

Drishya Tiny: building a small video generator from scratch, failures included

UDGAM LABS VERSION 1.0

Evidence labels: [MEASURED] our own runs or bills; [ESTIMATE] our projections; [EXTERNAL] other people's papers, cards or leaderboards, not reproduced by us; [EXPLORATION] ideas and options. A label covers the sentence, list or table it closes or heads.

Summary

Drishya Tiny by Udgam Labs is a 212M-parameter video generator trained from scratch, using the frozen Wan2.2 VAE and Flan-T5-Base text encoder. It turns a text prompt, or a text prompt plus a first frame, into a 2-second 256×256 clip. This report is the lab notebook behind it: five stages, what each one was for, what it measured, what failed and what it cost.

- **What it is.** A 211.92M-parameter diffusion transformer, trained from random initialisation with rectified flow on 475,157 two-second windows of open-domain web video (VidGen-1M, whose clips come from HD-VILA-100M). The text encoder (109.6M) and the VAE (704.7M) are borrowed and stay frozen; the whole text-to-video system is about 877M parameters at inference. The release run took 15 hours on one L40S. [MEASURED]
- **Video quality is close to larger models; prompt-following is about half.** On VBench, video quality is 75.6, level with the 256px VideoCrafter-0.9 (74.9) and close to ModelScope (78.1), both built on Stable Diffusion; our generator is about a sixth the size of ModelScope's 1.35B UNet. Prompt-following (semantic) is 32.3, about half theirs (65-67). Our numbers are our own sampling and scoring with the official VBench code, not a certified leaderboard entry. [MEASURED; peers EXTERNAL]
- **The controlled finding.** At equal compute, fixing the captions raised prompt-following by 3.0 points (a paired measure, Section 9) at no quality cost; adding photos as well raised it by 4.3 against the same control. The photos' own gain was in *what* things are (object class, scene) and none in composition (multiple objects, spatial relations), at a cost of +104 held-out FVD. Data fixes at this size move prompt-following by 3-4 points, not 30. [MEASURED]
- **What it cost.** About \$21 of GPU time for the release model's own path (data preparation \$7.40, training \$13.76), and \$57.43 for everything across 55 rented-machine rows, every failed attempt included. [MEASURED]
- **What did not work.** None of three training techniques beat the plain recipe at equal compute; our MeanFlow implementation diverged; 32 of the 55 machine rows in our cost ledger were failed or stopped attempts; we first trusted a dataset's Apache tag that its source clips do not support. All of it is in this report and in the published cost ledger. [MEASURED]
- **Where it falls short.** It almost never draws two named objects (2.9%), faces and hands smear, text melts, image-to-video drifts within the 2 seconds, and the weights are non-

commercial because the training clips are research-only. [MEASURED]

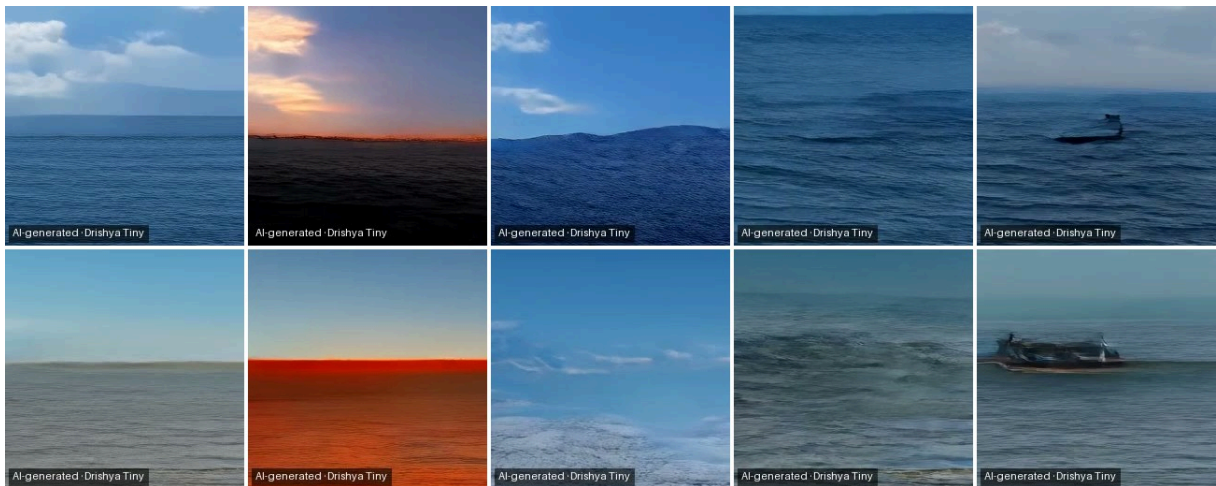


FIGURE 1. ONE FRAME FROM EACH OF TEN TEXT-TO-VIDEO CLIPS MADE BY THE RELEASE MODEL: FIVE OF OUR OWN PROMPTS, TWO SEEDS EACH, NOTHING ELSE PICKED.

1. Why a tiny video model

Video generators are usually large and expensive to train. We asked a smaller question: with one rented GPU and a budget that fits on a credit card, can a generator trained from random weights make clips worth watching? And once it works, what actually moves its ability to follow a prompt?

We did not try to train everything. A video model has three jobs: reading the prompt, compressing video into a small grid of numbers (and back), and generating that grid from noise. The first two have good open parts: Flan-T5-Base reads the prompt, and the Wan2.2 VAE compresses 17 frames of 256×256 video into a $48 \times 5 \times 16 \times 16$ latent. We kept both frozen and trained only the generator in between. That split is what makes the budget possible, and it is why every sentence in this report says "generator trained from scratch", never "model trained from scratch".

Drishya Tiny is a research release. Before release, our rules ask for a measured edge over the newest models of the same size. A fresh scan on 25 October 2026 found no claim we could make on quality, and one we had planned on cost was false: MiniDance, released on 14 October, trained a scenery-only 120M generator for about \$2 (Section 13) [EXTERNAL]. So we release Drishya Tiny for what it is: a reproducible small-budget baseline with its code, logs, evaluations, a controlled ablation and the full cost of getting there.

The name: Drishya (दृश्य) means "a scene, something seen".

2. What we release

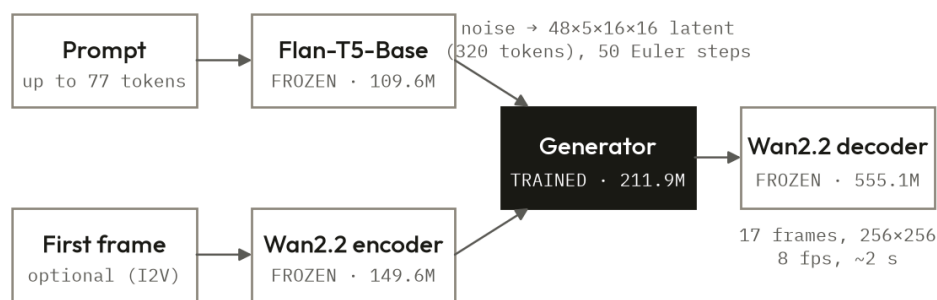
ITEM	WHAT IT IS	LICENCE	WHERE
Drishya-Tiny	The release model: Stage 3's final EMA weights (step 180,080) as safetensors, with the config, and a browser version in <code>onnx/</code>	CC BY-NC 4.0	huggingface.co/udgamlabs/Drishya-Tiny
Code	The generator, the trainer of every stage, data preparation, FVD and VBench evaluation, the Stage 5a decision script, the inference package, the browser version (text and photo)	Apache-2.0	github.com/udgamlabs/drishya-tiny
Results and logs	Every stage's results, its training logs, VBench's per-dimension output and <code>COST.md</code> , every machine we rented	Apache-2.0 (with the code)	the same repository, <code>results/</code>
Clip IDs	The VidGen-1M clip IDs and window starts used for training and held out, with the preparation scripts; no clips, images or captions	—	the same repository, <code>data/</code>
Space	A web demo on a GPU	—	huggingface.co/spaces/udgamlabs/Drishya-Tiny
Model page	The results, clips, and a version that runs in the visitor's browser	—	udgamlabs.com/models/drishya-tiny

The released weights are the EMA weights of the training checkpoint whose sha256 is listed in the release facts; the checkpoint also holds the raw weights and the optimiser state, which do not ship. [MEASURED]

3. The system

Drishya Tiny at inference

One part trained from random weights; the rest borrowed and frozen



PARAMETER COUNTS FROM THE SAFETENSORS HEADERS · UD GAM LABS, OCT 2026

FIGURE 2. WHAT RUNS WHEN YOU MAKE A CLIP. THE DARK BOX IS THE ONLY PART WE TRAINED; PARAMETER COUNTS FROM THE SAFETENSORS HEADERS.

