MOBILEWAN: Closing the Quality Gap for Mobile Video Diffusion

Mohsen Ghafoorian* Denis Korzhenkov* Adil Karjauv* Ioannis Lelekas*
Noor Fathima* Spydron Stasis Animesh Karnewar Amirhossein Habibian
Qualcomm AI Research

(a) Neodragon [VBench: 81.61] (ICLR 2026)



(b) MOBILEWAN [VBench: **83.79**] (Ours)

Figure 1: We significantly advance the quality of Mobile Video Generation compared to the previous state-of-the-art (a) and produce high quality videos (b) comparable to the ones produced by server models. Our user study preferred MOBILEWAN videos **80%** of the time over Neodragon [32].

Abstract

Recent advances in video diffusion have been driven by scaling transformer-based architectures to billions of parameters, substantially improving visual fidelity and motion coherence. In contrast, existing mobile video diffusion models remain limited to relatively small parameter budgets, typically 0.4–1.8B, restricting generation quality. In this work, we show that high-quality mobile video generation does not require small models. Instead, we demonstrate that a server-scale 5B-parameter video diffusion transformer can be deployed efficiently on memory-constrained mobile hardware through recurrent reformulation and structured compression.

Starting from Wan2.2-5B, we rely on a recurrence distillation framework that converts video generation into a chunk-wise autoregressive process with constant-memory attention computation. Combined with causal linear attention, the model operates as an RNN at inference time while preserving temporal coherence across chunks. We further propose a learnable attention head pruning method based on binary per-head gates optimized end-to-end using a noise-biased sparsity objective and distillation-based finetuning. Together with sampling-step distillation and memory-optimized VAE decoding, MOBILEWAN becomes the first 5B-scale video diffusion model deployable on a commercial mobile device. Our system generates 5-second 480×832 videos at 16 FPS with end-to-end latencies ranging from a projected 20s under W8A8 to ∼80s under W8A16, depending on the quantization configuration, and a VBench score of 83.79 for the FP model, establishing a new state of the art in mobile video generation. Please find the released DiT checkpoint and the sampling code in the project page: `https://qualcomm-ai-research.github.io/MobileWan`

*Core equal contributors

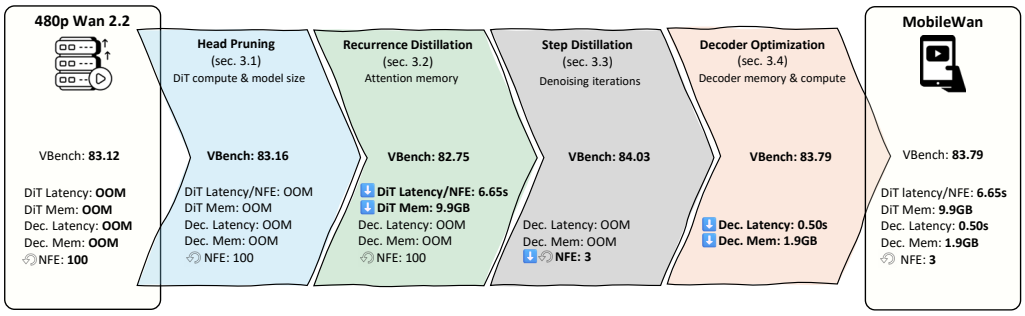

Figure 2: **MOBILEWAN**: We apply a sequence of complementary optimizations, targeting both the diffusion transformer and the decoder, to the Base Wan 2.2 5B model which requires a typical Server GPU like 80GB-A100 and transform it into MOBILEWAN running natively on a mobile device powered by Snapdragon® 8 Gen. 5 NPU, as estimated with W8A8 quantization.

1 Introduction

Video diffusion models have rapidly become a central paradigm for generative visual modeling, enabling applications ranging from creative content production and video editing to simulation and interactive visual generation. Their recent progress has been driven by a transition toward transformer-based architectures (DiT) operating on spatio-temporal tokens, which scale effectively with data and model capacity [6, 28, 58, 65].

Despite rapid progress, mobile video generation still lags substantially behind server-scale systems. Deploying video diffusion models on phones and edge devices is fundamentally constrained by peak memory consumption, the quadratic complexity of self-attention, decoder overhead, and stringent end-to-end latency requirements. As a result, existing mobile video diffusion models are typically restricted to much smaller backbones, commonly ranging from 400M to 1.8B parameters [3, 32, 35, 61, 62, 74]. While such compact architectures satisfy the practical constraints of mobile deployment, they inevitably limit generation quality, leading to an increasingly pronounced gap between mobile and server-scale video generation. As a concrete example, Neodragon [32], the current state-of-the-art video DiT for mobile, still exhibits a clear quality gap compared to server-scale models, a gap that this work seeks to bridge, as illustrated in Fig. 1.

Bridging this gap is challenging because the bottlenecks are not confined to a single module. A natural target is self-attention, whose cost grows quadratically with the number of video tokens. However, replacing softmax attention with linear attention often either requires substantial retraining from scratch [10] or incurs an expressiveness gap relative to full attention. More recently, works on hybrid attention [22, 23] have shown that pretrained softmax-based video diffusion models can be adapted more efficiently, narrowing the quality gap to the original model. Yet these approaches still fall short of converting the entire backbone into a fully recurrent architecture suitable for low-memory mobile deployment, leaving significant memory and compute bottlenecks unresolved.

In this work, we challenge the assumption that mobile video generation requires small diffusion models. Instead, we show that a server-scale 5B-parameter video DiT can be deployed efficiently on memory-constrained mobile hardware through recurrent reformulation and structured compression. Starting from Wan2.2-5B [58], we introduce a recurrence distillation framework that converts a pretrained transformer into a chunk-wise autoregressive model with near-constant memory attention computation. Rather than attending over the full video sequence, the model generates videos chunk-by-chunk while restricting attention to a small local token set. To preserve temporal coherence across chunks, we combine this formulation with a causal linear attention mechanism, enabling the diffusion transformer to operate as an RNN during inference.

We further introduce a learnable head pruning method tailored for video DiT's. Our approach relies on binary per-head gates optimized end-to-end with sparsity regularization and a noise-biased learning strategy that improves the pruning robustness. Combined with recurrent distillation, aggressive sampling-step distillation, and memory-optimized VAE decoding, these optimizations together enable, for the first time, deployment of a 5B-scale video diffusion model on a commercial mobile device. We present MOBILEWAN, a high-quality on-device video diffusion system built on these principles.

Our contributions are summarized as follows:

- We introduce MOBILEWAN, a novel on-device video diffusion system that brings the 5B-parameter Wan2.2 model to mobile hardware, substantially narrowing the quality gap between mobile and server-scale video generation. Through system-level optimizations, we demonstrate the generation of 5-second 480×832 videos at 16 fps in 20-80 seconds based on the underlying model quantization, achieving 83.79 VBench.
- We show that by distilling all transformer blocks into recurrent structures, the DiT inference can be formulated as an RNN over frames, reducing the peak memory footprint as required for deploying large models on mobile devices.
- We propose a noise-biased, learnable head-pruning method that combines per-head gating, sparsity regularization, and targeted timestep sampling, enabling more aggressive and effective pruning than heuristic alternatives under stringent compression settings.

2 Related Work

We review the most relevant prior work on mobile video diffusion models here, and defer the additional coverage of DiT pruning, efficient attention, step distillation, and decoder optimization to Appendix A.1.

Running video diffusion models on mobile devices is severely constrained by memory, compute, and latency limitations, leading most prior work to emphasize aggressive model reductions. Earlier on-device approaches were primarily built on UNet-based architectures. *AMD-Hummingbird* [29] applied visual-feedback-guided pruning to reduce model size with limited quality degradation, while *MobileVD* [3] adapted Stable Video Diffusion [4] through spatial downscaling, multi-scale temporal processing, and structured pruning, achieving over $500\times$ efficiency gains. *SnapGen-V* [62] combined temporal-layer Neural Architecture Search with adversarial step distillation for mobile deployment, and *MoViE* [31] targeted mobile video editing via architectural simplifications and single-step adversarial distillation.

DiT-based VDMs have more recently begun to transition toward on-device execution. *On-device Sora* [34] demonstrates training-free mobile text-to-video generation using step skipping, temporal token merging, and dynamic module loading. *MVDiT* [61] subsequently shows that mobile DiT-based video generation is feasible via extreme VAE compression, KD-guided pruning, and adversarial step distillation. Building on these directions, *Neodragon* [32] introduces a mobile-oriented pyramidal video DiT that integrates block-level pruning with extended distribution-matching and asymmetric decoder distillation, enabling 2-second video generation on mobile NPUs. Finally, *S^2DiT* [74] presents aggressive step distillation for video diffusion transformers, serving as a general acceleration mechanism underlying many later mobile deployments. All the aforementioned mobile models are smaller than 2B parameters and exhibit notable gaps in the quality and/or size of the video they generate, with respect to the server models. Our work focuses on narrowing the *quality gap* by optimizing and enabling a 5B model from Wan, a state-of-the-art and arguably the most widely used open-source video diffusion model family [58].

3 Methods

3.1 Head Pruning

Previous work on pruning diffusion transformers has focused primarily on block pruning [32, 56, 63]. Upon an evaluation of a representative block-pruning strategy adapted from a state-of-the-art efficient video generation model [32], we found that Wan2.2 5B is highly sensitive to this form of compression. Please refer to the supplementary A.2 for more details. This observation motivates a more fine-grained pruning strategy.

We therefore focus on pruning self-attention heads. In Wan2.2 5B, self-attention is one of the most expensive components in terms of FLOPs, which makes it a particularly relevant target for efficient, and especially on-device, generation. We study two approaches: a heuristic pruning method and a learned gating method.

Heuristic head pruning. Our first approach ranks heads according to an importance score and removes the least important ones. For each self-attention head, we measure its importance through its contribution to the final attention output after the output projection. More specifically, let $h_{b,i}(x_t)$ denote the output of head i in block b, and let $W_{b,i}^{\text{out}}$ be the slice of the output projection corresponding to that head. We define the head contribution $c_{b,i}$ as $c_{b,i}(x_t) = W_{b,i}^{\text{out}} h_{b,i}(x_t)$. This can also be interpreted as the difference between the full attention output and the attention output obtained after removing head i. We then compute the raw importance score as the mean squared norm of this contribution: $s_{b,i} = \mathbb{E}_{x_t,t}\|c_{b,i}(x_t)\|_2^2$, where the expectation is taken over a sample set and diffusion timesteps. We compute importance scores over a fixed set of calibration timesteps spanning the diffusion trajectory, with a higher density of timesteps in the high-noise region.

Since score magnitudes may vary substantially across blocks, we normalize them within each block:

$$\tilde{s}_{b,i} = \frac{s_{b,i}}{\frac{1}{H_b}\sum_{j=1}^{H_b} s_{b,j}},$$

where H_b is the number of heads in block b. This normalization allows scores to be compared globally across blocks. Given a target pruning ratio, we remove the bottom $k\%$ of heads according to this global ranking.

