Video Killed the Energy Budget:
Characterizing the Latency and Power Regimes of
Open Text-to-Video Models

Julien Delavande
Hugging Face
ENS Paris-Saclay
julien.delavande@ens-paris-saclay.fr

Regis Pierrard
Hugging Face
regis.pierrard@huggingface.co

Sasha Luccioni
Hugging Face
sasha.luccioni@huggingface.co

Abstract

Recent advances in text-to-video (T2V) generation have enabled the creation of high-fidelity, temporally coherent clips from natural language prompts. Yet these systems come with significant computational costs, and their energy demands remain poorly understood. In this paper, we present a systematic study of the latency and energy consumption of state-of-the-art open-source T2V models. We first develop a compute-bound analytical model that predicts scaling laws with respect to spatial resolution, temporal length, and denoising steps. We then validate these predictions through fine-grained experiments on WAN2.1-T2V, showing quadratic growth with spatial and temporal dimensions, and linear scaling with the number of denoising steps. Finally, we extend our analysis to six diverse T2V models, comparing their runtime and energy profiles under default settings. Our results provide both a benchmark reference and practical insights for designing and deploying more sustainable generative video systems.

1 Introduction

Text-to-video (T2V) generation has rapidly become one of the most compelling frontiers of generative AI. Proprietary systems such as OpenAI's *Sora* [Brooks et al., 2024] and DeepMind's *Veo* [DeepMind, 2025] have showcased remarkable progress in realism and temporal consistency. At the same time, the open-source community is closing the gap, releasing increasingly powerful models [Guo et al., 2024, Yang et al., 2025, HaCohen et al., 2024, Team, 2024, Wan et al., 2025] that can be executed on commodity GPUs. As these systems transition from research prototypes to real-world applications used in creative tools and production-grade video synthesis APIs, it becomes crucial to understand not only their quality, but also their computational costs and environmental impacts.

Generating even a few seconds of coherent video typically requires dozens of denoising steps, high spatial resolutions, and hundreds of frames. This leads to substantial energy consumption and long inference times. Yet, most evaluations of T2V models emphasize perceptual metrics such as sample fidelity, FID scores, or motion smoothness, while largely overlooking latency and energy efficiency. In an era where democratization and sustainability are key, these overlooked dimensions deserve systematic study.

In this paper, we make the following contributions:

- **Theoretical analysis.** We develop a compute-bound analytical model of latency and energy for WAN2.1-T2V [Wan et al., 2025], decomposing FLOPs by operator and predicting scaling laws with respect to spatial resolution, temporal length, and denoising steps.
- **Empirical validation.** We perform fine-grained microbenchmarks on WAN2.1-T2V to test these predictions, revealing quadratic scaling in spatial and temporal dimensions, and linear scaling in steps.
- **Cross-model benchmarking.** We extend our analysis to six open-source T2V models, comparing their latency and energy profiles under default generation settings.
- **Implications.** We discuss the consequences of these findings for efficient deployment, sustainable model design, and future directions such as diffusion caching and quantization.

Together, these contributions provide both a modeling framework and empirical evidence for understanding the structural inefficiencies of T2V pipelines, offering actionable insights for balancing quality and sustainability in generative video systems.

2   Related Work

The environmental costs of machine learning are a new but growing field of scholarship, starting with the pioneering study of Strubell et al., which was the first to quantify the carbon footprint of training a large language model (LLM) [2019]. The subsequent years were marked by more work on the carbon footprint of different types of machine learning (ML) models and the factors that influence them Patterson et al. [2021], Luccioni et al. [2022], Gupta et al. [2021], Wu et al. [2022]. While much of the initial work was focused on ML model training – given that it presents a larger up-front cost in terms of energy and carbon – recent work has increasingly focused on inference, given the ubiquity of deploying different kinds of ML models in practice. Notably, Luccioni et al. [2024] carried out the first large-scale study on the energy and carbon costs for different tasks and approches, including image generation.

While there is limited existing work on the energy demands of video generation, recent work by Li et al. [2024], studied the energy needed to generate videos by the Open-Sora model Zheng et al. [2024]. They analyzed the energy required to generate 2-second videos at 240p resolution, and found that not only is video generation significantly more energy-intensive than text generation (which corroborates the findings of Luccioni et al. [2024]), but also that *"the primary source of emissions stemming from iterative diffusion denoising"*. They also found that the energy requirements of video generation scales near-quadratically with video resolution. This is the only existing published study on the energy requirements of video-generation, which is nonetheless limited to a single model and type of output (i.e. video length and resolution), emphasizing the importance of having a better understanding of this important topic. This was the motivation for our own study, which we describe in the following section.

3   Theoretical Model of Latency and Energy

To ground our analysis, we focus on the **WAN2.1-T2V-1.3B** model [Wan et al., 2025], which serves as our reference architecture. WAN2.1 is representative of modern latent text-to-video diffusion systems: a pretrained text encoder provides conditioning, a timestep embedding MLP injects the diffusion step index, a large DiT (Diffusion Transformer) performs the bulk of spatio-temporal denoising, and a VAE decoder maps latent tensors back to pixel space. This structure is shown in Figure 1. The same framework can be applied to other recent models with minor adjustments. WAN2.1 is also the most downloaded text-to-video model on the Hugging Face Hub at the time of writing, motivating its selection for an in-depth study.

We are then able to derive a compute-bound analytical model of WAN2.1 inference, decomposing FLOPs by operator and predicting latency and energy as explicit functions of resolution (H, W), number of frames T, and denoising steps S.

3.1   Compute vs. Memory-Bound Regimes

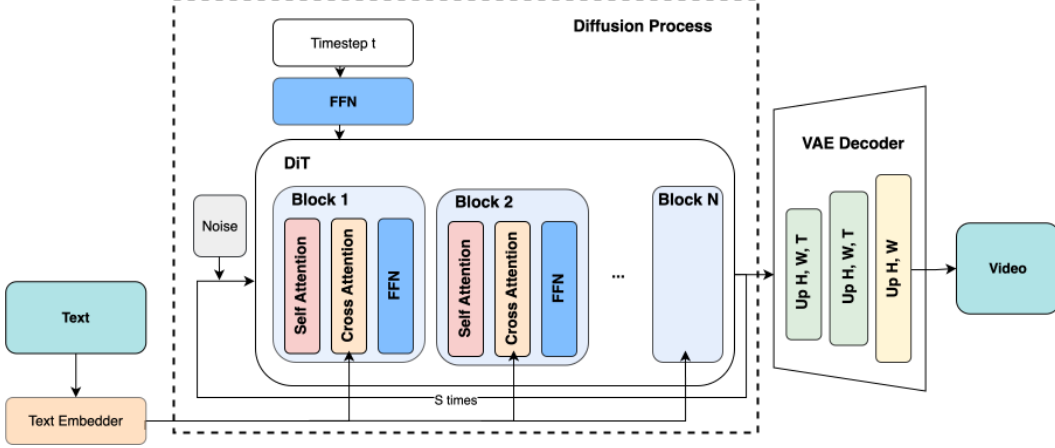On modern GPUs such as the NVIDIA H100, inference kernels can be either:

Figure 1: Simplified architecture of WAN2.1-T2V-1.3B.

