng, Tianjun Xiao, and Tong He. Consistent video-to-video transfer using synthetic dataset. In *ICLR*, 2024.

Min Jin Chong and David Forsyth. Effectively unbiased FID and Inception score and where to find them. In *CVPR*, 2020.

Arnab Choudhury, Yang Wang, Tuomas Pelkonen, Kutta Srinivasan, Abha Jain, Shenghao Lin, Delia David, Siavash Soleimanifard, Michael Chen, Abhishek Yadav, et al. MAST: Global scheduling of ML training across Geo-Distributed datacenters at hyperscale. In *Proceedings from 18th USENIX Symposium on Operating Systems Design and Implementation*, 2024.

Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. w2v-BERT: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021.

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

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *ICLR*, 2024.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

Aram Davtyan, Sepehr Sameni, and Paolo Favaro. Efficient video prediction via sparsely conditioned flow matching. In *ICCV*, 2023.

Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim M Alabdulmohsin, et al. Patch n' Pack: NaViT, a vision transformer for any aspect ratio and resolution. In *NeurIPS*, 2024.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. ArcFace: Additive angular margin loss for deep face recognition. In *CVPR*, 2019.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021.

Shangzhe Di, Jiang, Sihan Liu, Zhaokai Wang, Leyan Zhu, Zexin He, Hongming Liu, and Shuicheng Yan. Video background music generation with controllable music transformer. In *ACM MM*, 2021.

John R Dormand and Peter J Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 1980.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16×16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

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

Klas Dykhoff. Non-diegetic sound effects. *The New Soundtrack*, 2(2):169–179, 2012.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018.

ElevenLabs. ElevenLabs. https://elevenlabs.io/app/sound-effects.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. *arXiv preprint arXiv:2302.03011*, 2023.

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

Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. *Image analysis*, 2003.

FFmpeg Developers. FFmpeg. https://ffmpeg.org/.

Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.

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

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

Songwei Ge, Aniruddha Mahapatra, Gaurav Parmar, Jun-Yan Zhu, and Jia-Bin Huang. On the content bias in Fréchet video distance. In *CVPR*, 2024.

Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *ICASSP*, 2017.

Michal Geyer, Omer Bar-Tal, Shai Bagon, and Tali Dekel. TokenFlow: Consistent diffusion features for consistent video editing. *arXiv preprint arXiv:2307.10373*, 2023.

Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Text-to-audio generation using instruction-tuned LLM and latent diffusion model. *arXiv preprint arXiv:2304.13731*, 2023.

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y. Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 2014.

Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

Yuwei Guo, Ceyuan Yang, Anyi Rao, Yaohui Wang, Yu Qiao, Dahua Lin, and Bo Dai. AnimateDiff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023.

Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Li Fei-Fei, Irfan Essa, Lu Jiang, and José Lezama. Photorealistic video generation with diffusion models. *arXiv preprint arXiv:2312.06662*, 2023.

Xuanhua He, Quande Liu, Shengju Qian, Xin Wang, Tao Hu, Ke Cao, Keyu Yan, Man Zhou, and Jie Zhang. ID-Animator: Zero-shot identity-preserving human video generation. *arXiv preprint arXiv:2404.15275*, 2024a.

Zecheng He, Bo Sun, Felix Juefei-Xu, Haoyu Ma, Ankit Ramchandani, Vincent Cheung, Siddharth Shah, Anmol Kalia, Harihar Subramanyam, Alireza Zareian, Li Chen, Ankit Jain, Ning Zhang, Peizhao Zhang, Roshan Sumbaly, Peter Vajda, and Animesh Sinha. Imagine yourself: Tuning-free personalized image generation. *arXiv preprint arXiv:2409.13346*, 2024b.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

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. *arXiv preprint arXiv:2210.02303*, 2022a.

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

Tomlinson Holman. *Sound for film and television*. Routledge, 2012.

Susung Hong, Junyoung Seo, Heeseong Shin, Sunghwan Hong, and Seungryong Kim. Direct2v: Large language models are frame-level directors for zero-shot text-to-video generation. *arXiv preprint arXiv:2305.14330*, 2023.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Li Hu, Xin Gao, Peng Zhang, Ke Sun, Bang Zhang, and Liefeng Bo. Animate Anyone: Consistent and controllable image-to-video synthesis for character animation. In *CVPR*, 2024.

Qingqing Huang, Daniel S Park, Tao Wang, Timo I Denk, Andy Ly, Nanxin Chen, Zhengdong Zhang, Zhishuai Zhang, Jiahui Yu, Christian Frank, et al. Noise2Music: Text-conditioned music generation with diffusion models. *arXiv preprint arXiv:2302.03917*, 2023.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *CVPR*, 2024.

Ideogram. Ideogram v2, 2024. https://ideogram.ai/.

Jaeyong Kang, Soujanya Poria, and Dorien Herremans. Video2Music: Suitable music generation from videos using an affective multimodal transformer model. *arXiv preprint arXiv:2311.00968*, 2024.

Ozgur Kara, Bariscan Kurtkaya, Hidir Yesiltepe, James M Rehg, and Pinar Yanardag. Rave: Randomized noise shuffling for fast and consistent video editing with diffusion models. *arXiv preprint arXiv:2312.04524*, 2023.

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

Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2Video-Zero: Text-to-image diffusion models are zero-shot video generators. *arXiv preprint arXiv:2303.13439*, 2023.

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A metric for evaluating music enhancement algorithms. *arXiv preprint arXiv:1812.08466*, 2018.

Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. AudioCaps: Generating captions for audios in the wild. In *NAACL-HLT*, 2019.

Jihwan Kim, Junoh Kang, Jinyoung Choi, and Bohyung Han. FIFO-Diffusion: Generating infinite videos from text without training. *arXiv preprint arXiv:2405.11473*, 2024.

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

KlingAI. Kling AI, 2024. https://klingai.com/.

Jonas Kohler, Albert Pumarola, Edgar Schönfeld, Artsiom Sanakoyeu, Roshan Sumbaly, Peter Vajda, and Ali Thabet. Imagine Flash: Accelerating Emu diffusion models with backward distillation. *arXiv preprint arXiv:2405.05224*, 2024.

Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Rachel Hornung, Hartwig Adam, Hassan Akbari, Yair Alon, Vighnesh Birodkar, et al. VideoPoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023.

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

Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. AudioGen: Textually guided audio generation. *arXiv preprint arXiv:2209.15352*, 2022.

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved RVQGAN. In *NeurIPS*, 2024.

Gael Le Lan, Bowen Shi, Zhaoheng Ni, Sidd Srinivasan, Anurag Kumar, Brian Ellis, David Kant, Varun Nagaraja, Ernie Chang, Wei-Ning Hsu, et al. High fidelity text-guided music generation and editing via single-stage flow matching. *arXiv preprint arXiv:2407.03648*, 2024.

Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, and Wei-Ning Hsu. Voicebox: Text-guided multilingual universal speech generation at scale. *arXiv preprint arXiv:2306.15687*, 2023.

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

Sang-gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon. BigVGAN: A universal neural vocoder with large-scale training. *arXiv preprint arXiv:2206.04658*, 2022b.

Shenggui Li, Fuzhao Xue, Chaitanya Baranwal, Yongbin Li, and Yang You. Sequence parallelism: Long sequence training from system perspective. *arXiv preprint arXiv:2105.13120*, 2021.

Xin Li, Wenqing Chu, Ye Wu, Weihang Yuan, Fanglong Liu, Qi Zhang, Fu Li, Haocheng Feng, Errui Ding, and Jingdong Wang. VideoGen: A reference-guided latent diffusion approach for high definition text-to-video generation. *arXiv preprint arXiv:2309.00398*, 2023a.

Xirui Li, Chao Ma, Xiaokang Yang, and Ming-Hsuan Yang. VidToMe: Video token merging for zero-shot video editing. *arXiv preprint arXiv:2312.10656*, 2023b.

Xuanyi Li, Daquan Zhou, Chenxu Zhang, Shaodong Wei, Qibin Hou, and Ming-Ming Cheng. Sora generates videos with stunning geometrical consistency. *arXiv preprint arXiv:2402.17403*, 2024.

Zhen Li, Mingdeng Cao, Xintao Wang, Zhongang Qi, Ming-Ming Cheng, and Ying Shan. PhotoMaker: Customizing realistic human photos via stacked ID embedding. *arXiv preprint arXiv:2312.04461*, 2023c.

Feng Liang, Bichen Wu, Jialiang Wang, Licheng Yu, Kunpeng Li, Yinan Zhao, Ishan Misra, Jia-Bin Huang, Peizhao Zhang, Péter Vajda, and Diana Marculescu. FlowVid: Taming imperfect optical flows for consistent video-to-video synthesis. *arXiv preprint arXiv:2312.17681*, 2023.

Minghui Liao, Guan Pang, Jing Huang, Tal Hassner, and Xiang Bai. Mask TextSpotter v3: Segmentation proposal network for robust scene text spotting. In *ECCV*, 2020.

