MIRA Mini

Reproducing, Compressing, and Measuring a Multiplayer Interactive World Model

Hugo Hernández **Louis Develle**

Alakazam · Technical Report · July 2026

On July 6, 2026, General Intuition, Kyutai and Epic Games released MIRA [8, 5]: code, data [9], and a technical report for a 5B-parameter action-conditioned world model of four-player Rocket League. The weights stayed private. We reproduce the full recipe at the 1B ablation scale on the released data: codec (125k steps, PSNR 28.6 vs. the paper's 29.7), single-player world model (52k steps, gFID 12.8 at 52% of the paper's step budget vs. their 10.7), and the four-player multiplayer fine-tune (warm-started per the paper's budget-split sweep), on preemptible GPUs. We build what the release does not include: a production serving stack that lets four humans play the model from a browser. We contribute a training-free controllability instrument (a seed-controlled action-divergence ladder), use it to isolate a multiplayer-specific early-window controllability lag, and show that inference-time action guidance (classifier-free guidance applied to the action stream, exploiting the recipe's own action dropout) moves the multiplayer model across the felt-agency threshold with no retraining. A companion program compresses the result: a bit-exact CUDA-graph runtime (2.75× single-view), a post-hoc few-step distillation whose warm-start construction we contribute, a small decoder (8× cheaper), and a 364M student that plays the full game on a 2021 MacBook. Every measured number traces to a logged artifact; projections are marked as estimates.

Demo: play.alakazam.gg
Blog: alakazam.gg/blog/mira-mini
Code + weights: github.com/Alakazam-studios/alakazam-mira-mini **Report sources:** logged artifacts per section

Contents

1 Introduction and philosophy

MIRA's release is unusually complete (code, data, configs, and a 59-page technical report [8]) but withholds weights, which makes it an ideal test of a question we care about commercially and scientifically: is a frontier world-model recipe *actually* reproducible by a small team on public infrastructure, and what does it cost?

Five days after the release (2026-07-11), the repository showed 355 stars, 17 forks, and no public reproduction. The gap between "open" and "reproduced" is operational, not intellectual: the recipe's cost is dominated not by GPU-hours but by distributed-training failure modes, evaluation discipline, and the unglamorous machinery between a checkpoint and a playable system. This report documents that machinery.

Our working rules, fixed before training started:

1. **R1: Recipe fidelity.** Deviate from the published recipe only on the paper's own evidence, and record every deviation (§3). Where the released config and the paper text disagreed, the paper text won.
2. **R2: Verification by artifact.** Every claimed state change must be evidenced by an artifact (a checkpoint hash, an eval JSON, a probe frame dump, a healthz payload), never by the absence of an error message.
3. **R3: Gates before spend.** Each phase (codec → SP → MP) had numeric go/no-go gates scored against the paper's tables before the next phase's GPUs were rented.
4. **R4: Honest deltas.** We report where we are behind the paper and why, at the same prominence as where we match it.
5. **R5: Playability is an eval.** Automated metrics gate spend; human hands-on-keyboard sessions gate deployment. Several findings in Sections 9–10 were discovered by play, not metrics.

Is a frontier world-model recipe reproducible by a small team on public infrastructure, and what does it cost?

That is the question the release lets us test, and the report answers it in the affirmative at ablation scale: the recipe reproduces on the public data for roughly $10k of preemptible GPU time (§8). The cost is dominated by distributed-training failure modes and the machinery between a checkpoint and a playable system, not by GPU-hours alone. Five rules fixed before training (recipe fidelity, verification by artifact, gates before spend, honest deltas, and playability as an eval) govern every claim that follows.

2 The MIRA recipe

We summarize [8] only as needed; readers should consult the original.

MIRA sits in the lineage of learned game simulators (World Models [6], GameNGen's real-time DOOM [14], DIAMOND's CS:GO world model [1], and Decart's playable Oasis [3]) and extends it to *joint multi-agent* conditioning. It operates in the latent space of a representation autoencoder codec (RAEv2): a frozen DINOv3-L/16 encoder [12] with a learned bottleneck (32-channel latents, /32 spatial, temporal downsampling $t_d = 2$, i.e. 10 Hz latents for 20 fps video), trained for reconstruction. The world model is a ~1B-parameter (ablation scale; the demo model is 5B) flow-matching diffusion-forcing transformer that predicts the next latent

conditioned on a rolling context window and per-player action embeddings over the paper's canonical nine-key keyboard vocabulary (six directional keys derived from three-valued axes, plus jump/boost/slide; no mouse conditioning, per §3.1/§4.5 of [8]).