- **Compute-bound**, when execution is limited by arithmetic throughput (FLOP/s).
- **Memory-bound**, when limited by memory bandwidth.

Profiling shows that the main operators of WAN2.1 inference (self-attention, cross-attention, MLPs, VAE convolutions) are predominantly compute-bound. GPU utilization remains saturated, and power traces indicate negligible CPU-induced idle time. We therefore adopt a compute-bound model, following the classic roofline formulation [Williams et al., 2009], where latency is proportional to total FLOPs divided by sustained throughput. This approximation is consistent with prior studies of large-scale transformer workloads [Shoeybi et al., 2019, Narayanan et al., 2021, Hagemann et al., 2024, Jiang et al., 2024, Pavani et al., 2025].

3.2 Notation and Constants

We follow the HPC convention where one multiply-add corresponds to two FLOPs. Throughout, $H \times W$ denotes the spatial resolution, T the number of frames, S the number of denoising steps, N the number of DiT layers, d the hidden size, f the MLP expansion factor, m the text conditioning length, g the number of classifier-free guidance (CFG) passes, and ℓ the latent token length seen by the DiT. A complete list of symbols, constants, and hardware parameters is provided in Appendix A.

The DiT token length ℓ grows with the spatial (H, W) and temporal (T) dimensions of the latent grid:

$$\ell = \left(1 + \frac{T}{4}\right) \frac{H}{16} \frac{W}{16}.$$

3.3 Operation-Level FLOP Breakdown

The total FLOPs per video generation can be decomposed into contributions from the text encoder, timestep MLP, the diffusion transformer (DiT), and the VAE decoder, see table 1. A full derivation of these FLOP formulas is provided in Appendix A, where we detail each operator (self-attention, cross-attention, MLP, VAE, text encoder, timestep MLP).

3.4 Total FLOPs

The total FLOPs for generating a video of spatial size $H \times W$, T frames, and S steps is:

$$F_{\text{total}} = F_{\text{text}} + F_{\text{VAE,conv}} + F_{\text{VAE,mid-attn}} + Sg \cdot \left(F_{\text{self}} + F_{\text{cross}} + F_{\text{mlp}} + F_\tau\right).$$

We define μ as the ratio between sustained and peak throughput:

$$\mu = \frac{F_{\text{total}}/D_{\text{measured}}}{\Theta_{\text{peak}}}.$$

3

Table 1: FLOP cost of WAN2.1-T2V-1.3B components. Top: once per video. Bottom: per denoising step (to be multiplied by gS). Symbols are defined inline in Section 3, with the complete list deferred to Appendix A.

Component	FLOPs	Notation
Once per video		
Text encoder (T5)	$p_{\text{text}}\, L_{\text{text}} \left(8 m d_{\text{text}}^2 + 4 m^2 d_{\text{text}} + 4 f_{\text{text}} m d_{\text{text}}^2 \right)$	F_{text}
VAE decoder convolutions	$\sum_{j=1}^{N_{\text{dec,conv}}} 2\, k_t^{(j)} k_h^{(j)} k_w^{(j)}\, C_{\text{in}}^{(j)} C_{\text{out}}^{(j)}\, T^{(j)} H^{(j)} W^{(j)}$	$F_{\text{VAE,conv}}$
VAE decoder 2D "middle" attention	$T_* \left(8\, C_*^2 L_* + 4\, L_*^2 C_* \right)$	$F_{\text{VAE,mid-attn}}$
Per denoising step (multiply by gS)		
DiT		
Self-attention (N layers)	$N \left(8 \ell d^2 + 4 \ell^2 d \right)$	F_{self}
Cross-attention (N layers)	$N \left(4 \ell d^2 + 4 m d^2 + 4 \ell m d \right)$	F_{cross}
MLP (N layers)	$N \left(4 f \ell d^2 \right)$	F_{mlp}
Timestep MLP (shared across layers)	$2\, d_\tau d\ +\ 14\, d^2$	F_τ

Assuming compute-bound execution with empirical efficiency μ, and letting Θ_{peak} denote the GPU's theoretical peak throughput in dense BF16, the total latency D_{total} of generating a video can be approximated as:

$$D_{\text{total}} \approx \frac{F_{\text{total}}}{\mu\, \Theta_{\text{peak}}}.$$

In practice, the H100 provides a dense BF16 peak of $\Theta_{\text{peak}} = 989\,\text{TFLOP/s}$ (NVIDIA datasheet), but this level is unattainable. The empirical efficiency μ thus acts as a correction factor, reflecting both hardware under-utilization (tile misalignment, kernel overheads, memory-bound ops) and approximations of our latency model. For WAN2.1 – after performing the experiments explained in section 4 – we obtain $\mu \approx 0.456$, consistent with sustained FLOP utilization of 30–63% reported for large-scale transformer inference on H100s [Hagemann et al., 2024, Jiang et al., 2024, Pavani et al., 2025]. We calibrated μ by linear regression of measured latencies against theoretical FLOPs across our experiments, which yielded $\mu = 0.456$ with negligible overhead and $R^2 = 0.998$.

Compute-bound regime. On the H100, main operators such as self-attention and MLPs become compute-bound above sequence lengths of $\ell \approx 295$ and $\ell \approx 590$, respectively. Since all configurations studied here operate at much higher token counts (ℓ is typically in the 10^4-10^5 range even for moderate resolutions such as 480×720 and a few seconds of video), these blocks are firmly compute-bound. For very short ℓ, the MLP dominates latency and energy, but such regimes are far below our operating range. Full derivation and extensions to other hardware showing the same trends are given in Appendix B.

3.5 Energy Model

Since sustained GPU power remains close to P_{max} during inference, the total energy consumed E_{total}:

$$E_{\text{total}} \approx P_{\text{max}} \cdot D_{\text{total}}.$$

where P_{max} denotes the GPU's maximum power draw (here $\sim 700\,\text{W}$). Thus, energy and latency scale proportionally.

