Mobile Video Diffusion

Haitam Ben Yahia*† Denis Korzhenkov*‡ Ioannis Lelekas‡
Amir Ghodrati† Amirhossein Habibian‡

‡Qualcomm AI Research

{dkorzhen, ilelekas, ahabibia}@qti.qualcomm.com

Abstract

Video diffusion models have achieved impressive realism and controllability but are limited by high computational demands, restricting their use on mobile devices. This paper introduces the first mobile-optimized image-to-video diffusion model. Starting from a spatio-temporal UNet from Stable Video Diffusion (SVD), we reduce the computational cost by reducing the frame resolution, incorporating multi-scale temporal representations, and introducing two novel pruning schemas to reduce the number of channels and temporal blocks. Furthermore, we employ adversarial finetuning to reduce the denoising to a single step. Our model, coined as MobileVD, can generate latents for a $14 \times 512 \times 256$ *px clip in* 1.7 *seconds on a Xiaomi-14 Pro, with negligible quality loss. Our results are available at* https://qualcomm-ai-research.github.io/mobile-video-diffusion

1. Introduction

Video diffusion models are making significant progress in terms of realism, controllability, resolution, and duration of the generated videos. Starting from zero-shot video models [11, 26, 30, 36], which deploy pretrained image diffusion models to generate consistent frames, *e.g*., through cross-frame attention, modern video diffusion models rely on spatio-temporal denoising architectures, *i.e*., 3D UNets [2, 3, 16] or 3D DiTs [33, 68, 74]. This involves inflating image denoising models by adding temporal transformers and convolutions to denoise multiple frames simultaneously. Despite their impressive generation qualities, spatio-temporal denoising architectures demand high memory and computational power, which limits their usage to clouds with high-end GPUs. This hinders the wide adoption of video generation technology for many applications that require generating content locally on mobile devices.

Prior work on accelerating video diffusion models has mostly focused on reducing the number of sampling steps [62, 72]. By extending the consistency models [58] and adversarial distillation [54] to video diffusion models, they managed to reduce the number of denoising steps from 25 to only 4 [62] and 1 step [72], which tremendously accelerates video generation. However, step distillation alone does not reduce the memory usage of the model, which is the key setback in deploying video diffusion models on mobile devices.

This paper is the first attempt to build image-to-video diffusion models for mobile. Starting from model of Stable Video Diffusion (SVD) [2, 3], as a representative for video diffusion models, we conduct a series of optimizations on its spatio-temporal UNet to build a *mobile-friendly UNet*: driven by the lower resolution needs for user-generated content on phones, we first opt for using a smaller latent resolution for generating 512×256 px frames. Additionally, instead of preserving the number of frames throughout the denoising UNet, we introduce additional temporal down- and up-sampling operations to extend the multi-scale representation both in space and time, which reduces the memory and computational cost with minimal loss in quality. Moreover, we discuss how naive visual conditioning through cross-attention leads to significant computational overhead that can be avoided without damaging visual quality.

We further accelerate the mobile-friendly UNet by reducing its parameters using a novel channel compression schema, coined *channel funneling*, and a novel technique to prune the temporal transformers and temporal residual blocks from the UNet. Finally, following Zhang et al. [72], we reduce the number of denoising steps to a single step using adversarial finetuning. This results in the first mobile video diffusion model called MobileVD, which is able to generate 14 latent frames of resolution 512×256 in 1.7 seconds on a Xiaomi 14-Pro smartphone at a slightly worse quality in terms of FVD, 149 vs. 171.

*Equal contribution
†Work done while at Qualcomm AI Research
‡Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc. Snapdragon and Qualcomm branded products are products of Qualcomm Technologies, Inc. and/or its subsidiaries.

2. Related work

Video generation. Fueled by advancements in generative image modeling using diffusion models, there has been notable progress in the development of generative video models [2, 3, 13, 14, 21, 25, 32, 45, 52, 75]. These video models generally evolve from image models by incorporating additional temporal layers atop spatial blocks or by transforming existing 2D blocks into 3D blocks to effectively capture motion dynamics within videos. Although these advancements have paved the way for the generation of high-resolution videos, the significant computational demands make them impractical for use on low-end devices. In this work, we address this by optimizing a representative of video generative model, SVD [2], to make it accessible to a broader range of consumer-graded devices.

Diffusion optimization. The problem of making diffusion models efficient naturally consists of the following two parts: (i) reducing the number of denoising steps and (ii) decreasing the latency and memory footprint of each of those steps. Reducing number of steps is achieved by using higher-order solvers [39, 40, 69], distilling steps to a reduced set using progressive step distillation [34, 41, 53], straightening the ODE trajectories using Rectified Flows [35, 37, 76], mapping noise directly to data with consistency models [38, 57, 58], and using adversarial training [54, 55, 63, 72]. To decrease computational cost of each step, research has been done in weight quantization [19, 46, 56] and pruning [8, 34] as well as architectural optimization of the denoiser [9, 17, 31, 49, 66]. In this work, following Zhang et al. [72] we reduce number of steps to one using adversarial training and optimize the UNet denoiser using multiple novel techniques.

On-device generation. On-device generation has attracted interest due to its ability to address privacy concerns associated with cloud-based approaches. There have been advancements in running text-to-image generation on mobile devices and NPUs [5, 7, 8, 22, 34, 73]. Although there has been progress in the video domain with fast zero-shot video editing models [27, 28, 64, 71], by now the only reported deployment of a video generative model to a device is concurrent work of SnapGen-V [65]. However, unlike SnapGen-V, we finetune a publicly released image-to-video SVD [2] checkpoint rather than train from scratch, which makes our method suitable for low-compute practitioners. It is noteworthy that, while most state-of-the-art generative models pivoted to a DiT-based transformer architecture [48], deploying DiT to device is more challenging due to its quadratic memory need. Therefore, all the mentioned mobile-targeting approaches stick with UNet-based architecture.

3. Mobile Video Diffusion

In this section, we propose a series of optimizations to obtain a fast and lightweight version of an off-the-shelf image-to-video diffusion model [2], suitable for on-device deployment.