The multiplayer wrapper (§4.5 of [8]) tiles four per-player views into one joint latent with shared attention, and trains with per-player action dropout so any seat can be driven by the model instead of a controller ("theory of mind", §6.8). Inference is streaming: a 20-latent rolling window with KV caching; sampling steps per latent frame trade quality for latency (§5 of [8]).

The paper's key training-budget finding (§6.6) drives our schedule: at a fixed budget, pretraining single-player and continuing on multiplayer beats multiplayer-from-scratch (which collapses at small budgets), and a *modest* single-player share is best. Their sweep's optimum sits near ~50k SP steps at the 1B/100k-step scale, the basis for our warm-start point (§6).

What in the MIRA recipe drives how we spend a fixed budget?

The model is a flow-matching diffusion-forcing transformer over a DINOv3-based representation-autoencoder codec, wrapped for four players with shared attention and per-player action dropout. Its decisive scheduling finding (§6.6 of the paper) is that pretraining a single-player model and continuing on multiplayer beats multiplayer-from-scratch, which collapses at small budgets, with a modest single-player share best. That finding sets our phase order (codec, then single-player, then a warm-started multiplayer fine-tune) and fixes the warm-start point near ~50k single-player steps.

3 Reproduction setup

3.1 Data

The released dataset [9] (CC BY-NC-SA 4.0; Rocket League content used with Epic Games' permission): ~15.8k bot-vs-bot matches (≈2,000 match-hours; 20 fps video with per-player controller logs and privileged state). The paper's flagship models train on the full ~83k-match / ~10,000-hour corpus, a ~5.3× match-count gap to the release, which is the ablation-scale corpus. All our models train on the release.

3.2 Hardware

GCP `a3-highgpu-8g` nodes (8×H100 80 GB, NVLink intra-node), zones us-east4-a/b:

Phase	Nodes	Provisioning	Measured pace
Codec (125k steps)	1	spot	~48 h wall
SP world model (52k)	1	spot	1.37 s/step
MP smoke (2-node)	2	spot	1.88 s/step
MP fine-tune (4-node, 32 GPU)	4	3× spot + 1× DWS flex-start	2.55–3.6 s/step (§8)

Table 1: Hardware and measured pace by phase.

Cross-node communication runs all-TCP NCCL (`NCCL_P2P_DISABLE=1`, `NCCL_SHM_DISABLE=1`, GDR off) over GVNIC, a deliberate trade of bandwidth for robustness after early NVLink/PCIe faults (§8).

3.3 Code and deviations from the release

Code pinned at [5] commit `6aae8d3`. Complete list of deviations:

1. **Clip length 80, not 40.** The released world-model YAML defaults to 40-frame clips; the paper's ablation text specifies 80-frame training clips. Paper text won (R1). This materially changes the difficulty of the task (4 s of context vs 2 s).
2. **Loader robustness patch.** A ~10-line patch to skip undecodable clips (the released loader crashes the epoch on a corrupt sample). No effect on data distribution beyond dropping unreadable files.
3. **All-TCP NCCL** (Hardware, above): infrastructure, not recipe.
4. **Checkpoint cadence** every 1,000 steps with `keep_recent=2` plus off-node archival (§8), vs. the release default: operational, not recipe.

Everything else (optimizer, LR schedule, batch shapes, codec architecture, diffusion forcing config, validation protocol) is the release's ablation configuration unchanged.

What exactly are we reproducing, and where do we knowingly diverge?

We train on the released ~15.8k-match ablation corpus (roughly one-fifth of the flagship's ~83k-match data) on GCP 8×H100 spot nodes, pinning the released code at commit `6aae8d3`. Only four deviations exist, and every one is recorded: 80-frame clips (paper text over the released YAML), a loader-robustness patch, an all-TCP NCCL posture for fault-robustness, and an operational checkpoint cadence. Optimizer, schedule, batch shapes, codec architecture, and validation protocol are the release's ablation configuration unchanged.

4 Codec

125k steps of the paper's codec ablation recipe (the LR schedule anneals fully by 125k in the ablation config, and the paper's own codec comparisons evaluate at the 125k checkpoint, so our comparison is step-matched). Training: ~48 h on one spot node, zero preemptions.