PART	PARAMETERS	TRAINED BY US?
Generator: VideoDiT "s210" (width 768, depth 16, 12 heads, patch 1×2×2)	211.92M	Yes, from random initialisation
Text encoder: Flan-T5-Base encoder (with its token embedding)	109.63M	No, frozen
Wan2.2 VAE (from Wan-AI/Wan2.2-TI2V-5B-Diffusers)	704.69M: encoder 149.63M, decoder 555.05M	No, frozen

[MEASURED, counted from tensor shapes in the safetensors headers]

Text-to-video at inference uses the generator, the T5 encoder and the VAE decoder: 876.6M parameters. Image-to-video adds the VAE encoder for the given frame: 1,026.2M. [MEASURED]

3.1 Latents and tokens

A clip is 17 frames at 8 frames per second, about 2 seconds, at 256×256. The Wan2.2 VAE compresses it 16× in each spatial direction and 4× in time, to 48 channels × 5 latent frames × 16 × 16. With a patch of 1 × 2 × 2 that is **320 tokens per clip**: each token stands for a 32 × 32-pixel square over about four frames. [MEASURED]

We chose the Wan2.2 VAE on day 0, on the Mac, for \$0. On 16 real clips it reconstructed video slightly better than the Wan2.1 VAE (33.99 against 33.69 dB PSNR) while giving the generator 4× fewer tokens (576 against 2,304 at the 33-frame length we tested) [MEASURED]. Fewer tokens

is the single biggest lever on compute for a transformer, and it is also the main limit on detail (Section 12).

3.2 The generator

The generator extends our Chitra Tiny image transformer to video. Each of its 16 blocks has:

1. self-attention over all 320 tokens, with adaLN-zero modulation and QK-norm (RMSNorm on queries and keys);
2. cross-attention to the Flan-T5 token states of the prompt;
3. an MLP (4× width, GELU-tanh) with adaLN modulation.

Positions use a fixed 3D sine-cosine embedding over time, height and width. The adaLN modulation is driven by the timestep embedding plus the mean-pooled prompt embedding, so the prompt reaches every block twice: once as a summary, once token by token. Prompts are cut at 77 Flan-T5 tokens. [MEASURED, `training/vmodel.py`]

3.3 Rectified flow

Training follows rectified flow (flow matching). For a clean latent x_0 and Gaussian noise ϵ , the model sees the straight-line mix

$$x_t = (1 - t) \cdot x_0 + t \cdot \epsilon$$

and learns to predict the velocity $\epsilon - x_0$, with a mean-squared error. The time t is drawn as the sigmoid of a standard normal (logit-normal), which spends more of training at middle noise levels. Sampling runs Euler steps from $t = 1$ (noise) to $t = 0$ (a clip), on a time grid warped by a shift s : $t' = s \cdot t / (1 + (s - 1) \cdot t)$, which spends more steps at high noise. Classifier-free guidance comes from dropping the caption on 10% of training steps; at sampling, the model is run with and without the prompt and the difference is scaled by the guidance weight (CFG). The release defaults are 50 Euler steps, CFG 5 and shift 5 (Section 8.2). [MEASURED, `training/train_t2v.py`]

3.4 Image-to-video by first-frame replacement

One model does both text-to-video and image-to-video. On 25% of training steps, the clean latent of the clip's first frame replaces the noisy one, a learned embedding marks its tokens, and its loss is masked out, so the model learns to continue a given frame (the frame-replacement recipe of STIV and SANA-Video [EXTERNAL]). At inference the given frame is re-imposed at every step and kept as given. Without a frame, the same model does text-to-video. [MEASURED]

4. Data

Source. VidGen-1M (arXiv 2408.02629), a curated, captioned set of web video clips taken from Microsoft's HD-VILA-100M [EXTERNAL]. We used zips 0-599, about 300K clips, and held out

zips 2040-2047, never trained on. [MEASURED]

Windows. From each clip we cut a 2-second window at the start and, for long clips, a second one mid-clip, and encoded each to Wan2.2 latents once. The trainer loaded **475,157 training windows**; the data step's own log counted 474,356 when it finished, and we use the number the trainer actually loaded. The held-out set gave 2,048 scored clips and 3,918 validation windows. [MEASURED]

Captions. VidGen-1M's own captions, about 80 words on average; the model reads the first 77 tokens. No photos, no synthetic data and no other video set are in the release model. [MEASURED]

Licence, and a mistake. HD-VILA-100M's terms allow research use only, so the weights are released under CC BY-NC 4.0, and the clips themselves are not redistributed: the repository lists clip IDs and window starts, with the preparation scripts. When we chose the data, we went by the uploader's Apache-2.0 tag on VidGen-1M and treated the licence as no constraint. Reading the original source's terms later showed the weights could not be Apache. The lesson we now apply: read the licence of the source clips, not the tag of the copy. [MEASURED]

The data step. Encoding 600 zips took about 6.2 hours on one RTX 5090 (\$7.40). Five attempts failed before it: three machines never became reachable, one could not reach PyPI, and one ran at only about 5 windows per second and was stopped for an encoder fix (\$0.80). At this scale, preparing video data cost more than half as much as training on it (Section 10). [MEASURED]

5. Day 0 and Stage 1: checking the method on skies

Day 0 (\$0, on the Mac). Three checks before any spend: the VAE comparison above; a one-batch overfit, where the generator memorised 8 of 8 clips (a learnability check: a falling loss alone proves little); and the FVD harness. FVD (Fréchet Video Distance) compares features of generated and real clips; lower is better. Our implementation matched the published method on test distributions, but showed two known weaknesses. It reacts weakly to motion: freezing every clip on its first frame raised FVD only 1.74× over the real-against-real baseline, while blurring with $\sigma = 4$ raised it 2.4×. And it is biased at small sample sizes: two halves of the same real set score 669 at 200 clips each. So every final score uses 2,048 clips per side, and every FVD comes with a **motion ratio** (generated motion over real motion) and, once there are prompts, a **CLIP gap** (how much better a clip matches its own caption than a shuffled one). [MEASURED]

Stage 1 (18 October, \$7.14). Before paying for open-domain data, we checked that the recipe learns at all, on SkyTimelapse, the standard unconditional FVD benchmark: 1,965 training clips, 36,481 windows, no prompts. Same 211.9M generator, two hours on one RTX 5090: 43,449 steps at batch 64, 2.78M clip samples, 386 clips per second. [MEASURED]

CHECK (WRITTEN BEFORE THE RUN)	RULE	RESULT
P1	FVD-2048 below 200	167.7 (pass)
P2	below real frames frozen still (215.1; real against real 72.2)	167.7 (pass)
P3	motion ratio 0.5-2.0	0.89 (pass)
P4	held-out loss at the end below that at 10%	0.3565 against 0.3608 (pass)

[MEASURED, results/stage1/final.json]

At sampling shift 3 the same model scores 160.7, and animating a real first frame scores FVD-512 139.0. For context only, published SkyTimelapse numbers in another harness are 59.82 for Latte-XL and 79.52 for StyleGAN-V [EXTERNAL, Latte paper]; our FVD code was later shown to match StyleGAN-V's official code (Section 7.1), but the protocols still differ, so we draw no comparison.