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *WACV*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

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

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. AudioLDM: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023b.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023c.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

LumaLabs. Dream Machine, 2024. https://lumalabs.ai/dream-machine.

Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. CLIP4Clip: An empirical study of CLIP for end to end video clip retrieval and captioning. *Neurocomputing*, 508:293–304, 2022.

Simian Luo, Chuanhao Yan, Chenxu Hu, and Hang Zhao. Diff-Foley: Synchronized video-to-audio synthesis with latent diffusion models. In *NeurIPS*, 2024.

Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liang Wang, Yujun Shen, Deli Zhao, Jingren Zhou, and Tieniu Tan. Videofusion: Decomposed diffusion models for high-quality video generation. *arXiv preprint arXiv:2303.08320*, 2023.

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*, 2024a.

Ze Ma, Daquan Zhou, Chun-Hsiao Yeh, Xue-She Wang, Xiuyu Li, Huanrui Yang, Zhen Dong, Kurt Keutzer, and Jiashi Feng. Magic-me: Identity-specific video customized diffusion. *arXiv preprint arXiv:2402.09368*, 2024b.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

Navonil Majumder, Chia-Yu Hung, Deepanway Ghosal, Wei-Ning Hsu, Rada Mihalcea, and Soujanya Poria. Tango 2: Aligning diffusion-based text-to-audio generations through direct preference optimization. *arXiv preprint arXiv:2404.09956*, 2024.

Ilaria Manco, Emmanouil Benetos, Elio Quinton, and György Fazekas. MusCaps: Generating captions for music audio. In *IJCNN*, 2021.

Shivam Mehta, Ruibo Tu, Jonas Beskow, Éva Székely, and Gustav Eje Henter. Matcha-TTS: A fast TTS architecture with conditional flow matching. In *ICASSP*, 2024.

Xinhao Mei, Varun Nagaraja, Gael Le Lan, Zhaoheng Ni, Ernie Chang, Yangyang Shi, and Vikas Chandra. FoleyGen: Visually-guided audio generation. *arXiv preprint arXiv:2309.10537*, 2023.

W. Menapace, A. Siarohin, I. Skorokhodov, E. Deyneka, T. Chen, A. Kag, Y. Fang, A. Stoliar, E. Ricci, J. Ren, and S. Tulyakov. Snap Video: Scaled spatiotemporal transformers for text-to-video synthesis. In *CVPR*, 2024.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022.

Midjourney. Midjourney, 2024. https://www.midjourney.com/.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on GPU clusters using Megatron-LM. In *SC*, 2021.

NVIDIA. Context parallelism overview. https://docs.nvidia.com/megatron-core/developer-guide/latest/api-guide/context_parallel.html, 2024.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. In *NeurIPS*, 2018.

OpenAI. Dall-E 3, 2024. https://openai.com/index/dall-e-3/.

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

Daniil Ostashev, Yuwei Fang, Sergey Tulyakov, Kfir Aberman, et al. MoA: Mixture-of-attention for subject-context disentanglement in personalized image generation. *arXiv preprint arXiv:2404.11565*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Pika Labs. Pika labs. https://www.pika.art/.

Ed Pizzi, Sreya Dutta Roy, Sugosh Nagavara Ravindra, Priya Goyal, and Matthijs Douze. A self-supervised descriptor for image copy detection. In *CVPR*, 2022.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

KR Prajwal, Bowen Shi, Matthew Le, Apoorv Vyas, Andros Tjandra, Mahi Luthra, Baishan Guo, Huiyu Wang, Triantafyllos Afouras, David Kant, et al. MusicFlow: Cascaded flow matching for text guided music generation. In *ICLR*, 2024.

PySceneDetect Developers. PySceneDetect. https://www.scenedetect.com/.

Bosheng Qin, Juncheng Li, Siliang Tang, Tat-Seng Chua, and Yueting Zhuang. InstructVid2Vid: Controllable video editing with natural language instructions. *arXiv preprint arXiv:2305.12328*, 2023.

Haonan Qiu, Menghan Xia, Yong Zhang, Yingqing He, Xintao Wang, Ying Shan, and Ziwei Liu. FreeNoise: Tuning-free longer video diffusion via noise rescheduling. *arXiv preprint arXiv:2310.15169*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *JMLR*, 2020.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. *arXiv preprint arXiv:1910.02054*, 2020.

Sai Saketh Rambhatla and Ishan Misra. SelfEval: Leveraging the discriminative nature of generative models for evaluation. *arXiv preprint arXiv:2311.10708*, 2023.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. ZeRO-Offload: Democratizing billion-scale model training. *arXiv preprint arXiv:2101.06840*, 2021.

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

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. DreamBooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, 2023a.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Wei Wei, Tingbo Hou, Yael Pritch, Neal Wadhwa, Michael Rubinstein, and Kfir Aberman. HyperDreamBooth: HyperNetworks for fast personalization of text-to-image models. *arXiv preprint arXiv:2307.06949*, 2023b.

RunwayML. Gen-2, 2023. https://research.runwayml.com/gen2.

RunwayML. Gen-3 Alpha, 2024. https://runwayml.com/research/introducing-gen-3-alpha.

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

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *NeurIPS*, 2016.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. LAION-5B: An open large-scale dataset for training next generation image-text models. In *NeurIPS*, 2022.

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

Kai Shen, Zeqian Ju, Xu Tan, Yanqing Liu, Yichong Leng, Lei He, Tao Qin, Sheng Zhao, and Jiang Bian. NaturalSpeech 2: Latent diffusion models are natural and zero-shot speech and singing synthesizers. *arXiv preprint arXiv:2304.09116*, 2023.

Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu edit: Precise image editing via recognition and generation tasks. In *CVPR*, 2024.

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

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-A-Video: Text-to-video generation without text-video data. In *ICLR*, 2023.

Uriel Singer, Amit Zohar, Yuval, Shelly Sheynin, Adam Polyak, Devi Parikh, and Yaniv Taigman. Video editing via factorized diffusion distillation. In *ECCV*, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Robynn J Stilwell. The fantastical gap between diegetic and nondiegetic. 2007.

Kun Su, Judith Yue Li, Qingqing Huang, Dima Kuzmin, Joonseok Lee, Chris Donahue, Fei Sha, Aren Jansen, Yu Wang, Mauro Verzetti, and Timo I. Denk. V2Meow: Meowing to the visual beat via video-to-music generation. In *AAAI Conference on Artificial Intelligence*, 2023.

Siu-Lan Tan, Matthew P Spackman, and Elizabeth M Wakefield. The effects of diegetic and nondiegetic music on viewers' interpretations of a film scene. *Music Perception: An Interdisciplinary Journal*, 34(5):605–623, 2017.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, et al. UL2: Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*, 2022.

Team Gemini. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Zeyue Tian, Zhaoyang Liu, Ruibin Yuan, Jiahao Pan, Xiaoqiang Huang, Qi fei Liu, Xu Tan, Qifeng Chen, Wei Xue, and Yi-Ting Guo. VidMuse: A simple video-to-music generation framework with long-short-term modeling. *arXiv preprint arXiv:2406.04321*, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and finetuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. FVD: A new metric for video generation. *ICLR Workshops*, 2019.

Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *NeurIPS*, 2017.

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

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

Apoorv Vyas, Bowen Shi, Matthew Le, Andros Tjandra, Yi-Chiao Wu, Baishan Guo, Jiemin Zhang, Xinyue Zhang, Robert Adkins, William Ngan, et al. Audiobox: Unified audio generation with natural language prompts. *arXiv preprint arXiv:2312.15821*, 2023.

Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. ModelScope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023a.

Qixun Wang, Xu Bai, Haofan Wang, Zekui Qin, and Anthony Chen. InstantID: Zero-shot identity-preserving generation in seconds. *arXiv preprint arXiv:2401.07519*, 2024a.

Tan Wang, Linjie Li, Kevin Lin, Yuanhao Zhai, Chung-Ching Lin, Zhengyuan Yang, Hanwang Zhang, Zicheng Liu, and Lijuan Wang. DisCo: Disentangled control for realistic human dance generation. In *CVPR*, 2024b.

Wen Wang, kangyang Xie, Zide Liu, Hao Chen, Yue Cao, Xinlong Wang, and Chunhua Shen. Zero-shot video editing using off-the-shelf image diffusion models. *arXiv preprint arXiv:2303.17599*, 2023b.

Wenjing Wang, Huan Yang, Zixi Tuo, Huiguo He, Junchen Zhu, Jianlong Fu, and Jiaying Liu. VideoFactory: Swap attention in spatiotemporal diffusions for text-to-video generation. *arXiv preprint arXiv:2305.10874*, 2023c.

Xi Wang, Nicolas Dufour, Nefeli Andreou, Marie-Paule Cani, Victoria Fernandez Abrevaya, David Picard, and Vicky Kalogeiton. Analysis of classifier-free guidance weight schedulers. *arXiv preprint arXiv:2404.13040*, 2024c.

Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data. In *ICCV*, 2021.

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

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.

Yujie Wei, Shiwei Zhang, Zhiwu Qing, Hangjie Yuan, Zhiheng Liu, Yu Liu, Yingya Zhang, Jingren Zhou, and Hongming Shan. DreamVideo: Composing your dream videos with customized subject and motion. In *CVPR*, 2024.

Yuxiang Wei, Yabo Zhang, Zhilong Ji, Jinfeng Bai, Lei Zhang, and Wangmeng Zuo. ELITE: Encoding visual concepts into textual embeddings for customized text-to-image generation. In *ICCV*, 2023.

Bichen Wu, Ching-Yao Chuang, Xiaoyan Wang, Yichen Jia, Kapil Krishnakumar, Tong Xiao, Feng Liang, Licheng Yu, and Péter Vajda. Fairy: Fast parallelized instruction-guided video-to-video synthesis. *arXiv preprint arXiv:2312.13834*, 2023a.

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

Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-A-Video: One-shot tuning of image diffusion models for text-to-video generation. In *ICCV*, 2023b.

Jay Zhangjie Wu, Xiuyu Li, Difei Gao, Zhen Dong, Jinbin Bai, Aishani Singh, Xiaoyu Xiang, Youzeng Li, Zuwei Huang, Yuanxi Sun, et al. CVPR 2023 text guided video editing competition. *arXiv preprint arXiv:2310.16003*, 2023c.

Tao Wu, Yong Zhang, Xintao Wang, Xianpan Zhou, Guangcong Zheng, Zhongang Qi, Ying Shan, and Xi Li. CustomCrafter: Customized video generation with preserving motion and concept composition abilities. *arXiv preprint arXiv:2408.13239*, 2024.

Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pre-training with feature fusion and keyword-to-caption augmentation. In *ICASSP*, 2023d.

Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Xintao Wang, Tien-Tsin Wong, and Ying Shan. DynamiCrafter: Animating open-domain images with video diffusion priors. *arXiv preprint arXiv:2310.12190*, 2023.

Yazhou Xing, Yingqing He, Zeyue Tian, Xintao Wang, and Qifeng Chen. Seeing and hearing: Open-domain visual-audio generation with diffusion latent aligners. In *CVPR*, 2024.

Hu Xu, Saining Xie, Xiaoqing Ellen Tan, Po-Yao Huang, Russell Howes, Vasu Sharma, Shang-Wen Li, Gargi Ghosh, Luke Zettlemoyer, and Christoph Feichtenhofer. Demystifying CLIP data. *arXiv preprint arXiv:2309.16671*, 2023.

Manjie Xu, Chenxing Li, Yong Ren, Rilin Chen, Yu Gu, Wei Liang, and Dong Yu. Video-to-audio generation with hidden alignment. *arXiv preprint arXiv:2407.07464*, 2024a.

Zhongcong Xu, Jianfeng Zhang, Jun Hao Liew, Hanshu Yan, Jia-Wei Liu, Chenxu Zhang, Jiashi Feng, and Mike Zheng Shou. MagicAnimate: Temporally consistent human image animation using diffusion model. In *CVPR*, 2024b.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. ByT5: Towards a token-free future with pre-trained byte-to-byte models. In *TACL*, 2022.

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. VideoGPT: Video generation using VQ-VAE and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

Wilson Yan, Andrew Brown, Pieter Abbeel, Rohit Girdhar, and Samaneh Azadi. Motion-conditioned image animation for video editing. *arXiv preprint arXiv:2311.18827*, 2023.

Shuai Yang, Yifan Zhou, Ziwei Liu, , and Chen Change Loy. Rerender A Video: Zero-shot text-guided video-to-video translation. In *SIGGRAPH Asia*, 2023.

Danah Yatim, Rafail Fridman, Omer Bar Tal, Yoni Kasten, and Tali Dekel. Space-time diffusion features for zero-shot text-driven motion transfer. *arXiv preprint arXiv:2311.17009*, 2023.

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. IP-Adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

Jae Shin Yoon, Lingjie Liu, Vladislav Golyanik, Kripasindhu Sarkar, Hyun Soo Park, and Christian Theobalt. Pose-guided human animation from a single image in the wild. In *CVPR*, 2021.

Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. *arXiv preprint arXiv:2110.04627*, 2021.

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

Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023b.

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

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. SoundStream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2022.

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

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023a.

Shiwei Zhang, Jiayu Wang, Yingya Zhang, Kang Zhao, Hangjie Yuan, Zhiwu Qin, Xiang Wang, Deli Zhao, and Jingren Zhou. I2VGen-XL: High-quality image-to-video synthesis via cascaded diffusion models. *arXiv preprint arXiv:2311.04145*, 2023b.

Yiming Zhang, Yicheng Gu, Yanhong Zeng, Zhening Xing, Yuancheng Wang, Zhizheng Wu, and Kai Chen. FoleyCrafter: Bring silent videos to life with lifelike and synchronized sounds. *arXiv preprint arXiv:2407.01494*, 2024.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588*, 2024.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *ECCV*, 2022.

Ye Zhu, Kyle Olszewski, Yuehua Wu, Panos Achlioptas, Menglei Chai, Yan Yan, and S. Tulyakov. Quantized GAN for complex music generation from dance videos. *arXiv preprint arXiv:2204.00604*, 2022.

Le Zhuo, Zhaokai Wang, Baisen Wang, Yue Liao, Stanley Peng, Chenxi Bao, Miao Lu, Xiaobo Li, and Si Liu. Video background music generation: Dataset, method and evaluation. In *ICCV*, 2022.

Liu Ziyin, Tilman Hartwig, and Masahito Ueda. Neural networks fail to learn periodic functions and how to fix it. In *NeurIPS*, 2020.

Appendix

A Additional Model Training Details

A.1 Text encoders

As described in Section 3.1.4 we use text embeddings from three text encoders. We provide details on the text encoders and how they are used for visual-text generation.

Long-prompt MetaCLIP training. We train our own CLIP-style (Xu et al., 2023) model that can process long text prompts up to 256 text tokens. We utilized synthetic image captions generated by an image captioning model (Dubey et al., 2024) to finetune the MetaCLIP (Xu et al., 2023) text encoder. We expanded the position embedding in the text encoder to 256 tokens, allowing it to handle longer input sequences. However, we found that finetuning all parameters in the text encoder led to rapid overfitting and suboptimal text encoding performance. We hypothesize that this is due to the uniform text style in the synthetic long image captions. To address this issue, we froze both the image and text encoders and finetuned the position embedding and all biases in the text encoder using a mix of long synthetic, short synthetic, and human-annotated captions.

Visual text generation. We implemented several key modifications to the input and prompt rewriting, model architecture, and data processing steps to enable visual text generation. Specifically, our approach consists of three main components. First, we assume that text enclosed within quotation marks (**“ ”**) in the input text prompt needs to be generated as visual-text. We also employ prompt rewriting (see Section 3.4.1) to automatically identify such text at inference time. Second, we use the character-level ByT5 encoder for encoding the text within the quotation marks. Finally, we ensure that our pre-training data has a good mix of visual text that covers 10-50% of the image area by leveraging OCR detection models.

A.2 Model scaling and training efficiency

For memory capacity constrained models like Movie Gen Video with a context length of 73K, sub-optimal performance may be achieved through memory-saving techniques like activation checkpointing (AC). AC reduces peak memory demand by trading off FLOPs and memory, which can be sometimes be more effective than fully-optimized parallelisms due to physical training system constraints, *e.g.*, the FLOPs/sec and HBM capacity of each GPU, and the intra- and inter- node GPU-GPU interconnect bandwidths.

Overlapping communication and computation.

While the parallelism techniques mentioned in Section 3.1.6—which aim to partition training FLOP and memory demands across GPUs at the cost of added communication—are successful at enabling the 0-to-1 ability to train such large sequence transformer models, their direct implementation and composition come with overheads and inefficiencies with respect to memory and communication. As a result, under such overheads optimal *realized* performance for a model which is memory capacity constrained, as-is Movie Gen Video with a context length of 73K, may actually be achieved through the use of other memory-saving techniques in addition to, or even in place of, the above parallelisms.

Given that AC can never provide strong scaling due to strictly increasing the amount of work needed to compute a training step, we focused our efforts on improving the scaling characteristics of model parallelism techniques. Specifically, we aimed to have a final model parallelism implementation which achieved as close to strong scaling as possible for both: 1) activation memory size (to reduce the usage of AC), and 2) forward/backward step time.

As the four existing model parallel techniques discussed in Section 3.1.6 (TP, SP, CP, FSDP) all already achieve strong theoretical FLOP scaling, we began by: 1) determining the scaling characteristics of the activation sizes of our model under these techniques as well as 2) building an analytical framework to model the change and interdependence of the compute and communication times of our model execution. Using both, we: 1) identified all occurrences of duplicated activations, and 2) identified which inter-GPU communications are required as well as exposed.