Model	NFE	FVD ↓	TFLOPs ↓	Latency (ms) ↓	
				GPU	Phone
Resolution 1024 × 576					
SVD	50	149	45.43	376	OOM
AnimateLCM*	8	281	45.43	376	OOM
UFOGen*	1	1917	45.43	376	OOM
LADD*	1	1894	45.43	376	OOM
SF-V*	1	181	45.43	376	OOM
MobileVD-HD (ours)	1	184	23.63	227	OOM
Resolution 512 × 256					
SVD (original)	50	476	8.60	82	OOM
SVD (finetuned)	50	196	8.60	82	OOM
SF-V (our implement.)	1	168	8.24	76	3630
MobileVD (ours)	1	171	4.34	45	1780

Table 1. **Comparison with recent models.** FLOPs and latency are provided for a single function evaluation (NFE) with batch size of 1. For rows marked with asterisk* FVD measurements were taken from Zhang et al. [72], while performance metrics are based on our measurements for UNet used by SVD. For consistency with these results, FVD for SVD and our MobileVD model was measured on UCF-101 dataset at 7 frames per second.

3.1. Preliminaries

Stable Video Diffusion. We adopt Stable Video Diffusion (SVD) [2] as the base model for optimization. The SVD released checkpoint[1] is an image-to-video model that by default generates 14 frames at the resolution 1024×576 with 25 sampling steps. To generate a video from an image, the input image is first mapped into a latent code of resolution 128×72 using a Variational Auto-Encoder (VAE). Then it is duplicated 14 times and concatenated with a noise latent of spatiotemporal resolution $14 \times 128 \times 72$. The combined latent is then denoised by a conditional UNet through an iterative process. Additionally, the input image is encoded with a CLIP image embedding for use in cross-attention layers [50]. The UNet denoiser consists of four downsampling blocks, one middle block and four upsampling blocks. To handle the dynamics of video sequences, the model employs temporal blocks after spatial blocks. Notably, up- and downsampling is conducted across spatial axes only, and temporal resolution of the latent is kept constant to 14 throughout the UNet. The UNet denoiser as-is is too resource-intensive for on-device use, requiring 45.43 TFLOPs and 376 ms per denoising step on a high-end A100 GPU. Using FLOPs and latency as proxy metrics, we propose a series of optimizations to make the model suitable for on-device deployment.

Baseline model. We observe that the original high-resolution SVD can not be compiled for on-device usage. The main reason is that Android NPU has extremely low memory (only 8 MB) and restricted set of instructions and

[1] https://huggingface.co/stabilityai/stable-video-diffusion-img2vid/tree/9cf024d

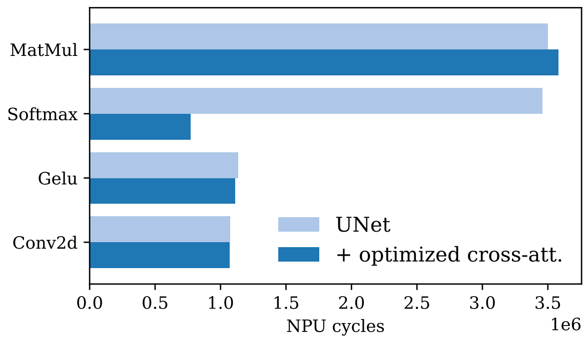


Figure 1. **Effect of optimized cross-attention for a mobile device.** We show the number of cycles of the top-4 operations on mobile hardware for an input resolution of 128×128. Note that removing the no-op similarity map computation in cross-attention layers reduces cycles on softmax operations by roughly 80%.

operations (no 5D operations, sub-optimal batch inference, slow scalar processing required for activation functions and softmax). To satisfy the memory constraints, we decrease the spatial resolution of the model to 512×256, resulting in latent size of $14 \times 64 \times 32$. This decreased resolution leads to 4.5x smaller feature maps and reduced peak memory. While the released SVD checkpoint supports multiple resolutions, we found out that the original model demonstrates deteriorated quality for our target spatial resolution as reported in Tab. 1. Therefore, we finetuned the diffusion denoiser at our target spatial size. With this optimization, computational cost and GPU latency is reduced to 8.60 TFLOPS and 82 ms respectively. We consider this low resolution, finetuned model as the baseline in our work.

Adversarial finetuning. In addition to the high cost of the UNet, the iterative sampling process in video diffusion models further slows them down. For example, with the conventional 25-step sampling it takes 11 seconds to generate a video on a high-end A100 GPU. Also, due to the usage of classifier-free guidance, each step results in two UNet forward passes on a device leading to the total number of UNet evaluations (NFE) of 50 [20]. To reduce the cost, we follow SF-V framework [72] and use adversarial finetuning, enabling our models to generate videos in a single forward pass. Namely, we initialize the discriminator feature extractor with an encoder part of the denoising UNet and do not train it. After each block of this backbone, the extracted feature maps are passed to the two projection discriminator heads, one spatial and one temporal [43]. The heads also use the frame index and the CLIP embedding of the conditional image as input. At each training step, we apply the pseudo-Huber (Charbonnier) [6] and non-saturating adversarial loss [15] to the generator output to update its weights. To regularize the discriminator, the R_1 penalty is used [42]. For further details please refer to the original SF-V work.

3.2. Mobile-friendly UNet

Our first modification to SVD architecture regards the feature resolution that affects both GPU and mobile latency. Then we highlight some lossless optimizations that have a large effect on mobile latency. These are the first optimizations that allow us to run the UNet on device.

Temporal multiscaling. To further reduce computational burden, one might lower the input resolution more heavily. However, this significantly degrades visual quality. Instead, we can additionally downscale the feature maps by a factor of 2 along either spatial or temporal axis after the first downsampling block of the UNet. To maintain the same output shape, this is accompanied by the corresponding nearest-neighbor upscaling operation before the last upsampling block. We refer to these two additions as multiscaling. In terms of computational cost, spatial multiscaling results in a 51% reduction in FLOPs and 33% in GPU latency, while temporal multiscaling reduces FLOPs and GPU latency by 34% and 22%, respectively. For our final deployed model, we use temporal multiscaling as it offers a better trade-off between quality and efficiency, as reported in Sec. 4.3.1.