3.6 Predicted Scaling Regimes

From these equations, we can anticipate distinct computational regimes:

- **Quadratic scaling in spatial and temporal dimensions.** Since the DiT token length ℓ grows linearly with H, W, and T, the self- and cross-attention terms contribute $\mathcal{O}(\ell^2)$ FLOPs, leading to quadratic growth in latency and energy as resolution or frame count increases.
- **Linear scaling in denoising steps.** Each step applies the same sequence of N transformer layers, so the ideal cost scales as $\mathcal{O}(S)$.
- **Negligible contributions from auxiliary components.** The text encoder is run once per video, and the timestep MLP adds only a small overhead per step. Likewise, the VAE decoder scales linearly with voxel count $T \times H \times W$ and is quickly dominated by the quadratic DiT cost.

In summary, the theoretical model predicts that WAN2.1 inference is *transformer-dominated and compute-bound*, with quadratic regimes in spatial and temporal dimensions, linear dependence on denoising steps, and minor overhead from conditioning networks. These predictions will be validated against empirical measurements in Section 5.

4 Methodology

Our methodology combines two complementary perspectives. First, we perform controlled micro-benchmarks on **WAN2.1-T2V-1.3B**, our reference model, to validate the scaling regimes predicted by the theoretical model (Section 3). Second, we benchmark a diverse set of recent open-source text-to-video models under default settings, to situate WAN2.1 within the broader ecosystem.

4.1 Hardware and Measurement Protocol

All experiments were conducted on a dedicated NVIDIA H100 SXM GPU (80GB HBM3) paired with an 8-core AMD EPYC 7R13 CPU, with no co-scheduled jobs. We measured GPU and CPU energy using `CodeCarbon` [Courty et al., 2024], which interfaces with NVML and pyRAPL, and estimated RAM energy using CodeCarbon's default heuristic[1].

To reduce noise, each measurement included two warmup iterations, followed by five repeated runs. Inference used the Hugging Face `Diffusers` library von Platen et al. [2022] with default generation parameters. We relied on the standard optimizations provided by recent PyTorch releases, such as fused kernels and FlashAttention [Dao, 2023], which are automatically enabled.

4.2 Controlled Scaling Experiments on WAN2.1-T2V-1.3B

To validate the theoretical model, we systematically varied the three key structural parameters: resolution, number of frames, and denoising steps. Since the text encoder always pads or truncates prompts to a fixed length of 512 tokens, the specific choice of prompt does not affect runtime. We therefore fixed a single prompt and applied the same warmup-and-repetition protocol as above to isolate structural scaling laws.

- **Spatial resolution:** from 256×256 to 3520×1980, both dimensions divisible by 8 (model constraint). Frames and steps fixed.
- **Temporal length (frames):** from 4 to 100 in increments of 4 (model constraint). Resolution and steps fixed.
- **Denoising steps:** from 1 to 200. Resolution and frames fixed.

For each configuration we logged total latency (seconds) and energy for each hardware component (GPU / CPU / RAM).

4.3 Cross-Model Benchmark

To provide a bird's-eye view of energy and latency costs across current systems, we selected a diverse set of models spanning different architectures and parameter scales (Table 2), focusing on those that are among the most downloaded and trending on the Hugging Face Hub at the time of writing.

---

[1]`https://mlco2.github.io/codecarbon/methodology.html#ram`

For this benchmark, we generated **50 different prompts** per model. Each prompt was measured with the protocol above (2 warmups, 5 runs), yielding robust averages and standard deviations that capture both runtime noise and input variability.

- **AnimateDiff** [Guo et al., 2024](License) - lightweight motion-layer diffusion.
- **CogVideoX-2b/5b** [Yang et al., 2025] (License) - cascaded base + refiner stages.
- **LTX-Video-0.9.7-dev** [HaCohen et al., 2024](License) - autoregressive temporal modeling.
- **Mochi-1-preview** [Team, 2024](License) - large-scale diffusion optimized for motion realism.
- **WAN2.1-T2V (1.3B and 14B)** [Wan et al., 2025](License) - high-resolution latent diffusion with DiT backbone.

Table 2: Default generation settings for each model (from Hugging Face model cards).

Model	Steps	Resolution (HxW)	Frames	FPS
AnimateDiff	4	512×512	16	10
CogVideoX-2b	50	480×720	49	8
CogVideoX-5b	50	480×720	49	8
LTX-Video	40	512×704	121	24
Mochi-1-preview	64	480×848	84	30
WAN2.1-T2V-1.3B	50	720×1280	81	15
WAN2.1-T2V-14B	50	720×1280	81	15

We did not assess perceptual quality to isolate compute behavior; instead, these experiments confront the predicted quadratic and linear regimes (Section 3) with actual scaling laws and scheduler-induced deviations. All code, prompts, and configurations are available in an anonymized repository at GitHub repo, and all generated videos are released on the Hugging Face Hub at HF org.

5 Empirical Findings

We now compare the theoretical predictions of Section 3 with empirical measurements – first by conducting a fine-grained validation on **WAN2.1-T2V-1.3B** and comparing measured energy and latency against theoretical curves as resolution, temporal length, and denoising steps vary. We then situate these results in the broader context of other open-source video generation models.

5.1 Validation on WAN2.1-T2V-1.3B

In this section we focus exclusively on *GPU energy and latency*, since GPU accounts for 80–90% of the total consumption and dominates inference cost. Figures show theoretical predictions (stacked areas by operator: self-attention, cross-attention, MLP, VAE, text encoder, timestep MLP) with empirical measurements overlaid as points with error bars.
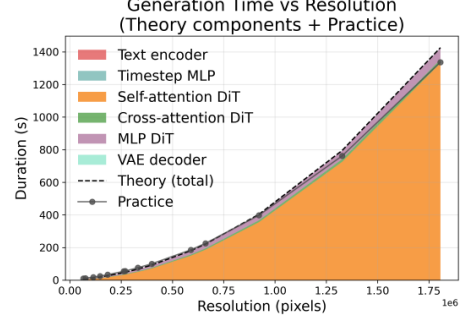
5.1.1 Spatial Resolution

Increasing the resolution from 256×256 to 3520×1980 (frames - 81 and steps - 50 fixed) *causes both latency and energy to grow quadratically*. Theoretical predictions (stacked by operator) and empirical measurements are compared in Figure 2. The agreement remains strong across the entire range, with modest deviations at high resolutions (see Table 3). The VAE contribution remains minor compared to the DiT blocks.