What it told us. The held-out loss was best at 50% of the run (0.3535) and worse at 75% (0.3542). The model had seen each window about 76 times: it was limited by data, not compute. That shaped the next two stages. [MEASURED]

What went wrong. Stage 1 cost \$7.14 against about \$3.20 planned [ESTIMATE in the plan]. Community GPUs had no stock, so the run used Secure at \$1.19 an hour instead of \$0.69. Computing statistics of the real videos took 50 minutes instead of about 10, because 16 loader threads each spawned PyTorch's own thread pool on a 120-core machine (fix: one PyTorch thread per loader thread). On that host the VAE failed our bf16 precision rule, so evaluation decoded in fp32 at 0.92 seconds a clip instead of 0.42. Two bad hosts and our own threshold bug cost about \$0.30 more. [MEASURED]

6. Stage 2: three techniques against the plain recipe

Papers report several ways to train diffusion transformers faster or better. We tested three of them against the plain recipe on SkyTimelapse, four arms in all, each given exactly 60 GPU-minutes on one RTX 5090 and scored by FVD-2048 (shift 3, 50 Euler steps, EMA weights). To know what counts as a difference, we sampled the control twice with different seeds: 160.7 and 148.1. They differ by 12.6, so the pre-registered margin was twice that, ± 25.1 FVD. [MEASURED]

ARM	CLIPS PER SECOND	CLIPS SEEN	FVD-2048	MOTION	HELD-OUT LOSS	VERDICT
A0, plain recipe (control)	383	1.38M	160.7	0.88	0.3551	control
A1, + joint image and video	320	1.15M	182.4	0.90	0.3621	tie (inside the margin)
A3, + TREAD (50% token routing)	532	1.92M	171.2	0.95	0.3602	tie (inside the margin)
A2, MeanFlow (few-step)	125	0.45M	4,420 at 4 steps	3.9 (noise)	0.3811	diverged

[MEASURED, `results/stage2/summary.json` ; Stage 2 cost \$7.20]

Stage 2: no technique beat the plain recipe

FVD-2048, lower is better · SkyTimelapse · 60 GPU-minutes each · margin ± 25.1

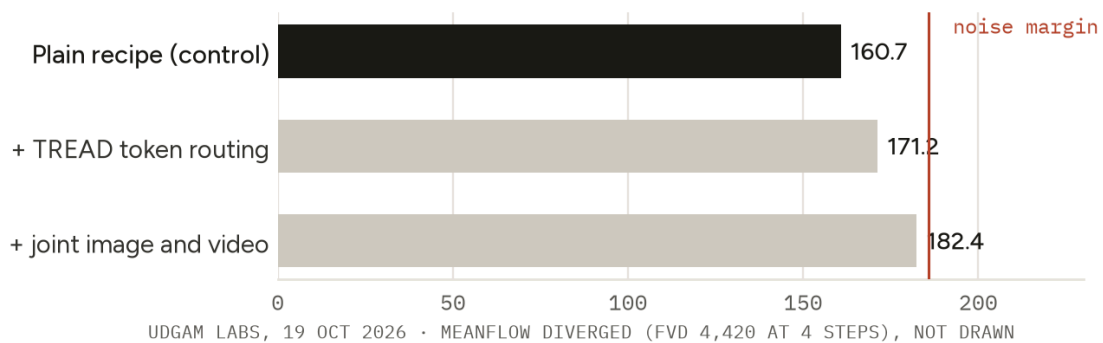


FIGURE 3. FVD-2048 OF THE THREE ARMS THAT TRAINED NORMALLY; THE RED LINE IS THE CONTROL PLUS THE NOISE MARGIN. MEANFLOW IS OFF THE SCALE.

No technique beat the plain recipe, so the plain recipe went into the release run unchanged. In detail:

- The control at 60 minutes matched Stage 1 at 120 minutes (160.7 both, at shift 3). More compute on the same 1,965 clips bought nothing, which confirmed the data limit and also exposed a flaw in the stage itself: a bake-off of compute-saving techniques cannot show a gain on a proxy that is already data-limited. We would now run such a comparison after the open-domain data exists. [MEASURED]
- TREAD routes half the tokens past the middle blocks. It gave 39% more throughput but no quality gain at equal time. At 320 tokens per clip, attention is a small share of the compute, so skipping tokens saves time without helping; it may matter for longer or larger clips. [MEASURED; EXPLORATION for the reason]

- **Joint image and video training** did not reproduce the gain reported in Latte's ablation at our scale and budget; the image batches cost 17% of video throughput. [MEASURED]
- **Our MeanFlow implementation diverged. This is not a verdict on the method.** MeanFlow trains the model to predict the average velocity over an interval, so that one to four steps suffice. The logged average-velocity error climbed from about 1.6 to 300-1,300; the adaptive loss weight $1/(\text{error} + c)$ then silenced those samples, so the few-step path never learned, and 1-4-step samples were noise (FVD 6,771 at 1 step, 11,241 at 2, 4,420 at 4). The plain-flow part of the same arm learned normally for its smaller sample count (FVD-512 289, motion 0.97). For scale, the control sampled with only 4 Euler steps gives FVD-512 294.8. [MEASURED]
- Likely causes, untested: (a) our time embedding scales t by 1,000, so the time derivative and the target $-(t - r) \cdot du/dt$ come out about $1,000\times$ too large; (b) the Jacobian-vector product ran under bf16 autocast; (c) the attention path that supports it ran at a third of the throughput, so the arm saw a third as many clips. A later "improved MeanFlow" paper targets exactly this kind of instability [EXTERNAL]. [ESTIMATE]
- **Infrastructure.** Community RTX 5090 hosts failed repeatedly, once handing us the same broken machine twice; page-locking 4.5 GB of memory failed on one host and not another, so the trainer no longer pins large tensors; the first MeanFlow memory probe left out the evaluation networks and under-counted by about 6 GB, which the dry run caught (out of memory at 30.8 of 31.4 GB). [MEASURED]

7. Stage 3: the release run

Run. 20 October 2026, 02:41-18:21 IST, on one NVIDIA L40S 48 GB (Shadeform, Massed Compute). 900 training minutes, 180,080 steps at batch 64, 11.53M clips seen, about 24 passes over the 475,157 windows, at about 213 clips per second. The recipe is Stage 1's, plus captions. [MEASURED]

SETTING	VALUE
Objective	Rectified flow; 25% of steps image-to-video by first-frame replacement; 10% caption dropout
Optimiser	MuonAdamW: Muon (learning rate 0.02) for the matrices, AdamW ($2e-4$) for the rest
Schedule	warm-up over 1,000 steps, learning-rate decay over the final 25%
Weights kept	EMA with decay 0.9999; the release model is the final EMA
Precision	bf16 autocast; gradient clipping at 1.0

[MEASURED, `training/train_t2v.py` and the run's frozen config]

Stage 3: the release run

Flow-matching loss by step (batch 64) · 1× L40S, 900 minutes · learning-rate decay in the last 25%

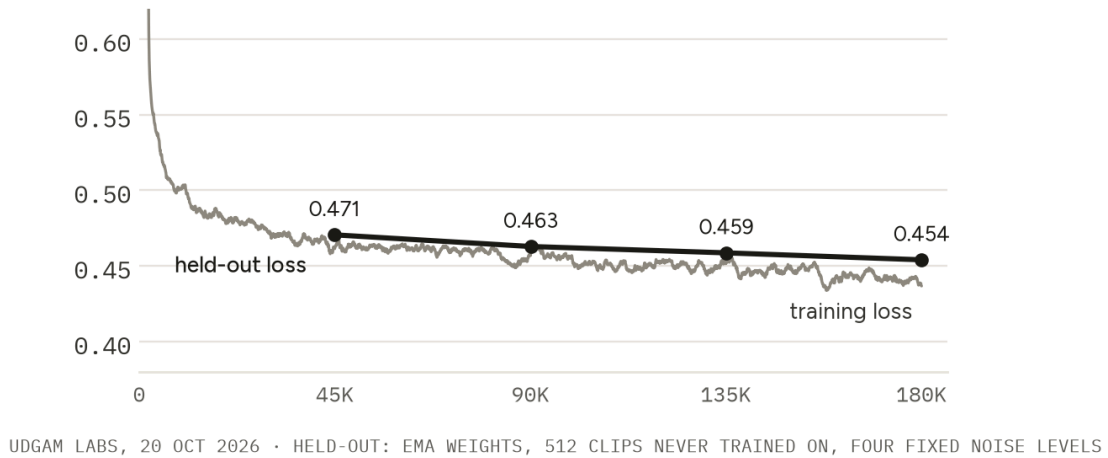


FIGURE 4. TRAINING LOSS (RUNNING MEAN) AND THE HELD-OUT LOSS AT 25%, 50%, 75% AND THE END OF THE RUN. RUN OF 20 OCTOBER 2026.