Optimizing cross-attention. In SVD, each cross-attention layer integrates information from the conditioning image into the generation process. The attention scores are computed similarly to self-attention layers, $\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(QK^T/\sqrt{d}\right)V$, but the key and value pair (K, V) comes from the context tokens. However, the context in the cross-attention blocks always consists of a single token, namely, the CLIP embedding vector of the conditional image. Consequently, each query token attends to only a single key token. Therefore, computation of a similarity map QK^T and softmax becomes a no-op, and query and key projection matrices can be removed without any difference in results. While this loss-less optimization reduces GPU latency only by 7%, we found that it significantly impacts on-device behavior. In detail, at target resolution of 512×256, the model runs out of memory (OOM) on the device without the described modification of cross-attention. And at a smaller resolution of 128×128 this optimization reduces mobile latency by 32%. The gain is attributed to the time-consuming nature of softmax operation on device, as shown in Fig. 1.

3.3. Channel funnels

Channel size, which refers to the width of neural network layers, is crucial for scaling models. Increasing channel size generally enhances model capacity but also increases the number of parameters and computational cost. Research has focused on compressing the number of channels by either discarding less informative channels through channel pruning [12, 44] or representing weight matrices as low-rank products of matrices using truncated singular decomposition [70]. However, these could have sub-optimal tradeoffs

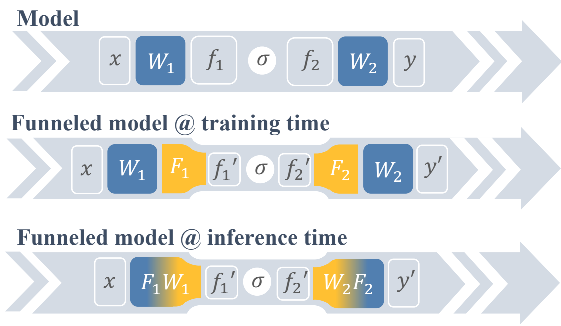
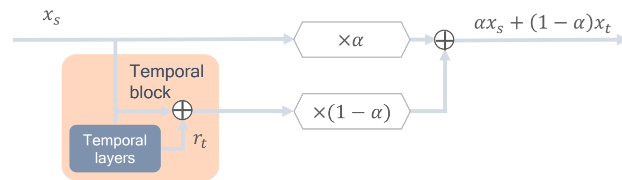


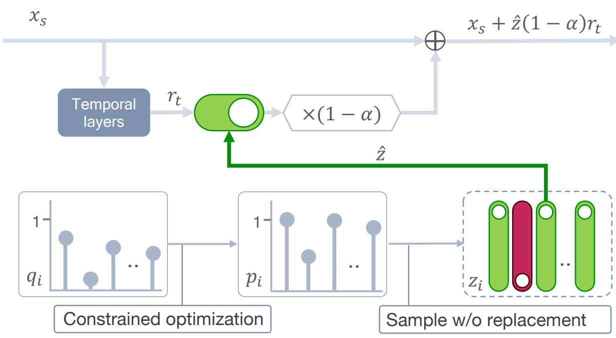
Figure 2. **Channel funnels.** We show an example of channel funnels applied to a couple of layers within the model. At training time, funnels serve as adaptors reducing model width. At inference, they are merged with corresponding weight matrices without loss of quality.

in quality and efficiency when deployed on device. While low-rank decomposition is relatively straightforward to implement, it only reduces the number of parameters and computational complexity if the rank is reduced by more than half for feed forward layers. Additionally, not all layers of neural network types benefit equally from low-rank factorization. Moreover, this method does not reduce the size of output feature maps, which can cause significant memory overhead on mobile devices. In this part, we propose *channel funnels*, a straightforward method to reduce the number of channels at inference time, with negligible quality degradation. Intuitively, a channel funnel is placed between two affine layers, reducing the intermediate channel dimensionality to save computation. Consider two consecutive linear layers $y = W_2\sigma(W_1x)$ and the non-linear function σ in between, where $W_1 \in \mathbb{R}^{c_{\text{inner}} \times c_{\text{in}}}$, $W_2 \in \mathbb{R}^{c_{\text{out}} \times c_{\text{inner}}}$. We introduce two funnel matrices, $F_1 \in \mathbb{R}^{c' \times c_{\text{inner}}}$ and $F_2 \in \mathbb{R}^{c_{\text{inner}} \times c'}$, where $c' < c_{\text{inner}}$, and rewrite our network as $y' = W_2F_2\sigma(F_1W_1x)$. The F-weights, having fewer channels, can be merged during inference with their associated W-weights, resulting a weight matrix with smaller inner dimension c', see Fig. 2. We refer to the reducing factor of the inner rank of the layers, *i.e.* c'/c_{inner}, as the *fun-factor*.

Initialization. We propose to use *coupled singular initialization (CSI)* for funnel matrices F_1 and F_2 that improves the model results, as demonstrated below. In this method we make a simplifying assumption by ignoring the non-linearity, and consider the effective weight matrix $W_2F_2F_1W_1$ which in practice has rank of c'. For that reason, we aim to use such an initialization which mimics the best c'-rank approximation of the original effective matrix. As Eckart-Young-Mirsky theorem implies, this can be achieved by means of truncated singular decomposition [10]. Let $W_2W_1 = U\Sigma V^T$ be the singular vector decomposition, and $U_{c'}\Sigma_{c'}V_{c'}^T$ to be its truncated c'-rank version. Then it suffices to set $F_2 = W_2^{\dagger}U_{c'}\Sigma_{c'}^{1/2}$ and $F_1 = \Sigma_{c'}^{1/2}V_{c'}^TW_1^{\dagger}$ to obtain $W_2F_2F_1W_1 \approx U_{c'}\Sigma_{c'}V_{c'}^T$, where $\dagger$ means the Moore-Penrose pseudoinverse.



(a) Temporal blocks in the original architecture of SVD.



(b) A zero-one gate multiplier is sampled to each temporal block during training.

Figure 3. **Learned pruning of temporal blocks.** (a) Each temporal block in the base SVD model is implemented as a residual block w.r.t. its input x_s. The output of temporal layers r_t is summed with the input x_s, and after that once again averaged with x_s with learnable weight α. By reordering the terms, we derive the effective update rule $\alpha x_s + (1-\alpha)\, x_t = x_s + (1-\alpha)\, r_t$. (b) During training, we introduce a scalar gate $\hat{z} \in \{0, 1\}$ to the residual update rule of each block. We learn importance values $\{q_i\}_i$ of temporal blocks which are transformed to inclusion probabilities $\{p_i\}_i$ at each training step. Zero-one gate multipliers are sampled according to those probabilities. To enable end-to-end training, we use straight-through estimator trick. At inference, only n blocks with highest importance values are used.