After pruning, we perform two-stage fine-tuning. In the first stage, we use block-wise distillation: the dense teacher and the pruned student are optimized block by block, with each block trained in isolation [8]. In the second stage, we fine-tune the entire pruned model end-to-end. We refer to this procedure as the *heuristic head pruning* approach.

Learned head pruning via gates. Our second approach learns which heads to prune directly from gradient signals. For each self-attention head, we introduce a learnable scalar gate. If $h_{b,i}(x_t)$ denotes the output of head i in block b, we modulate its contribution as

$$\hat{c}_{b,i}(x_t) = g_{b,i}\, c_{b,i}(x_t), \qquad g_{b,i} = \text{sigmoid}\left(\frac{\alpha_{b,i}}{\eta}\right),$$

where $\alpha_{b,i}$ is a learnable logit and η is the temperature. We initialize each logit to 5.0, so that the corresponding gate value is close to 1, i.e., the model starts close to the dense solution. Throughout all phases of training, we jointly optimize both the gates and the model weights. This allows the remaining heads to gradually absorb useful computation from suppressed heads, leading to smoother optimization and substantially reducing the quality drop that would arise from hard pruning alone.

The model is trained with the standard flow-matching objective $\mathcal{L}_{\text{FM}}$ augmented with a sparsity regularizer on the gates: $\mathcal{L}_{\text{pruning}} = \mathcal{L}_{\text{FM}} + \lambda \sum_{b,i} g_{b,i}$, where λ controls the strength of pruning. The second term explicitly encourages lower gate values, and therefore promotes sparsity. As the temperature is annealed, this regularizer drives a subset of gates to collapse toward 0, while the remaining gates stay close to 1, yielding an increasingly discrete pruning decision.

We optimize these gates using a three-phase fine-tuning schedule. In the first phase, we set the temperature to $\eta = 1.0$, allowing the model to explore soft gate values. In the second phase, we exponentially anneal the temperature from 1.0 to 0.1. By the end of this phase, most gates are close to binary values. In the final phase, we binarize the gates, freeze them, and continue fine-tuning with a fixed pruned architecture under the standard noise sampling distribution.

A key observation in our experiments is that the noise distribution used during gate learning strongly affects the quality of the final pruned model. During the first two phases, we bias training toward higher noise levels by drawing σ as $\sigma \sim \text{LogitNormal}(1.5, 1)$. Intuitively, high-noise timesteps place greater emphasis on global structure and motion, and are therefore more informative for deciding which heads are essential. In the final phase, after the pruning decisions have been fixed, we return to the standard noise scheme, $\sigma \sim \text{LogitNormal}(0, 1)$. We find this high-noise-biased training to be critical for successful gate learning: at a representative pruning ratio of approximately 33%, it outperforms training with the standard and low-noise-biased distributions in quantitative metrics (see Table 1).

Comparison and efficiency. Across all pruning ratios we consider, the learned gating approach consistently outperforms the heuristic approach, and the gap becomes larger at more aggressive pruning levels (see Figure 3).

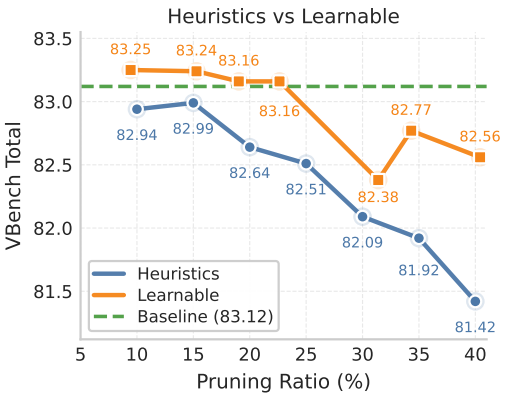

Figure 3: VBench comparison of head pruning methods. Learnable pruning with high-noise-biased training performs best, especially at aggressive pruning ratios.

Table 1: Effect of noise distribution during gate learning. High-noise-biased training yields the best VBench score and visual quality compared with standard and low-noise-biased training.

Model	VBench		
	Quality	Semantic	Total
Low Noise Biased	82.19	72.18	80.19
Standard	82.61	74.12	80.91
High Noise Biased	83.64	76.40	82.19

Self-attention head pruning yields a measurable reduction in FLOPs. While the absolute savings are moderate compared to more aggressive structural modifications, they come with only a limited loss in quality, which makes this trade-off attractive in practice. In particular, such savings are useful in resource-constrained and on-device settings, where even modest efficiency gains can meaningfully improve deployment feasibility. Please refer to the supplementary A.2 for more details.

3.2 Recurrence Distillation

Consider $x \in \mathbb{R}^{N\times D}$ denoting a sequence of N tokens, and a self-attention layer $A(x)$ represented as

$$A(x) = y = \text{softmax}\left(\frac{qk^\top}{\sqrt{D}}\right) v, \tag{1}$$

where queries, keys, and values are computed as linear projections as $q = xw_q,\ k = xw_k,\ v = xw_v$, with learnable parameters $w_q, w_k, w_v \in \mathbb{R}^{D\times D}$. Linear attention attempts to approximate this operation by applying the kernel trick to a rearranged formulation of the softmax attention:

$$y_i = \frac{\sum_{j=1}^{N} \text{sim}(q_i, k_j)\, v_j}{\sum_{j=1}^{N} \text{sim}(q_i, k_j)} \approx \frac{\phi(q_i)\sum_{j=1}^{N} \phi(k_j)\, v_j^\top}{\phi(q_i)\sum_{j=1}^{N} \phi(k_j)}, \tag{2}$$

where $\text{sim}(q_i, k_j) = e^{q_i k_j^\top}$ denotes the unnormalized exponential similarity (up to scaling and numerical stabilization), and the approximation assumes that the exponential kernel can be factorized using a non-negative feature map ϕ, i.e., $e^{q_i k_j^\top} \approx \phi(q_i)\phi(k_j)$. However, since the exponential dot-product kernel has infinite rank, any feature map $\phi(\cdot)$ that exactly reproduces the softmax attention kernel must be infinite-dimensional.

To circumvent the issue, following [22], we leverage a linear/softmax hybrid attention that benefits from the higher expressiveness of softmax and scalability of the linear attention. Specifically, let $x \in \mathbb{R}^{N\times D}$ denote a flattened latent representation obtained from an underlying tensor of shape (T, H, W, D), such that $N = THW$. We divide the temporal dimension into contiguous segments of T_c slices and, for each segment t, define a corresponding token block $X_t \in \mathbb{R}^{N'\times D}$, where $N' = T_c HW$. Equivalently, the full latent can be viewed as a reshaped tensor $X \in \mathbb{R}^{T'\times N'\times D}$, with $T' = N/N'$ denoting the number of temporal chunks; this chunk-indexed notation helps distinguish block-wise operations from single-token indexing (e.g., x_i).

Under this convention, the query matrix for chunk t is given by $Q_t \in \mathbb{R}^{N'\times D}$, and its corresponding linear-attention features by $\phi_q(Q_t) \in \mathbb{R}^{N'\times D'}$. Then for the hybrid attention output $\hat{y}_t \in \mathbb{R}^{N'\times D}$ of tokens in chunk t, we partition the total tokens $\mathcal{T} = \{1, 2..., N\}$ to attend to, into softmax attention tokens $\mathcal{T}_t^{\text{S}}$ and linear attention tokens $\mathcal{T}_t^{\text{L}}$, and compute it as:

Table 2: Comparison of VBench score and compute-burden of various hybrid recurrent models with different number of transformer blocks (out of 30) distilled to recurrence.

Method	VBench			Total Self-Attention TFLOPs
	Quality	Semantic	Total	
Wan2.2 5B	84.16	78.95	83.12	43.3
+ 15 Recurrent blocks	84.79	79.59	83.75	37.0
+ 20 Recurrent blocks	84.51	80.07	83.62	34.9
+ 25 Recurrent blocks	84.07	80.24	83.30	32.8
+ 30 Recurrent blocks	83.68	80.70	83.09	26.9

Table 3: VBench scores of our optimized Wan video DiT after step distillation.

Method	NFEs	VBench		
		Quality	Semantic	Total
DMD	3	83.77	80.90	83.20
D-DMD	4	83.41	80.14	82.75
	3	83.03	80.09	82.44
	2	82.80	80.07	82.25
DMD2	3	85.32	78.89	84.03
	2	85.23	78.83	83.95
	1	80.18	70.40	78.22

$$\hat{y}_t = \frac{\sum_{j\in\mathcal{T}_t^{\mathrm{S}}} \exp(Q_t k_j^\top/\sqrt{D} - c_t) v_j + \phi_q(Q_t)\Big(\sum_{j\in\mathcal{T}_t^{\mathrm{L}}} \phi_k(k_j)\, v_j^\top\Big)}{\sum_{j\in\mathcal{T}_t^{\mathrm{S}}} \exp(Q_t k_j^\top/\sqrt{D} - c_t) + \phi_q(Q_t)\Big(\sum_{j\in\mathcal{T}_t^{\mathrm{L}}} \phi_k(k_j)\Big)}, \tag{3}$$

where c_t is a stabilizing constant (typically the maximum exponent), and $\phi_q(\cdot)$ and $\phi_k(\cdot)$ denote the kernel feature maps for the linear component for queries and keys.

We partition the tokens attended by chunk t into a local set $\mathcal{T}_t^{\mathrm{S}} = \{j \mid tN' \le j < (t+1)N'\}$, handled by softmax attention, and its complement $\mathcal{T}_t^{\mathrm{L}} = \mathcal{T} \setminus \mathcal{T}_t^{\mathrm{S}}$, handled by linear attention. To improve temporal coherence, we allow softmax attention to span overlapping chunks by extending $\mathcal{T}_t^{\mathrm{S}}$ to include T_o neighboring temporal slices, i.e. $\mathcal{T}_t^{\mathrm{S}} = \{j \mid \max(tN' - T_o HW, 0) \le j < (t+1)N'\}$, while keeping $\mathcal{T}_t^{\mathrm{L}}$ unchanged.

Following [23], we employ learnable polynomial feature maps ϕ_q, ϕ_k, constructed via lightweight per-head embeddings and degree-wise expansions, to better approximate the exponential attention kernel. We first performed a block-wise distillation pretraining, where the only unfrozen parameters are the ϕ_qs and ϕ_ks, followed by a full DiT weights finetuning. *Note that once the model is trained, at inference time, we can turn it into a full RNN thanks to the causal linear modeling of the attention to the past chunks for all the transformer blocks, enabling constant memory regardless of the sequence length*. Further details on the two-stage training process and inference-time reformulation to a recurrent model are available in Appendix, A.3. Table 2 compares the baseline Wan model with various recurrent hybrid models with different numbers of transformer blocks converted to recurrence.