Evaluation (2,048 held-out clips, the paper's protocol: full-frame LPIPS-AlexNet [15]; we note the training-log LPIPS is VGG-on-crops and *not* comparable):

Metric	Ours (125k, ~15.8k-match release)	Paper Table 3 (125k, ~5× matches)	Δ
PSNR ↑	28.59	29.7	−1.1 dB
SSIM ↑	0.867	0.891	−0.024
LPIPS-Alex ↓	0.068	0.051	+0.017

Table 2: Codec reconstruction at the step-matched 125k checkpoint, against MIRA Table 3.

With step count matched, the remaining gap is most plausibly the ~5× corpus difference; we did not run a codec data ablation to isolate it, and record the attribution as plausible rather than demonstrated. Gate R3 was "within 2 dB PSNR": **pass**.

The codec's reconstruction floor matters downstream: §5's decomposition shows the world model at 45k steps is already *codec-limited*. A better codec, not a bigger WM, is the first lever for future quality (consistent with the paper's emphasis on codec quality, §6.3 of [8]).

Does the codec reproduce at ablation scale?

Yes. At the step-matched 125k checkpoint the codec reaches PSNR 28.59 against the paper's 29.7, a 1.1 dB gap most plausibly explained by the ~5× corpus difference, though we did not run the data ablation to prove it. The R3 gate of "within 2 dB PSNR" passes. The reconstruction floor is the binding downstream constraint: at 45k steps the single-player world model is already codec-limited, so a better codec is the first lever for future quality.

5 Single-player world model

1B parameters, frozen codec from §4, 80-frame clips, global batch 16 on one node. Launched 2026-07-09; warm-start checkpoint taken at 52k steps (§6).

Training trajectory (Fig. 1): from-scratch start at loss 9.39; validation (256 held-out samples) descends monotonically 1.176 → 0.385 over 5k → 50k steps with a stable train-val gap (no overfitting at this data scale).

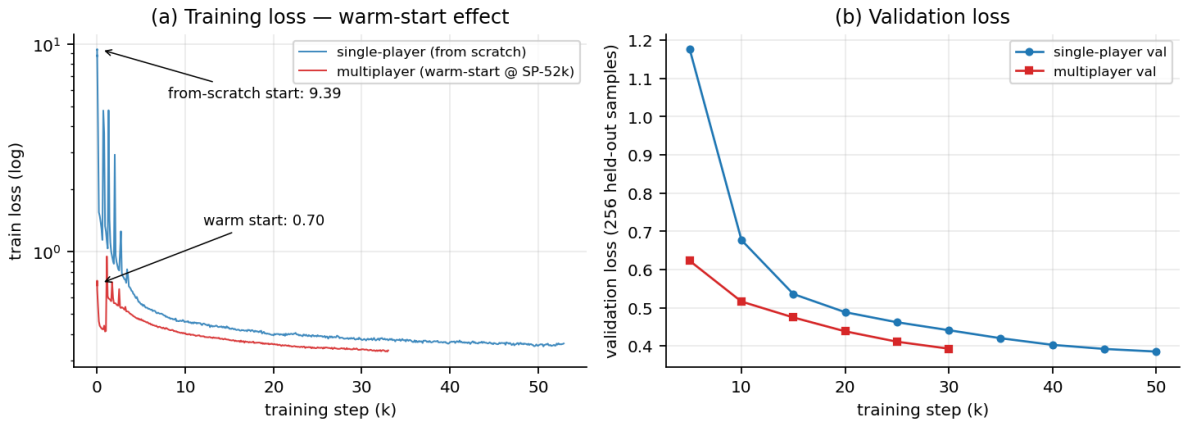

Figure 1: (a) Train loss; note the multiplayer run (red) warm-starts at 0.70 vs 9.39 from scratch, the §6.6 recipe visualized. Early SP spikes (<4k steps) are transient instabilities that self-recovered; no intervention was applied. (b) Validation loss, 256 held-out samples every 5k steps.