Training. We apply channel funnels in attention layers where query and key embedding W_q and W_k are used to compute the similarity map of $XW_q\,(XW_k)^T$. With funnel matrices F_q and F_k of size $c_{\text{inner}} \times c'$, we modify the aforementioned bilinear map as $XW_qF_q\,(XW_kF_k)^T = XW_qF_qF_k^TW_k^TX^T$. Similarly, funnels are applied to the pair of value and output projection matrices of a self-attention layer. In our ablations we also show the impact of channel funnel on convolutions in residual blocks. Unless specified otherwise, we use fun-factor of 50%.

3.4. Temporal block pruning

Motivation. The original UNet in the SVD model does not contain 3D convolutions or full spatiotemporal attention layers. Instead, to model motion dynamics, SVD incorporates temporal blocks after each spatial block. The output of such group is a linear combination of the spatial and temporal block outputs, x_s and x_t respectively, $\alpha x_s + (1-\alpha)\, x_t$,

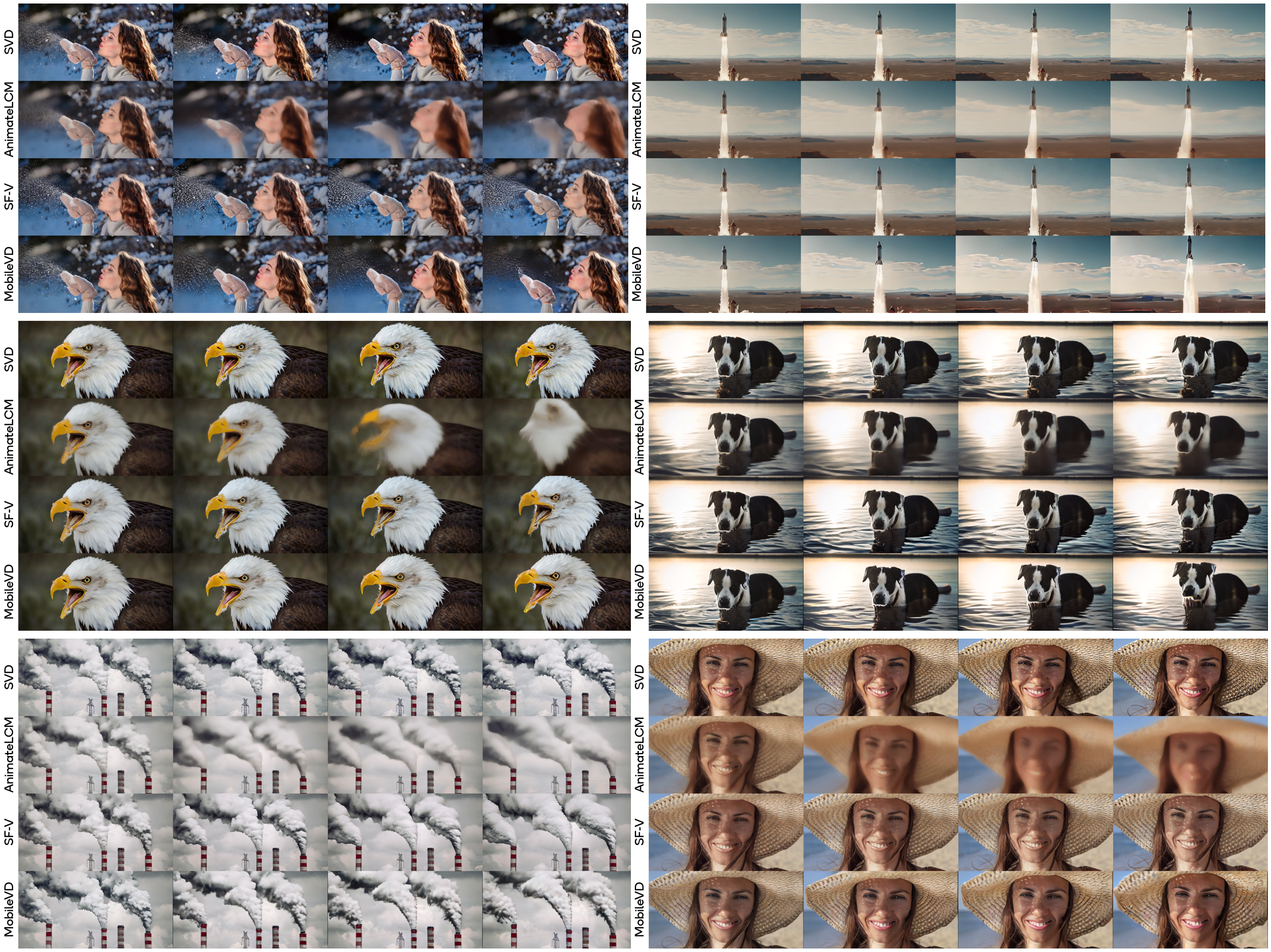


Figure 4. **Comparison with recent models.** We provide the 1st, 6th, 10th and 14th frames from the videos generated with different models[2]. For AnimateLCM [62] and SF-V [72] we downsampled the released high-resolution videos from Zhang et al. [72]. For SVD [2] and our MobileVD model, videos were generated at their native resolution, 1024×576 and 512×256 respectively.

where α is a weight scalar that emphasizes spatial features when higher and temporal features when lower, see Fig. 3a. While this approach leverages image model priors when extending the model to videos, it adds computational cost. Moreover, not all of these blocks are equally important for maintaining quality. Here, we propose a learnable pruning technique to remove less important temporal blocks while minimizing quality degradation. To this end, for each temporal block we define an *importance* value $q_i,\ 0 \leq q_i \leq 1$ where $i = 1, \ldots, N$ and N is number of temporal blocks. The values $\{q_i\}_i$ are trained to identify the blocks that are the most crucial for model performance. At inference time, only n blocks with the highest importance q_i are kept where n is the budget chosen in advance. In our experiments we found it possible to remove as many as 70% of all temporal blocks which leads to 14% reduction in FLOPS as compared to the model with optimizations from Sec. 3.2 applied.