7.1 The gate

The pass rule was written before the run and checked in code on the held-out clips (EMA weights, 50 Euler steps, shift 3, CFG 5):

CHECK	RULE	RESULT
G1	Text-to-video FVD-2048 below real clips frozen on frame 0 (660.4); motion 0.5-2.0	FVD 235.7, motion 0.73 (pass)
G2	CLIP gap at least 30% of the real clips' gap (25.73 → 7.72), and above 0	20.15, 78% of real (pass)
G3	Held-out loss at the end below that at 25%	0.454 against 0.471 (pass)
G4	Image-to-video FVD-512 below the still-frame FVD-2048 (660.4)	FVD-512 240.3, motion 0.89, CLIP gap 23.7 (pass)

[MEASURED, results/stage3/final.json . CLIP is OpenCLIP ViT-B/32 laion2b_s34b_b79k , not OpenAI CLIP. FVD-512 is biased upward against FVD-2048, so G4, written that way before the run, is only harder to pass; apart from that rule we never compare the two.]

During training the checks ran on 512 held-out clips:

	25%	50%	75%	FINAL
FVD	446.8	431.8	453.8	235.7
Motion ratio	0.76	0.82	0.85	0.73
CLIP gap	16.24	18.27	19.27	20.15
Held-out loss	0.4706	0.4629	0.4587	0.4541

[MEASURED. FVD, motion and CLIP gap on 512 held-out clips during training and 2,048 at the end. Held-out loss: EMA weights on the first 512 held-out clips at four fixed noise levels ($t = 0.2, 0.4, 0.6, 0.8$), fixed seed.]

[MEASURED]

The CLIP gap and the held-out loss improved steadily. FVD-512 stayed flat around 430-450 while the final FVD-2048 came out at 236: at 512 clips the metric is too biased and noisy to read a trend from, which is why the gate uses 2,048. [MEASURED]

What the clips look like (our impression, not a measurement). Scene layouts and colours follow the caption: an ocean horizon, a person at a table, an animal on grass, a red car, a road through trees. Motion is small. Faces and hands smear and fine detail is poor. In image-to-video the given frame is kept, and later frames keep the scene but drift, with blur growing over the 2 seconds.

What went wrong. Stage 3 cost \$24.71; the release path (data \$7.40, training \$13.76) was about \$21 of it. The other \$3.55: the five failed data-step attempts (Section 4); a dry run that needed three starts (two never came up; the third passed train, resume, every score, samples and uploads); and three Runpod main-run starts that never came up, two with no reachable machine and one whose container never started. Our start-up guard checked that SSH answered, not that the container was running, so each failure cost up to 20 minutes. The run then moved to an L40S on Shadeform after a \$0.28 speed probe there. [MEASURED]

8. Stage 4: evaluation

8.1 Checking our own FVD code

Before trusting FVD, we compared our implementation with StyleGAN-V's official FVD code on identical frames: 21.5059 against 21.5059 and 279.374 against 279.374, a relative difference of about $1e-10$. [MEASURED, `results/stage4/fvd_audit_result.json`]

8.2 Choosing the sampling settings

A pre-registered sweep on 512 held-out clips picked the setting with the highest CLIP gap among those within the baseline's FVD + 20 (the rule's allowance for sampling noise):

CFG	SHIFT	STEPS	FVD-512	CLIP GAP	MOTION
5	3	50 (the Stage 3 setting)	426.9	20.47	0.74
3	3	50	373.3	18.55	0.74
3	5	50	376.0	18.61	0.73
5	5	50 (chosen)	432.0	20.50	0.74
7	3	50	517.1	21.13	0.76
7	5	50	519.1	21.22	0.76
5	5	100	437.3	20.47	0.76

[MEASURED; the real clips' CLIP gap is 25.38]

Guidance trades one for the other: CFG 3 gives better FVD (373 against 432) with slightly weaker prompt-following, CFG 7 the reverse. 100 steps add nothing over 50.

8.3 VBench

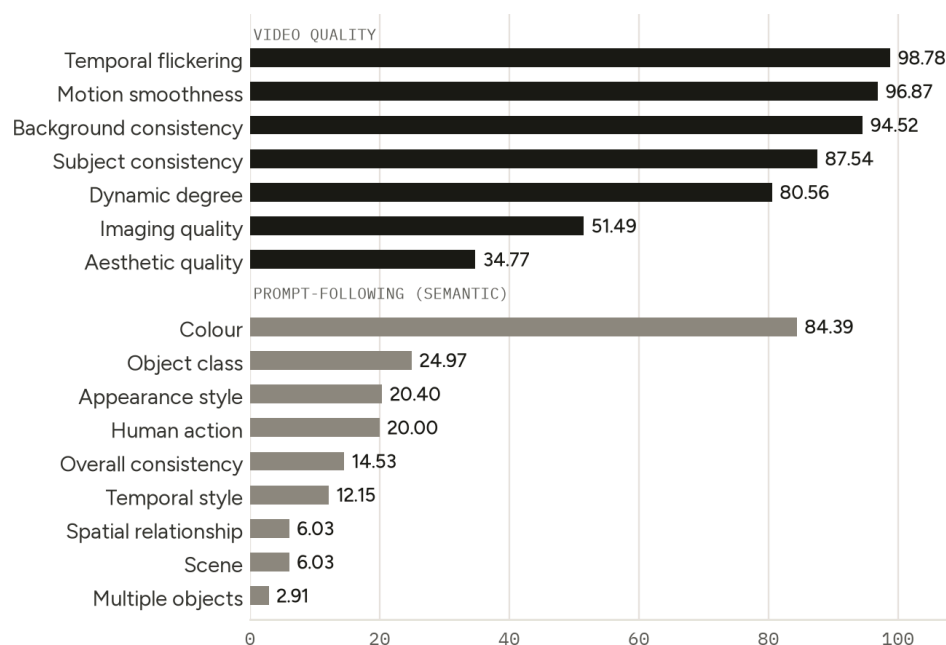
VBench scores a video model on many separate dimensions with fixed prompt lists. We generated every video ourselves (16,530 videos; one RTX A6000, 4 hours, \$2.25) and scored them with the official VBench code at commit fd18b3d055cb and its official normalisation (one A40, 6.8 hours, \$3.99). These are our own sampling and scoring with the official VBench code, not a certified leaderboard entry. [MEASURED]

SUITE	TOTAL	QUALITY	SEMANTIC / I2V
Text-to-video, standard prompts (946 prompts × 5 videos; temporal flickering on 25 per prompt after the static filter)	66.92	75.59	semantic 32.26
Text-to-video, VBench long prompts (GPT-enhanced)	not computed (flickering not run, by design)	—	semantic 33.03
Image-to-video, 1:1 crops (1,118 × 5)	82.62	75.00	I2V 90.24

[MEASURED, 21-22 October 2026; CFG 5, shift 5, 50 steps]

VBench text-to-video, every dimension

Score, % · Drishya Tiny · 946 prompts × 5 videos · CFG 5, shift 5, 50 steps



OUR OWN SAMPLING AND SCORING WITH THE OFFICIAL VBENCH CODE (FD18B3D055CB), 21-22 OCT 2026; NOT CERTIFIED

FIGURE 5. EVERY VBENCH TEXT-TO-VIDEO DIMENSION. DARK: VIDEO QUALITY; GREY: PROMPT-FOLLOWING.

The split is plain. The clips are smooth (motion smoothness 96.87%), steady (temporal flickering 98.78%), keep their subject and background (87.54%, 94.52%) and really move (dynamic degree 80.56%, so the high smoothness is not a still video gaming the score). They get colours right (84.39%). But a clip shows two named objects 2.91% of the time, a named scene or a spatial relation about 6% of the time, and the right object class about a quarter of the time (24.97%). Long, GPT-enhanced prompts barely help (33.03 against 32.26). [MEASURED]

Image-to-video per dimension: subject 93.64, background 95.92, camera motion 24.80, subject consistency 85.77, background consistency 93.63, motion smoothness 96.49, dynamic degree 69.35, aesthetic 50.87, imaging 63.43. [MEASURED]

Checking the image-to-video inputs. We took Vbench's image-to-video images from a Hugging Face mirror (`dc-ai/vbench_i2v`). To be sure they were the official ones, we compared all 355 1:1 crops we used with the official `crop.zip` that Vbench's download script names at the same commit: all 355 are byte-identical by sha256, none differ, none are missing. [MEASURED, 25 October 2026]

8.4 Against published 256px models

The fair peers are the models the VBench team sampled and scored themselves at the same size and length: 256×256, 16 frames, about 2 seconds. Every one has Stable Diffusion in its lineage. [EXTERNAL, official VBench leaderboard, verified 25 October 2026]

MODEL	PARAMETERS	STARTED FROM	TOTAL QUALITY SEMANTIC		
Drishya Tiny [MEASURED, self-run]	211.92M generator (~877M system)	random initialisation	66.92	75.59	32.26
ModelScope T2V	1.7B system (UNet ≈ 1.35B)	Stable Diffusion 2.1	75.75	78.05	66.54
VideoCrafter-0.9	not stated	Stable Diffusion (spatial layers)	73.02	74.91	65.46
InstructVideo	a reward fine-tune of ModelScope	ModelScope	76.61	81.56	56.81

[Peer rows EXTERNAL, official VBench leaderboard; Drishya Tiny MEASURED.]