5.1.2 Temporal Length (Frames)

Varying the number of frames from 4 to 100 (resolution - 720 × 1280 and steps - 50 fixed) also induces *quadratic growth* in both latency and energy, as shown in Figure 3. This behavior directly follows from the quadratic dependence of attention on the token count ℓ. The model closely tracks empirical results, with errors reported in Table 3.
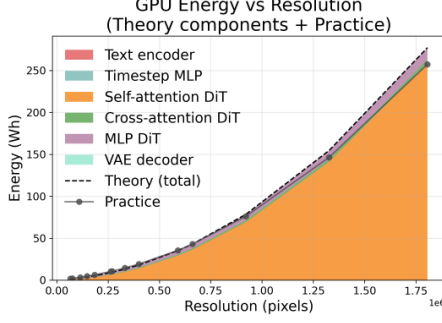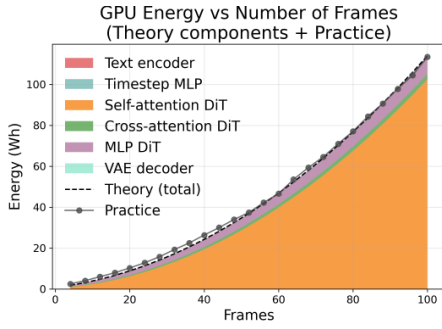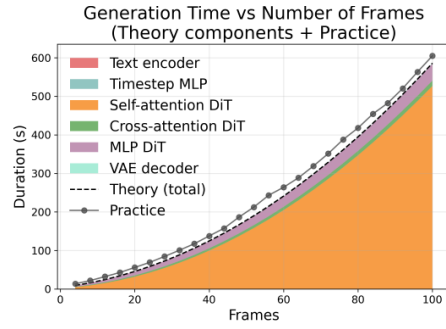
(a) GPU energy vs. spatial resolution

(b) Latency vs. spatial resolution

Figure 2: Empirical results (points) vs. theoretical predictions (stacked areas per operator) as a function of resolution. Both energy and latency follow the predicted quadratic regime.
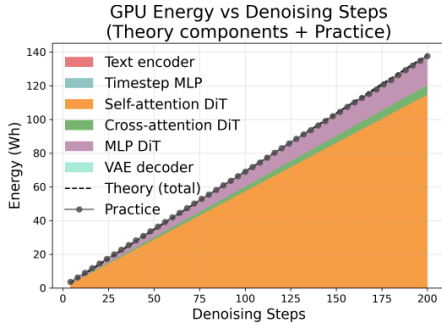

(a) GPU energy vs. number of frames
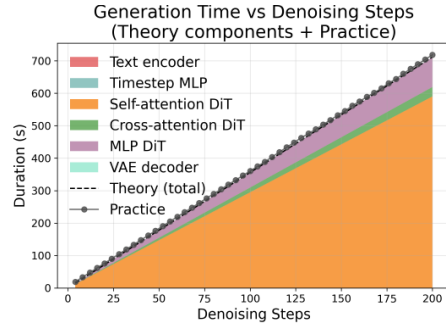
(b) Latency vs. number of frames

Figure 3: Empirical results (points) vs. theoretical predictions (stacked areas per operator) as a function of temporal length. Both metrics follow the quadratic regime predicted by the model.

5.1.3 Denoising Steps

In contrast to resolution and frame count (resolution - 720×1280 and frames - 81 fixed), scaling with the number of denoising steps is *perfectly linear*, exactly as predicted by the theoretical model. Each additional step applies the same N transformer layers, leading to a cost that grows proportionally with S. Figure 4 shows near-perfect alignment between predictions and measurements, with errors below 2% (Table 3).

(a) GPU energy vs. denoising steps

(b) Latency vs. denoising steps

Figure 4: Empirical results (points) vs. theoretical predictions (stacked areas per operator) as a function of denoising steps. Both energy and latency scale linearly with S, in near-perfect agreement with the compute-bound model.

7

Table 3: Mean percentage error (MPE) between theoretical predictions and empirical measurements.

	Energy	Latency
Resolution scaling	11.6%	14.0%
Temporal length	6.6%	10.5%
Denoising steps	1.9%	1.9%

5.2 Cross-Model Comparison

Finally, we compare average GPU energy consumption, latency, and component-wise energy shares across seven open-source text-to-video models under their default generation settings (Figure 5).

(a) GPU energy per model for one video

(b) Generation latency per model for one video

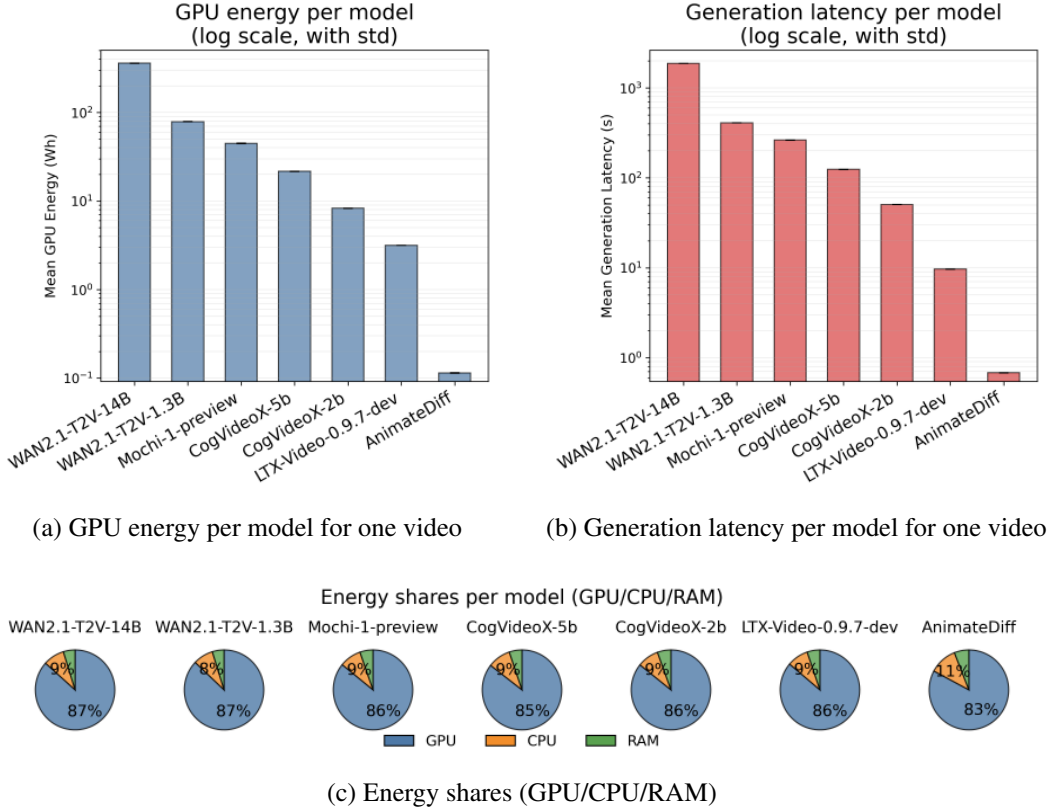(c) Energy shares (GPU/CPU/RAM)

Figure 5: Cross-model comparison of energy and latency. Top: GPU energy and latency (log scale, with std). Bottom: relative contributions of GPU, CPU, and RAM.