Training. At each training iteration, we sample randomly n blocks which participate in the computational graph. To this end, we define the indicator variable $z_i \in \{0, 1\}$ where $z_i = 1$ if i-th block is sampled for participation, and $z_i = 0$ otherwise. The sum of all participants should equal to the budget value *i.e.* $\sum_i z_i = n$. Note that $\mathbb{E}\left[\sum_i z_i\right] = \sum_i \mathbb{E}\left[z_i\right] = \sum_i p_i = n$, where p_i is the *inclusion probability* of the i-th block. We relate i-th importance value q_i to the inclusion probability of p_i using the constrained optimization problem

$$\min_{c, \{p_i\}_i} \sum\nolimits_i (p_i - c q_i)^2, \quad \text{s.t.} \quad \sum\nolimits_i p_i = n,\ 0 \leq p_i \leq 1,\ c \geq 0.$$

We obtain a closed-form solution of the above optimization with Lagrange multipliers which is differentiable w.r.t. q_i. In simple words, we find the proper set of inclusion probabilities $p_i \approx cq_i$, with importance values q_i and a proportionality coefficient c. After obtaining $\{p_i\}_i$ at each training iteration, we sample n blocks without replacement using Brewer's sampling [4, 60]. As such sampling is non-differentiable, we employ straight-through estimators (STE) [1] to enable end-to-end training. Namely, we define gate $\hat{z}_i$ as STE of the probability p_i, *i.e.* $\hat{z}_i = p_i + \texttt{stop\_gradient}(z_i - p_i)$. The output of a temporal block is multiplied by this gate as Fig. 3b shows. For practical aspects of training, please refer to SM.

4. Experiments

In this section, we describe our experimental setup, followed by a qualitative and quantitative evaluation of our model. Finally, we present ablations to justify our choices for the final model deployed on the device.

4.1. Implementation details

Dataset. For our experiments, we use a collected video dataset. We follow the data curation pipeline from Open-Sora [74, V.1.1], selecting videos with motion scores between 4 and 40, and aesthetic scores of at least 4.2. This results in a curated dataset of approximately 31k video files for finetuning.

Micro-conditioning. UNet used by SVD, has two conditioning parameters called *FPS* and *Motion bucket id*. To obtain videos with different FPS, we chose each k-th frame from the video with randomly sampled k, $1 \le k \le 4$, and adjusted the native FPS value of the video accordingly. The notion of motion bucket has not been properly documented at time of the model release. While it is connected with the speed of motion in the video, as described in the original SVD paper, the motion estimator has not been open-sourced. For that reason, we implemented our own definition of the motion bucket using a simple heuristic. For the sampled chunk of 14 frames, we converted them to gray color, spatially downsampled to the resolution of $14 \times 128 \times 64$, and reshaped to the matrix of size $14 \times (128 \cdot 64)$. After that, we computed the singular values of that matrix. Note that for a completely static video this matrix has a rank of 1, and therefore the only non-zero singular value. And the less similar frames are, the more singular components are needed to faithfully reconstruct the full video. Based on that observation, we re-defined the motion bucket as the area under the normalized cumulative sum of singular values.

Training. For training, we begin with the original SVD weights, apply all the optimizations (excluding temporal block pruning), and train the resulting UNet with a standard diffusion loss for 100k iterations on 4 A100 GPUs with a total batch size of 8. This UNet serves as the initialization for adversarial finetuning, where a good initialization is crucial for fast convergence. We found that training for 5k iterations on 2 GPUs suffices for the second stage. We implement temporal block pruning in the second stage as we observed that excessive pruning in the first stage hinders model performance. In this case, we train the second stage for 10k steps. Check SM for more details.

Metrics. We used DeepSpeed library [51, v0.14.2] to measure the number of FLOPs. For GPU latency, NVIDIA® A100 SXM™ 80GB GPU was used. To measure GPU latency, UNet model was compiled using the PyTorch [47, v2.0.1] compiler with default settings. Phone latencies are measured on a Xiaomi-14 Pro that uses Snapdragon® 8 Gen. 3 Mobile Platform with a Qualcomm® Hexagon™ processor. All performance metrics were measured for a single UNet evaluation with batch size of 1. For video quality metric, we follow existing works [2, 72] by using Fréchet Video Distance (FVD) [61] with I3D feature extractor [29]. We use the first frame of UCF-101 dataset [59] as the conditioning image, generating 14-frame clips at the model's native resolution. Unless stated otherwise, we set the FPS, an SVD micro-condition, to 25, matching the UCF-101 frame rate [2]. For motion bucket, for our models we used the median value at the specified frame rate across UCF-101 data. For SVD the default bucket of 127 was used. In addition, we evaluated our model using the recent VBench-I2V toolkit [23, 24]. For that purpose, we generated videos with different speeds to demonstrate the trade-off between motion degree and frame-wise plausibility.

4.2. Results

MobileVD. In Tab. 2 we compare MobileVD to the baseline model. As the results indicate, each optimization reduces speed of inference on a mobile phone. Optimized cross-attention unlocks on-device execution with a latency of 3.6 seconds. More specifically, temporal downsampling layers in UNet make inference 29% faster. Additionally, temporal blocks pruning reduces phone latency by 13%, and channel funneling further decreases it by 9%. Empirically, we found that a difference of up to 20 FVD units does not significantly affect visual quality and typically falls within the standard deviation when using different random seeds. Based on that, we see that our optimizations reduce on-device inference by 50% while having minimal impact on FVD. As Tab. 3 shows, by changing the motion bucket, we can generate videos with the same dynamic degree as original SVD model.

SOTA comparison. In Tab. 1 we compare to the recent works that similarly aim for accelerating SVD image-to-video model, namely, AnimateLCM [62], LADD [54], SF-

[2]Conditioning images are under MIT license © 2024 Fu-Yun Wang. https://github.com/G-U-N/AnimateLCM/blob/9a5a314/LICENSE.txt

Model	NFE	FVD ↓		TFLOPs ↓	Latency (ms) ↓	
		25 FPS	7 FPS		GPU	Phone
SVD (finetuned for 512×256)	50	194	196	8.60	82	OOM
+ optimized cross-attention	50	194	196	8.24	76	3630
+ adversarial finetuning	1	133	168	8.24	76	3630
+ temporal multiscaling	1	139	156	5.42	59	2590
+ temporal block pruning	1	127	150	4.64	47	2100
+ channel funneling	1	149	171	4.34	45	1780

Table 2. **Effect of our optimizations.** We successfully deployed the image-to-video model to a mobile device without significantly sacrificing the visual quality. FLOPs and latency are provided for a single function evaluation with batch size of 1. We call the model in the bottom row Mobile Video Diffusion, or MobileVD.