IMAGE-TO-VIDEO MODEL	I2V TOTAL
DynamiCrafter-256	85.25
ConsistI2V (from Stable Diffusion 2.1-base)	84.07
Drishya Tiny [MEASURED, self-run]	82.62
VideoCrafter-I2V	82.57

[Peer rows EXTERNAL: the VBench team's sampling and scoring on the official leaderboard, checked 25 October 2026. Drishya Tiny MEASURED, our own sampling and scoring.]

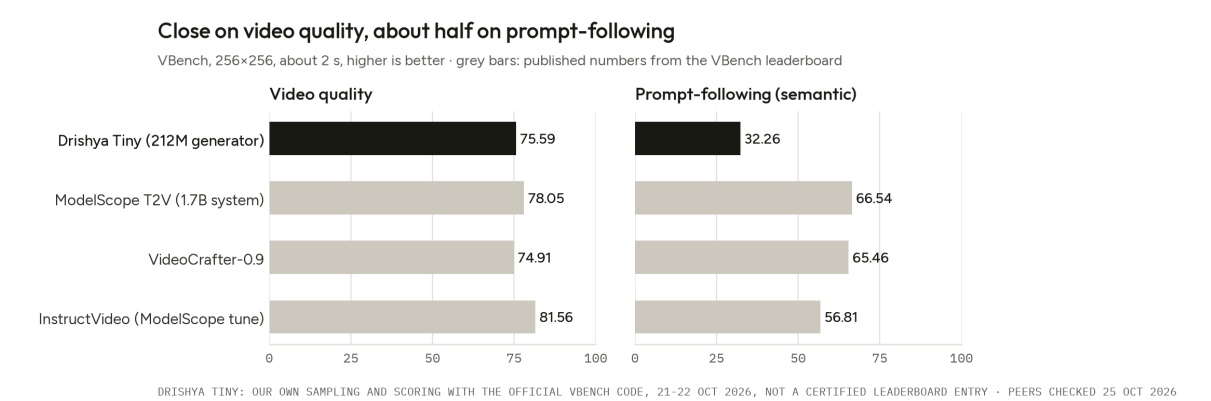


FIGURE 6. VIDEO QUALITY AND PROMPT-FOLLOWING AGAINST THE 256PX MODELS THE VBENCH TEAM SCORED. DRISHYA TINY IS THE DARK BAR: OUR OWN SAMPLING AND SCORING WITH THE OFFICIAL VBENCH CODE, 21-22 OCTOBER 2026. GREY BARS: PUBLISHED NUMBERS, CHECKED ON 25 OCTOBER 2026 [EXTERNAL].

What this supports. Video quality is level with VideoCrafter-0.9 and close to ModelScope, with a generator trained from scratch (on the frozen Wan2.2 VAE and Flan-T5-Base) and about a sixth the size of ModelScope's UNet. Prompt-following is about half theirs, and the total score is clearly lower. Image-to-video only matches VideoCrafter-I2V and trails the two leaders. We claim no total-score, semantic or image-to-video edge. [MEASURED against EXTERNAL]

What went wrong. Stage 4 cost \$8.48, of which generation and scoring were \$6.24. Sixteen of its 24 machines were failed attempts (\$1.66 of waste). Four early attempts on one provider failed on slow SSH, a provider error, a background job killed when the start-up script exited, and a smoke test that caught two real problems: an MKL/libgomp threading clash in VBench's subprocesses (fixed with `MKL_THREADING_LAYER=GNU`) and the `dreamsim` package that VBench's image-to-video scorer needs but does not install. VBench also depends on detectron2, which needs CUDA 12.1 or older, so its environment had to be compiled against torch 2.1.2; several builds and preflights failed before one passed. The evaluation stack cost more calendar time than training did. [MEASURED]

9. Stage 5a: what moves prompt-following?

The VBench split suggested a question worth a controlled test: is the prompt-following gap about the data? We trained three copies of the release model for 18,000 more steps each (about 10% of the release run), with the same schedule and evaluation, differing only in data. [MEASURED]

- **A0, control:** VidGen only, captions cut at 77 tokens, as in Stage 3.
- **A1, caption fix:** captions up to 128 tokens, plus 20% short captions (the first sentence), because VBench's prompts are much shorter than VidGen's ~80-word captions.
- **A2, caption fix + photos:** A1 plus 30% image steps (5,405 of 18,000) on CC12M photos with long captions from BLIP3o: 887,004 photos, after 20,934 were dropped for watermarks. The idea: photos show many objects and scenes our videos rarely do.

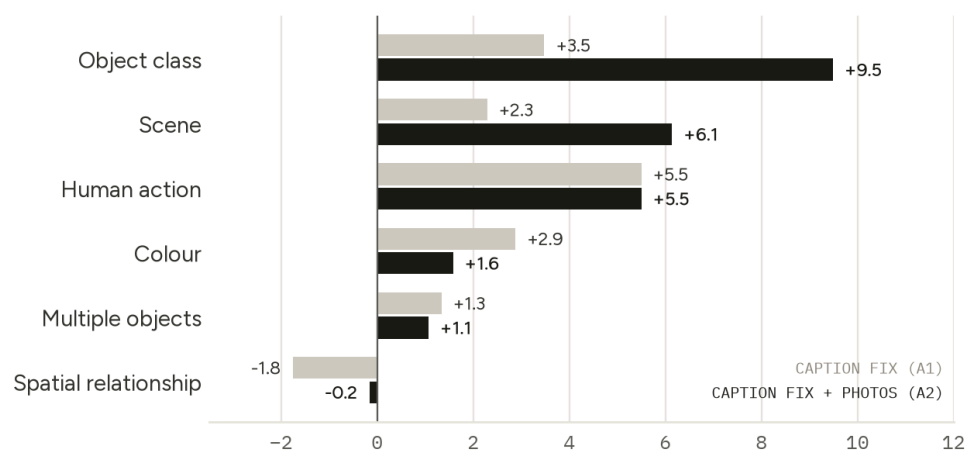
The metric, "mini-semantic", is the mean of six VBench semantic dimensions (object class, multiple objects, human action, colour, spatial relationship, scene) on their 516 prompts $\times$ 2 seeds. **The decision statistic** is a paired one against A0: per dimension, the mean per-prompt difference on the prompts both arms have per-video scores for, averaged over the six dimensions, with a paired bootstrap 95% interval. It pairs 466 prompts for A1 and 468 for A2, because VBench's colour scorer has per-video scores for only 35-37 of its prompts. So the paired statistic (+3.01, +4.34) is not the same as the gap between two arms' mini-semantic scores (+2.29, +3.94); the rule was written on the paired statistic. **Guards:** held-out FVD-512 and CLIP gap, a CLIP gap on 200 real user prompts from VidProM, and image-to-video FVD-256. **The rule, written before the run:** the winner is the arm with the highest mini-semantic score among those whose paired statistic is at least +3 with the interval above zero, a VidProM gap no lower than A0's, and FVD-512 no more than A0's + 20. [MEASURED]

ARM	MINI- SEMANTIC	PAIRED, AGAINST A0 [95% CI]	HELD-OUT FVD-512	HELD-OUT CLIP GAP	VIDPROM CLIP GAP	I2V FVD-256
A0 control	21.42	—	414.4	20.35	8.67	457.7
A1 caption fix	23.71	+3.01 [+0.93, +5.26]	413.3	20.53	8.85	441.1
A2 caption fix + photos	25.36	+4.34 [+1.99, +6.72]	518.7	21.97	10.47	481.4

[MEASURED, 22-23 October 2026; Stage 5a cost \$9.90]

Stage 5a: photos teach what, not where

Points against the control (A0) · six VBench semantic dimensions · 516 prompts × 2 seeds



UDGAM LABS, 22-23 OCT 2026 · PAIRED: A1 +3.01 [+0.93, +5.26], A2 +4.34 [+1.99, +6.72]; A2 FVD-512 +104

FIGURE 7. CHANGE AGAINST THE CONTROL ON EACH OF THE SIX DIMENSIONS (VBENCH AGGREGATES; THE PAIRED STATISTICS ARE IN THE NOTE). PHOTOS MOVE OBJECT CLASS AND SCENE; NOTHING MOVES MULTIPLE OBJECTS OR SPATIAL RELATIONS.