3.3 Step Distillation

To further improve the latency of our system, we applied step distillation to the optimized diffusion checkpoint. We opted for distribution matching distillation (DMD) [67] due to its proven performance for image- and video generation tasks. With DMD method, for the single-step student's prediction of the clean video $\hat{x}_\theta$, the forward diffusion process is applied with the noise level τ, $\hat{x}_\tau = (1-\tau)\,\hat{x}_\theta + \tau\varepsilon$ with the noise $\varepsilon \sim \mathcal{N}(0, I)$. Given the teacher velocity model $v(x_t, t)$ and the so-called *fake-score* velocity model $u_\varphi(x_t, t)$, gradient of the DMD loss w.r.t. the student's parameters θ is defined as follows, $\nabla_\theta L_{dmd} \propto \nabla_\theta L_{dmd}^{core} = \left(v^c(\hat{x}_\tau, \tau) - u_\varphi(\hat{x}_\tau, \tau)\right)^T \nabla_\theta \hat{x}_\theta$ where v^c denotes the teacher model output augmented with the classifier-free guidance (CFG) [25] with the scale value of c. The per-sample weight of DMD loss is usually defined as $w_{dmd} = \left\|v^c(\hat{x}_\tau, \tau) - (\varepsilon - \hat{x}_\theta)\right\|_1^{-1}$. Fake-score model u_φ is trained to denoise the outputs of forward diffusion process applied to the student's generations.

Liu et al. [41] have recently reconsidered the use of classifier-free guidance by the teacher model in their D-DMD work. They proposed to split the unnormalized gradient of L_{dmd}^{core} into two terms denoted as DM (distribution matching) and CA (CFG augmentation). It has been shown that the main work of step distillation is conducted by the CA loss which resembles a variation of SDS loss [44, 49] while the theoretically grounded DM loss mostly serves as a regularization term. Namely, the teacher's output can be expressed as $v^c(\hat{x}_\tau, \tau) = v(\hat{x}_\tau, \tau) + (c-1)\cdot\left(v(\hat{x}_\tau, \tau) - \tilde{v}(\hat{x}_\tau, \tau)\right)$, where $\tilde{v}$ means the output conditioned on a blank or negative prompt [18]. Substituting this into $\nabla_\theta L_{dmd}^{core}$ yields

$$\nabla_\theta L_{dmd}^{core} = \nabla_\theta L_{DM}^{core} + (c-1)\cdot\nabla_\theta L_{CA}^{core} \tag{4}$$

with

$$\nabla_\theta L_{DM}^{core} = \left(v\left(\hat{x}_\tau, \tau\right) - u_\varphi\left(\hat{x}_\tau, \tau\right)\right)^T \nabla_\theta \hat{x}_\theta, \tag{5}$$

$$\nabla_\theta L_{CA}^{core} = \left(v\left(\hat{x}_\tau, \tau\right) - \tilde{v}\left(\hat{x}_\tau, \tau\right)\right)^T \nabla_\theta \hat{x}_\theta. \tag{6}$$

Instead of sharing the same noise level τ among DM and CA terms, in D-DMD one samples different levels τ_{DM} and τ_{CA} as well as injected Gaussian noise terms ε_{DM} and ε_{CA} independently. In addition, Liu et al. [41] recommended using τ_{CA} which is cleaner than the noise level σ of the current student's input, i.e. $p\left(\tau_{CA} \mid \sigma\right) \propto p_{prior}\left(\tau_{CA}\right) \cdot \mathbb{1}\left[\tau_{CA} < \sigma\right]$. Sample-wise weights w_{term} for $term \in \{CA, DM\}$ are computed similarly to w_{dmd} but without CFG, $w_{term} = \left\|v\left(\hat{x}_{\tau_{term}}, \tau_{term}\right) - \left(\varepsilon_{term} - \hat{x}_\theta\right)\right\|_1^{-1}$. We used shifted uniform distribution [19] with parameter 5 as a prior for all sampled noise levels in our experiments.

In our experiments we initially conducted step distillation only with the commonly used DMD loss, following the FastVideo recipe [70]. The resulting few-step generator achieves good VBench metrics with just 3 denoising steps (NFEs), see Tab. 3. But visual analysis shows that the produced videos look slightly oversaturated to a human eye, and the dynamic degree drops in comparison with the original model (45.83 vs 68.33). Notably, after switching to the D-DMD loss, the color palette of the outputs changes drastically in an unfavorable way. We found that to solve this, it is sufficient to sample τ_{CA} independently from a student's input noise level σ while preserving the same marginal distribution $p\left(\tau_{CA}\right)$. While this makes the colors look more pleasant, the amount of motion still remains lower than in case of multi-step generator. Overall, despite having lower scores, D-DMD configuration produces visually appealing samples, albeit with a slightly lower dynamic degree compared to the diffusion model. Therefore, we used D-DMD model with 2 sampling steps for our user study (see below).

As an alternative approach to D-DMD, we tried the DMD2 pipeline, i.e. used both adversarial loss and L_{dmd}^{core} [66]. In our implementation of the adversarial loss we followed FastGen [45]: a lightweight convolutional discriminator used the features of $\hat{x}_\tau$ extracted by the teacher network. Note that in this experiment the ground-true visual data is also required while in other scenarios only text prompts are used. As shown, DMD2 increases the dynamic degree (74.72) and results in highly competitive VBench metrics with as few as 2 sampling steps.

3.4 Decoder optimization

Video VAEs are commonly constructed using causal 3D convolutional operators, where the generation of a frame F_t conditions on past frames (e.g., F_{t-1}). While effective, causal Conv3D layers introduce substantial computational and memory overhead, making them unsuitable for mobile deployment. To address this, open source approaches have developed efficient decoder designs based on carefully structured 2D convolutions that approximate the behavior of causal Conv3D layers [5, 15]. In this work, we build upon the open-source LightX2V [15], which is specifically trained for the Wan2.2 video model and is designed to be both compute- and memory-efficient. However, our analysis reveals that despite its efficiency, the resulting videos exhibit pronounced temporal artifacts, leading to degraded visual quality. We attribute this behavior to the limited temporal receptive field of the decoder, where each frame prediction depends on insufficient historical context. To mitigate this issue, we modify the efficient decoder architecture by extending the causal look-back window of each layer. Specifically, the prediction of frame F_t is conditioned on a longer temporal context, incorporating information from frames up to F_{t-4}, as illustrated in Fig. 4. This extended temporal conditioning significantly reduces temporal artifacts while preserving the efficiency advantages of the original design.

We freeze the VAE encoder of Wan2.2 and fine-tune the modified efficient decoder, as illustrated in Fig. 4. The decoder is partially initialized from the pretrained LightX2V VAE, while parameters introduced by the architectural modifications are randomly initialized. Video samples are first encoded using the frozen Wan2.2 encoder to obtain latent representations, and the decoder is trained to reconstruct the corresponding videos using a combination of an L_2 reconstruction loss and a perceptual loss. We train on an in-house synthetic dataset consisting of approximately 80K videos (details avaialble in A.5), each video is resized to a patch size of 480×480 and first 33 frames are used. The decoder is fine-tuned for 200K iterations on 8 NVIDIA A100 GPUs with an effective batch size of 8. For the first 2K training iterations, we use a learning rate of 10^{-5}, followed by 10^{-4} for the

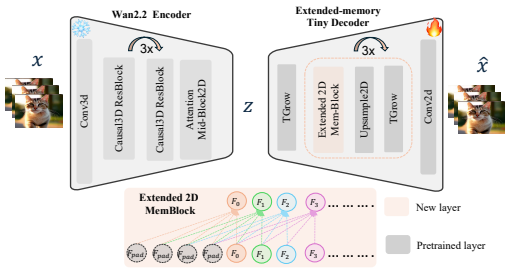

Figure 4: Overview of the VAE architecture. We retain the Wan2.2 encoder and modify the MemBlock from LightX2V [15] by extending the causal look-back window at each layer, enabling the prediction of frame F_t to leverage information from up to F_{t-4}.

Table 4: Quantitative comparison of VAE variants across quality and efficiency metrics. We report VBench scores and PSNR on 60 randomly selected DAVIS [7, 48] test videos $33 \times 512 \times 512$. We also report decoder parameter counts and mobile inference latency measured at target resolution $81 \times 480 \times 832$.

Model	VBench ↑			PSNR ↑	Dec. Params (M)	Mobile Time (ms)
	Total	Qual.	Sem.			
Wan 2.2 VAE	83.12	84.16	78.95	32.69	555.04	OOM
LightX2V VAE	82.44	83.54	78.01	25.01	9.9	72.60
Tiny VAE	82.31	83.34	78.23	26.77	9.9	72.60
Ours	82.93	83.99	78.70	28.85	167.86	501



Figure 5: Qualitative Evaluation of MOBILEWAN model. We visualize uniformly selected frames from the $81 \times 480 \times 832$ videos generated on Qualcomm Snapdragon® 8 Gen. 5 NPU.

remaining iterations. We use the Adam optimizer with $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay of 10^{-3}, and gradient clipping at 1.0.

In Table 4 we compare our proposed VAE against existing efficient decoder baselines and observe consistent improvements in reconstruction quality. Our architectural modifications and fine-tuning strategy lead to measurable gains across multiple metrics, including higher PSNR and improved VBench scores, compared to the efficient decoder counterparts.

4 End-to-end Integration

To further reduce computational cost, we first decrease the token count by downsampling the video's spatiotemporal resolution, and then fine-tune the model at the target setting of $81 \times 480 \times 832$ (16 FPS). Further details are provided in Appendix A.4.

Table 5 shows the one-by-one integration of the optimization components, i.e. head pruning, recurrent hybrid attention conversion, step distillation and decoder integration. As it can be observed, neither each of optimization components nor the whole set significantly drop the corresponding metrics. The detailed metrics are reported in Appendix A.7. Table 7 compares our model to the state-of-the-art efficient video diffusion models and specifically available on-device models that report VBench [27]. Evidently, our model remains pretty competitive.

Computational Efficiency We evaluate the contribution of each optimization to the overall computational efficiency of MOBILEWAN. In addition to proxy efficiency metrics, namely the number of parameters, computational cost (TFLOPs), and attention weight size (MHSA), we report measured memory usage (RAM) and latency (Runtime) with W8A8 latency estimates on a smartphone equipped with a Snapdragon® 8 Gen. 5 NPU. Further precision-specific deployment measurements are detailed in Appendix A.9.

Table 5: The generation quality impact of adding the optimization components one-by-one.

Method	VBench		
	Quality	Semantic	Total
Wan2.2 5B - Finetuned	84.16	78.95	83.12
+ Head Pruning (23%)	84.13	79.30	83.16
+ Recurrence Distil. (30 Blocks)	83.44	80.01	82.75
+ Step Distillation (3 steps)	85.32	78.89	84.03
+ Optimized Decoder	85.16	78.32	83.79

Table 6: Results of the method-blinded human visual preference study over 498 paired videos for comparison with Wan2.2 5B, and 361 for comparisons with Neodragon.