Model	Motion bucket	I2V Subject	I2V Background	Aesth. Quality	Imag. Quality	Dynamic Degree
SVD (original)	127	93.48	94.74	53.59	63.49	95.69
SVD (finetuned)	20	95.72	96.36	54.91	65.17	16.26
SVD (finetuned)	40	95.24	96.04	54.62	65.16	65.04
MobileVD	20	93.68	94.30	53.69	67.16	68.21
MobileVD	40	92.98	93.93	53.19	67.46	95.77

Table 3. **VBench-I2V evaluation.** By varying the motion bucket, we can generate videos with different dynamic degree. Faster videos generally have slightly worse visual quality, as framewise metrics show.

Spatial multiscaling	Temporal multiscaling	FVD ↓	TFLOPs ↓	Latency (ms) ↓	
				GPU	Phone
×	×	133	8.24	76	3630
×	✓	138	5.42	59	2590
✓	×	145	4.35	51	2280
✓	✓	163	3.39	48	—

Table 4. **Effect of additional multiscaling layers in UNet.** We observe that both temporal and spatial multiscaling has good impact on mobile latency without compromising much on FVD, while combining the two increases FVD by a noticeable amount.

V [72], and UFOGen [67]. As all these methods operate on 1024×576 resolution, for a fair comparison, we also trained a high-resolution version of our model called MobileVD-HD. It uses the whole set of our proposed optimizations, and training details are provided in Supplementary. For lower resolution, we observe that MobileVD leads to a comparable FVD to SVD and SF-V but requires significantly less computation.

Qualitative results. Following previous works [62, 72], we show qualitative results with a commonly used set of conditioning images. Sampled frames from the generated videos are presented in Fig. 4. For this visualization, we generated videos at 7 FPS and with spatial resolution of 512×256 using our MobileVD. Please refer to the Supplementary for the full videos. We observe that in general our method produces videos with sharp details and consistent motion.

4.3. Ablations

In this section, we evaluate our design choices through ablation experiments. Unless otherwise specified, we use the SVD checkpoint with low-resolution input, optimized cross-attention, and adversarial finetuning as the reference model, *cf*. Tab. 2.

4.3.1. Resolution impact in UNet

In Tab. 4 we compare different latent multiscaling optimizations proposed in Sec. 3.2. Specifically, we investigate the impact of inserting spatial or temporal multiscaling layers after the first UNet block in terms of FLOPs, latency, and FVD. Spatial multiscaling offers better FLOPs and latency than temporal multiscaling and it increases FVD by 12 units compared to 5 for temporal downsampling. While we typically do not see video degradation with this increase in FVD, we do see clear degradation in video quality when using spatial instead of temporal multiscaling. We hypothesize that this is because the model already enjoys multiple stages of spatial downsampling, while temporal downsampling was originally absent. Based on these results, we have opted for temporal downsampling for our mobile-deployed model. We hold similar conclusions for combining the two multiscaling approaches with spatiotemporal multiscaling.

4.3.2. Funnel finetuning

Fun-factor and funnel initialization. Reducing the width of affine layers in the model is a form of lossy compression, and overly aggressive fun-factor values will hurt the model performance. In Tab. 5, we observe the impact of the fun-factor. Reducing the fun-factor to 0.25 results in a performance loss of 22 FVD units compared to fun-factor of 1 (*i.e.*, no compression). To avoid performance degradation from stacking multiple optimizations described in Sec. 3, we set the fun-factor to 0.5 for the deployed model. Additionally, the results highlight that the proposed coupled singular initialization (CSI) is essential for optimal funnel behavior, whereas the standard He initialization [18] is suboptimal.

Funnel merging and low-rank layers. We compare channel funnels with multiple baselines in terms of FLOPs, on-device latency and FVD. All baselines are applied on the same attention layers, unless specified otherwise. The first baseline uses channel funnels but merges funnel and weight matrices during training, hence mimicking behavior at inference time as shown in Fig. 2. We report in Tab. 6 that keeping

Initialization	Fun-factor	FVD ↓
Coupled singular init. (CSI)	0.25	155
Coupled singular init. (CSI)	0.50	132
Coupled singular init. (CSI)	0.75	145
Coupled singular init. (CSI)	1.00	133
He init. [18]	0.50	332

Table 5. **Effect of funnel initialization and fun-factor.** Initialization funnels with CSI is crucial to getting good FVD as He initialization [18] obtains roughly 200 FVD units more. Additionally, we see that reducing the fun-factor beyond 0.5 starts to affect the performance.

Width reduction method	r	FVD ↓	TFLOPs ↓	Latency (ms) ↓
Original UNet	-	133	8.6	3630
+ Funnels	0.5	132	8.0	2870
+ Funnels (merge before finetune)	0.5	138	8.0	2870
+ Funnels (convolutions)	0.5	139	7.2	3400
+ Truncated singular decomposition	0.5	142	8.6	3482
+ Truncated singular decomposition	0.25	130	8.0	3345

Table 6. **Comparison of model width reduction methods.** We compare the proposed channel funneling (in grey) with finetuned low-rank approximation of individual attention layers with truncated singular decomposition. We additionally compare to Funnels applied to convolutions instead of attention. The reduction rate (referred to as fun-factor in case of funnels) is highlighted with r.

funnel and weight matrices separate at training performs equally well. The second baseline is applying funnels to convolutions in ResNet blocks instead of attention layers. While we obtain favourable FVD and even greater gain in TFLOPs (7.2 vs 8.6), it does not translate to the latency reduction we see with funnels on attention (3.40 vs 2.87 seconds). We hypothesize that the attention layers play a greater role in reducing latency on device than convolutions for this model. The last baselines employ the standard technique of truncated singular decomposition of individual layers [70]. This decomposition breaks down a weight matrix of a linear layer $W \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}}}$ into a low-rank product of two matrices $W_1 \in \mathbb{R}^{rc \times c_{\text{in}}}$ and $W_2 \in \mathbb{R}^{c_{\text{out}} \times rc}$ where r is the rank reduction factor, $r < 1$, and $c = \min(c_{\text{in}}, c_{\text{out}})$. Note that reduction in the number of parameters and FLOPs is achieved only if $r < 0.5$, while the size of the feature map after these two matrices remain intact in this approach. While truncated decomposition after finetuning performs well in terms of FVD for both $r = 0.25$ and $r = 0.5$, it is slower on device compared to channel funneling (3.35 and 3.48 vs. 2.87 seconds respectively). This difference is attributed to memory transfer overhead from introducing additional layers as well as not decreasing the original feature size, emphasizing the benefit of funnels.

