Mobile Video Diffusion

Supplementary Material

A1. Additional results

A1.1. Visual quality metrics

Although FVD is a widely used metric for visual quality in the community, its ability to assess the faithfulness of motion as opposed to the appearance of individual frames is sometimes argued [3, 4, 8]. For that reason, we also evaluate the quality of different models using a recently proposed JEDi (JEPA Embedding Distance) metric [7, v.0.1.3]. JEDi reportedly demonstrates much better correlation with the human preference than FVD. Since Luo et al. [7] have not recommended guidelines for comparison between generative models using their metric, we opted for the setup similar to FVD computation [2, 9]. In detail, we used the same set of clips from UCF-101, and similarly downsampled the videos to the resolution closest to 320×240, preserving the original aspect ratio, with the central crop afterwards. In Tab. A1 we provide the extended quantitative results. We observe that in general the new metric is better aligned with architecture optimizations that we apply.

In Tab. A2 we report the full set of metrics measured by VBench-I2V except for 'camera motion' dimension. The reason is that due to the lack of textual prompting, SVD-based models generate random camera trajectories.

A1.2. Decoding latents

As our approach stems from SVD, the presented MobileVD is also a latent model. Therefore, each generated latent code must be decoded to raw RGB pixels. For all the experiments reported in the main text, we used the native autoencoder weights released alongside the SVD model itself. The decoding was conducted independently for each frame. Notably, this model is relatively slow on device: it takes 91.5 ms per frame, resulting in 1,280 ms for decoding the full generated video. This timing is comparable with the latency of MobileVD (1,780 ms). As an alternative, we used the decoder from TAESD autoencoder[1]. It is significantly faster on a smartphone: 6.4 ms per frame, or 90 ms for the full video. At the same time, the difference in quality metrics is negligible, see Tab. A3.

A2. Additional details

A2.1. Training

For diffusion training, we used AdamW optimizer [6] with learning rate of 1×10^{-6} and weight decay 1×10^{-3}, while other hyperparameters were default. During adversarial finetuning, we also used AdamW. For generator the learning rate was equal to 1.25×10^{-6}, and for discriminator we set it 10 times higher. For logits of importance values, used for pruning of temporal blocks, learning rate was equal to 1×10^{-3}. Momentum weights for both optimizers we set as follows $\beta_1 = 0.5$, $\beta_2 = 0.999$. For generator, the weights for adversarial and pseudo-Huber loss were equal to 1 and 0.1 respectively. For discriminator, weight of R_1 penalty was 1×10^{-6}, and we applied it once per 5 iterations, as recommended in [5].

In addition to our main effort to port the SVD model on a mobile phone, we tested if our set of optimizations can be applied to a high-resolution model. For this purpose, we trained a model called MobileVD-HD which is capable of generating 14-frame videos with spatial size of 1024×576 px. Architecture hyperparameters used to finetune MobileVD-HD, *i.e.* temporal multiscaling factor, fun-factor and number of pruned temporal blocks, were the same as for our low-resolution MobileVD. As was shown in the main text, it achieves visual quality on par with SF-V while decreasing its GPU latency by 40%. For high-resolution training, the first (diffusion) stage lasted for 10k iterations with total batch size of 4. The second (adversarial) stage comprised 30k iterations with batch size of 2. The learning rates for adversarial finetuning were twice as lower as for low-resolution case, except for the learning rate of importance values. R_1 penalty was applied at each step. Other training aspects were the same both for MobileVD and MobileVD-HD.

A2.2. Channel funnels

Attention layers. Consider a query and key projection matrices in a self-attention similarity map computation, $XW_q \left(XW_k\right)^T$ with layer input X having a shape of $L \times c_{\text{in}}$, and W_q and W_k of $c_{\text{in}} \times c_{\text{inner}}$. With funnel matrices F_q and F_k of size $c_{\text{inner}} \times c'$, we modify the aforementioned bilinear map as $XW_qF_q \left(XW_kF_k\right)^T = XW_qF_qF_k^TW_k^TX^T$. We apply our coupled singular initialization (CSI) by setting $F_q = W_q^\dagger U_{c'} \Sigma_{c'}^{1/2}$ and $F_k = W_k^\dagger V_{c'} \Sigma_{c'}^{1/2}$. For value and output projections matrices the initialization is applied in the way discussed in the main text.