We observe orders-of-magnitude disparities: **AnimateDiff** requires only **0.14 Wh** in total, while **WAN2.1-T2V-14B** consumes over **415 Wh**, a factor of nearly $3000\times$. Latency follows a similar trend, with lightweight models producing clips in less than a second, while large-scale architectures such as WAN2.1-14B or Mochi require several minutes of inference. These differences stem from:

- **Model size:** larger models (WAN2.1-14B, Mochi) process more parameters per step.
- **Sampling steps:** AnimateDiff runs in 4 steps vs. 60–64 for others.
- **Video length:** frame count and FPS vary significantly.
- **Architectural complexity:** cascaded pipelines (CogVideoX) require multiple stages.

As shown in the bottom panel, GPU consistently dominates energy consumption ($>80\%$) across all models, confirming a compute-bound regime with high GPU utilization. CPU and RAM contributions remain secondary, though slightly more pronounced in cascaded or multi-stage pipelines.

Table 4: Cross-model average latency and energy consumption (default settings). All values are reported as mean $\pm$ std.

Model	Latency (s)	GPU (Wh)	CPU (Wh)	RAM (Wh)
WAN2.1-T2V-14B	1875 ± 2.1	359.7 ± 0.5	35.6 ± 4.0	19.8 ± 0.02
WAN2.1-T2V-1.3B	410 ± 0.5	78.8 ± 0.1	7.4 ± 0.4	4.3 ± 0.01
Mochi-1-preview	263 ± 0.5	44.7 ± 0.2	4.6 ± 0.01	2.8 ± 0.01
CogVideoX-5B	124 ± 0.4	21.6 ± 0.05	2.4 ± 0.03	1.3 ± 0.004
CogVideoX-2B	50.6 ± 0.2	8.3 ± 0.03	0.84 ± 0.04	0.53 ± 0.002
LTX-Video-0.9.7-dev	9.7 ± 0.01	3.16 ± 0.006	0.32 ± 0.002	0.19 ± 0.001
AnimateDiff	0.68 ± 0.002	0.115 ± 0.001	0.016 ± 0.0001	0.008 ± 0.00003

6   Discussion

Our results confirm that WAN2.1 inference operates in a **compute-bound regime**, where latency and energy scale quadratically with spatial (H, W) and temporal (T) dimensions, and linearly with denoising steps (S). The close match between theory and measurement validates the analytical model and provides clear guidance for practitioners.

Implications for efficiency.   Quadratic scaling in H, W, and T means that even modest increases in resolution or video length incur steep costs: doubling any of these dimensions in isolation yields $\sim 4\times$ more compute, while scaling multiple dimensions compounds multiplicatively (e.g., H and W doubled $\to 16\times$). Thus, *output size control* is a powerful lever: reducing spatial or temporal length often saves more than architectural changes. In practice, offering presets (e.g., "low resolution, low frames" vs. "high fidelity") balances user needs with energy cost.

Validated linear regime in steps.   In contrast, denoising steps scale linearly, with measured costs matching theoretical predictions once empirical efficiency μ is applied. This makes S a reliable knob for latency–quality trade-offs: halving steps roughly halves both latency and energy.

Opportunities for model-level improvements.   The public Hugging Face implementation of WAN2.1 lacks inference-time optimizations, but the original paper suggests effective techniques: (i) *diffusion caching*, reusing redundant attention/CFG activations for up to $1.62\times$ savings, and (ii) *quantization*, using FP8/INT8 mixed precision for $\sim 1.27\times$ speedup without loss. Other avenues include step pruning, low-rank attention, and kernel fusion to better exploit GPU tensor cores.

Broader implications.   Video diffusion is far more costly than text or image generation. Normalized per output, Luccioni et al. [Luccioni et al., 2024] report average costs of ~ 0.002 Wh for text classification, 0.047 Wh for text generation, and 2.9 Wh for image generation. By comparison, generating a single short video with WAN2.1–T2V–1.3B consumes nearly ~ 90 Wh. This places video diffusion roughly $30\times$ more costly than image generation, $2,000\times$ than text generation, and $45,000\times$ than text classification. At scale, the quadratic growth in (H, W, T) implies rapidly increasing hardware and environmental costs, highlighting the need for hardware-aware optimizations and sustainable model design. Theoretical thresholds derived in Appendix B suggest that compute-bound behavior extends to all tested accelerators, reinforcing the generality of our scaling model.

7   Limitations and Conclusion

Limitations.   Our analysis provides a detailed characterization of WAN2.1–1.3B using the open-source Hugging Face codebase. As such, it does not capture potential improvements from internal optimizations such as diffusion caching, quantization, or kernel fusion. The theoretical model also assumes uniform attention cost and ignores memory hierarchy effects, which may cause deviations for small inputs or extreme aspect ratios.

Energy measurements were conducted on a single hardware platform (NVIDIA H100 SXM). While Appendix B shows that the compute-bound regime and associated scaling trends should extend

to other accelerators for realistic token lengths, this remains to be confirmed experimenta. We deliberately excluded perceptual quality from our scope, leaving open the question of energy–fidelity tradeoffs. Finally, many production T2V systems (e.g., Veo) also generate audio, whose contribution to energy cost remains unexplored.

Conclusion. We presented a systematic study of latency and energy consumption in text-to-video generation. Through fine-grained experiments on WAN2.1, we validated a simple analytical model that predicts quadratic scaling with spatial and temporal dimensions, and linear scaling with denoising steps. Cross-model benchmarks confirmed that this compute-bound regime extends broadly across recent open-source systems, with orders-of-magnitude disparities in cost depending on model size, sampling strategy, and video length.

These findings highlight both the structural inefficiency of current video diffusion pipelines and the urgent need for efficiency-oriented design. Promising avenues include diffusion caching, low-precision inference, step pruning, and improved attention mechanisms. We hope this work serves as both a benchmark reference and a modeling framework to guide future research on sustainable generative video systems.

References

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL `https://openai.com/research/video-generation-models-as-world-simulators`.