From this we designed and implemented a model parallel solution for the MOVIE GEN VIDEO backbone based upon the principles of TP, SP, and CP, which: 1) achieves strong activation memory scaling, and 2) minimizes exposed communication time for our training cluster. This enabled us to achieve close-to-strong scaling even with model parallelism widths spanning multiple nodes. We achieved such performance through a custom implementation defining the execution of both the forward and backward paths of the MOVIE GEN VIDEO backbone block. Forgoing the use and ease of chaining smaller autograd operators together allowed us to precisely partition, control, and overlap compute and communication execution. This also enabled us to transparently apply techniques such as selective recomputation without any performance loss.

Our implementation is fully written in PyTorch and compiles into CUDAGraphs supporting the execution of variable sized inputs at the start of every training or inference job, dynamically based on the specified model and system configurations.

Sharding plan generation and selection. We expanded the performance modeling tools developed above to further estimate the memory utilization and latency of end-to-end model execution (*e.g.*, backbone blocks, text encoders, TAE). We then utilized this to generate multiple sharding and parallelism plans, which exhibit similar theoretical and estimated latencies, and are deemed valid as they are estimated to fit within the available GPU High-Bandwidth Memory (HBM).

This development: 1) enabled us to generate sharding plans in which different components and stages of the end-to-end model execution can be sharded by different strategies; and 2) allowed us to empirically identify training parallelism configurations which have an approximately neutral batch-size to step time scaling relationship.

The ability to accomplish the latter was important for the successful training of MOVIE GEN VIDEO due to the relationship of model parallel size to the global batch size of its corresponding training step. Specifically, both the number of GPUs over which parameters are disjointly sharded (TP) and the number of GPUs over which the input sequence of a single sample are disjointly sharded (SP, CP), proportionally reduce the effective global batch size—and impact the wall latency of—the corresponding training step. Identifying groups of sharding plans which have neutral batch-size to step time scaling allows us to effectively scale the number of optimizer steps, while holding the number of GPUs and total training data size constant. Although the training data throughput and end-to-end efficiency of such groups of sharding plans are similar, the number of optimizer steps taken to process varying amounts of training data across various stages of training can significantly impact the final model's quality.

The final sharding plan used during the the most expensive stage of MOVIE GEN VIDEO training, processing 768px video inputs with a per-sample token sequence length of 73K, was the following:

- **Text encoders.** Due to their relatively small size and weights being frozen, ByT5 and Long-Prompt MetaCLIP were replicated on all GPUs. UL2 however has significantly more parameters and memory overhead, yet it is still relatively small in terms of end-to-end execution latency, and was sharded FSDP-only across the DP-group of each TP-rank.
- **TAE.** Although containing a relatively small number of frozen parameters, the size of intermediate activations of the TAE can become prohibitively expensive as the input size grows. Furthermore, unlike the text encoders, the latency of the TAE is non-trivial with respect to the end-to-end step time, and unlike the backbone, it is non-trivial to efficiently partition and model parallelize the TAE's execution. These limitations resulted in us performing a data pre-processing step where the latents for high resolution video inputs were pre-computed and cached prior to their ingestion in the MOVIE GEN VIDEO backbone training pipeline.
- **Movie Gen Video backbone.** While a transformer block at its core, the MOVIE GEN VIDEO backbone has additional learned components, such as factorized positional embeddings and per-context embedders, each with not only their own memory and compute requirements but also their own input and output activation connections. This results in the backbone parameter sharding and input and output activation and gradient flow changing as the model moves through its different stages. The final backbone contained interconnected sections sharded: FSDP-only (*e.g.*, patchifier), FSDP+TP (*e.g.*, context embedders), FSDP+TP+SP (*e.g.*, cross-attention), and FSDP+TP+SP+CP (*e.g.*, self-attention).

Curation Step	Thresholds	Remaining volume in %
Duration	4s $\leq$ duration $\leq$ 120s	100
Resolution	width $\geq$ 768 and height $\geq$ 768	25
Aspect Ratio	width $\geq$ height	7
No Text	No sampled frame with word detection score * word recognition score $\geq$ 0.6	1.94
No Border	No videos with borders around them	1.87
No Scene Change	1 clip of duration 12s to 16s sampled from 1 scene in a video	1.78
Aesthetics	aesthetic score on middle frame of clip $\geq$ 4.0	1.57
No Slow Motion	motion score $>$ 2.0, motion vector average $>$ 0.5, motion vector average $<$ 7	1.32
No Jittery Motion	Number of shots per second $<$ 0.85	1.22
No Content Duplicates	Embedding cosine similarity $<$ 0.99	1.15
Concept Resampling	Volume per cluster: 1/sqrt(cluster size)	0.94

Table 38 Volume drop during high resolution set curation. Note that our data acceptance rate with these thresholds is less than 1%.

B Additional Data Details

B.1 Video Data Curation Thresholds

In this section, we share more details about models used in data curation and the corresponding thresholds used.

OCR model. Our internal OCR model samples frames adaptively, detects words within those sampled frames, and then recognizes the text of those detected words. We only retained videos where the word detection score multiplied by the word recognition score was below 0.6 for all sampled frames.

Border detection. We noticed that the presence of borders in training videos resulted in generated videos having black borders around them. This issue is particularly common in portrait-mode videos. We removed such videos by writing a simple border detector based on first order derivative calculations. We first detect pixels with large vertical and horizontal deltas and then apply a scanning line algorithm to find the borders.

Clip sampling. With an average duration of 28 seconds, our raw videos needed to be clipped into shorter segments to meet our Movie Gen Video model's training requirements of 4-16 second clips. However, we noticed that randomly sampling clips without considering scene boundaries leads to generating videos with frequent and abrupt scene changes. Thus, we used FFmpeg (FFmpeg Developers) to detect scene changes and sampled 1-2 scenes with duration exceeding 16 seconds from each video. We then randomly extracted a single clip from each scene, with a duration ranging from 4-16 seconds, to use as a training clip. More than 50% of our training clips have duration ranging from 15 seconds to 16 seconds.

Aesthetic filtering. We removed clips with poor aesthetic quality such as blurry or compressed clips by applying the public LAION aesthetics image model (Schuhmann et al., 2022) on the middle frame of each clip. We removed all clips with an aesthetic score less than 4, ensuring to have high-quality clips for training. We also calculated average aesthetic scores across multiple frames in a clip and observed that multi-frame aesthetic score didn't lead to a significant increase in the recall of poor-quality clips.

Jittery motion detection. FFmpeg (FFmpeg Developers) motion scores and motion vectors struggle to detect videos with frequent, jittery camera movements, which ultimately leads to our model generating videos with a jittery quality. We noticed that the Shot Boundary Detection (SBD) from PySceneDetect (PySceneDetect Developers) breaks down jittery videos into numerous false-positive shots. To identify and remove jittery videos, we used the number of shots detected per second, removing clips with a rate exceeding 0.85 shots per second.

Data volume per filter. We analyzed the data volume drop at each filtering step when using our most strict curation thresholds in Table 38. These thresholds were used to curate our high resolution set.

B.2 Camera Motion Control types

Here, we explain in more detail the different kinds of camera motion control that we train our model for. As described in Section 3.2.1 in the technical report, to enable cinematic camera motion control, we train a camera motion classifier to predict 16 different camera motion types. The predictions from this classifier are prefixed to the training captions. The 16 camera motion control types are: zoom in, zoom out, push in, pull out, pan right, pan left, truck right, truck left, tilt up, tilt down, pedestal up, pedestal down, arc shot, tracking shot, static shot, and handheld shot. As detailed in Section 3.3, during supervised finetuning, we label 6 additional camera motion and position types in the finetuning set. These include: wide angle, close-up, aerial, low angle, over the shoulder, and first person view.

C Additional Evaluation Details

C.1 Annotation Variance Study on Text-to-Video Evaluation

To ensure trustworthy human annotation results and assess the significance of the winning or losing outcomes, we analyze the annotation variance across each evaluation axis. Specifically, we repeated the same annotation tasks four times using a subset of 381 prompts for text-faithfulness, quality, and realness & aesthetics A/B tests. We calculate the standard deviation of the net win rate (win% - lose%) for each evaluation axis. This estimation is detailed in Table 39. In Section 3.6.1 we use these standard deviations to gauge the statistical significance of the results.

As shown in Table 39, the overall quality axis exhibits higher variance than text-faithfulness, mainly due to the subjectivity introduced by combining different evaluation signals within the overall quality axis. Among the quality axes, frame consistency displays a higher variance than the others, as determining which video has greater distortion is more challenging than judging which has larger or more natural overall motion. Furthermore, realness demonstrates less variance than aesthetics, as it is generally more objective to identify *generated looking* content (for the realness axis) than to align on a universally pleasing aesthetic definition (for the aesthetic axis).

Text Faithfulness	Overall	Frame consistency	Motion Completeness	Motion Naturalness	Realness	Aesthetics
3.74%	5.07%	4.08%	3.98%	1.68%	2.52%	4.84%

Table 39 Evaluation Variance. Each number is acquired by sending the same A/B testing four times and compute standard deviation on the net win rate of each evaluation metric.