Convolutional layers. Applying the same initialization to a pair of convolutional layers is not straightforward. Weight tensors of 2D convolutional layers are 4-dimensional, and therefore computation of the effective weight tensor is not obvious. However, we make use of the following observation. Consider input tensor X with shape $h \times w \times c_{\text{in}}$. We refer to its 2-dimensional pixel coordinate as $\vec{p}$, and therefore $X_{\vec{p}}$

[1]https://huggingface.co/madebyollin/taesd/tree/614f768

Model	NFE	FVD ↓		JEDi ↓		TFLOPs ↓	Latency (ms) ↓	
		25 FPS	7 FPS	25 FPS	7 FPS		GPU	Phone
Resolution 1024 × 576								
SVD	50	140	149	0.61	0.59	45.43	376	OOM
MobileVD-HD	1	126	184	0.96	1.75	23.63	227	OOM
Resolution 512 × 256								
SVD	50	366	476	1.05	1.14	8.60	82	OOM
+ low-resolution finetuning	50	194	196	0.71	0.65	8.60	82	OOM
+ optimized cross-attention	50	194	196	0.71	0.65	8.24	76	3630
+ adversarial finetuning	1	133	168	0.66	0.71	8.24	76	3630
+ temporal multiscaling	1	139	156	0.83	0.81	5.42	59	2590
+ temporal block pruning	1	127	150	0.97	1.32	4.64	47	2100
+ channel funneling	1	149	171	1.07	1.21	4.34	45	1780

Table A1. **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. The model trained with the same hyperparameters but intended for high-resolution generations is referred to as MobileVD-HD.

Model	Motion bucket	I2V Subject	I2V Background	Subject Consistency	Background Consistency	Aesthetic Quality	Imaging Quality	Temporal Flickering	Motion Smoothness	Dynamic Degree
SVD (original)	127	93.48	94.74	96.76	96.84	53.59	63.49	95.35	97.29	95.69
SVD (finetuned)	20	95.72	96.36	98.87	98.23	54.91	65.17	98.47	99.16	16.26
SVD (finetuned)	40	95.24	96.04	98.29	97.70	54.62	65.16	97.34	98.78	65.04
MobileVD	20	93.68	94.30	97.22	96.57	53.69	67.16	97.06	98.43	68.21
MobileVD	40	92.98	93.93	96.18	95.95	53.19	67.46	96.00	97.96	95.77

Table A2. **Full VBench-I2V evaluation.**

Decoder	FVD ↓		JEDi ↓		Latency (ms) ↓
	25 FPS	7 FPS	25 FPS	7 FPS	
Original decoder	149	171	1.07	1.21	1280
TAESD decoder	149	179	1.05	1.21	90

Table A3. **Impact of latent decoder.** While being significantly faster on device, decoder from TAESD has little to no impact on visual quality as measured by FVD and JEDi.

is a vector with c_{in} entries. Let W be a convolutional kernel with size $k_h \times k_w \times c_{\text{out}} \times c_{\text{in}}$, and we refer to its spatial 2-dimensional coordinate as $\vec{q}$, while $\vec{q} = 0$ is a center of a convolutional filter. For j-th output channel, $W_{\vec{q},j}$ is also a vector with c_{in} entries. The layer output $Y \in \mathbb{R}^{h \times w \times c_{\text{out}}}$ can be computed as

$$Y_{\vec{p},j} = \sum\nolimits_{\vec{q}} \langle W_{\vec{q},j},\, X_{\vec{p}+\vec{q}} \rangle\,, \tag{A1}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. Simply speaking, this means that convolution can be treated as a linear layer with weight matrix of shape $c_{\text{out}} \times (k_h \cdot k_w \cdot c_{\text{in}})$ applied to each flattened input patch.

At the same time, another way of reshaping the kernel is also possible. Consider a tensor E of shape $k_h \times k_w \times c_{\text{out}} \times h \times w$ defined as

$$E_{\vec{q},j,\vec{p}} = \langle W_{\vec{q},j},\, X_{\vec{p}} \rangle\,. \tag{A2}$$

In other words, the convolution kernel reshaped as $(k_h \cdot k_w \cdot c_{\text{out}}) \times c_{\text{in}}$ is multiplied by features of each input pixel. Then Eq. (A1) can be rewritten as

$$Y_{\vec{p},j} = \sum\nolimits_{\vec{q}} E_{\vec{q},j,\vec{p}+\vec{q}} = \sum\nolimits_{\vec{q}} \langle E_{\vec{q},j},\, \delta_{\vec{p}+\vec{q}} \rangle\,, \tag{A3}$$

where δ is a 4-dimensional identity tensor, *i.e.* $\delta_{\vec{u},\vec{v}} = 1$ if $\vec{u} = \vec{v}$ and 0 otherwise.