Baseline	Human Preference %		
	MOBILEWAN	No preference	Baseline
Wan2.2 5B - FT	22	25	53
Neodragon	80	11	9

Table 7: Comparisons with SOTA generic and on-device video diffusion models on VBench.

Up-to-5B Server Models	Total↑	Quality↑	Semantic↑
Open-Sora Plan V1.3 [39]	77.23	80.14	65.62
CogVideoX 5B [65]	81.91	83.05	77.33
CogVideoX1.5 5B [65]	82.01	82.72	79.17
Open-Sora V1.2 [77]	79.76	81.35	73.39
LTX-Video [24]	80.00	82.30	70.79
Mobile Video DiT - Server [61]	83.09	84.65	76.86
CogVideoX 2B [65]	81.55	82.48	77.81
PyramidalFlow [30]	81.72	84.74	69.62
M4V [26]	81.91	83.36	76.10
STA [71]	83.00	85.37	73.52
VSA [70]	82.77	83.60	79.47
SANA-Video [10]	83.71	84.35	81.35
Attention Surgery [23]	83.21	85.19	75.25
Wan2.2 5B* [58]	83.12	84.16	78.95
Mobile Models			
SnapGenV [62]	81.14	83.47	71.84
Mobile Video DiT - Mobile [61]	81.45	83.12	74.76
Neodragon [32]	81.61	83.68	73.36
S^2DiT - AR [74]	83.26	**85.63**	73.79
MOBILEWAN(Ours)	**83.79**	85.16	**78.32**

Table 8: Computational efficiency of DiT reported for $1\times$ NFE generating $81 \times 480 \times 832$ video on a Snapdragon® 8 Gen. 5 NPU at W8A8 precision

Model	TFLOPs	Params (B)	MHSA (GB)	RAM (GB)	Runtime (s)
Wan2.2 5 B	98	5.0	15.7	OOM	NA
+ Head Pruning	88	**4.7**	12.1	OOM	NA
+ Recurrent Distillation	**73**	4.8	**0.1**	**9.9**	**6.6**

As shown in Table 8, the original DiT denoiser from Wan2.2 5B fails to run on mobile hardware primarily due to its excessive memory footprint. Although head pruning yields substantial reductions in TFLOPs and attention weight size, it alone is insufficient to enable on-device deployment. In contrast, recurrent distillation makes mobile inference feasible by restricting attention computation to a limited set of tokens, resulting in a drastic reduction in attention weights. This enables a denoising step over the entire video to be completed in 6.6 seconds. Measurements assume W8A8.

Human Evaluation We extend our quantitative evaluation with two internal user studies involving 24 participants, using the two-step D-DMD checkpoint described in Section 3.3. In the first survey, we compare our MOBILEWAN to the original diffusion checkpoint finetuned for our spatiotemporal resolution. In the second, we evaluate the proposed system against the recent Neodragon pipeline [32]. We excluded prompts describing static content and presented the GPT-enhanced long prompts used for generation. Model identities were hidden, and the left/right ordering of the videos was randomized. Participants were shown a text prompt, randomly sampled from the VBench set, and two videos generated by the models compared. They needed to choose the better video or declare that they had no preference. In total, we collected 859 paired comparisons: 498 against the 50-step Wan2.2 5B baseline and 361 against Neodragon. The results are summarized in Tab. 6. MobileWan was preferred in 22% of comparisons against Wan2.2, with no preference in another 25%, yielding a combined preference-or-tie rate of 47%. Against Neodragon, MobileWan was preferred in 80% of comparisons. Qualitative examples are presented in Figure 1 and Figure 5, with additional samples and comparisons in the appendix and supplementary material.

5 Conclusion

We presented MOBILEWAN, a high-quality on-device video diffusion model that narrows the gap to server-scale video generation under mobile constraints. Starting from Wan2.2, we combined a learning-based self-attention head pruning method with a fully recurrent linear/softmax hybrid reformulation of self-attention, together with decoder and distillation improvements, to enable efficient mobile deployment of a 5B-scale video diffusion model. Our final system generates 5 seconds of 480×832 video at 16 fps, with latency ranging from a projected 20 seconds under W8A8 to approximately 80 seconds under W8A16, while achieving a VBench score of 83.79, competitive

with the corresponding server-scale Wan model. Overall, our results show that large video diffusion transformers can be made practical on mobile devices through fine-grained compression and recurrent reformulation, without requiring an extreme loss in quality. Nevertheless, our model still demonstrates some limitations. First, the quality of some scene types, e.g. human faces in the middle distance, is suboptimal. Second, some generations may demonstrate oversaturated colors or reduced motion (depending on the chosen step distillation method). Third, in certain cases our RNN reformulation leads to temporal discontinuities in the generated scenes. Finally, a combination of step distillation and optimized decoder causes flickering visible in some videos. We plan to improve these aspects in future work.

References

[1] Hanno Ackermann, Hong Cai, Mohsen Ghafoorian, and Amirhossein Habibian. HLA: Hadamard Linear Attention, 2026. URL https://arxiv.org/abs/2602.12128. 16

[2] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. 16

[3] Haitam Ben Yahia, Denis Korzhenkov, Ioannis Lelekas, Amir Ghodrati, and Amirhossein Habibian. Mobile video diffusion. In *ICCV*, 2025. 2, 3, 16

[4] 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. URL https://arxiv.org/abs/2311.15127. 3

[5] Ollin Boer Bohan. Taehv: Tiny autoencoder for hunyuan video. https://github.com/madebyollin/taehv, 2025. 7

[6] 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. https://openai.com/index/video-generation-models-as-world-simulators/, 2024. OpenAI technical report, accessed 2026-05-05. 2

[7] Sergi Caelles, Alberto Montes, Kevis-Kokitsi Maninis, Yuhua Chen, Luc Van Gool, Federico Perazzi, and Jordi Pont-Tuset. The 2018 davis challenge on video object segmentation. *arXiv:1803.00557*, 2018. 8

[8] Keshigeyan Chandrasegaran, Michael Poli, Daniel Y Fu, Dongjun Kim, Lea M. Hadzic, Manling Li, Agrim Gupta, Stefano Massaroli, Azalia Mirhoseini, Juan Carlos Niebles, Stefano Ermon, and Li Fei-Fei. Exploring diffusion transformer designs via grafting. In *NeurIPS*, 2026. URL https://openreview.net/forum?id=CaSQgef484. 4

[9] Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Song Han, and Enze Xie. Sana-sprint: One-step diffusion with continuous-time consistency distillation. In *ICCV*, 2025. 16

[10] Junsong Chen, Yuyang Zhao, Jincheng Yu, Ruihang Chu, Junyu Chen, Shuai Yang, Xianbang Wang, Yicheng Pan, Daquan Zhou, Huan Ling, et al. Sana-video: Efficient video generation with block linear diffusion transformer. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026. 2, 9, 16

[11] Yu Cheng and Fajie Yuan. Leanvae: An ultra-efficient reconstruction vae for video diffusion models, 2025. URL https://arxiv.org/abs/2503.14325. 16

[12] Zhenglin Cheng, Peng Sun, Jianguo Li, and Tao Lin. Twinflow: Realizing one-step generation on large models with self-adversarial flows. In *ICLR*, 2026. URL https://openreview.net/forum?id=fBc9v8CVvm. 16

[13] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. URL https://arxiv.org/abs/1904.10509. 16

[14] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH. 16

[15] LightX2V Contributors. Lightx2v: Light video generation inference framework. https://github.com/ModelTC/lightx2v, 2025. 7, 8

[16] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *ICLR*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec. 16

[17] Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Re. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In *NeurIPS*, 2022. URL https://openreview.net/forum?id=H4DqfPSibmx. 16

[18] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. In *NeurIPS*, 2020. 6

[19] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. URL https://openreview.net/forum?id=FPnUhsQJ5B. 7

[20] Xiangyu Fan, Zesong Qiu, Zhuguanyu Wu, Fanzhou Wang, Zhiqian Lin, Tianxiang Ren, Dahua Lin, Ruihao Gong, and Lei Yang. Phased dmd: Few-step distribution matching distillation via score matching within subintervals, 2026. URL https://arxiv.org/abs/2510.27684. 16

[21] Gongfan Fang, Kunjun Li, Xinyin Ma, and Xinchao Wang. Tinyfusion: Diffusion transformers learned shallow. In *CVPR*, 2025. 16

[22] Mohsen Ghafoorian and Amirhossein Habibian. Rehyat: Recurrent hybrid attention for video diffusion transformers. In *CVPR*, 2026. 2, 5, 16

[23] Mohsen Ghafoorian, Denis Korzhenkov, and Amirhossein Habibian. Attention surgery: An efficient recipe to linearize your video diffusion transformer. In *CVPR*, 2026. 2, 6, 9, 16, 18

[24] Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024. 9, 17

[25] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI. 6

[26] Jiancheng Huang, Gengwei Zhang, Zequn Jie, Siyu Jiao, Yinlong Qian, Ling Chen, Yunchao Wei, and Lin Ma. M4v: Multi-modal mamba for text-to-video generation. *arXiv preprint arXiv:2506.10915*, 2025. 9

[27] 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. 8

[28] Hunyuan Foundation Model Team. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024. 2

[29] Takashi Isobe, He Cui, Dong Zhou, Mengmeng Ge, Dong Li, and Emad Barsoum. Amd-hummingbird: Towards an efficient text-to-video model. *arXiv preprint arXiv:2503.18559*, 2025. 3

[30] 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 *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=66NzcRQuOq. 9, 16

[31] Adil Karjauv, Noor Fathima, Ioannis Lelekas, Fatih Porikli, Amir Ghodrati, and Amirhossein Habibian. Movie: Mobile diffusion for video editing. *arXiv preprint arXiv:2412.06578*, 2024. 3

[32] Animesh Karnewar, Denis Korzhenkov, Ioannis Lelekas, Noor Fathima, Adil Karjauv, Mohsen Ghafoorian, and Amirhossein Habibian. Neodragon: Mobile video generation using diffusion transformer. In *ICLR*, 2026. 1, 2, 3, 9, 16, 17

[33] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: fast autoregressive transformers with linear attention. In *ICML*, 2020. 16

[34] Bosung Kim, Kyuhwan Lee, Isu Jeong, Jungmin Cheon, Yeojin Lee, and Seulki Lee. On-device sora: Enabling training-free diffusion-based text-to-video generation for mobile devices. *arXiv preprint arXiv:2502.04363*, 2025. 3

[35] Bosung Kim, Kyuhwan Lee, Isu Jeong, Jungmin Cheon, Yeojin Lee, and Seulki Lee. Mobile-oriented video diffusion: Enabling text-to-video generation on mobile devices without retraining, compression, or pruning. In *WACV*, 2026. 2