C.2 T2V comparison to prior work

In Section 3.6.1 in the main paper, we compare to prior works for text-to-video generation. Here, we provide extra details on how we obtain generated videos for each method and on how we post-process our generated videos to ensure fair comparison and reduce annotator bias. The size parameters for the videos from prior work that we use for comparion are shown in Table 40. We assume that many of the black box industry models that we compare to are being updated and improved over time. Hence, we include the dates on which we collected the videos from website.

OpenAI Sora. Our only option for comparing to OpenAI Sora is by using the prompts and videos from their publicly released website (158 videos in total). We note that for these closed source methods, the only videos released publicly on their website are likely to only represent their "best" samples, obtained through some unknown amount of cherry picking. As discussed in Section 3.6.1 in the main paper, for fair comparison to OpenAI Sora we hence also select samples from Movie Gen Video using what we consider a modest amount of cherry picking. Specifically, for each prompt, we generate 5 different videos from Movie Gen Video using different random seeds. From each of these 5 videos, we manually pick the "best". The videos released by OpenAI Sora are at a variety of different resolutions. Specifically, a small number of videos are 1080p HD, whilst the majority are at 1280×720, whereas all videos from Movie Gen Video are 1080p HD. For fair comparison, and to reduce annotator bias in the human evaluation, we adaptively spatially downsample our

	Generation Specs				
	Runway Gen3	LumaLabs	OpenAI Sora	Kling1.5	Movie Gen
Resolution	1280×768	1360×752	1920×1080	1920×1080	1920×1080
Duration	10s	5s	10s	10.43s	10.67s
#Video frames	256	121	300	313	256
FPS	24	24	30	30	24

Table 40 We compare with the above generation specification with external works. Runway Gen3 videos are collected on September 23th, 2024. LumaLabs videos are collected on September 23th, 2024. Kling1.5 videos are collected on September 25th, 2024. The videos released from OpenAI Sora are at a variety of different resolutions and durations. We note that the max resolutions of the released videos is 1920×1080, whereas the majority of videos are at 1280×720 and 10s in duration.

generated videos such that they are the same resolution as the corresponding OpenAI Sora video for each prompt (we do this for all prior work comparisons). The videos released by OpenAI Sora are at a variety of different durations, with the majority being 10s, whereas all videos from Movie Gen Video are 10.66s. For fair comparison, we adaptively temporally center crop either our or OpenAI Sora's videos such that they are the same duration for each prompt, whilst retaining the original frame rate. For the OpenAI Sora comparisons, we sampled from Movie Gen Video using 500 linear steps.

To allow for fair comparison to Movie Gen Video for future work, we hope to release random (non cherry picked) generated videos for the Movie Gen Video Bench (see Section 3.5 in the main paper).

C.3 Correlations between audio-based objective and subjective metrics

We show the relationship between the objective metrics and subjective metrics presented in Section 6.3.1. Using around 10,000 annotations over 53 evaluations for each of the Video-Audio Alignment and Audio Quality tasks, we explore (1) how well we can ascertain system-level net win rate on a subjective metric from the objective metrics, and (2) on an item level, how subjective scores depend on objective metrics.

For brevity we choose to focus on a single aspect of both tasks: "Overall" audio quality for the Audio Quality pairwise task and the "Correctness" aspect of the Video-SFX alignment task. For Audio quality, we observe a high degree of correlation between the aspects: "overall", "professional" and "naturalness" have Pearson correlation coefficients of 0.9 or higher. For the Video-SFX alignment task, the correctness and synchronization aspects have similarly high observed correlation of 0.76.

For the Text-Alignment task, we combine annotated precision and recall into an F_1 score; the 1-5 scores for both are mapped to (20%, 40%, 60%, 80% and 100%) for both precision and recall. The number of systems evaluated for the Text-Alignment task is small, so we do not consider system-level correlations.

C.3.1 System-level correlations

Figure 36 shows how system-level pairwise performance based on subjective evaluations (using net win rate) compares to the mean difference of the item-level objective metrics.

We find that objective metrics are predictive of subjective measures but with caveats. For instance, there is a significant amount of model-specific bias, meaning two model pairs with the same mean objective score difference may have different or opposing net win rates, which means relying solely on differences in objective metrics to make superiority claims is risky. Pairwise comparisons of Movie Gen Audio with external baselines (non-ablation ones) show larger net win rates on Audio Quality at a given difference in audio quality score, which may indicate that Movie Gen Audio improves aspects of perceived audio quality not captured by audio quality score alone.

If we rely on ablations alone and consider model pairs (33 evaluations) where differences in objective scores are statistically significant, we find that significant differences in audio quality score correctly predict overall audio quality preferences in 21 out of 24 comparisons (87.5% of the time, with a 95% CI: 71.7 - 96%). ImageBind score was similarly predictive of overall audio quality preference but differences in ImageBind score were

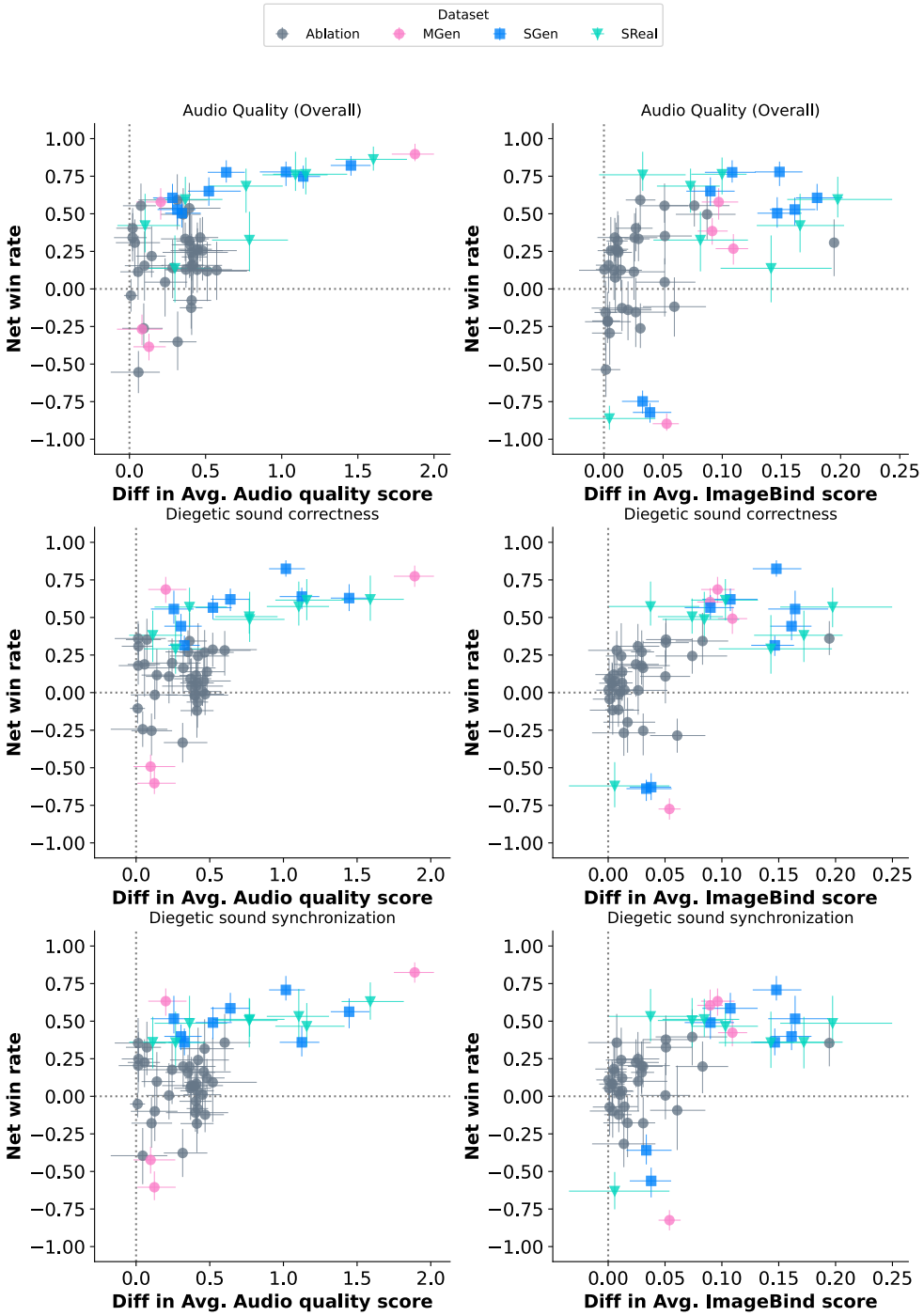


Figure 36 Comparing subjective and objective metrics at the system level. Each scatter point is a pair of models evaluated on a given dataset. The x-axis shows the absolute value of the mean item-level difference in objective metric and the y axis shows the net win rate for the model with the higher mean item-level difference in objective scores. Grey scatter points show model ablation comparisons for Movie Gen Audio, other points show pairwise comparisons between Movie Gen Audio and an external baseline.