Benoit Courty, Victor Schmidt, Sasha Luccioni, Goyal-Kamal, MarionCoutarel, Boris Feld, Jérémy Lecourt, LiamConnell, Amine Saboni, Inimaz, supatomic, Mathilde Léval, Luis Blanche, Alexis Cruveiller, ouminasara, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, Niko Laskaris, Edoardo Abati, Douglas Blank, Ziyao Wang, Armin Catovic, Marc Alencon, Michał Stęchły, Christian Bauer, Lucas Otávio N. de Araújo, JPW, and MinervaBooks. mlco2/codecarbon: v2.4.1, May 2024. URL `https://doi.org/10.5281/zenodo.11171501`.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

DeepMind. Veo: Advanced video generation model. `https://storage.googleapis.com/deepmind-media/veo/Veo-3-Tech-Report.pdf`, 2025. Accessed: 2025-08-08.

Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning, 2024. URL `https://arxiv.org/abs/2307.04725`.

Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. Chasing carbon: The elusive environmental footprint of computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 854–867. IEEE, 2021.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. Ltx-video: Realtime video latent diffusion, 2024. URL `https://arxiv.org/abs/2501.00103`.

Johannes Hagemann, Samuel Weinbach, Konstantin Dobler, Maximilian Schall, and Gerard de Melo. Efficient parallelization layouts for large-scale distributed model training, 2024. URL `https://arxiv.org/abs/2311.05610`.

Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, Yulu Jia, Sun He, Hongmin Chen, Zhihao Bai, Qi Hou, Shipeng Yan, Ding Zhou, Yiyao Sheng, Zhuo Jiang, Haohan Xu, Haoran Wei, Zhang Zhang, Pengfei Nie, Leqi Zou, Sida Zhao, Liang Xiang, Zherui Liu, Zhe Li, Xiaoying Jia, Jianxi Ye, Xin Jin, and Xin Liu. Megascale: Scaling large language model training to more than 10,000 gpus, 2024. URL `https://arxiv.org/abs/2402.15627`.

Baolin Li, Yankai Jiang, and Devesh Tiwari. Carbon in motion: Characterizing open-sora on the sustainability of generative ai for video generation. *ACM SIGENERGY Energy Informatics Review*, 4(5):160–165, 2024.

Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model, 2022. URL `https://arxiv.org/abs/2211.02001`.

Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of ai deployment? In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, page 85–99. ACM, June 2024. doi: 10.1145/3630106.3658542. URL `http://dx.doi.org/10.1145/3630106.3658542`.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on GPU clusters. *CoRR*, abs/2104.04473, 2021. URL `https://arxiv.org/abs/2104.04473`.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training, 2021. URL `https://arxiv.org/abs/2104.10350`.

Jessica Pavani, Rosangela Helena Loschi, and Fernando Andres Quintana. Modeling temporal dependence in a sequence of spatial random partitions driven by spanning tree: an application to mosquito-borne diseases, 2025. URL `https://arxiv.org/abs/2501.04601`.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053, 2019. URL `http://arxiv.org/abs/1909.08053`.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. *CoRR*, abs/1906.02243, 2019. URL `http://arxiv.org/abs/1906.02243`.

Genmo Team. Mochi 1. `https://github.com/genmoai/models`, 2024.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL `https://arxiv.org/abs/2503.20314`.

Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, April 2009. ISSN 0001-0782. doi: 10.1145/1498765.1498785. URL `https://doi.org/10.1145/1498765.1498785`.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable AI: Environmental implications, challenges and opportunities. *Proceedings of machine learning and systems*, 4: 795–813, 2022.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer, 2025. URL `https://arxiv.org/abs/2408.06072`.

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

A Detailed FLOP Derivations and Scaling Laws

Conventions. We follow the HPC convention where one multiply–add equals two FLOPs. Matrix multiplications of shape $(a \times b) \cdot (b \times c)$ therefore cost $2abc$ FLOPs. Bias additions, activations, layer norms, and softmax are lower order and omitted unless stated. All results below apply per forward pass.

Table 5: Complete set of WAN2.1-T2V-1.3B hyperparameters and constants. This table provides the full notation, including VAE layer-wise symbols (instantiated explicitly in Appendix A.8).

Symbol	Value	Meaning
Global video parameters		
T	variable	Number of frames
$H \times W$	variable	Input spatial resolution
S	variable	Number of denoising steps
g	2	CFG passes per step (cond + uncond)
v_t, v_s	4, 8	Temporal and spatial downsampling factors of the VAE
p_h, p_w	2, 2	Spatial patch size in the DiT latent grid
Diffusion Transformer (DiT)		
N	32	Number of DiT layers
d	2048	Hidden size
f	4	MLP expansion factor ($8192 = 4d$)
ℓ	$(1 + \frac{T}{4})\frac{H}{16}\frac{W}{16}$	Token length of latent grid
Text encoder (T5-XXL)		
m	512	Output tokens per video (conditioning length)
p_{text}	2	Calls per video (cond + uncond)
d_{text}	4096	Hidden size
L_{text}	24	Encoder layers
f_{text}	2.5	MLP expansion factor
Timestep embedding		
d_τ	256	Hidden width of timestep MLP
VAE (layer-wise; values in App. A.8)		
j	$1, \ldots, N_{\text{dec,conv}}$	Layer index along the VAE **decoder** path
$N_{\text{dec,conv}}$	11	Number of 3D conv layers in the VAE decoder
$k_t^{(j)}, k_h^{(j)}, k_w^{(j)}$	–	3D kernel sizes of decoder layer j
$C_{\text{in}}^{(j)}, C_{\text{out}}^{(j)}$	–	In/out channels at decoder layer j
$T^{(j)}, H^{(j)}, W^{(j)}$	–	Output grid sizes at decoder layer j
C_*	384	Channel width at middle attention block
T_*, H_*, W_*	$\lceil T/4 \rceil, H/8, W/8$	Grid sizes at middle resolution
L_*	$H_* W_*$	Spatial token length per frame (2D middle attention)
Hardware / efficiency constants		
μ	0.456	Empirical efficiency (fraction of Θ_{peak})
Θ_{peak}	989×10^{15} FLOP/s	Peak GPU throughput (H100)
P_{max}	700 W	Sustained GPU power
D_{total}	$F_{\text{total}}/(\mu \Theta_{\text{peak}})$	Total latency

A.1 Latent Tokenization and Shapes

Let the video have T frames and spatial size $H \times W$ in pixels. The VAE downsamples temporally by a factor v_t and spatially by v_s, and the DiT operates on spatial patches of size $p_h \times p_w$ in the latent

grid. The token length ℓ seen by the DiT is

$$\ell = \left(1 + \frac{T}{v_t}\right) \frac{H}{v_s\, p_h} \frac{W}{v_s\, p_w}. \tag{1}$$

In WAN2.1 we use $(v_t, v_s, p_h, p_w) = (4, 8, 2, 2)$, hence the shorthand $\ell = (1 + \frac{T}{4}) \frac{H}{16} \frac{W}{16}$ used in the main text.