Having said that, a sequence of two convolutions can be presented as (i) flattening the input patches; (ii) matrix multiplication by the first kernel reshaped according to Eq. (A1); followed by (iii) matrix multiplication by the second kernel reshaped as in Eq. (A2); and with (iv) independent of kernels operation described by Eq. (A3) afterwards. Therefore, coupled singular initialization can be applied to the product of matrices used in steps (ii) and (iii). We follow this approach and introduce funnels to the pairs of convolutions within the same ResNet block of denoising UNet.

A2.3. Pruning of temporal adaptors

Practical considerations. In the main text we described that we transform the update rule of the temporal block as follows $x_s + \hat{z}\,(1-\alpha)\,r_t$, where x_s and r_t are outputs of the spatial and temporal layers respectively, α is the learnable weight, and $\hat{z}$ is a zero-one gate multiplier. Note that if the temporal block is pruned, *i.e.* $\hat{z}_i = 0$, then the gradient of the loss function w.r.t. the temporal block's learnable parameters equals zero. This affects the gradient momentum buffers used by optimizers. For that reason, we do no update the momentum of the temporal block's parameters in case it has been pruned by all the devices at current iteration of multi-GPU training.

In the network, we parametrize the importance values q_i with the sigmoid function with fixed temperature value of 0.1. In the same way weight coefficients α_i were reparametrized. For faster convergence, each value q_i is initialized with the weight of the corresponding temporal block, *i.e.* $1-\alpha_i$. We also found necessary to set the learning rate for the logits of importance values q_i significantly higher than for the other parameters of the denoising UNet.

Constrained optimization. As discussed in the main text, we relate the importance values of temporal blocks $\{q_i\}_{i=1}^N$ to their inclusion probabilities for sampling without replacement $\{p_i\}_{i=1}^N$ by solving the following constrained optimization problem:

$$\begin{aligned} \min_{c,\{p_i\}_i} \quad & \sum\nolimits_i (p_i - cq_i)^2, \\ \text{s.t.} \quad & \sum\nolimits_i p_i = n, \\ & 0 \le p_i \le 1. \end{aligned} \tag{A4}$$

To solve it, we employ the common method of Lagrange multipliers assuming that all q-values are strictly positive. *W.l.o.g.* we consider the case of sorted values $\{q_i\}_i$, *i.e.* $q_1 \ge q_2 \ge \dots \ge q_N > 0$. In detail, we define a Lagrangian

$$\begin{aligned} & L(c, \{p_i\}_i, \lambda, \beta, \{\gamma_i\}_i, \{\delta_i\}_i) \\ & \quad = \lambda \sum\nolimits_i (p_i - cq_i)^2 \\ & \qquad + \beta \left(\sum\nolimits_i p_i - n\right) \\ & \qquad + \sum\nolimits_i \gamma_i (-p_i) \\ & \qquad + \sum_i \delta_i (p_i - 1) \end{aligned} \tag{A5}$$

and aim to solve the following system of equalities and inequalities

$$\frac{\partial L}{\partial p_i} = 2\lambda (p_i - cq_i) + \beta - \gamma_i + \delta_i = 0 \quad \forall i, \tag{A6}$$

$$\frac{\partial L}{\partial c} = 2\lambda \sum\nolimits_i (cq_i - p_i)\, q_i = 0, \tag{A7}$$

$$\sum\nolimits_i p_i = n, \tag{A8}$$

$$\gamma_i p_i = 0 \quad \forall i, \tag{A9}$$

$$\delta_i (p_i - 1) = 0 \quad \forall i, \tag{A10}$$

$$\gamma_i,\, \delta_i \ge 0 \quad \forall i, \tag{A11}$$

$$\lambda^2 + \beta^2 + \sum\nolimits_i \left(\gamma_i^2 + \delta_i^2\right) > 0. \tag{A12}$$