PER DIMENSION (%)	OBJECT CLASS	MULTIPLE OBJECTS	HUMAN ACTION	COLOUR	SPATIAL	SCENE
A0	23.18	1.18	15.50	75.80	6.41	6.47
A1	26.66	2.52	21.00	78.68	4.65	8.76
A2	32.67	2.25	21.00	77.38	6.25	12.61

[MEASURED]

By the rule, A1 wins, only just (+3.01 against a +3.00 threshold; the plain gap between mini-semantic scores is +2.29, so the margin is thin on either reading). A2 has the larger gain but fails the FVD guard by a wide margin (+104 against an allowed +20).

What we read from it.

- The caption fix is worth 2-3 points for free: paired +3.01 (interval +0.93 to +5.26), mostly human action (+5.5) and object class (+3.5), with no quality cost (FVD 413 against 414). [MEASURED]
- Photos teach *what* things are, not how to arrange them. What photos add on top of the caption fix (A2 against A1) is in object class (+6.0) and scene (+3.9), concepts our videos rarely show; against A0 the two dimensions rose by 9.5 and 6.1. Multiple objects and spatial relations did not move: composition is not learned from single-subject photos at this scale and compute. And the gain was smaller than we had estimated before the run (+4.3 paired against A0, against +5 to +15 estimated; +1.65 in mini-semantic over the caption fix alone). [MEASURED; the estimate was ESTIMATE]
- Photos cost video quality by FVD (+104 held-out, +24 image-to-video). The samples do not look worse to us, and objects look cleaner (an impression from 16 prompts, not measured). One possible cause is a style shift from video frames toward stock-photo stills, which FVD on VidGen clips penalises; we have not tested that. [ESTIMATE]
- The gains are against the fair control, not over the release model. The Stage 4 VBench run scored the release model at 24.06 on the same six dimensions, unpaired, with five different seeds; A0 scores 21.42 and A1 23.71 [MEASURED]. So continued training on VidGen alone may have lowered prompt-following a little while improving FVD (414 against 432); with different seeds and seed counts this is not a paired comparison [ESTIMATE]. None of the three arms is the release model.
- The scale of the gap. The best arm reaches 25.4 on these dimensions, far below what the 1-1.7B peers reach. With a 212M-parameter generator and about a tenth of the release run's compute, data fixes move prompt-following by 3-4 points, not 30. Looking real is cheap; following the prompt is what costs. [MEASURED]

This is a finding at one size, one resolution, one length and one budget. We would like to see it challenged.

10. Cost: every machine, failures included

All-in, the project cost \$57.43 of GPU time across 55 rented-machine rows (56 machines: one row covers two that never came up), every failed attempt included, plus \$0.03 of non-GPU research on day 0: \$57.46 in total. The release model's own path was **about \$21**: the Stage 3 data step (\$7.40, one RTX 5090, about 6.2 hours, Runpod) and the training run (\$13.76, one L40S, 15 hours 40 minutes of machine time including its evaluations, Shadeform). [MEASURED]

Every dollar, failures included

GPU cost by stage · \$57.43 across 55 machine rows · release path \$7.40 data + \$13.76 training

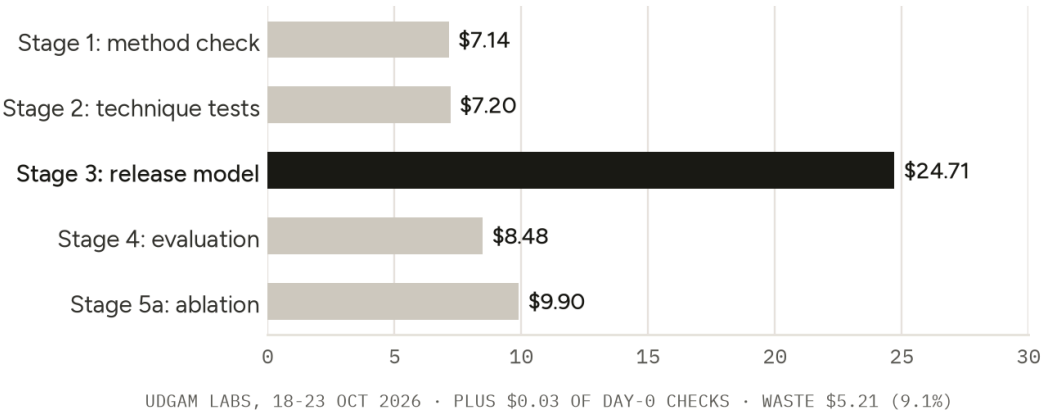


FIGURE 8. GPU COST BY STAGE. THE DARK BAR HOLDS THE RELEASE MODEL'S DATA AND TRAINING.

STAGE	GPU COST	LEDGER ROWS	WASTE	WHAT IT BOUGHT
Day-0 checks (Mac, a Colab probe, research)	\$0.03	—	—	feasibility
Stage 1: SkyTimelapse method check	\$7.14	5	\$0.29	the recipe works
Stage 2: three techniques against the plain recipe	\$7.20	6	\$0.45	keep the plain recipe
Stage 3: data and training of the release model	\$24.71	13	\$2.72	the release model
Stage 4: sampling sweep and VBench	\$8.48	24	\$1.66	the evaluation
Stage 5a: three controlled arms and scoring	\$9.90	7	\$0.09	the ablation
Total	\$57.46	55	\$5.21 (9.1%)	

[MEASURED, results/COST.md . "Waste" is money that produced no kept result: failed starts, idle or hung time, runs stopped by our own bug.]

- **How the rows were measured.** Stages 1-3 and the first Stage 4 rows are provider bills, or rate × time where our log marks them approximate; the later rows (our job runner, Stages 4-5a) are rate × minutes. Prices per GPU-hour at the time: RTX 5090 \$0.69 (Runpod Community) and \$1.19 (Secure); L40S \$0.88 (Shadeform) and \$1.09 (Runpod); RTX A6000 \$0.57; A40 \$0.59.
- **Not included:** the pretraining of the borrowed, frozen Wan2.2 VAE and Flan-T5-Base (not ours), people's time, the electricity of the Mac used for checks, and storage.

- **Dollars are not compute.** An L40S hour is not an L4 hour; the ledger gives the GPU type and minutes of every machine, so the work can be compared on compute as well as price.
- **Where the failures were.** Most failures happened at start-up: machines that never became reachable, containers that never started, environments that would not build, jobs that died in their first minutes. The money lost was small (\$5.21 in all) but the calendar time was not: the evaluation stage took at least 27 hours of wall-clock time against about 7 hours planned. Two lessons we now apply: a guard must check a real value (a container that is running, not a shell that answers), and the evaluation environment needs the same prebuilt, preflighted image as training. [MEASURED]
- **Data cost more than half of training.** The data step (\$7.40, \$8.72 with its failed attempts) cost more than half as much as the training run (\$13.76). At video scale the encoder's throughput deserves its own cheap pilot before anything is sized. [MEASURED]

11. Running it: Python and the browser

Python. The inference package (`python -m drishya` , or `Drishya.from_pretrained`) runs on CUDA, Apple silicon or a CPU and defaults to the evaluation settings (50 steps, CFG 5, shift 5). On a Mac mini (Apple silicon, MPS), a clip takes 2.9 seconds of sampling plus 5.0 seconds of decoding with the Wan2.2 VAE, about 8 seconds, after a one-time load of 12.8 seconds. Every clip it writes carries a small visible "AI-generated" mark and provenance metadata. [MEASURED, 28 October 2026]