A.2 Self-Attention in the DiT

Let d be the model width and h the number of heads (with $d_h = d/h$). For a sequence of length ℓ:

$$
\begin{aligned}
\text{Q,K,V projections:} \quad & 3 \times 2\,\ell d^2 \;=\; 6\,\ell d^2 \\
\text{Attention logits } (QK^\top): \quad & 2\,\ell^2 d \\
\text{Weighted sum } (AV): \quad & 2\,\ell^2 d \\
\text{Output projection:} \quad & 2\,\ell d^2 .
\end{aligned} \tag{2}
$$

Summing on all N DiT layers yields

$$F_{\text{self}} = N \times (8\,\ell d^2 + 4\,\ell^2 d). \tag{3}$$

(The head count h cancels out, since $h \cdot d_h = d$.)

A.3 Cross-Attention (Video $\rightarrow$ Text)

Let m be the number of text tokens and d the shared width. Assuming no KV cache (K,V recomputed each denoising step as it is done in the current official implementation) and one cross-attention block per DiT layer:

$$
\begin{aligned}
\text{Query from video:} \quad & 2\,\ell d^2 \\
\text{Keys/values from text:} \quad & 4\,m d^2 \quad \text{(K and V)} \\
\text{Attention products:} \quad & 2\,\ell m d + 2\,\ell m d = 4\,\ell m d \\
\text{Output projection:} \quad & 2\,\ell d^2 .
\end{aligned} \tag{4}
$$

Hence over the N layers

$$F_{\text{cross}} = N \times (4\,\ell d^2 + 4\,m d^2 + 4\,\ell m d). \tag{5}$$

With KV caching, the $4md^2$ term becomes once-per-video while the $4\ell m d$ products remain per step. *With windowed or factorized attention*, ℓ or m may be replaced by the effective window size.

A.4 Transformer MLP

With expansion factor f and sequence length ℓ, a two-layer MLP $d \rightarrow fd \rightarrow d$ costs over all DiT layers

$$F_{\text{mlp}} = N \times 4f\,\ell d^2 . \tag{6}$$

A.5 Stacking Across S Steps, and CFG

Let g denote the number of conditional forward passes (CGF) per denoising step ($g = 2$ under classifier-free guidance). Combining (3)–(6), the DiT cost is

$$F_{\text{DiT}}(T, H, W; S, N, d, f, m, g) = g\, S \times \left(F_{\text{self}} + F_{\text{cross}} + F_{\text{mlp}} \right), \tag{7}$$

with ℓ given by (1).

A.6 Text Encoder

For a L_{text}-layer encoder (e.g., T5/CLIP-like) with width d_{text}, expansion f_{text}, and m tokens:

$$
\begin{aligned}
\text{Self-attn per layer:} \quad & 8\,m d_{\text{text}}^2 + 4\,m^2 d_{\text{text}} \\
\text{FFN per layer:} \quad & 4 f_{\text{text}}\, m d_{\text{text}}^2 .
\end{aligned} \tag{8}
$$

For p_{text} forward passes per video (e.g., $p_{\text{text}} = 2$ for conditional and unconditional prompts),

$$F_{\text{text}} = p_{\text{text}}\, L_{\text{text}} \left(8\,m d_{\text{text}}^2 + 4\,m^2 d_{\text{text}} + 4 f_{\text{text}}\, m d_{\text{text}}^2 \right). \tag{9}$$

This term is once-per-video, independent of S.

A.7 Timestep Embedding MLP

Mapping a scalar diffusion step to a d-dim vector and injecting it into each block via a small MLP with hidden width d_τ:

$$F_\tau = gS(2\,d_\tau\,d\ +\ 14\,d^2).\tag{10}$$

A.8 VAE: Convolutions and Middle Attention

We account for the VAE cost as the sum of (i) all convolutional layers along the decoder and (ii) a 2D self-attention "middle" block evaluated independently per time slice.

Convolutional layers. For a 3D convolution with kernel $(k_t^{(j)}, k_h^{(j)}, k_w^{(j)})$, channels $C_{\text{in}}^{(j)} \to C_{\text{out}}^{(j)}$ and output size $T^{(j)} \times H^{(j)} \times W^{(j)}$, the cost is

$$F_{\text{conv3d}}^{(j)} = 2\,k_t^{(j)}k_h^{(j)}k_w^{(j)}\,C_{\text{in}}^{(j)}C_{\text{out}}^{(j)}\,T^{(j)}H^{(j)}W^{(j)}\,.\tag{11}$$

Summing over the decoder path gives $F_{\text{VAE,conv}} = \sum_{j=1}^{N_{\text{dec,conv}}} F_{\text{conv3d}}^{(j)}$, with concrete per-layer shapes provided in Table 6. WAN-2.1 VAE include a 2D self-attention middle block evaluated independently on each time slice ($L_* = H_*W_*$, channel width C_*):

$$F_{\text{VAE,mid-attn}} = T_*\big(8\,C_*^2 L_*\ +\ 4\,L_*^2 C_*\big).\tag{12}$$

Middle self-attention (2D, per time slice). Let C_* be the channel width at the middle resolution, and T_*, H_*, W_* the temporal/spatial sizes (thus $L_* = H_*W_*$ tokens per time slice). Using the derivation in Appendix A.2, the middle attention cost is

$$F_{\text{VAE,mid-attn}} = T_*\big(8\,C_*^2 L_*\ +\ 4\,L_*^2 C_*\big),\tag{13}$$

where the final $2C_*^2 L_*$ term arises from the output projection and is included in the $8C_*^2 L_*$ term above.

WAN2.1 decoder instantiation (values). In WAN2.1, the VAE decoder starts from a latent grid $(T_0, H_0, W_0) = \big(\lceil T/4 \rceil, H/8, W/8\big)$ with $z=16$ channels. A causal $3\times3\times3$ convolution expands this to 384 channels, followed by a "middle" block consisting of two residual $3\times3\times3$ convolutions and a 2D self-attention layer applied independently per time slice. The decoder then progressively upsamples: two *temporal+spatial* upsamplings (doubling T, H, W and halving channels), followed by one purely *spatial* upsampling (doubling H, W and halving channels). Residual blocks (three per stage) refine features at each resolution, and a final $3\times3\times3$ convolution produces the RGB output at (T, H, W).

Table 6 summarizes the dominant operators for FLOP accounting. Applying Eq. (11) across these layers yields $F_{\text{VAE,conv}}$, while Eq. (13) gives the middle-attention cost.