Case $\lambda = 0.$ In this case $\forall i\ \gamma_i - \delta_i = \beta = \text{const}$. If $\beta > 0$, then $\forall i\ \gamma_i > \delta_i \ge 0 \Rightarrow \gamma_i > 0 \Rightarrow p_i = 0$, which leads to a contradiction. Cases $\beta < 0$ and $\beta = 0$ also trivially lead to contradictions.
Case $\lambda = 1.$ First, we derive that $c \sum_i q_i^2 = \sum_i p_i q_i$, and thus

$$c = \frac{\sum_i p_i q_i}{\sum_i q_i^2} > 0. \tag{A13}$$

Since $\forall i\ \frac{\partial L}{\partial p_i} = 0$, then $\sum_i q_i \frac{\partial L}{\partial p_i} = 0$.

$$\sum\nolimits_i q_i \frac{\partial L}{\partial p_i} \tag{A14}$$

$$= 2\sum\nolimits_i q_i (p_i - cq_i) + \beta \sum\nolimits_i q_i - \sum\nolimits_i q_i (\gamma_i - \delta_i) \tag{A15}$$

$$= \beta \sum\nolimits_i q_i - \sum\nolimits_i q_i (\gamma_i - \delta_i) \tag{A16}$$

$$= 0, \tag{A17}$$

and therefore,

$$\beta = \frac{\sum_i q_i (\gamma_i - \delta_i)}{\sum_i q_i}. \tag{A18}$$

Lemma A2.1. *For all indices i it holds true that $\gamma_i = 0$.*

Proof. Proof is given by contradiction. Let us assume that $\exists k\ \gamma_k > 0 \Rightarrow p_k = 0 \Rightarrow \delta_k = 0 \Rightarrow -2cq_k + \beta - \gamma_k = 0 \Rightarrow \beta = 2cq_k + \gamma_k > 0$. Then for any index j such that $j > k$ and, consequently, $q_j \le q_k$, the following equality holds true:

$$2(p_j - cq_j) + \beta - \gamma_j + \delta_j \tag{A19}$$

$$= 2(p_j - cq_j) + 2cq_k + \gamma_k - \gamma_j + \delta_j \tag{A20}$$

$$= 2p_j + 2c(q_k - q_j) + (\gamma_k - \gamma_j) + \delta_j \tag{A21}$$

$$= 0. \tag{A22}$$

All the terms in the last sum are known to be non-negative except for $\gamma_k - \gamma_j$. Therefore, $\gamma_k \le \gamma_j \Rightarrow \gamma_j > 0 \Rightarrow p_j = 0$. We define index s as the largest index for which $\gamma_s = 0$, *i.e.* $\gamma_1 = \dots = \gamma_s = 0,\ \gamma_{s+1} > 0$. Note that $s > n$, since otherwise the equality $\sum_i p_i = n$ cannot be satisfied. Also, $p_j = 0$ for $j > s$. Now we can rewrite Eq. (A18) as follows

$$\beta = \frac{1}{\sum_i q_i}\left(-\sum_{i\colon i\leq s} q_i\delta_i + \sum_{i\colon i>s} q_i\gamma_i\right) \tag{A23}$$

$$= \frac{1}{\sum_i q_i}\left(-\sum_{i\colon i\leq s} q_i\delta_i + \sum_{i\colon i>s} q_i\left(\beta - 2cq_i\right)\right) \tag{A24}$$

$$= \frac{1}{\sum_i q_i}\left(-\sum_{i\colon i\leq s} q_i\delta_i - 2c\sum_{i\colon i>s} q_i^2\right) + \frac{\beta \sum\limits_{i\colon i>s} q_i}{\sum_i q_i}. \tag{A25}$$

After moving the last term from RHS to LHS, we obtain

$$\frac{\beta \sum\limits_{i\colon i\leq s} q_i}{\sum_i q_i} = \frac{1}{\sum_i q_i}\left(-\sum_{i\colon i\leq s} q_i\delta_i - 2c\sum_{i\colon i>s} q_i^2\right). \tag{A26}$$

Note that LHS is obviously strictly positive, while RHS is non-positive. □

Since $\forall i\ \gamma_i = 0$, we rewrite Eq. (A18),

$$\beta = -\frac{\sum_i q_i\delta_i}{\sum_i q_i}. \tag{A27}$$

If $\forall i\ \ \delta_i = 0$, then it is also true that $\beta = 0$, leading to $\forall i\ p_i = cq_i$. This is the case when inclusion probabilities are *exactly* proportional to the importance values. However, this is possible if and only if the maximum value q_1 is not too large in comparison with other values, since otherwise $p_1 > 1$.
In this last case $\exists k\ \delta_k > 0 \Rightarrow p_k = 1 \Rightarrow 2\left(1 - cq_k\right) + \beta + \delta_k = 0 \Rightarrow \beta = 2\left(cq_k - 1\right) - \delta_k$. For any index j such that $j < k$ we have