Generation quality at 52k steps (gFID [7] / gFDD at the 4 s horizon, following the paper's §6.2 protocol but on 512 samples where the paper evaluates 2,048, an eval-cost concession that adds sampling noise to our side of the comparison):

Metric	Ours @52k (52% budget)	Paper @100k	Ratio
gFID ↓	12.8	10.7	~1.2×
gFDD ↓	0.45	0.55	0.82× (better)
Rollout PSNR @4s ↑	17.3	–	–
Rollout LPIPS @4s ↓	0.32	–	–

Table 3: Single-player generation quality at the shipped 52k checkpoint against the paper's 100k reference.

The 52k row is the shipped checkpoint's eval. The same pipeline re-run on the earlier 45k checkpoint reproduces its mid-training result (gFID 13.46 vs the previously reported 13.4; gFDD 0.466 vs 0.47), which is the protocol-validation behind the swap.

Decomposing FDD: the world-model-above-codec component (0.21) is *below* the codec floor (0.26), i.e., at 45k steps the WM saturates its codec. This is the quantitative basis for the §4 claim

that codec quality is the binding constraint.

Qualitative playability timeline (R5; single human, browser, 8 sampling steps): 5k = coherent for ∼1 s; 10k = world persists, clock ticks, steering registers weakly; 20k = multi-second coherence; 45k = ball persists across the full window, opponent renders, "action response present but not snappy", consistent with the paper's finding that controllability converges *after* visual quality (§6.5 of [8], and §10 below).

Ball-persistence investigation. Early sessions showed balls/opponents vanishing mid-view. Three hypotheses were tested and all confirmed as contributing, with under-training dominant: (T1) codec reconstruction floor on small distant objects, (T2) too-few sampling steps at inference, (T3) training depth. At 45k with 8 steps the phenomenon is resolved in single-player. (Working notes: `tasks/diag-ball-persistence.md`.)

Does the single-player world model reproduce, and what limits it?

At 52k steps (52% of the paper's budget) the shipped single-player checkpoint reaches gFID 12.8 against the paper's 10.7 at 100k, and gFDD 0.45 against 0.55 (better on that axis). Decomposing FDD shows the world-model-above-codec component (0.21) already below the codec floor (0.26): the model saturates its codec, so codec quality, not model depth, is the binding constraint. Hands-on play tracks the metrics (coherent worlds by 20k, ball persistence and rendered opponents by 45k) with action response present but not yet snappy, matching the paper's finding that controllability converges after visual quality.

6 The warm-start decision

The paper's §6.6 sweep answers "how should a fixed budget split between SP and MP?" Its measured optimum at the 1B scale is a modest SP share; in absolute steps their best split pretrains SP for roughly ∼50k steps. Their *demo* model warm-started at 30k, but from the 5B run: a scheduling convenience, not the sweep's optimum. We therefore took our warm start at 52k (the checkpoint nearest the sweep optimum when the MP fleet was ready), rather than imitating the demo's 30k.

The effect is Fig. 1a: multiplayer training opens at loss 0.70 (vs 9.39 from scratch). The remap of single-player weights into the four-view wrapper transfers rendering and dynamics wholesale, leaving multiplayer training to learn what is genuinely new: joint-view consistency and per-player action attribution.

6.1 Reflection (post-hoc, added after the early-window plateau appeared)

The choice above was contested in the moment: the lead advocated matching the demo's 30k, and we argued for 52k. Recording the reasoning and its honest re-assessment, because the plateau (§10) later put the decision under scrutiny.

What we argued for 52k. (i) The demo's 30k carries no optimality claim: it is the 5B model on ∼5× data, its schedule explicitly "well short of the ceiling," never swept. (ii) Converting the §6.6 sweep's ∼25%-single-player optimum to absolute steps at the larger (3.2M-view) budget sits near ∼50k SP steps, and 52k sat on it. (iii) Warm-start transfers at any depth: constant LR after warmup (any step is a valid seed), the multiplayer-specific parameters initialize fresh regardless, and a smoke fine-tune fell 1.15 → 0.88 loss within 100 steps. (iv) Against a fixed Monday freeze, MP steps are the scarce resource. (v) SP-30k survived in a GCS serving tar as a live ∼2-hour fallback.

Where the argument over-reached. "We are *on* the sweet spot" was stronger than the evidence. The

25%-share optimum maps to *different* absolute SP steps by total budget: SP-25k at the plotted 1.6M-view sweep (right beside the demo's 30k), SP-50k only at the 2× budget; we invoked the arm that flattered 52k. And every number in that sweep is gFID: the paper never measured controllability against warm-start depth, so it cannot adjudicate 30k vs 52k for *steering*, the axis we now care about.