[36] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB. 16

[37] Denis Korzhenkov, Adil Karjauv, Animesh Karnewar, Mohsen Ghafoorian, and Amirhossein Habibian. PyramidalWan: On making pretrained video model pyramidal for efficient inference. In *CVPR*, 2026. 16

[38] Zongjian Li, Bin Lin, Yang Ye, Liuhan Chen, Xinhua Cheng, Shenghai Yuan, and Li Yuan. WF-VAE: Enhancing video vae by wavelet-driven energy flow for latent video diffusion model. In *CVPR*, 2025. 16

[39] Bin Lin, Yunyang Ge, Xinhua Cheng, et al. Open-sora plan: Open-source large video generation model. *arXiv preprint arXiv:2412.00131*, 2024. URL https://arxiv.org/abs/2412.00131. 9

[40] Shanchuan Lin, Xin Xia, Yuxi Ren, Ceyuan Yang, Xuefeng Xiao, and Lu Jiang. Diffusion adversarial post-training for one-step video generation. In *ICML*, 2025. URL https://proceedings.mlr.press/v267/lin25m.html. 16

[41] Dongyang Liu, Peng Gao, David Liu, Ruoyi Du, Zhen Li, Qilong Wu, Xin Jin, Sihan Cao, Shifeng Zhang, Steven HOI, and Hongsheng Li. Decoupled DMD: CFG augmentation as the spear, distribution matching as the shield. In *ICLR*, 2026. URL https://openreview.net/forum?id=jBztvOiCKE. 6, 7

[42] Haosong Liu, Yuge Cheng, Wenxuan Miao, Zihan Liu, Aiyue Chen, Jing Lin, Yiwu Yao, Chen Chen, Jingwen Leng, Minyi Guo, and Yu Feng. Astraea: A token-wise acceleration framework for video diffusion transformers. In *ICLR*, 2026. URL https://openreview.net/forum?id=e8P4Oo8S6U. 16

[43] Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-Instruct: A Universal Approach for Transferring Knowledge From Pre-trained Diffusion Models. In *NeurIPS*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/f115f619b62833aadc5acb058975b0e6-Paper-Conference.pdf. 16

[44] David McAllister, Songwei Ge, Jia-Bin Huang, David W. Jacobs, Alexei A. Efros, Aleksander Holynski, and Angjoo Kanazawa. Rethinking Score Distillation as a Bridge Between Image Distributions. In *NeurIPS*, 2024. URL https://openreview.net/forum?id=I8PkICj9kM. 6

[45] Weili Nie, Julius Berner, Chao Liu, and Arash Vahdat. Nvidia fastgen: Fast generation from diffusion models, 2026. URL https://github.com/NVlabs/FastGen. 7

[46] Weili Nie, Julius Berner, Nanye Ma, Chao Liu, Saining Xie, and Arash Vahdat. Transition matching distillation for fast video generation, 2026. 16

[47] NVIDIA. Cosmos world foundation model platform for physical ai, 2025. URL https://arxiv.org/abs/2501.03575. 16

[48] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 8

[49] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. In *ICLR*, 2023. URL https://openreview.net/forum?id=FjNys5c7VyY. 6

[50] Haonan Qiu, Shikun Liu, Zijian Zhou, Zhaochong An, Weiming Ren, Zhiheng Liu, Jonas Schult, Sen He, Shoufa Chen, Yuren Cong, Tao Xiang, Ziwei Liu, and Juan-Manuel Perez-Rua. HiStream: Efficient high-resolution video generation via redundancy-eliminated streaming, 2025. 16

[51] Lingmin Ran and Mike Zheng Shou. TPDiff: Temporal pyramid video diffusion model. In *ICLR*, 2026. URL https://openreview.net/forum?id=Eg3KqoI9tS. 16

[52] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 2021. URL https://aclanthology.org/2021.tacl-1.4/. 16

[53] Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation. In *NeurIPS*, 2026. URL https://openreview.net/forum?id=pzHuesCvcO. 16

[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 Conference Papers*, 2024. URL https://doi.org/10.1145/3680528.3687625. 16

[55] Shitong Shao, Hongwei Yi, Hanzhong Guo, Tian Ye, Daquan Zhou, Michael Lingelbach, Zhiqiang Xu, and Zeke Xie. MagicDistillation: Weak-to-strong video distillation for large-scale few-step synthesis, 2025. 16

[56] Shitong Shao, Yufei Gu, and Zeke Xie. FastLightGen: Fast and light video generation with fewer steps and parameters. In *CVPR*, 2026. 3, 16

[57] Dor Shmilovich, Tony Wu, Aviad Dahan, and Yuval Domb. Liteattention: A temporal sparse attention for diffusion transformers, 2025. 16

[58] Team Wan et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. URL https://arxiv.org/abs/2503.20314. 2, 3, 9, 17

[59] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020. URL https://arxiv.org/abs/2006.04768. 16

[60] Pingyu Wu, Kai Zhu, Yu Liu, Liming Zhao, Wei Zhai, Yang Cao, and Zheng-Jun Zha. Improved video vae for latent video diffusion model. In *CVPR*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wu_Improved_Video_VAE_for_Latent_Video_Diffusion_Model_CVPR_2025_paper.html. 16

[61] Yushu Wu, Yanyu Li, Anil Kag, Ivan Skorokhodov, Willi Menapace, Ke Ma, Arpit Sahni, Ju Hu, Aliaksandr Siarohin, Dhritiman Sagar, Yanzhi Wang, and Sergey Tulyakov. Taming diffusion transformer for real-time mobile video generation. *arXiv preprint arXiv:2507.13343*, 2025. URL https://arxiv.org/abs/2507.13343. 2, 3, 9, 16

[62] Yushu Wu, Zhixing Zhang, Yanyu Li, Yanwu Xu, Anil Kag, Yang Sui, Huseyin Coskun, Ke Ma, Aleksei Lebedev, Ju Hu, et al. Snapgen-v: Generating a five-second video within five seconds on a mobile device. In *CVPR*, 2025. 2, 3, 9

[63] Enze Xie, Junsong Chen, Yuyang Zhao, Jincheng YU, Ligeng Zhu, Yujun Lin, Zhekai Zhang, Muyang Li, Junyu Chen, Han Cai, Bingchen Liu, Daquan Zhou, and Song Han. SANA 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. In *ICML*, 2025. URL https://openreview.net/forum?id=27hOkXzy9e. 3, 16

[64] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *AAAI*, 2021. 16

[65] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Zhang Yuxuan, 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. URL https://openreview.net/forum?id=LQzN6TRFg9. 2, 9

[66] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T. Freeman. Improved Distribution Matching Distillation for Fast Image Synthesis. In *NeurIPS*, 2024. URL https://openreview.net/forum?id=tQukGCDaNT. 7

[67] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step Diffusion with Distribution Matching Distillation. In *CVPR*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Yin_One-step_Diffusion_with_Distribution_Matching_Distillation_CVPR_2024_paper.html. 6, 16

[68] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf. 16

[69] Jintao Zhang, Haoxu Wang, Kai Jiang, Shuo Yang, Kaiwen Zheng, Haocheng Xi, Ziteng Wang, Hongzhou Zhu, Min Zhao, Ion Stoica, Joseph E. Gonzalez, Jianfei Chen, and Jun Zhu. SLA: Beyond sparsity in diffusion transformers via fine-tunable sparse–linear attention. In *ICLR*, 2026. URL https://openreview.net/forum?id=eD8IPvNoZB. 16

[70] Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric P Xing, and Hao Zhang. Faster video diffusion with trainable sparse attention. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. 7, 9

[71] Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention. In *ICML*, 2025. URL https://openreview.net/forum?id=U74MOXPEJd. 9

[72] Yang Zhang, Er Jin, Wenzhong Liang, Yanfei Dong, Ashkan Khakzar, Philip Torr, Johannes Stegmaier, and Kenji Kawaguchi. Learnable sparsity for vision generative models. In *ICLR*, 2026. URL https://openreview.net/forum?id=9pNWZLVZ4r. 16

[73] Zhixing Zhang, Yanyu Li, Yushu Wu, yanwu xu, Anil Kag, Ivan Skorokhodov, Willi Menapace, Aliaksandr Siarohin, Junli Cao, Dimitris N. Metaxas, Sergey Tulyakov, and Jian Ren. SF-v: Single forward video generation model. In *NeurIPS*, 2024. URL https://openreview.net/forum?id=PVgAeMm3MW. 16

[74] Lin Zhao, Yushu Wu, Aleksei Lebedev, Dishani Lahiri, Meng Dong, Arpit Sahni, Michael Vasilkovsky, Hao Chen, Ju Hu, Aliaksandr Siarohin, et al. S2dit: Sandwich diffusion transformer for mobile streaming video generation. *arXiv preprint arXiv:2601.12719*, 2026. 2, 3, 9

[75] Sijie Zhao, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Muyao Niu, Xiaoyu Li, Wenbo Hu, and Ying Shan. CV-VAE: A compatible video VAE for latent generative video models. In *NeurIPS*, 2024. URL https://openreview.net/forum?id=8z4isrqbcf. 17

[76] Kaiwen Zheng, Yuji Wang, Qianli Ma, Huayu Chen, Jintao Zhang, Yogesh Balaji, Jianfei Chen, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Large scale diffusion distillation via score-regularized continuous-time consistency. In *ICLR*, 2026. URL https://openreview.net/forum?id=2uNlM353RI. 16

[77] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024. 9, 16

[78] Junhan Zhu, Hesong Wang, Mingluo Su, Zefang Wang, and Huan Wang. OBS-diff: Accurate pruning for diffusion models in one-shot. In *ICLR*, 2026. URL https://openreview.net/forum?id=eYdPMpW5os. 16

[79] Ya Zou, Jingfeng Yao, Siyuan Yu, Shuai Zhang, Wenyu Liu, and Xinggang Wang. Turbo-vaed: Fast and stable transfer of video-vaes to mobile devices. In *AAAI*, 2026. 17

A Technical appendices and supplementary material

A.1 Extended Related Work

DiT Pruning. Model pruning has emerged as one of the main tools for reducing the cost of diffusion transformers, both in image and video generation. In image generation, a number of recent works have explored structured pruning at different granularities, including block pruning, and channel- or head-level sparsification, often combined with distillation or post-pruning fine-tuning to recover quality [21, 63, 72, 78]. In the video domain, however, pruning remains relatively underexplored, and most existing approaches have focused on structured architectural reduction rather than a detailed study of fine-grained pruning behavior. In particular, recent efficient video diffusion systems, such as [61] and [32], employ structured pruning as part of broader efficiency pipelines. [61] explores pruning at multiple levels, including blocks, attention heads, and FFN dimensions, while [32] relies on block-level pruning. While such approaches can provide substantial efficiency gains, they also make relatively rigid pruning decisions at the block level or combine several compression mechanisms at once, making it difficult to isolate the role of finer-grained pruning. Our work is motivated by the observation that whole-block removal can be too coarse for modern large-scale video diffusion transformers such as Wan2.2 5B. We therefore focus specifically on self-attention head pruning and compare both heuristic and learnable pruning strategies under the same video generation setting.