In the browser, on your own device. The model page runs Drishya Tiny inside the web page with onnxruntime-web 1.30.0 on WebGPU, with a WebAssembly fallback, from a text prompt or from a photo plus a prompt. The prompt is tokenised with transformers.js, and neither the prompt nor the photo leaves the device. The files download once (665 MB for text-to-video; 3 MB more on the first photo) and stay in the browser's cache:

FILE	SIZE	WHAT IT IS
<code>drishya_tiny_dit_cond.onnx</code>	426 MB	the generator, with classifier-free guidance folded in (a batch of two: prompt and empty prompt) and a switch between text-to-video and image-to-video
<code>flan_t5_base_encoder.onnx</code>	219 MB	the Flan-T5-Base encoder (from the Xenova export)
<code>taew2_2_decoder.onnx</code>	20 MB	TAEW2.2, a tiny decoder for Wan2.2 latents (madebyollin/taehv, MIT)
<code>taew2_2_encoder.onnx</code>	3 MB	TAEW2.2's encoder, for the photo in image-to-video

[MEASURED]

- **Why different VAE parts.** The full Wan2.2 decoder has 555M parameters and its encoder 150M; TAEW2.2 is a small encoder and decoder trained by its author to approximate them [EXTERNAL]. On the same latents its decoder reaches 28.9-31.9 dB PSNR against the full decoder: the same scene, visibly softer. On three test photos its encoder's first-frame latents have a cosine similarity of 0.98 with the Wan2.2 encoder's, and the first frame matches the photo at 24.6-26.9 dB, against 24.9-27.3 dB with the full encoder. [MEASURED]
- **Image-to-video** follows the Python pipeline: the photo is centre-cropped and resized to 256×256 (a JavaScript copy of Pillow's bicubic filter, byte-identical on the test photos), encoded to the first-frame latent and written into latent frame 0 before every step, as in Section 3.4. [MEASURED]
- **Precision.** All the files store their weights in fp16 and compute in fp32; full fp16 compute gave NaNs on WebGPU. [MEASURED]
- **Parity.** Noise is seeded the same way in the page and in our Python ONNX pipeline, so a seed gives the same clip in both: 87.7 dB PSNR between the browser's clip and Python's for text-to-video, 74.6-82.5 dB for image-to-video. [MEASURED]
- **Speed.** At 25 steps, 3.3-3.6 seconds a clip in Chrome on a Mac mini once loaded, for text or a photo (5.5-6.0 seconds from opening the page with cached files); on the CPU fallback about 23-24 seconds for text and about 27.5 seconds for a photo. Not tested: Windows, NVIDIA GPUs, Safari, Firefox and phones. [MEASURED, 28 October 2026]

12. Where it falls short

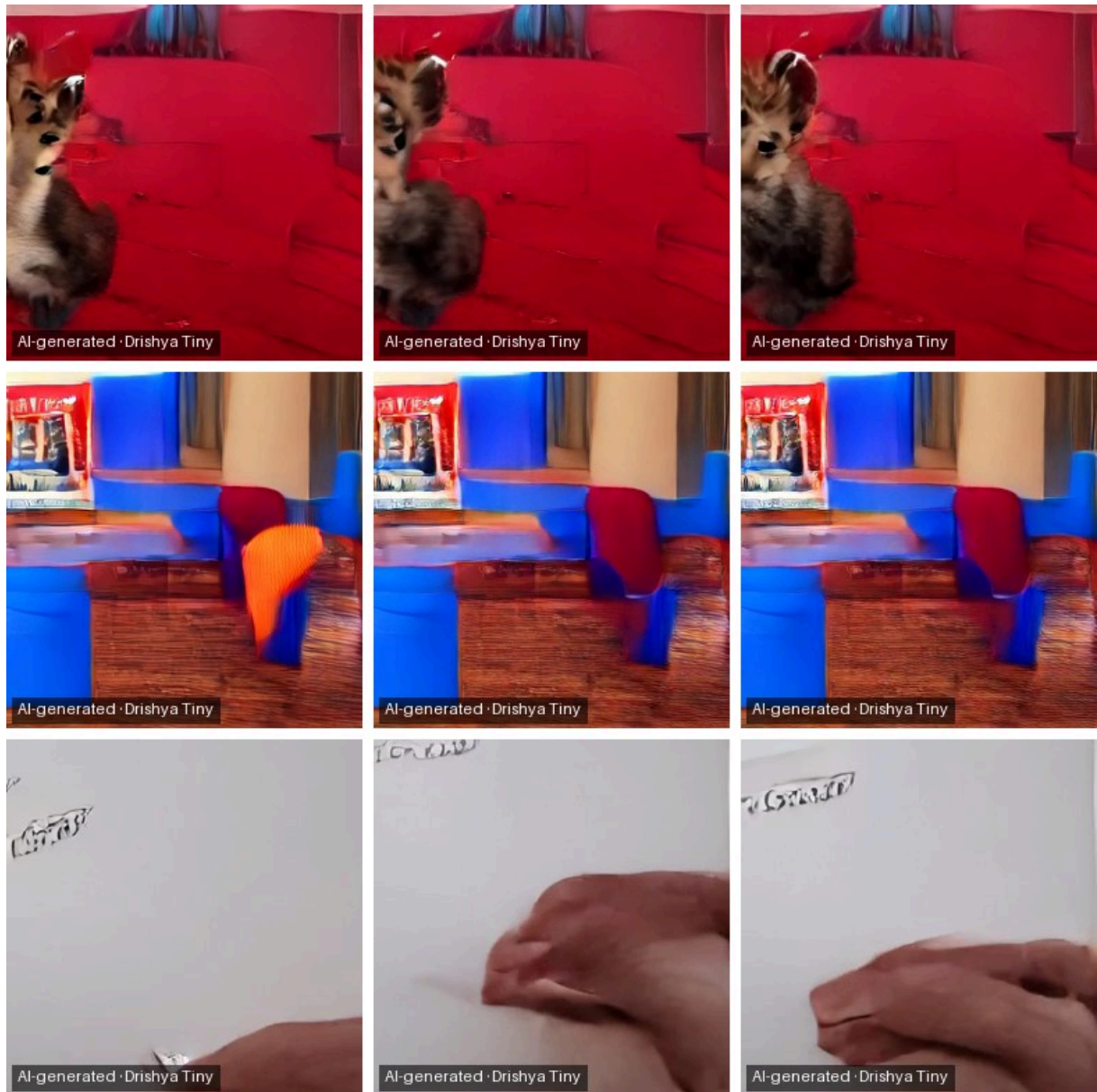


FIGURE 9. FRAMES 1, 9 AND 17 OF THREE CLIPS (SEED 0). TOP: “A CAT AND A DOG SITTING TOGETHER ON A RED SOFA” (THEY MERGE INTO ONE ANIMAL). MIDDLE: “A RED CUP TO THE LEFT OF A BLUE BOOK ON A WOODEN TABLE” (NEITHER APPEARS). BOTTOM: “A PERSON WRITING THE WORD HELLO ON A WHITEBOARD” (NO WORD GETS WRITTEN).

- Prompt-following is about half the peers'. VBench semantic 32.3 against about 65-67 for the 1-1.7B 256px models. It almost never draws two named objects (multiple objects 2.9%), rarely gets spatial relations (6.0%) or named scenes (6.0%), and gets the object class right about a quarter of the time (25.0%). [MEASURED]
- Data fixes alone will not close that gap at this size. Stage 5a moved prompt-following by 3-4 points. [MEASURED]
- Faces and hands smear. Text melts. Each token covers a 32×32 -pixel square over about four frames, which leaves little room for fine detail.

- **Short and small:** 2 seconds, 256×256, 8 frames a second, and the motion is smaller than in real video (motion ratio 0.73). [MEASURED]
- **Image-to-video drifts:** later frames lose detail and blur over the 2 seconds; faces and text melt. Its VBench score only matches the weakest of the 256px image-to-video peers. [MEASURED]
- **Visual quality is modest:** aesthetic 34.8%, imaging 51.5%. [MEASURED]
- **It works best on nature and scenery** (water, waves, sky, snow, boats, fire); people degrade. This is our impression from looking at clips, not a measurement.
- **Not certified:** the VBench numbers are our own sampling and scoring with the official code.
- **Non-commercial weights:** the training clips come from HD-VILA-100M, whose terms allow research use only.
- **The browser version is softer** than the Python one, because of its tiny decoder (and, for a photo, its tiny encoder).