4.3.3. Temporal blocks pruning

As mentioned in Sec. 3.4, in our experiments we found it possible to reduce up to 70% of all temporal blocks in the UNet. Notably, even with such a high pruning rate the quality is comparable to the original model: it achieves FVD of 127 and requires 14% less FLOPs, while the original model has FVD of 139. However, pruning even further seems far from straightforward. FVD degrades to the values above 200 with a pruning rate of 80%, and visual quality drops drastically.

Blocks pruned (%)	FVD ↓	TFLOPs ↓	Latency GPU (ms) ↓
Our method			
90	201	4.06	42
80	245	4.35	44
70	127	4.64	47
L_1 regularization			
70	207	4.67	48
53	165	5.17	52

Table 7. **Impact of temporal blocks pruning.** Our pruning outperforms the L_1- regularization which does not have explicit control over the number of removed blocks. We use the checkpoint, optimized up to the temporal block pruning stage, as the starting point.

Additionally, we compare our method with another pruning baseline. As described in Sec. 3.4, the output of a spatiotemporal block in SVD is a linear combination of the spatial and temporal blocks x_s and x_t, respectively, $\alpha x_s + (1-\alpha)x_t$, where α is a weight scalar. The baseline aims to minimize the influence of temporal blocks, by adding a loss term during finetuning: $\mathcal{L} = \lambda \sum_i (1 - \alpha_i)$. After training, blocks with the highest weight scalar α_i are pruned. However, this method lacks explicit control over the desired pruning rate, as only weight hyperparameter λ can be adjusted. While effective for small pruning rates, this approach did not allow us to remove as many blocks as our method, achieving only a 7% reduction in FLOPs with acceptable FVD level, see Tab. 7.

5. Conclusion

This paper introduced the first mobile-optimized image-to-video diffusion model, addressing the high computational demands that have limited their use on mobile devices. By optimizing the spatio-temporal UNet from Stable Video Diffusion and employing novel pruning techniques, we significantly reduced memory and computational requirements. Our model, MobileVD, achieves substantial efficiency improvements with minimal quality loss, making video diffusion technology feasible for mobile platforms.

Limitations. Despite the impressive acceleration, the output is currently limited to 14 frames at a resolution of 256×512 pixels. The next step involves leveraging more efficient autoencoders to achieve higher spatial and temporal compression rates, enabling the generation of larger and longer videos at the same diffusion latent generation cost. Also, our model inherits the lack of textual control from the released SVD version, which had been finetuned by its authors from a text-to-video to an image-to-video model. We plain to solve this in future either by reverse finetuning, or by switching to another pretrained backbone.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. 6

[2] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets, 2023. 1, 2, 5, 6

[3] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023. 1, 2

[4] Kenneth E. W. Brewer. A Simple Procedure for Sampling πpswor[1]. *Australian Journal of Statistics*, 17(3):166–172, 1975. 6

[5] Thibault Castells, Hyoung-Kyu Song, Tairen Piao, Shinkook Choi, Bo-Kyeong Kim, Hanyoung Yim, Changgwun Lee, Jae Gon Kim, and Tae-Ho Kim. Edgefusion: On-device text-to-image generation, 2024. 2

[6] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging . In *ICIP*, 1994. 3

[7] Yu-Hui Chen, Raman Sarokin, Juhyun Lee, Jiuqiang Tang, Chuo-Ling Chang, Andrei Kulik, and Matthias Grundmann. Speed is all you need: On-device acceleration of large diffusion models via gpu-aware optimizations. In *CVPR*, 2023. 2

[8] Jiwoong Choi, Minkyu Kim, Daehyun Ahn, Taesu Kim, Yulhwa Kim, Dongwon Jo, Hyesung Jeon, Jae-Joon Kim, and Hyungjun Kim. Squeezing large-scale diffusion models for mobile, 2023. 2

[9] Tim Dockhorn, Robin Rombach, Andreas Blatmann, and Yaoliang Yu. Distilling the knowledge in diffusion models. In *CVPRW*, 2023. 2

[10] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. 4

[11] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. In *ECCV*, 2023. 1

[12] Gongfan Fang, Xinyin Tan Ma, Michael Bi Mi, and Xinchao Wang. Isomorphic Pruning for Vision Models. In *ECCV*, 2024. 3

[13] Genmo. Mochi-1, 2024. 2

[14] Rohit Girdhar, Mannat Singh, Andrew Brown, Quentin Duval, Samaneh Azadi, Sai Saketh Rambhatla, Akbar Shah, Xi Yin, Devi Parikh, and Ishan Misra. Factorizing text-to-video generation by explicit image conditioning. In *ECCV*, 2024. 2

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 3

[16] Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. AnimateDiff: Animate Your Personalized Text-to-Image Diffusion Models without Specific Tuning. In *ICLR*, 2023. 1

[17] Amirhossein Habibian, Amir Ghodrati, Noor Fathima, Guillaume Sautiere, Risheek Garrepalli, Fatih Porikli, and Jens Petersen. Clockwork diffusion: Efficient generation with model-step distillation. In *CVPR*, 2024. 2

[18] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 7, 8

[19] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptqd: Accurate post-training quantization for diffusion models. In *NeurIPS*, 2024. 2

[20] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 3

[21] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models, 2022. 2

[22] Dongting Hu, Jierun Chen, Xijie Huang, Huseyin Coskun, Arpit Sahni, Aarush Gupta, Anujraaj Goyal, Dishani Lahiri, Rajesh Singh, Yerlan Idelbayev, Junli Cao, Yanyu Li, Kwang-Ting Cheng, S.-H. Chan, Mingming Gong, Sergey Tulyakov, Anil Kag, Yanwu Xu, and Jian Ren. SnapGen: Taming High-Resolution Text-to-Image Models for Mobile Devices with Efficient Architectures and Training, 2024. 2