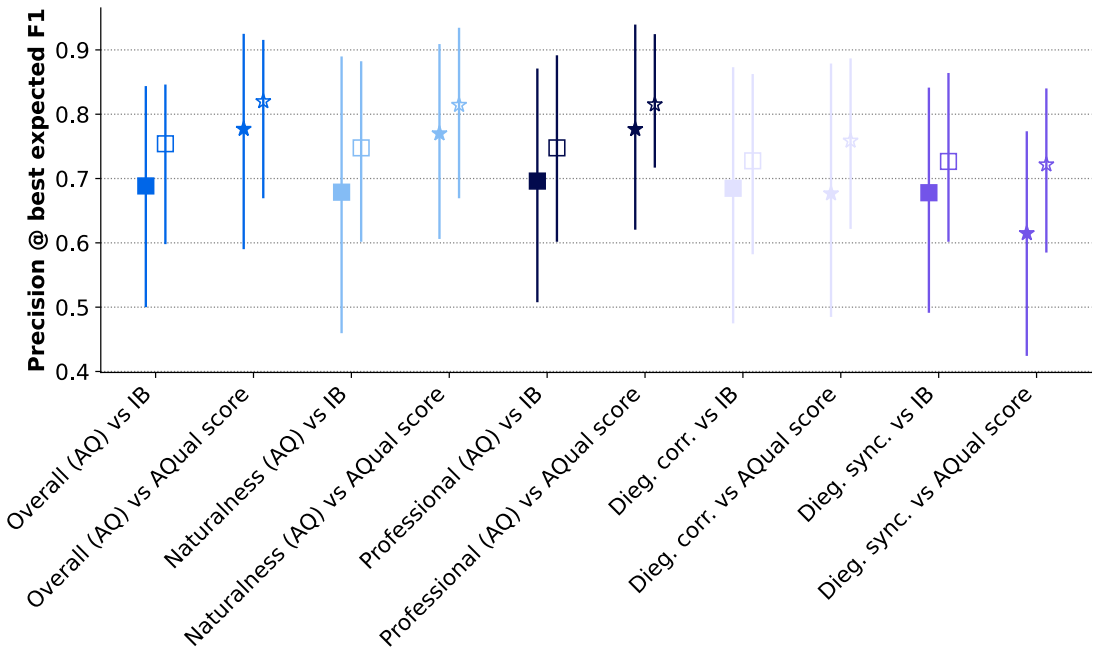


Figure 37 Precision at the system level: fraction of model pairs for which a given objective metric correctly predicts the human preference. 95% CI obtained via bootstrap resampling of both items and model pairs. Filled markers represent only considering model ablations, and unfilled markers consider both model ablations and comparisons between Movie Gen Audio and external baselines.

smaller, so only 17 out of 33 pairs had statistically significant differences in ImageBind scores, and of these 82.3% (95% CI: 63.6% - 94.5%) correctly predicted preferences on overall audio quality.

Figure 37 shows the precision (the fraction of model pairs where the average difference between the objective metric correctly predicts the subjective preference) for the best expected F1 score for each (objective metric, subjective metric) pairing. We also show 95% confidence intervals obtained from bootstrap resampling of both items and model pairs. We find due to the limited sample that precision estimates are quite uncertain but that (1) audio quality score tends to be a better predictor of audio quality preference than ImageBind, while both ImageBind and audio quality score seem to be comparably good predictors of Video-SFX Alignment aspects, and (2) both metrics are more predictive for the sample of model pairs comparing Movie Gen Audio and an external model.

C.3.2 Item-level correlations

Figure 38 shows how the (log) odds of a model generation being preferred depend on various objective metrics. The value shown on the y-axis is obtained by fitting a linear model with a Bradley-Terry likelihood (Bradley and Terry, 1952) on the observed subjective evaluations. We model the latent quality vector for model m and item i $z_{mi} = \beta_m^{(0)} + \sum_r \beta_{r,g_i} x_{mir}$ where $\beta_m^{(0)}$ is an offset parameter for each evaluated model, g_i indicates the group of the i-th item (in this case the presence or absence of non-diegetic music), and r is the index of the regression variable – in this case the r-th bin of the objective metric, and $x_{mir} = 1$ when the item i for model m has an objective metric that falls in the r-th bin and 0 otherwise. We then show β_{r,g_i} in the y axis of Figure 38. Note that for each subjective measure, we only regress on one objective metric for these visualizations.

The relative difference in the latent quality corresponds to the log-odds of one item being preferred to the other.

We make several observations

- Both audio quality score and ImageBind score are predictive of overall audio quality, with a generation

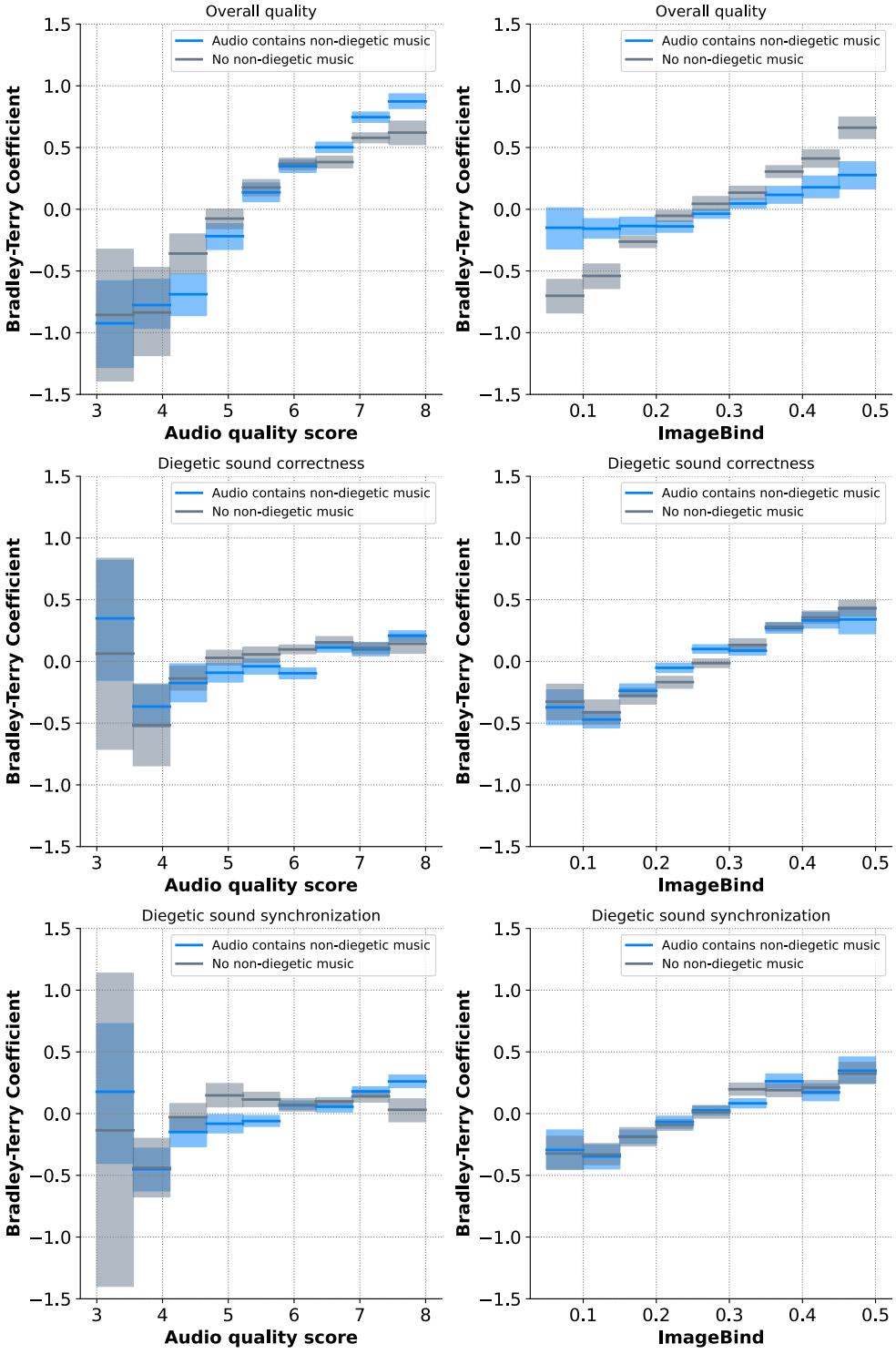


Figure 38 We show a regression coefficient for a given subjective preference metric as a function of objective scores. The difference between the coefficients of two items indicates the log-odds of the item being preferred on a given subjective aspect.