The load-bearing correction. The 30k-vs-52k choice **did not cost a single MP step.** the SP-30k checkpoint already existed; selecting it would have launched MP at the same wall-clock time with the same ~63k-by-freeze budget. Only the seed prior differed. Therefore warm-start depth is **not** a cause of the early-window weakness in §10; that is set by MP step count and model scale, both independent of this choice. The one channel by which 52k could still matter is a "stiffer prior" (a more-converged SP being slower to re-attribute actions across four cars); it is untested and falsifiable by the cheap SP-30k warm-start ablation, whose checkpoint already existed (§12).

At what single-player depth should the multiplayer fine-tune warm-start, and did the choice cost anything?

We warm-started at 52k steps (the checkpoint nearest the §6.6 sweep optimum when the fleet was ready) rather than copying the 5B demo's unswept 30k, and multiplayer training duly opened at loss 0.70 instead of 9.39 from scratch. The post-hoc reflection is candid that the "we are on the sweet spot" claim over-reached, since the sweep's optimum maps to different absolute steps by budget and was measured only in gFID, never in controllability. The load-bearing correction is that the choice cost zero multiplayer steps (the SP-30k checkpoint already existed), so warm-start depth cannot explain the early-window steering weakness; the one remaining channel, a stiffer prior, is untested and falsifiable with the existing SP-30k checkpoint.

7 Multiplayer fine-tune

Four-player MultiWrapper (§4.5 of [8]), four nodes / 32×H100, global batch 32 tiled views, checkpoint every 1,000 steps. Launched 2026-07-10 (Friday, ~20–23:00 UTC); the run froze at ~63k steps on 2026-07-13 (~14:00 UTC).

Validation (256 held-out samples):

Step	Val loss
5k	0.623
10k	0.516
20k	0.438
30k	0.393
40k	0.361
50k	0.340
60k (freeze −3k)	0.331

Table 4: Multiplayer validation loss (256 held-out samples) through the 63k freeze (2026-07-13); monotone with a stable ≈0.08 train-val gap.

Full series at 5k intervals, from the rank-0 trainer logs: 35k 0.3755, 45k 0.3524, 55k 0.3413; the last validation before the 63k freeze is 60k. The 50k→55k uptick of +0.0013 is within the eval's noise; the trend is monotone.

Monotone, no plateau, stable train-val gap ≈0.08 throughout. A 512-sample gFID at the 4 s horizon against the paper's multiplayer references (9.4–9.9 at larger budgets) is running at the time of writing (protocol and result recorded in §12).

Playtest milestones (R5): 5k: four coherent per-seat views of one arena, HUD state diverges within seconds ("first light"). 10k: per-seat world quality ≈ early SP; steering weak. 20k: "much better steering" (single-blind human report); HUD agreement improves; multi-ball episodes persist (§9). 30k: steering authority measurable but weak (§10); cross-view binding still the dominant defect.

Did the multiplayer fine-tune converge by the freeze?

Through the 63k freeze (2026-07-13) the four-player validation loss descends monotonically from 0.623 at 5k to 0.331 at 60k (the last validation before the freeze) with no plateau and a stable ≈ 0.08 train-val gap, warm-started from the single-player checkpoint. Play tracks the curve: coherent per-seat views and diverging HUDs by 5k, "much better steering" and persistent multi-ball episodes by 20k, measurable but still-weak steering authority by 30k, with cross-view binding the dominant remaining defect. The 512-sample gFID against the paper's 9.4–9.9 multiplayer references is running at the time of writing (recorded in §12).

8 Operations

What reproduction actually costs is dominated by operations rather than GPU-hours; this section records both the failure modes and the dollar total.

8.1 Hardware faults are model-invisible

Early codec attempts crashed with NCCL failures that pattern-matched to config error but were bad silicon: recurring Xid 94/137 on the same PCIe address across restarts. Diagnosis rule that held: *same-device recurrence = hardware; migrate zones/regions rather than debugging software.* The all-TCP NCCL posture (§3) traded ~bandwidth for elimination of the NVLink/P2P fault surface entirely; at 1B scale the training remains compute-bound enough that the trade was strictly positive for wall-clock-per-dollar.