Efficient Attention. Scaling video diffusion transformers to long spatiotemporal token sequences makes quadratic self-attention a major bottleneck, motivating progress along (i) structured sparsity, (ii) sub-quadratic approximations, and (iii) implementation-level acceleration. Structured sparse patterns (e.g., local, blockwise, or global+window connectivity) reduce attention cost while maintaining or approximating long-range interactions [2, 13, 52, 68]. Complementarily, linearized, kernelized, or low-rank approximations replace full softmax attention with factorized computation to achieve sub-quadratic (often linear) scaling [14, 33, 36, 59, 64]. In parallel, exact attention can be substantially faster in practice via I/O-aware fused kernels that reduce memory traffic and avoid materializing the full attention matrix [16, 17]. Within diffusion/video generation, these directions appear as (a) linear/hybrid attention replacements [1, 10, 22, 23, 69], (b) sparse or temporal-sparse attention [57], and (c) complementary token- and resolution-scheduling pipelines [30, 37, 42, 51]. In MOBILEWAN, we follow this theme by retaining a pretrained video DiT backbone while incorporating mobile-friendly efficient attention components (including ReHyAt) under tight compute budgets.

Step Distillation. Turning pretrained diffusion models into few-step generators, often referred to as 'step distillation' is crucial for creating efficient generative models. While most of the methods for such optimization are still being developed and probed on image models [9, 12, 53, 54], recent works have demonstrated their applicability to video generators as well. SF-V [73] and MobileVD [3] pipelines sucessfully applied latent adversarial distillation to an image-to-video model, while APT [40] scaled it to text-to-video generators. Arguably, most of the current works rely on the DMD [43, 67] method that aims to match the distribution of multi-step teacher's outputs with the distribution of data points produced by a single-step student model [20, 46, 50, 55, 56]. Note that in practice a few-step sampler is typically used, since getting acceptable quality with a single step at inference time is still challenging. It is noteworthy that DMD serves as an additional loss in other methods such as rCM [76] which adapted continuous consistency distillation to video generation. In our work we also opted for DMD pipeline to reduce the number of denoising steps in the final system.

Efficient Decoders. Latent video diffusion models employ video VAEs to compress frames into low-dimensional latent codes for video generation. Early approaches simply extended 2D image VAEs (e.g. Stable Diffusion) with causal Conv3D layers in the decoder, so each frame is generated using only past context. While these decoders preserve causal ordering and reduce flicker, they introduce substantial compute and memory overhead and do not fully exploit temporal redundancy. Open-source latent diffusion models like Open-Sora [77, v1.2] and NVIDIA's Cosmos [47] factorize video latents into separate spatial and temporal modules, effectively replacing heavy 3D convs with chains of 2D convs for improved efficiency. LeanVAE replaces learned upsampling with wavelet-based multiscale decoding[11], and WF-VAE [38] integrates Haar wavelet decomposition to offload low-frequency content from the neural decoder. These designs reduce FLOPs and memory footprints (often by one or two orders of magnitude) while maintaining reconstruction fidelity. Another challenge is maintaining temporal coherence under high compression. Recent designs extend the decoder's temporal receptive field to mitigate artifacts. IV-VAE [60] uses group causal convolution, allowing a look-back window at each layer. This yields smoother frame-to-frame quality without sacrificing the ability to process

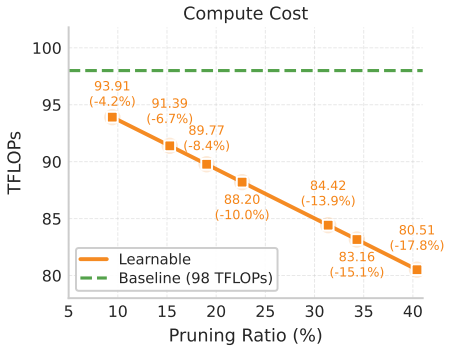

Figure 6: Head Pruning FLOPs vs Pruning Ratio. Number of FLOPs and the corresponding reduction at different pruning ratios with respect to the baseline.

Table 9: Block Pruning VBench Scores. VBench scores for block-pruned models after Stage 1 and Stage 2 (following [32]) with different numbers of removed blocks.

Stage	# Blocks Pruned	VBench ↑		
		Quality	Semantic	Total
1	10	81.24	69.28	78.85
	8	82.66	76.07	81.34
	5	82.87	78.25	81.95
	3	83.51	78.27	82.46
2	10	81.25	68.83	78.76
	8	82.04	74.73	80.58
	5	82.62	76.66	81.43
	3	83.12	77.29	81.95

long videos sequentially. Similarly, Wan2.2 VAE [58] employs overlapping frame blocks or caching strategies to preserve continuity across chunked decoding. Hardware-aware decoders also emerged. LTX-Video [24] and Turbo-VAED [79] specifically target edge devices: they shrink decoder networks via low-rank or depthwise convs and eliminate mobile-unfriendly operations like 3D pixel-shuffle, often by knowledge distillation from a larger model. Lastly, CV-VAE [75] focuses on latent-space compatibility i.e., it adds a regularization term to align the video VAE's latent distribution with an existing image VAE's space, enabling direct fine-tuning from pretrained image diffusion models. This approach ensures efficiency in terms of training cost (reusing weights and data), though not directly addressing runtime speed.

A.2 Details on Model Pruning

Block pruning vs head pruning. When we applied a block-pruning strategy adapted from *Neodragon* [32] to Wan2.2 5B, we found that the model is highly intolerant to this type of compression. Following [32], we first estimate the importance of each transformer block. However, instead of using the original scoring procedure, we define block importance based on the discrepancy between the full model prediction and the prediction obtained when a single block is skipped. Specifically, for each block, we measure the mean squared error between the full-pass velocity prediction (with no block skipped) and the skip-pass velocity prediction (with only that block skipped), averaged over diffusion timesteps and a calibration set. Intuitively, a larger error indicates that removing the block causes a greater deviation from the original model behavior, and therefore suggests higher block importance.

We then rank blocks according to this importance score. When selecting blocks to prune, we additionally enforce a non-neighboring constraint, i.e., pruned blocks are not allowed to be adjacent, in order to reduce the risk of excessive damage to the model. After removing the selected blocks, we fine-tune the pruned model in two stages. In the first stage, we perform end-to-end fine-tuning to recover the quality lost due to block removal. In the second stage, we continue fine-tuning with additional teacher supervision, including both output distillation and intermediate representation distillation, following [32].

The resulting VBench scores are reported in Table 9. As the results show, quality degrades rapidly as more blocks are removed. Pruning three blocks still gives marginally acceptable quality, but at a comparable FLOPs reduction, self-attention head pruning yields noticeably better performance. Based on this observation, we focus on self-attention head pruning in the main paper.

Self-attention head pruning FLOPs gain. Figure 6 shows the FLOPs reduction at different head pruning ratios. The computational cost decreases steadily as more self-attention heads are removed. Although the savings are moderate, they are achieved with relatively limited quality loss, making head pruning attractive for resource-constrained and on-device deployment.

A.3 Details on recurrent hybrid attention

A.3.1 Inference-time RNN Reformulation

A convenient way to view causal linear attention is as a mechanism that maintains a small set of running aggregates. Instead of recomputing attention against the entire past at every step, one can update a fixed-size state that summarizes all previously seen keys and values.

To make such a recurrent implementation valid, the linear-attention path must be strictly causal. In our hybrid layout, causality can be enforced by choosing the temporal index split, following the notation of subsection 3.2

$$\mathcal{T}_t^{\mathrm{L}} = \{j \mid j < \max(tN' - T_o HW, 0)\} \tag{7}$$

With this causal partition, sampling can be organized as a chunked recurrent procedure: the model advances in chunk steps and emits T_c temporal slices per step. The softmax term acts locally within the current chunk, while all information from earlier chunks is carried forward through a compact state. Concretely, we maintain (i) a matrix accumulator $s_t \in \mathbb{R}^{D' \times D}$ that stores the running sum of feature-mapped key–value outer products, and (ii) a vector accumulator $z_t \in \mathbb{R}^{D' \times 1}$ that stores the corresponding running sum for normalization. Using these two variables, the output for chunk t can be computed as

$$s_0 = 0, \tag{8}$$

$$z_0 = 0, \tag{9}$$

$$y_t = \frac{\sum_{j \in \mathcal{T}_t^{\mathrm{S}}} \exp(Q_t k_j^\top / \sqrt{D} - c_t) v_j + \phi_q(Q_t)\, s_t}{\sum_{j \in \mathcal{T}_t^{\mathrm{S}}} \exp(Q_t k_j^\top / \sqrt{D} - c_t) + \phi_q(Q_t)\, z_t}, \tag{10}$$

$$s_{t+1} = s_t + \sum_{j \in \mathcal{T'}_t^{\mathrm{S}}} \phi_k(k_j)\, v_j^\top, \tag{11}$$

$$z_{t+1} = z_t + \sum_{j \in \mathcal{T'}_t^{\mathrm{S}}} \phi_k(k_j), \tag{12}$$

where we use $\mathcal{T'}_t^{\mathrm{S}}$ for updating the linear attention summaries for the next step, to account for the shifted softmax tokens boundary for the next chunk, defined as:

$$\mathcal{T'}_t^{\mathrm{S}} = \{j \mid \max(tN' - T_o HW, 0) \le j < (t+1)N' - T_o HW\}. \tag{13}$$

This presentation highlights an implementation-oriented separation of concerns: intra-chunk interactions are handled by the softmax component, whereas all pre-chunk context is compressed into (s_t, z_t) and accessed through the linear feature maps.

Practically, the model does not need to be optimized in this recurrent form. One may train using the equivalent causal, non-recurrent computation graph and then switch to the above stateful execution at sampling time, which yields the same outputs but enables constant-memory generation. Under this chunked-recurrent implementation, computation scales linearly with the generated sequence length, i.e., $\mathcal{O}(N)$, while the memory footprint is governed only by the fixed-size states (s_t, z_t) and therefore does not grow with video duration.

A.3.2 Two-stage Training

We follow the efficient training recipe from [23], and break the training into two-stages of block-wise attention distillation and light-weight finetuning.