$$2\left(p_j - cq_j\right) + \beta + \delta_j \tag{A28}$$

$$= 2\left(p_j - cq_j\right) + 2\left(cq_k - 1\right) - \delta_k + \delta_j \tag{A29}$$

$$= 2p_j + 2c\left(q_k - q_j\right) + \left(\delta_j - \delta_k\right) - 2 \tag{A30}$$

$$= 0. \tag{A31}$$

By regrouping the terms, we obtain

$$2p_j + \delta_j = 2c\left(q_j - q_k\right) + \delta_k + 2 > 2, \tag{A32}$$

and since $p_j \leq 1$, this means that $\delta_j > 0 \Rightarrow p_j = 1$. Therefore, if for some index k it turns out that $\delta_k > 0$, then for all smaller indices j, $\delta_j > 0$, and consequently $p_j = 1$. Again, let us define the index t as the least index with zero δ coefficient, $\delta_{t-1} > 0$, $\delta_t = \delta_{t+1} = \cdots = 0$. Note that $t \leq n+1$, since more that n inclusion probabilities cannot be equal to 1.

For $i \geq t$, $2\left(p_i - cq_i\right) + \beta = 0 \Rightarrow p_i = cq_i - \frac{\beta}{2}$. Therefore,

$$\sum_i p_i = \sum_{i\colon i<t} p_i + \sum_{i\colon i\geq t} p_i = t - 1 + \sum_{i\colon i\geq t}\left(cq_i - \frac{\beta}{2}\right) = n. \tag{A33}$$

Similarly,

$$c\sum_i q_i^2 = \sum_i p_i q_i = \sum_{i\colon i<t} q_i + \sum_{i\colon i\geq t} q_i\left(cq_i - \frac{\beta}{2}\right) \tag{A34}$$

For any given $t = 2, \ldots, n$ two last equations allow us to compute the values of c and β.

$$\begin{pmatrix} \sum\limits_{i\colon i\geq t} q_i & -\left(N - t + 1\right) \\ \sum\limits_{i\colon i<t} q_i^2 & \sum\limits_{i\colon i\geq t} q_i \end{pmatrix} \cdot \begin{pmatrix} c \\ \frac{\beta}{2} \end{pmatrix} = \begin{pmatrix} n - t + 1 \\ \sum\limits_{i\colon i<t} q_i \end{pmatrix}, \tag{A35}$$

where N is the total number of important values. The solution exists and is unique for each t since, obviously,

$$\det\begin{pmatrix} \sum\limits_{i\colon i\geq t} q_i & -\left(N - t + 1\right) \\ \sum\limits_{i\colon i<t} q_i^2 & \sum\limits_{i\colon i\geq t} q_i \end{pmatrix} > 0.$$

In practice, we solve this matrix equation for each $2 \leq t \leq n$, test if the solution satisfies all the constraints, and after that select the solution that delivers the minimum value of our objective function. At least one proper solution always exists, since for $t = n + 1$ inclusion probabilities are equal to 1 for n largest importance values and equal to 0 for all the rest indices.
The solution of the system is differentiable w.r.t. all the q_i, leading to differentiable probabilities p_i. However, as mentioned earlier, we use only p_i computed with these equations for $i \geq t$, while for $i < t$ we set $p_i = 1$. Therefore, we employ a straight-through estimator for these indices [1].

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. 4

[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

[3] Tim Brooks, Janne Hellsten, Miika Aittala, Ting chun Wang, Timo Aila, Jaakko Lehtinen, Ming-Yu Liu, Alexei A Efros, and Tero Karras. Generating Long Videos of Dynamic Scenes. In *NeurIPS*, 2022. 1

[4] Songwei Ge, Aniruddha Mahapatra, Gaurav Parmar, Jun-Yan Zhu, and Jia-Bin Huang. On the Content Bias in Frechet Video Distance. In *CVPR*, 2024. 1

[5] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 1

[6] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 1

[7] Ge Ya Luo, Gian Favero, Zhi Hao Luo, Alexia Jolicoeur-Martineau, and Christopher Pal. Beyond FVD: Enhanced Evaluation Metrics for Video Generation Quality, 2024. 1

[8] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. StyleGAN-V: A Continuous Video Generator With the Price, Image Quality and Perks of StyleGAN2. In *CVPR*, 2022. 1

[9] 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