8.2 Fleet-consistent resume on preemptible nodes

Spot preemption is not an edge case; it is the steady state (4 preemptions of the same node inside 36 h). Our protocol keeps a multi-node fleet consistent through arbitrary node loss with zero step loss beyond the checkpoint interval:

- Rank-0 checkpoints every 1,000 steps; an uploader daemon (self-selecting via lockfile + identity tag) ships completed checkpoint dirs to GCS and flips a monotonic `LATEST` marker only after confirmed upload (ledger-after-flip).
- Every (re)launch derives its `continue_from` from `LATEST`: local checkpoints *ahead* of `LATEST` are pruned (they cannot be fleet-consistent), missing ones are pulled. A node refuses to start fresh if local state exists without a marker.
- Launch requires bucket reachability; supervisors retry with backoff.

Observed across the MP run: 4 preemptions, 3 uploader-daemon deaths (a quiesce-without-restart lifecycle hole, since automated), one boot-time configuration landmine (a stale systemd unit rewriting the run spec on boot; final fix: delete the unit *and* `chattr +i` the spec), and one 20-minute watcher-budget miss. Zero training divergence events; zero steps lost beyond the interval. Loss curves are continuous across every resume (Fig. 1).

Elastic-launch pathologies worth recording: torchrun c10d *ignores* `--node-rank`, so rank 0 is assigned to the lexicographically-first hostname, which silently relocates the "saving node" if fleet names change; and staggered supervisor relaunches split the rendezvous into disjoint rounds (fleet up, 0% GPU), fixed by a synchronized fleet-wide kill so supervisors relaunch inside one window. Both are now handled by a VM-resident guardian (10-minute cycle: resurrect terminated nodes, revive the uploader, synchronized kick on stale steps) after we learned that laptop-resident watchers die with the operator's laptop sleep.

8.3 Topology luck is real

Identical config, identical nodes, three launches: 2.55 s/step, then 3.2–3.6 s/step after two rendezvous re-rolls, a $\sim$25–40% throughput spread attributable to NCCL ring construction over TCP (member ordering changes the ring; some rings traverse the cross-zone link more). At this scale we accepted the draw; larger fleets should pin rendezvous order or measure-and-reroll at launch.

8.4 Capacity

Spot H100 capacity in us-east4 was intermittently zero. DWS flex-start (queue-based provisioning, $\leq$7,200 s request validity, $\leq$80 h lease, no mid-lease preemption, $\approx$2$\times$ spot price) filled in minutes on weekdays and not at all across a weekend Saturday despite a three-zone automatic refile loop. Plan capacity assuming DWS is a weekday instrument.

8.5 Cost accounting

The multiplayer leg completed at the 63k freeze (2026-07-13); the figures below are the through-freeze estimates, with final invoice reconciliation pending.

Item	Compute	Est. cost
Codec, 125k steps	1 node $\times$ $\sim$48 h spot	$\approx$\$1,100
SP world model $\rightarrow$ 52k	1 node $\times$ $\sim$20 h spot	$\approx$\$460
MP 2-node smoke + kill-drills	2 nodes $\times$ $\sim$2 h	$\approx$\$90
MP 4-node fine-tune $\rightarrow$ freeze	3 spot + 1 DWS $\times$ $\sim$68 h	$\approx$\$7,700
Offline evals (gFID/gFDD runs)	restarted node-hours	$\approx$\$150
Serving (B200 probe/playtest sessions)	scale-to-zero, per-session	$\approx$\$100–150
Total		$\approx$\$9.6–9.7k

Table 5: Training and serving cost through the 63k multiplayer freeze (2026-07-13); final invoice reconciliation pending.

For scale: the originating lab's disclosed funding is \$454M [13]. We state the two numbers and let the reader divide, without implying parity: their flagship is 5B on $\sim$5$\times$ the matches with evaluations we have not attempted, and funding buys far more than one training run.

What does reproducing the recipe on preemptible hardware actually cost, in dollars and in operational discipline?

Roughly $9.6–9.7k of preemptible GPU time through the multiplayer freeze, but the dollar figure is the smaller story. The steady state is failure: four preemptions of one node in 36 h, model-invisible bad silicon (recurring Xid on the same PCIe address), c10d rendezvous pathologies, and a ∼25–40% throughput spread from NCCL ring-topology luck. All of it was absorbed by a fleet-consistent resume protocol that lost zero steps beyond the checkpoint interval. Capacity itself is a scheduling problem: DWS flex-start is a weekday instrument that vanished across a Saturday. The cost is dominated by the machinery that keeps a multi-node fleet consistent through arbitrary node loss, not by GPU-hours.