Block-wise recurrence distillation. The goal here is to independently distill the bidirectional softmax attention of each transformer block into the corresponding student recurrent hybrid attention. For a smpled prompt p, noise ϵ and denoising iteration i, the objective is to minimize the L_1 difference between the student and teacher attention outputs.

$$\boldsymbol{\phi}^l = \boldsymbol{\phi}^l - \eta \nabla_{\boldsymbol{\phi}_l} \left(\mathbb{E}_{\substack{\epsilon \in \mathcal{N} \\ p \in \mathcal{P} \\ i \in \mathcal{S}}} \left| y^{(l,\epsilon,p,i)} - \hat{y}^{(l,\epsilon,p,i)} \right| \right). \tag{14}$$

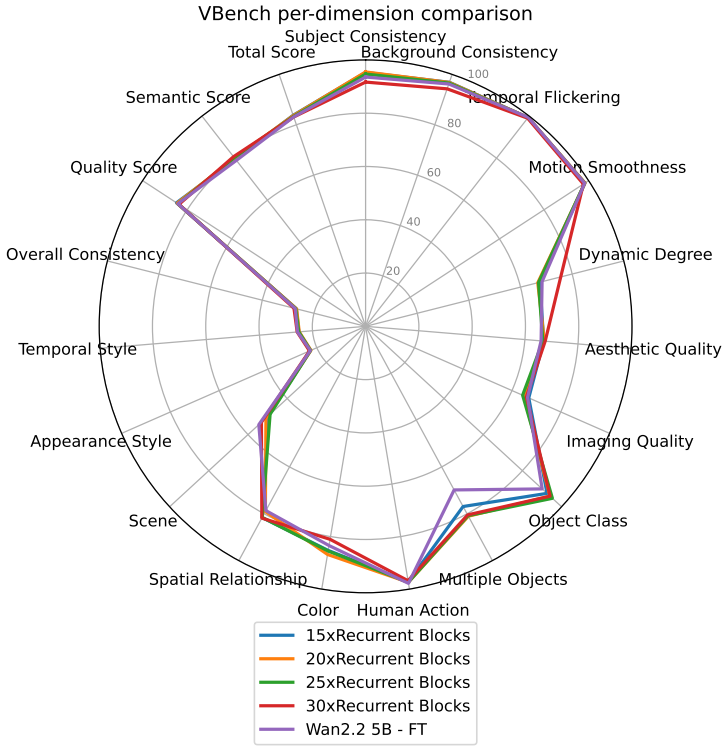

Figure 7: Radar plot comparing a subset of our hybrid models with the original Wan2.2 5B model on the full VBench set and 480×832 resolution

As specified above, in this phase we train only the linear attention kernels of each layer $l \in \{0, ...N\}$ given by $\phi^l = (\phi_q^l, \phi_k^l)$. More specifically, we pre-train the linear attention kernels for 30k iterations with the AdamW optimizer with a learning rate of 1e-3.

Block selection. We observed that converting the first few blocks have a more significant impact on the quality of the model. Therefore, we simply avoid converting the first 3 blocks in variations with lower than 27 blocks, i.e. 15, 20, and 25 blocks. More specifically, for the variant targeting n recurrent blocks, we hybridize the blocks in the range $[\min(3, 30-n), \min(3, 30-n)+n)$.

Hybrid Model Architectural Details. The final model we select for the on-device porting and evaluation has the following architectural specifications: All 30/30 blocks are converted to recurrence, resulting in a full RNN graph. Two-layer MLPs with polynomial degree of 3 is used for the ϕ_q and ϕ_k linear attention kernels. Recurrence latent chunk size T_s is 4 and chunk overlap T_o is set to 2.

Light-weight Finetuning. After pretraining the linear attention kernels, we finetune the whole DiT model for 10k iterations with the flow-matching loss with the AdamW optimizer with a mini-batch size of 16 and learning rate of 4e-5, on a dataset of 80k videos synthetically generated by Wan2.1 14B. Figure 7 provides more details on the per-dimension VBench comparison of the various recurrent hybrid model configurations to the finetuned Wan2.2 5B model. Note that the notably higher dynamic degree metric for full-RNN model (30 converted blocks) appears to be due to higher temporal jitters. The following step for teacher-based DMD step-distillation and finetuning resolves the artifacts observed.

A.3.3 Overall Architecture

Figure 8 describes our final optimized pipeline which still remains an instantiation of the latent video diffusion framework at the core, but replaces the Scheduler, DiT and the Decoder components with our optimized versions.

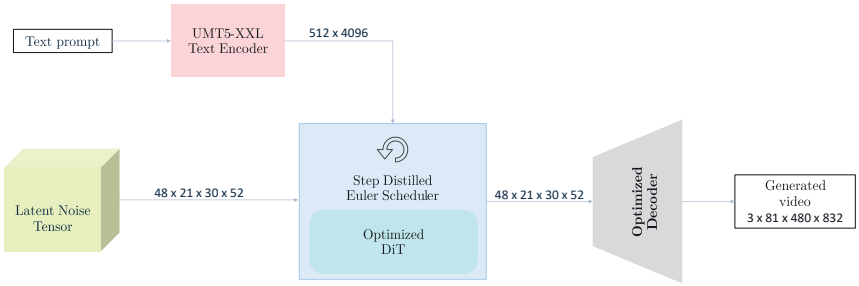

Figure 8: Our optimized pipeline matches exactly the base Wan 2.2 model's instantiation of the latent video diffusion framework, but replaces the DiT and the Decoder components with our optimized versions.

A.4 Target Resolution Finetuning

To reduce the number of tokens for our DiT, we opt to bring down the spatio-temporal resolution of the videos from 121×704×1280 (24 FPS) to 81×480×832 (16 FPS). However, the original Wan2.2 5B model, unlike the 14B model, does not support the 480p resolution, resulting in noticeably subpar generation quality and shaky camera artifacts, perhaps partially due to mismatched RoPE embeddings. Moreover, sampling 81 frames instead of 121 on the released checkpoint, results in lower dynamic contents as it resembles stretching a 3.3 second generation into 5 second by just accordingly decreasing the frame-rate. To alleviate these issues, prior to applying our optimization steps, we finetuned the model on a dataset of 5-second 480p videos at 16 FPS. The dataset consists of 80K videos synthetically generated by Wan2.1 14B. The model was finetuned for 2k iterations with the AdamW optimizer with a batch-size of 6 and learning rate of 1e-5, minimizing the flow-matching loss with a logit normal timestep distribution. Table 10 shows the impact of the finetuning step in terms of quality and compute.

Table 10: Comparison of generation quality of the original Wan2.2 5B model before and after finetuning for our target resolution.

Method	Resolution	#Tokens	VBench		
			Quality	Semantic	Total
Original Wan2.2 5B	121×704×1280	27280	85.03	76.28	83.28
Original Wan2.2 5B	81×480×832	8190	82.46	76.64	81.30
Wan2.2 5B - Finetuned	81×480×832	8190	84.16	78.95	83.12

A.5 Synthetic Training Dataset

We found that the real-world video datasets we considered were often unsuitable for our training setup due to limited resolution, short clip lengths, or undesirable content such as overlaid text and subtitles. We therefore used the publicly available Wan2.1 14B model to generate a synthetic corpus of 80K videos, each containing 81 frames at a resolution of 480×832. We followed the recommended sampling configuration, using a classifier-free guidance scale of 6, the UniPC sampling scheduler (`unipcm`), and 50 denoising steps. Human appearance, motion, and behavior remain particularly challenging for mid-size video diffusion models and are important to perceived generation quality. We therefore oversampled human-centered prompts so that they account for approximately half of the generated videos.

A.6 Qualitative Results and Failure Cases

We include additional qualitative comparisons here (see Figures 9,10,11,12,13, 14). Note that MOBILEWAN generates videos of competitive or better quality than previous state-of-the-art server and mobile models of comparable size. Please refer to the supplementary for the videos.

Although our recurrent formulation generally preserves temporal coherence, we occasionally observe reduced temporal smoothness at transitions between temporal chunks, resulting in visible discontinuities in the generated videos. Representative examples are available in the Failure Cases section of our project page: `https://qualcomm-ai-research.github.io/MobileWan/`.

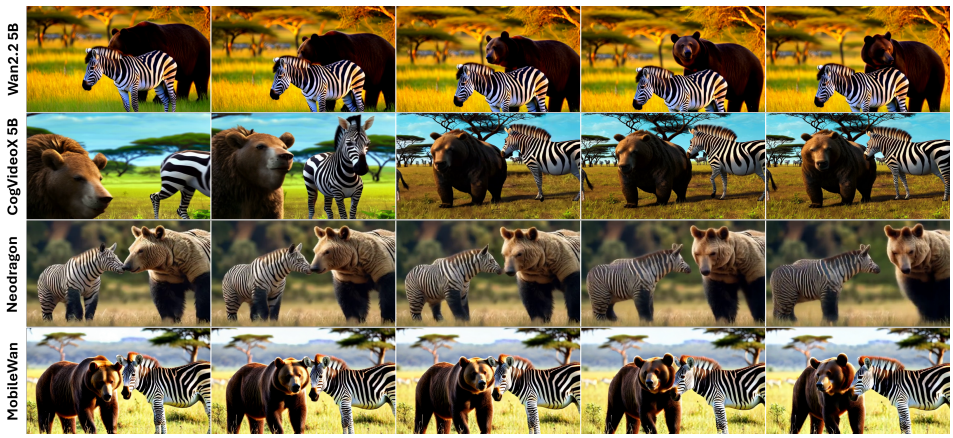

Figure 9: **Prompt:** A bear and a zebra.

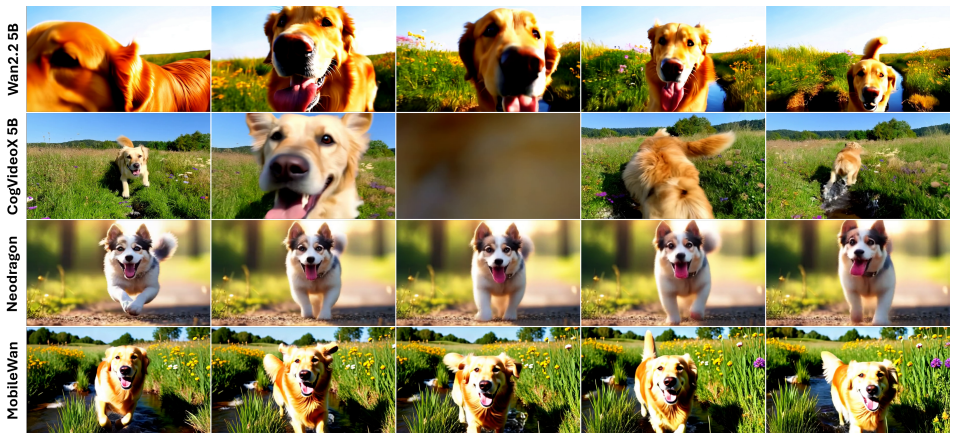

Figure 10: **Prompt:** A dog running happily.



Figure 11: **Prompt:** A happy fuzzy panda playing guitar nearby a campfire, snow mountain in the background.



Figure 12: **Prompt:** A person eating a burger.



Figure 13: **Prompt:** A person is pushing up.



Figure 14: **Prompt:** A teddy bear is swimming in the ocean.