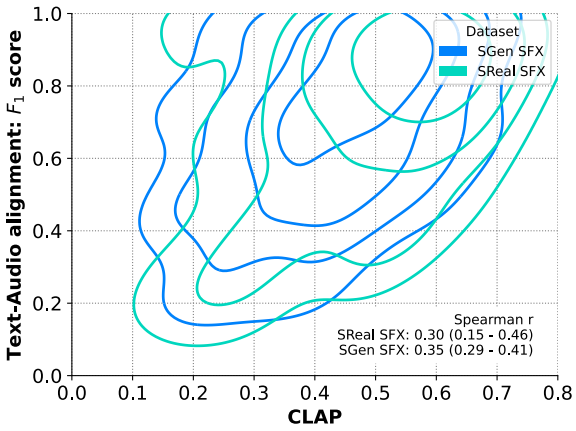


Figure 39 Above shows the correlation between consensus subjective evaluations of Text-Audio alignment and CLAP. We find moderate correlation (Spearman r of ~ 0.3 for single-scene (real) videos and ~ 0.35 for generated single-scene videos).

with the best observed ImageBind score having $\sim 4 - 4.5$ to 1 odds in favor of being preferred to a generation with the lowest observed ImageBind score.

- However, when non-diegetic music is present, the ImageBind score becomes less predictive of preferences for overall audio quality.
- A bad (< 4) audio quality score does indicate a slightly lower odds of an item showing stronger Video-SFX alignment ("correctness" aspect) compared to an item with a very high audio quality score ($\sim$1.5 to 1), but the impact of audio quality score improving past a $\sim 4 - 5$ has little to no further impact.
- Gains in overall audio quality tend to saturate at audio quality scores above ~ 6.5, but only for items without non-diegetic music. Items with non-diegetic music continue to see higher preferences for audio quality scores beyond the ~ 6.5 point, which is also reflected in direct tests shown in Table 32.
- Increases in ImageBind score tend to have a stronger relative impact on overall audio quality than on Video-SFX alignment.
- ImageBind scores do tend to monotonically increase Video-SFX correctness without saturating.
- Diegetic synchronization does seem to be correlated with ImageBind like diegetic correctness, however this is likely not causal based on findings that IB score is insensitive to shifts in audio. Instead, model changes that improve correctness likely also improve synchronization, leading to a spurious correlation between synchronization and IB score.

For the Text-Audio alignment subjective evaluations, where we measure both recall and precision in 20% increments, we report correlations with CLAP (Wu et al., 2023d) scores in Figure 39. We find that on the item level, there is moderate correlation. To compute confidence intervals via bootstrap we utilize a two-stage approach to also account for annotation error; we first resample evaluated items, then resample the individual annotations for each item and compute consensus score on the bootstrap sample. We find nominally that single-scene generated videos have a higher Spearman correlation between CLAP and human evaluations compared to single-scene real videos, however this difference is not statistically significant.

D Additional Results

D.1 Additional Audio Generation Results

D.1.1 Additional Metrics for Main Results on Sound Effect Generation

We present results on objective metrics and additional subjective metrics for the "sound effect generation" experiments in 6.4.1. Table 41 should be compared against Table 29.

Dataset	Baseline	Type	Subjective		Objective		
			TA Align		Quality	VA Align	TA Align
			Recall (%) ↑	Precision (%) ↑	AQual ↑	IB ↑	CLAP ↑
SReal SFX	Diff-Foley	V2A	-	-	5.68	0.28	0.36*
	FoleyCraft	V2A	-	-	6.03	0.31	0.35*
	VTA-LDM	V2A	-	-	5.97	0.30	0.35*
	Seeing&Hearing	V2A	-	-	5.21	0.38	0.33*
	Seeing&Hearing	TV2A	$63.0_{\pm 6.7}$	$56.6_{\pm 6.5}$	5.70	0.34	0.47
	PikaLabs	V2A	-	-	6.45	0.18	0.28*
	PikaLabs	TV2A	$64.5_{\pm 7.7}$	$62.4_{\pm 8.3}$	6.69	0.21	0.43
	ElevenLabs	T2A	$82.1_{\pm 5.9}$	$78.8_{\pm 6.6}$	6.53	0.24	0.54
	Movie Gen Audio	TV2A	$84.7_{\pm 4.8}$	$75.5_{\pm 5.0}$	6.76	0.38	0.51
SGen SFX	Diff-Foley	V2A	-	-	5.48	0.18	0.17*
	FoleyCraft	V2A	-	-	6.03	0.24	0.25*
	VTA-LDM	V2A	-	-	6.03	0.22	0.25*
	Seeing&Hearing	V2A	-	-	5.21	0.38	0.27*
	Seeing&Hearing	TV2A	$72.3_{\pm 3.8}$	$67.2_{\pm 3.8}$	5.49	0.37	0.42
	PikaLabs	V2A	-	-	6.18	0.16	0.26*
	PikaLabs	TV2A	$63.0_{\pm 4.6}$	$68.7_{\pm 5.1}$	6.20	0.18	0.41
	ElevenLabs	T2A	$71.0_{\pm 3.8}$	$68.7_{\pm 3.9}$	6.39	0.19	0.45
	Movie Gen Audio	TV2A	$73.9_{\pm 3.7}$	$71.8_{\pm 3.8}$	6.47	0.34	0.46

Table 41 Additional sound effect generation subjective and objective evaluations. This table reports additional metrics on the SReal and SGen SFX benchmarks. TA and VA are abbreviations for text-audio and video-audio respectively. We put "*" on CLAP results when text is not used at inference.

We first note that on text-audio alignment, Movie Gen Audio outperforms all baselines supporting text input on recall (how many sound effects described in the text caption are generated), which is the main metric we concern for the text-audio alignment axis. It also outperforms most baselines on precision and is on par with ElevenLabs on the real videos. It should also be noted that CLAP score between TV2A are similar, which does not correlate strongly with the relative ranking. However, CLAP is much higher for TV2A models compared to V2A models, which shows its discriminative power when the delta is large enough, as we expect V2A models to generate audio that has much worse alignment with text not they are not conditioned on.

For audio quality, we find reasonable correlation between AQual and subjective pairwise comparison. On pairwise subjective tests Movie Gen Audio outperforms all baselines, and PikaLabs and ElevenLabs have the smallest gaps. On the objective metrics, Movie Gen Audio also has the highest score, followed by those two models.

On video-SFX alignment, the correlation between objective and subjective is much weaker. We first note that Seeing&Hearing directly use ImageBind for classifier-guidance which maximizes that. Therefore, both the V2A and TV2A variant show IB score on par with Movie Gen Audio, despite that the subjective tests reveal that the video-SFX alignment is much worse compared to Movie Gen Audio. Next, we observe that Movie Gen Audio achieves the higher IB score compared to other baselines, and also outperforms them on subjective evaluations. However the ranking for the baselines in terms of relative performance to Movie Gen Audio on subjective tests does not correlate highly with IB ranking.

Dataset	Baseline	Type	Objective		
			Quality AQual ↑	VA Align IB ↑	TA Align CLAP ↑
Movie Gen Audio Bench SFX	Diff-Foley	V2A	5.26	0.21	0.19*
	FoleyCraft	V2A	5.73	0.25	0.24*
	VTA-LDM	V2A	5.74	0.22	0.17*
	Seeing&Hearing	V2A	5.21	0.37	0.25*
	Seeing&Hearing	TV2A	5.52	0.33	0.43
	ElevenLabs	T2A	6.17	0.20	0.47
	Movie Gen Audio	TV2A	6.39	0.36	0.45

Table 42 Sound effect generation objective evaluation on Movie Gen Audio Bench. TA and VA are abbreviations for text-audio and video-audio respectively. We put "*" on CLAP results when text is not used at inference. Similar trends as in tables 29 and 41 are observed.

Objective results on Movie Gen Audio Bench for sound effect generation are shown in Table 42. We note that similar trends are observed, where Movie Gen Audio leads in AQual and IB (except when compared to Seeing&Hearing which optimizes IB in inference), and on par with baselines on CLAP.

D.1.2 Control audio quality through text prompts

Figure 40 and Figure 41 present examples for audio quality control via text prompts for sound effect generation and joint music and sound effect generation respectively. When conditioned on prompts indicating low quality (*i.e.*, quality score of 5.0), Movie Gen Audio generates natural but low quality audio that contains for example wind noises or music with broken bass.

D.1.3 Control music style through text prompts

Figure 42 shows an example of varying music captions for the same video.