9 Serving: from checkpoint to four humans in a browser

The release ships no serving system; the one described here is original work: to our knowledge the only production *serving stack* for a multiplayer world model (Odyssey's Agora-1 [11] and the original MIRA demo both operate as research previews).

9.1 Architecture

Browser (React, canvas) ↔ *room relay* (FastAPI; one room = one GPU session; seats, tokens, key-state fan-in, frame fan-out) ↔ *GPU engine* (Modal "flash" WebSocket container, B200 or H100; one session per container, scale-to-zero). The engine wraps the release's streaming inference (20-latent rolling window plus KV cache) with: warmup-before-ready (the first-session compile killed cold connections otherwise), per-deploy context clips (priming on *real* gameplay frames, since synthetic gray context produces immediate rollout garbage), deploy-time environment pinning (checkpoint path, app name, deploy id), and health introspection used by the deployment gates.

Every deploy passes two gates before it is considered live. **Gate A**: the healthz endpoint must report *this* deploy id serving *this* checkpoint path (a still-warm previous container otherwise reports stale success). **Gate B**: a protocol-level probe drives a four-seat session and verifies frame integrity (JPEG magic bytes, per-view lengths) and throughput, dumping frames and a stats JSON as the artifact of record.

Delivered throughput against sampling steps (Fig. 2, all points measured by Gate-B probes): one-player views run 10.4 fps at 8 steps, up to 22.1 fps at 2 steps on B200. But 2-step sampling, while producing clean stills, *dissolves temporally* in autoregressive rollout (compounding error), making 8 steps the current quality floor and 4 steps marginal. Four tiled views cost ∼ 30%: 7.5 fps at 8 steps. Latent-frame decode is a constant ≈ 37 ms and is excluded from the sampling trade.

9.2 Playing all four seats: autopilot and live seat reassignment

The paper trains per-player action dropout so that masked seats are *imagined* by the model (§4.5, §6.8 of [8]). We expose this in production: unclaimed seats default to model-driven ("autopilot"), and the engine accepts live seat-mode changes mid-session: joining players reclaim their car from the model, leavers hand it back, and any player can click another car to drive it (their old car reverts to autopilot). Each client receives all four views (the grid UI); input routing follows the player, not the socket. Autopilot opponents chase, contest, and score with no scripted behavior. The motion prior *is* the bot.

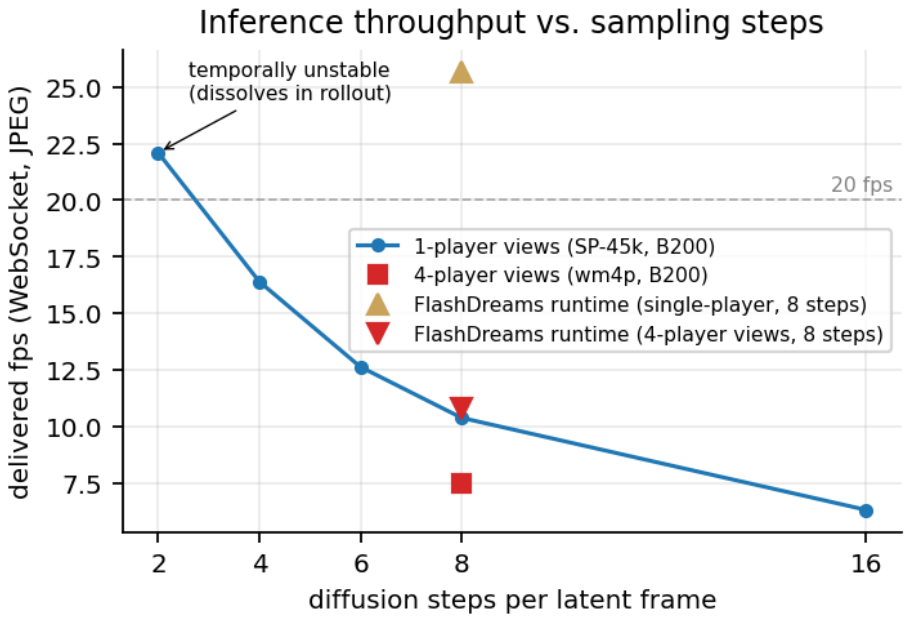

Figure 2: Delivered fps (WebSocket, JPEG-encoded) against diffusion steps per latent frame, measured end-to-end. The FlashDreams point (25.7 fps, single-player view, 8 steps, bit-exact) is the companion optimization work [10]; the four-view multiplayer fast path measures ~ 10.8 fps (§10.4).