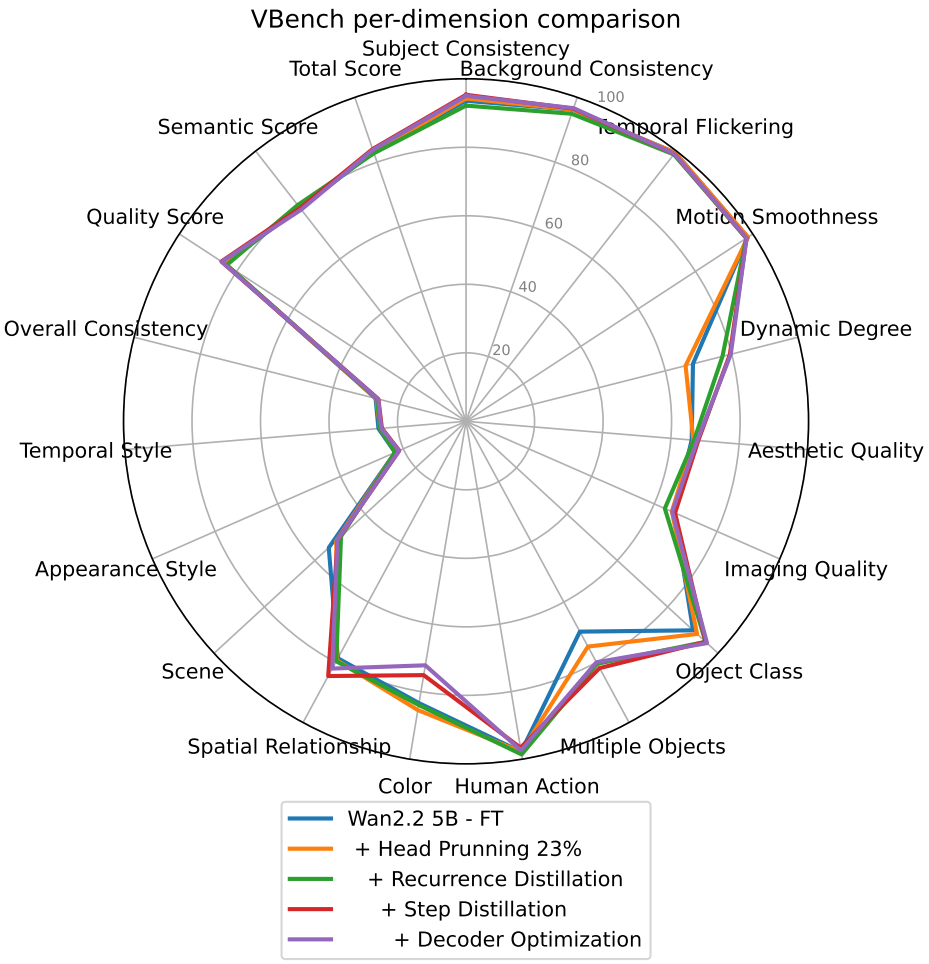

Figure 15: Full set of VBench dimensions for optimizations steps used by MOBILEWAN.

A.7 Details on end-to-end integration

We provide an extended version of Table 5 in Figure 15 to illustrate the impact of every optimization step. While step distillation introduces some drop in the 'Color' dimension, this does not affect the visual appeal of the produced videos. In fact, we found in our experiments that VBench toolkit gives higher scores to slightly oversaturated colors, and this does not always align with the human preference.

A.8 Quantization and mixed-precision training.

We consider deployment configurations with 8-bit weights and different activation precisions. W8A16 retains activations at 16-bit precision and supports straightforward post-training quantization (PTQ). For mixed-precision W8A[8/16] execution, most activations are quantized to 8 bits, while sensitive activations in selected layers, including residual connections, are retained at 16 bits. The extended mixed-precision configuration retains a larger set of sensitive activations at 16 bits.

We initialize quantization using PTQ with the FastForward framework. For the mixed-precision configuration, PTQ alone introduces noticeable degradation in video quality. We therefore apply quantization-aware training (QAT) using a two-stage distillation procedure. First, we perform block-wise distillation, training individual quantized DiT blocks to match the outputs of their floating-point counterparts. We then fine-tune the entire quantized DiT using teacher–student distillation to match the outputs of the floating-point model. This procedure reduces the quality gap introduced by quantization while retaining the deployment benefits of predominantly 8-bit activations.

A.9 Details on deployments and on-device measurements

Reported latencies in Table 11 are measured on research prototype devices with a Qualcomm Snapdragon® 8 Gen. 5 Mobile Platform and a HexagonTM processor, for running inference on the platform's NPU component.

As expected, DiT is the clear run-time bottleneck of the on-device pipeline. The high number of parameters, context lengths and consequently large activation maps lead to a strenuous edge inference use-case based on limited resources of the mobile environment. Thus, porting attempts of the entire base graph result in OOM, hence a split-based porting alternative was adopted, with relevant number of splits - blocks and run-times reported. We differentiate between "pure" run-time and initialization time of the graph; as an example, porting the $6 \times 5-$block Wan2.2 5B - Finetuned, requires 6×5-block$_{\text{run_time}}$ + 5×5-block$_{\text{init_time}}$.

For Table 11 we do consider the initial split always loaded in memory. Head-pruning brings about the first relaxation of the on-device strain, enabling the porting of 3×10-block splits. Full recurrence unlocks the final - claimed runtime, further boosting mobile-friendliness by effectively suppressing context lengths with its recurrent formulation. Similar conclusions can be deduced by comparing full recurrence variants, only this time we are able to port the full 30-block graph, hence leading to no additional initialization overheads (under the assumption of pre-loaded first split).

Comparing similar split-level configurations, across Base - Pruned and full recurrence variants, we see that full recurrence performs best on both initialization and run-times. From conducted run-trace analysis, its superiority can be attributed to its mobile-friendliness. Based and pruned (to a smaller extent) graphs impose significantly heavier DMA traffic, for moving data in-out the NPU's tightly coupled memory, i.e. the Spill-Fill operations, rendering the graph entirely idle in certain periods waiting on the completion of these transfers. On the other hand, full recurrence manages to alleviate those transfers and leads to a better utilized execution graph. This is also reflected by the relevant Spill-Fill numbers.

Table 12 lifts the assumption of pre-loaded splits to provide complete overview of runtimes on different quantization schemes. The more aggressive 8bit setup constitutes the lower runtime-memory bound, leading however to higher quantization error. On the contrary, W8A16/W8AFP16 alleviates quantization-induced performance regressions, at the expense of inflated graph footprint. To balance this trade-off mixed precision variants are also provided for the DiT, where most of the graph (weights and activations) remain on 8bit with a small set of sensitive activation quantizers kept at 16 bit. For the "extended" setup we further widen this sensitive set of quantizers. Inflated graph footprint necessitates introducing more splits for some cases. A final remark on Text encoder's splits, with splits 2-4 carrying each the marked also the table 8 blocks and the first split carrying only the embedding.

B Broader impacts

This paper presents a method for converting a server-scale text-to-video generative model into a system that runs offline on a mobile device. Such capabilities may increase access to generative video technologies and support useful applications in mobile creativity, education, and assistive tools.

However, broader accessibility also introduces potential risks. In particular, efficient on-device generation may reduce the cost of producing misleading or malicious synthetic media, including deepfakes, impersonation, and disinformation.

We believe these risks should be addressed through continued research on robust digital watermarking and content authentication.

C The use of assets

Datasets:

- In-house synthetic data generated by Wan2.1 14B – no license.
- DAVIS – GPL-3.0 license.

Models:

Component	#Splits	#Blocks	Init (s)	Run (s)	Total (s)	RAM (GB)	Spill/Fill (GB)
Text Encoder	-	-	-	0.2	0.2	9.7	0.7
Wan Decoder	-	-	-	-	-	OOM	-
LiteX2V Decoder	-	-	-	0.07	0.07	1.4	1.8
Our Decoder	-	-	-	0.5.	0.5	1.9	10.6
DiT - variants							
Wan2.2 5B - Finetuned			11.9	9.1	21.	4.1	313.2
+ Head Pruning (23%)	6	5	10.	7.4	17.4	3.5	259.8
+ Full recurrence			1.6	6.5.	8.1	1.5	91.2
Wan2.2 5B - Finetuned			-	-	-	OOM	-
+ Head Pruning (23%)	3	10	7.5	7.7	15.2	5.2	294.
+ Full recurrence			1.1	6.5	7.6	2.7	89.4
Wan2.2 5B - Finetuned			-	-	-	OOM	-
+ Head Pruning (23%)	1	30	-	-	-	OOM	-
+ Full recurrence			0.	6.7	6.7	9.9	87.6

Table 11: On device measurements, corresponding to processing of a single $81 \times 480 \times 832$ sample, on Snapdragon® 8 Gen. 5 NPU. $\#$Splits and corresponding $\#$Blocks are reported for split-based profilings, Peak RAM, along with initialization and run-times yielding the total run-time. Finally, Spill and Fill corresponds to data transfers from and back to NPU's tightly coupled memory respectively, to make room for graph operations.

	#Splits	#Blocks	Init (s)	Run (s)	Total (s)	RAM (GB)
Text encoder						
W8AFP16	4	8	3.6	1.1	4.7	2.4
W8A8	-	-	0.1	0.2	0.3	9.7
Decoder						
W8A16	-	-	0.4	1.1	1.5	4.
W8A8	-	-	0.3	0.5	0.8	1.9
DiT						
W8A16	3	10	6.5	19.9	26.4	5.6
W8A8	1	30	1.9	6.7	8.5	9.9
Mixed precision	3	10	4.4	10.8	15.2	4.1
Mixed precision extended	3	10	5.8	14.2	20.	4.4

Table 12: On device measurements for single $81 \times 480 \times 832$ sample processing across component variants with different quantization schemes. DiT timings are per NFE.

- Wan2.1 14B – Apache License 2.0.
- Wan2.2-TI2V-5B – Apache License 2.0.
- LightX2V lighttaew2_2 – Apache License 2.0.
- VBench toolkit – Apache License 2.0.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The on-device latency and peak memory measurements are provided in Sec. 4 and Appendix A.6. The quality of the proposed system is reported in Sec. 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in Sec. 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides the details of finetuning at different stages or cites the open-sourced training recipes. The details of deployment to the final device have been reported as well.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We have not presented any new datasets. We are planning to release the inference code in future.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: See Question 4.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We follow the common practice in the generative models community. Due to the high computational costs of experiments and the multi-stage nature of the proposed system, accurate computation of statistical significance is not feasible. The baseline methods have not reported their error bars either.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Question 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We comply with the Code of ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader impacts are discussed in Appendix B.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: We are not releasing our model at time of submission. We plan to require that users adhere to the guidelines after the checkpoint is released.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The licences are provided in Appendix C.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the DiT checkpoint and sampling code through the project page. The accompanying documentation describes model usage, training, licensing, and limitations.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: The details are provided in Sec. 4.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: There is no equivalent of IRB for our research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [No]

    Justification: LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.