13. Related work

Small and cheap from-scratch video generators [EXTERNAL, scan of 25 October 2026].

MiniDance (released 14 October 2026) is the closest recipe to ours and cheaper: a 119.5M generator trained from random initialisation on the same frozen Wan2.2 VAE, with a frozen Qwen3-0.6B text encoder, on about 40K nature-scenery clips (people, animals, vehicles and text removed), 33 frames at 12 fps, for about ¥12 (~\$1.7) and about 11 hours on one L4, fully open under Apache-2.0. It reports no VBench scores. MiniDance trained a scenery-only model for even less than we did (about \$2); Drishya Tiny learns from open-domain web video, does text-to-video and image-to-video in one model, and publishes VBench scores and a controlled ablation. **ZYVIDEO 0.1** (18 October 2026) applies MiniDance's recipe to the open-domain OpenVid-1M on one L4; its base training cost is not stated, so we do not compare costs with it. **Movetoy** trains a 51M pixel-space generator *and* its own text encoder from scratch, at 32×32. None of the three reports VBench.

The 256px models we compare with. ModelScope T2V, VideoCrafter-0.9, InstructVideo, DynamiCrafter and ConsistI2V (Section 8.4) all have Stable Diffusion in their lineage [EXTERNAL]. Their VBench numbers are the VBench team's own.

Larger open models, for scale [EXTERNAL, self-reported unless marked]. NOVA 0.6B trained its own text-to-image model from scratch and then video, in 342 A100-days (VBench 78.48). SANA-Video 2B continued from a text-to-image model on 64 H100s for about 12 days (83.71). Wan2.1-T2V-1.3B scores 83.31 (the VBench team's row). Open-Sora 1.2 used 35K H100-hours and started from PixArt-Σ, so it is not a from-scratch model. These are different leagues of

compute and resolution, listed so the reader can place a 212M generator making 2-second, 256px clips.

Methods we built on. Diffusion transformers with adaLN-zero (DiT); rectified flow and flow matching, and the logit-normal timestep sampling and time shift used for rectified-flow transformers (Stable Diffusion 3); classifier-free guidance; first-frame replacement for image-to-video (STIV, SANA-Video); Latte's ablations of joint image-video training; TREAD token routing; MeanFlow; the Muon optimiser; VBench; FVD with StyleGAN-V's reference implementation. Our sibling Chitra Tiny, a 212M text-to-image model, supplied the block design.

Where the idea of an open, fully costed small run comes from. Andrej Karpathy's nanochat showed the value of a small model whose every step and dollar is public. Drishya Tiny tries to do that for video, with the failures left in.

14. What we would test next

Directions only; none is planned or costed here. [EXPLORATION]

- **Where the prompt-following gap lives.** Is it in understanding the prompt or in the size of the generator? Pairing the same small generator with a larger frozen text encoder, scored on the same VBench semantic dimensions, would split the two.
- **Composition data.** Photos taught objects and scenes but not arrangement; data that shows several named things together, with captions that say where they are, is the obvious next test.
- **Commercially usable weights.** Retraining the same recipe on a clip set whose terms allow it, such as OpenVid-1M (CC BY 4.0 according to MiniDance's card [EXTERNAL]), would remove the non-commercial limit.
- **A fair MeanFlow test** with the three likely causes fixed, and TREAD at clip lengths where attention dominates.
- **Measuring the cheaper peers in our own harness**, so that comparisons with MiniDance and ZYVIDEO rest on the same VBench sampling and scoring as ours.

15. Licences and credits

Drishya Tiny. Weights under CC BY-NC 4.0 with a use policy, because the training clips come from HD-VILA-100M, whose terms allow research use only. Code, recipe and the inference package under Apache-2.0. Every clip the package or the browser version makes carries an "AI-generated" mark.

Borrowed parts. The Wan2.2 VAE (Wan team, Apache-2.0), Flan-T5-Base (Google, Apache-2.0) and, in the browser version, TAEW2.2 (Ollin Boer Bohan, MIT); onnxruntime-web and

transformers.js for running it in a page.

Data. VidGen-1M and the HD-VILA-100M clips it comes from (research use only; clip IDs released, clips not redistributed); SkyTimelapse for Stage 1; CC12M photos with BLIP3o captions in Stage 5a's A2 arm only, not in the release model; VidProM user prompts for Stage 5a's evaluation.

Thank you to the VidGen-1M and HD-VILA-100M authors for the clips, the Wan team for the Wan2.2 VAE, Google for Flan-T5, Ollin Boer Bohan for TAEW2.2, and the VBench team for a benchmark anyone can run.

16. Citation

```
@misc{udgamlabs2026drishya,  
  title = {Drishya Tiny: building a small video generator from scratch,  
failures included},  
  author = {Prasad, Jitesh K},  
  year = {2026},  
  howpublished = {Udgam Labs},  
  url = {https://huggingface.co/udgamlabs/Drishya-Tiny}  
}
```

17. References

1. Z. Tan et al. VidGen-1M: a large-scale dataset for text-to-video generation. arXiv 2408.02629, 2024.
2. H. Xue et al. Advancing high-resolution video-language representation with large-scale video transcriptions (HD-VILA-100M). CVPR 2022.
3. Wan team. Wan: open and advanced large-scale video generative models. arXiv 2503.20314, 2025; Wan2.2 VAE (Wan-AI/Wan2.2-TI2V-5B-Diffusers).
4. H. W. Chung et al. Scaling instruction-finetuned language models (Flan-T5). 2022.
5. W. Peebles and S. Xie. Scalable diffusion models with transformers (DiT). ICCV 2023.
6. X. Liu et al. Flow straight and fast: learning to generate and transfer data with rectified flow. ICLR 2023. Y. Lipman et al. Flow matching for generative modeling. ICLR 2023.
7. P. Esser et al. Scaling rectified flow transformers for high-resolution image synthesis (Stable Diffusion 3). 2024.
8. J. Ho and T. Salimans. Classifier-free diffusion guidance. 2022.

9. Z. Lin et al. STIV: scalable text and image conditioned video generation. 2024. J. Chen et al. SANA-Video. arXiv 2509.24695, 2025.
10. X. Ma et al. Latte: latent diffusion transformer for video generation. 2024.
11. F. Krause et al. TREAD: token routing for efficient architecture-agnostic diffusion training. 2025.
12. Z. Geng et al. Mean flows for one-step generative modeling (MeanFlow). 2025.
13. K. Jordan et al. Muon: an optimizer for hidden layers in neural networks. 2024.
14. Z. Huang et al. VBench: comprehensive benchmark suite for video generative models. CVPR 2024; code at github.com/Vchitect/VBench, commit fd18b3d055cb.
15. T. Unterthiner et al. Towards accurate generative models of video (FVD). 2018. I. Skorokhodov et al. StyleGAN-V. CVPR 2022.
16. J. Wang et al. ModelScope text-to-video technical report. arXiv 2308.06571, 2023. Y. He et al. LVDM / VideoCrafter. arXiv 2211.13221. H. Yuan et al. InstructVideo. arXiv 2312.12490. J. Xing et al. DynamiCrafter. arXiv 2310.12190. W. Ren et al. ConsistI2V. arXiv 2402.04324.
17. MiniDance (huggingface.co/JuniverseLi/MiniDance), 2026; ZYVIDEO 0.1 (huggingface.co/caikybaldo999/zyvideo-0.1), 2026; Movetoy (huggingface.co/Kompaktion/Movetoy), 2026.
18. H. Deng et al. NOVA: autoregressive video generation without vector quantization. arXiv 2412.14169. Open-Sora 1.2 report (hpcaitech/Open-Sora).
19. O. Boer Bohan. TAEHV / TAEW2.2 tiny autoencoders (github.com/madebyollin/taehv).
20. A. Karpathy. nanochat. github.com/karpathy/nanochat, 2025.
21. VidProM (W. Wang and Y. Yang, 2024); CC12M (S. Changpinyo et al., 2021); BLIP3o (J. Chen et al., 2025); SkyTimelapse (W. Xiong et al., 2018).

About Udgam Labs. Udgam Labs is an independent AI research lab. We work from first principles: before we build, we understand why things work, whether we are training a model, fine-tuning one or applying AI to a new problem. Udgam (उद्गम) is Sanskrit for origin. udgamlabs.com