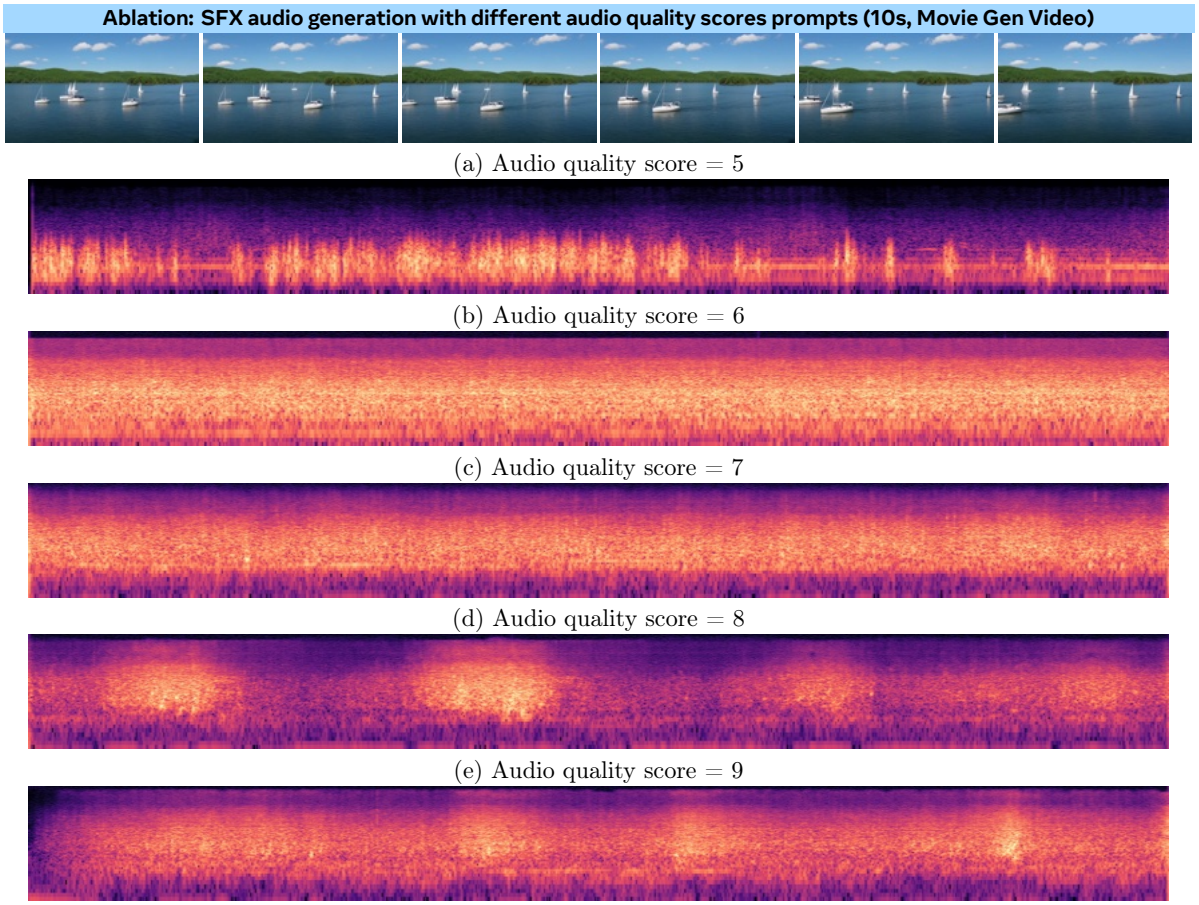


Figure 40 Examples of generated audio with different audio quality score in the text prompt with Movie Gen Audio. For these samples, higher audio quality scores tends to produced audio without wind noises compared to lower audio quality scores. Videos in this Figure found at https://go.fb.me/MovieGen-Figure40.

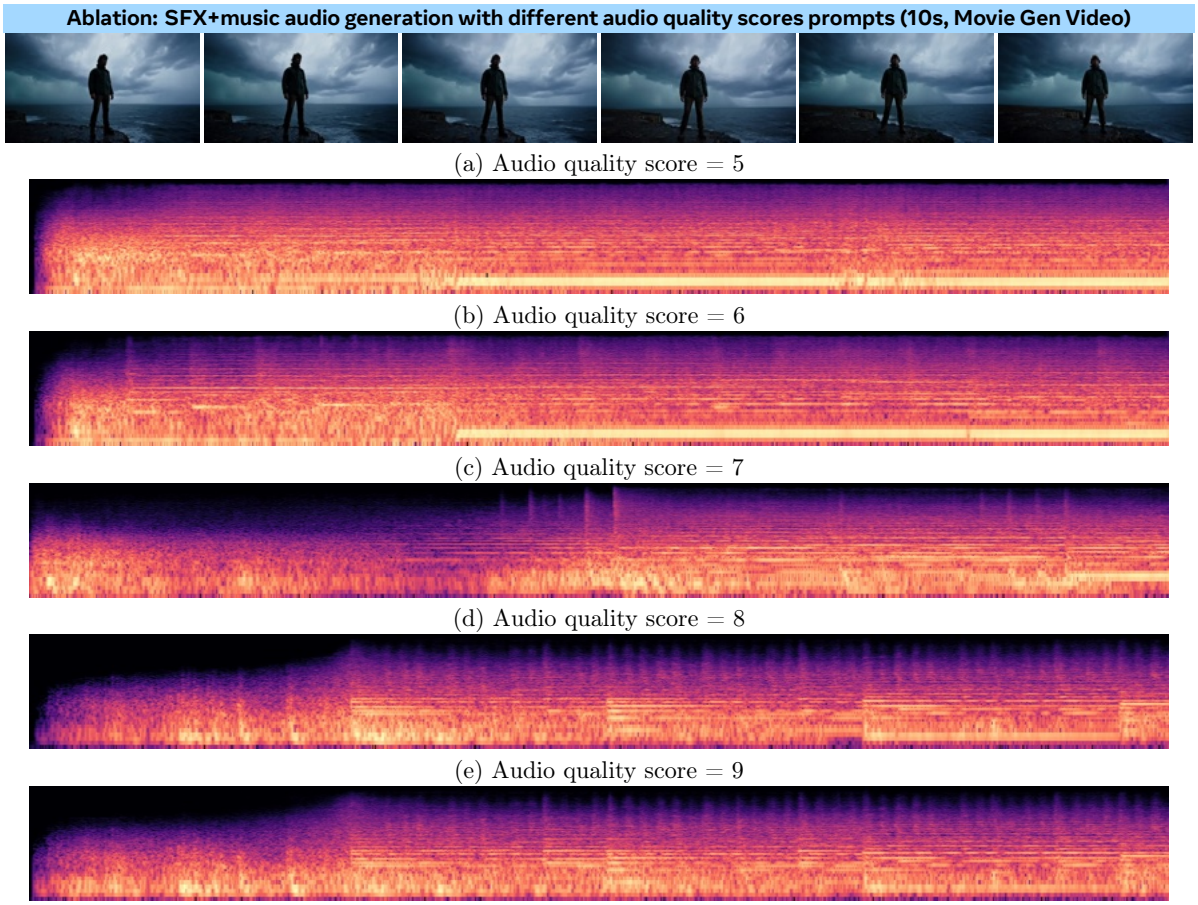


Figure 41 Examples of generated audio with different audio quality score in the text prompt with Movie Gen Audio. Movie Gen Audio controls the generated audio output quality using the input text prompts (refer to 26). As we can observed in the spectrogram plot, lower quality scores contains some high frequency noises and by gradually increasing the quality scores, we got a cleaner audio spectrogram. Videos in this Figure found at https://go.fb.me/MovieGen-Figure41.

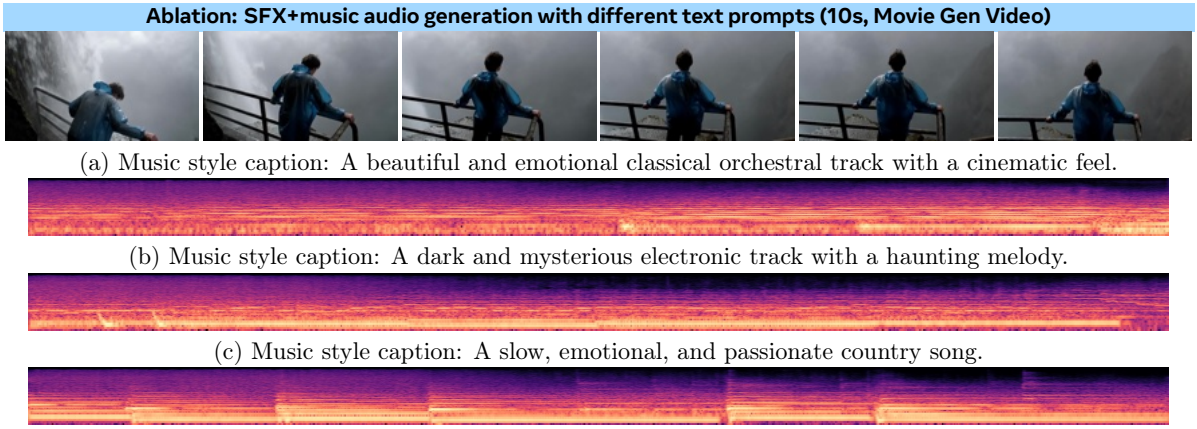


Figure 42 Examples of generated audio with different music description in the text prompt with Movie Gen Audio. Videos in this Figure found at https://go.fb.me/MovieGen-Figure42.