A.9 Total FLOPs and Leading-Order Scaling

We finally obtain

$$F_{\text{total}}(H, W, T, S) = F_{\text{text}} + F_{\text{VAE,conv}} + F_{\text{VAE,mid-attn}} + F_\tau + F_{\text{DiT}}\,,\tag{14}$$

with components given by (9), (11), (13), (10), and (7). Since ℓ grows linearly with H, W, and T (Eq. 1), the $\ell^2 d$ and $\ell m d$ terms in F_{DiT} dominate for typical settings ($\ell \gg m$), yielding quadratic growth in H, W, and T, and linear growth in S.

Scope and caveats. (i) FlashAttention and fused kernels reduce memory traffic and constants but do not change FLOP counts. (ii) KV caching changes only the cross-attention $4md^2$ term from per-step to once-per-video. (iii) Windowed or factorized attention replaces ℓ (or m) by an effective window size, altering quadratic scaling. (iv) If activations or norms become bandwidth-bound, the proportionality between FLOPs and latency weakens; our WAN2.1 measurements on H100 indicated compute-bound behavior over the operating points considered.

Table 6: VAE decoder: representative dominant operators for FLOP accounting (layer j). It mirrors the encoder; $z{=}16$, $C_*{=}384$, middle resolution ($\lceil T/4 \rceil, H/8, W/8$).

Stage j	Op type	Kernel (k_t, k_h, k_w)	$C_{\text{in}}^{(l)} \to C_{\text{out}}^{(l)}$	$T^{(l)}$	$H^{(l)}$	$W^{(l)}$
D0	conv3d	$(3,3,3)$	$z \to 384$	$\lceil T/4 \rceil$	$H/8$	$W/8$
Middle (RBs)	conv3d	$(3,3,3)$	$384 \to 384$	$\lceil T/4 \rceil$	$H/8$	$W/8$
Middle (attn 2D)	attn-2D	–	$384 \to 384$	$\lceil T/4 \rceil$	$H/8$	$W/8$
D1 (RBs)	conv3d	$(3,3,3)$	$384 \to 384$	$\lceil T/4 \rceil$	$H/8$	$W/8$
Up (time)	conv3d (time)	$(3,1,1)$	$384 \to 2{\times}384$	$\lceil T/2 \rceil$	$H/8$	$W/8$
Up (space)	conv2d (space)	$(1,3,3)$	$384 \to 192$	$\lceil T/2 \rceil$	$H/4$	$W/4$
D2 (RBs)	conv3d	$(3,3,3)$	$192 \to 384$	$\lceil T/2 \rceil$	$H/4$	$W/4$
Up (time)	conv3d (time)	$(3,1,1)$	$384 \to 2{\times}384$	T	$H/4$	$W/4$
Up (space)	conv2d (space)	$(1,3,3)$	$384 \to 192$	T	$H/2$	$W/2$
D3 (RBs)	conv3d	$(3,3,3)$	$192 \to 192$	T	$H/2$	$W/2$
Up (space)	conv2d (space)	$(1,3,3)$	$192 \to 96$	T	H	W
Head	conv3d	$(3,3,3)$	$96 \to 3$	T	H	W

B   Theoretical Compute-Bound Thresholds for DiT Blocks

We estimate the arithmetic intensity (FLOP per byte transferred between HBM and registers) for the main operations in DiT: the self-attention block (with FlashAttention) and the MLP. We then derive the compute-bound threshold $\ell^\star$ at which the operation's intensity matches the hardware balance $\beta = \Theta_{\text{peak}}/B$.

Let s be the byte size of a scalar (e.g., $s = 2$ for BF16), and assume a fully optimized implementation that reads inputs and writes outputs only once from HBM, so each tensor contributes twice to memory traffic (read + write).

FlashAttention (forward).   We include only the matrix multiplications $QK^\top$ and PV (not projections). The total FLOPs scale as $F_{\text{attn}} = 4\ell^2 d$, and total memory transfer as $D_{\text{attn}} = 2\ell ds$ (read inputs Q, K, V and write output of size ℓd).

$$\text{AI}_{\text{attn}}(\ell) = \frac{F_{\text{attn}}}{D_{\text{attn}}} = \frac{4\ell^2 d}{2\ell ds} = \frac{2\ell}{s} \quad \Rightarrow \quad \ell^\star_{\text{attn}} = \frac{s\beta}{2}$$

MLP block (GEMM)   The total FLOPs are $F_{\text{mlp}} = f\ell d^2$, and the memory transfer is $D_{\text{mlp}} = (fd^2 + \ell d + f\ell d)s$.

$$\text{AI}_{\text{mlp}}(\ell) = \frac{F_{\text{mlp}}}{D_{\text{mlp}}} = \frac{f\ell d}{(fd + \ell(1+f))s} \quad \Rightarrow \quad \ell^\star_{\text{mlp}} = s\beta$$

For $d = 2048$, $s = 2$, and $\beta = 295$ (H100 BF16), we find:

$$\ell^\star_{\text{attn}} = \frac{2 \cdot 295}{2} = \mathbf{295}, \quad \ell^\star_{\text{mlp}} = 2 \cdot 295 = \mathbf{590},$$

Thus, all MLP are compute-bound for $\ell > 590$, and attention becomes compute-bound for $\ell > 290$. In our WAN2.1 runs, $\ell \gg 10^4$, so both blocks operate far in the compute-bound regime.

Caveat.   These thresholds assume peak theoretical performance. In practice, we observe an empirical efficiency $\mu \approx 0.4$ for compute throughput on the H100. Similarly, the effective memory throughput often remains well below B due to irregular access patterns and latency bottlenecks.

Other hardware.   Table 7 reports β and the corresponding compute-bound thresholds for both attention and MLP blocks across a range of accelerators.

Table 7: Approximate FLOP-to-bandwidth ratios ($\beta = \Theta_{\text{peak}}/B$) and corresponding compute-bound thresholds $\ell^\star$ for DiT blocks (BF16).

Accelerator	Θ_{peak} (TFLOP/s)	B (TB/s)	β (FLOP/byte)	$\ell^\star_{\text{attn}} / \ell^\star_{\text{mlp}}$
NVIDIA H100 SXM	989	3.35	295	295 / 590
NVIDIA A100 SXM	312	2.0	156	156 / 312
RTX 4090	330	1.0	330	330 / 660
NVIDIA L4	121	0.3	605	605 / 1210
TPU v6	918	1.6	574	574 / 1148
AMD M3250X	2500	6.0	417	417 / 834
Intel Gaudi3	1678	3.7	453	453 / 906

All realistic settings in WAN2.1 yield $\ell \gg 10^4$, even for low-resolution and short-duration inputs. Thus, both MLP and attention blocks operate well beyond the compute-bound threshold on all tested accelerators.