[23] 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. 6

[24] Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, Yaohui Wang, Xinyuan Chen, Ying-Cong Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench++: Comprehensive and Versatile Benchmark Suite for Video Generative Models, 2024. 6

[25] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong MU, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In *ICLR*, 2025. 2

[26] Kumara Kahatapitiya, Adil Karjauv, Davide Abati, Fatih Porikli, Yuki M Asano, and Amirhossein Habibian. Object-centric diffusion for efficient video editing. In *ECCV*, 2024. 1

[27] 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. In *CVPR*, 2024. 2

[28] Adil Karjauv, Noor Fathima, Ioannis Lelekas, Fatih Porikli, Amir Ghodrati, and Amirhossein Habibian. MoViE: Mobile diffusion for video editing, 2024. 2

[29] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman,

and Andrew Zisserman. The kinetics human action video dataset, 2017. 6

[30] 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. In *CVPR*, 2023. 1

[31] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. Bk-sdm: A lightweight, fast, and cheap version of stable diffusion. In *ECCV*, 2024. 2

[32] Kuaishou. Kling, 2024. 2

[33] PKU-Yuan Lab and Tuzhan AI etc. Open-sora-plan, 2024. 1

[34] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. In *NeurIPS*, 2023. 2

[35] Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport, 2022. 2

[36] Shaoteng Liu, Yuechen Zhang, Wenbo Li, Zhe Lin, and Jiaya Jia. Video-p2p: Video editing with cross-attention control. In *CVPR*, 2024. 1

[37] Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023. 2

[38] Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In *ICLR*, 2025. 2

[39] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *NeurIPS*, 2022. 2

[40] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models, 2022. 2

[41] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, 2023. 2

[42] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which Training Methods for GANs do actually Converge? In *ICML*, 2018. 3

[43] Takeru Miyato and Masanori Koyama. cGANs with Projection Discriminator. In *ICLR*, 2018. 3

[44] Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Bhuminand Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact Language Models via Pruning and Knowledge Distillation. In *NeurIPS*, 2024. 3

[45] OpenAI. Sora, 2024. 2

[46] Nilesh Prasad Pandey, Marios Fournarakis, Chirag Patel, and Markus Nagel. Softmax bias correction for quantized generative models. In *CVPR*, 2023. 2

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. 6

[48] William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. In *ICCV*, 2023. 2

[49] Elia Peruzzo, Adil Karjauv, Nicu Sebe, Amir Ghodrati, and Amir Habibian. ADAPTOR: Adaptive Token Reduction for Video Diffusion Transformers. In *CVPRW*, 2025. 2

[50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, 2021. 2

[51] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *SIGKDD*, 2020. 6

[52] Runway. Introducing gen-3 alpha, 2024. 2

[53] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022. 2

[54] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. In *SIGGRAPH Asia*, 2024. 1, 2, 6

[55] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *ECCV*, 2024. 2

[56] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *CVPR*, 2023. 2

[57] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *ICLR*, 2024. 2

[58] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, 2023. 1, 2

[59] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild, 2012. 6

[60] Yves Tillé. *Sampling algorithms*. Springer, New York, 2006. 6

[61] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. FVD: A new metric for video generation, 2019. 6

[62] Fu-Yun Wang, Zhaoyang Huang, Weikang Bian, Xiaoyu Shi, Keqiang Sun, Guanglu Song, Yu Liu, and Hongsheng Li. AnimateLCM: Computation-Efficient Personalized Style Video Generation without Personalized Video Data. In *SIGGRAPH Asia 2024 Technical Communications*, 2024. 1, 5, 6, 7

[63] Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-gan: Training gans with diffusion. In *ICLR*, 2023. 2

[64] Bichen Wu, Ching-Yao Chuang, Xiaoyan Wang, Yichen Jia, Kapil Krishnakumar, Tong Xiao, Feng Liang, Licheng Yu, and Peter Vajda. Fairy: Fast parallelized instruction-guided video-to-video synthesis. In *CVPR*, 2024. 2

[65] Yushu Wu, Zhixing Zhang, Yanyu Li, Yanwu Xu, Anil Kag, Yang Sui, Huseyin Coskun, Ke Ma, Aleksei Lebedev, Ju Hu, Dimitris Metaxas, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. SnapGen-V: Generating a Five-Second Video within Five Seconds on a Mobile Device. In *CVPR*, 2025. 2

[66] Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. SANA: Efficient high-resolution text-to-image synthesis with linear diffusion transformers. In *ICLR*, 2025. 2

[67] Yanwu Xu, Yang Zhao, Zhisheng Xiao, and Tingbo Hou. UFOGen: You Forward Once Large Scale Text-to-Image Generation via Diffusion GANs. In *CVPR*, 2024. 7

[68] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan.Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. CogVideoX: Text-to-Video Diffusion Models with An Expert Transformer. In *ICLR*, 2025. 1

[69] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *ICLR*, 2023. 2

[70] X. Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *PAML*, 2015. 3, 8

[71] Youyuan Zhang, Xuan Ju, and James J Clark. FastVideoEdit: Leveraging consistency models for efficient text-to-video editing. In *WACV*, 2025. 2

[72] Zhixing Zhang, Yanyu Li, Yushu Wu, Yanwu Xu, Anil Kag, Ivan Skorokhodov, Willi Menapace, Aliaksandr Siarohin, Junli Cao, Dimitris Metaxas, Sergey Tulyakov, and Jian Ren. SF-V: Single Forward Video Generation Model. In *NeurIPS*, 2024. 1, 2, 3, 5, 6, 7

[73] Yang Zhao, Yanwu Xu, Zhisheng Xiao, and Tingbo Hou. Mobilediffusion: Subsecond text-to-image generation on mobile devices. In *ECCV*, 2024. 2

[74] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, 2024. 1, 6

[75] Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. Magicvideo: Efficient video generation with latent diffusion models, 2022. 2

[76] Yuanzhi Zhu, Xingchao Liu, and Qiang Liu. Slimflow: Training smaller one-step diffusion models with rectified flow. In *ECCV*, 2024. 2