9.3 Cross-view consistency: the observable frontier

The signature multiplayer defect at partial training is state divergence between views: each seat's view is locally plausible while the *shared* world disagrees. Most vividly, several seats each dream their own ball ("everybody gets a ball"), and HUD scoreboards and clocks drift apart. Qualitatively this improves with training (at 5k, scoreboards diverge within seconds; at 20–30k, clocks typically agree within seconds and score disagreements are intermittent).

Quantifying it is subtler than it looks. Our first metric is the mean pairwise cross-seat correlation of (a) the HUD score region and (b) the whole downsampled frame, over the Gate-B probe dumps. It is confounded: whole-frame similarity measures *view* likeness, not *state* agreement, and better-trained models legitimately diverge in viewpoint as players spread across the arena. The measured series (Fig. 3: HUD 0.834 / 0.797 / 0.797, scene 0.480 / 0.418 / 0.379 at 10k / 20k / 30k, $n = 8$ frame-sets each) is therefore reported as inconclusive: a caution against naive consistency metrics rather than evidence of regression. The correct instrument is the paper's game-state probe (§6.2 of [8]): read ball and car positions out of each view's latents and score *state* agreement directly, planned against the 40k/freeze checkpoints.

Can four humans play a 1B multiplayer world model from a browser?

Yes. The serving stack (a room relay over a scale-to-zero GPU engine, behind two deploy gates) runs four tiled views at 7.5 fps at 8 steps, with unclaimed seats on a model-driven autopilot and live seat reassignment mid-session. The open frontier is cross-view *state* binding: it is still emerging at 30k, and naive pixel-space metrics conflate viewpoint diversity with state disagreement, so the honest measurement waits on the paper's game-state probe.

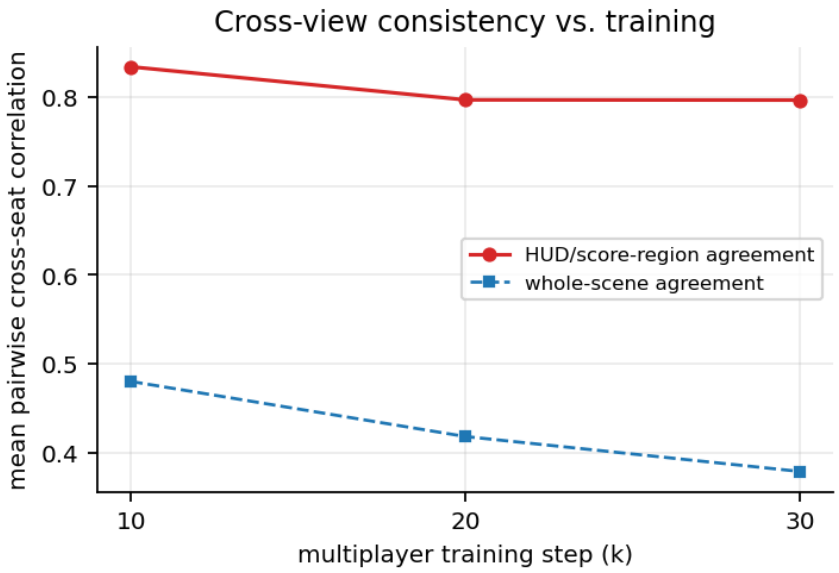

Figure 3: A cautionary result: naive pixel-space cross-view agreement conflates viewpoint diversity with state disagreement and cannot resolve the binding trend; see §9.

10 Controllability

The paper's central controllability instrument is the Action Recoverability Ratio (ARR, §6.2/§6.5 of [8]): a trained action probe recovers commanded inputs from generated video, validated against human judgment (Spearman $\rho = 0.93$ against Elo). Their 1B model's aggregate ARR climbs $0.63 \rightarrow 0.85 \rightarrow 0.90$ over 25k $\rightarrow$ 50k $\rightarrow$ 100k steps: controllability converges *after* visual quality, and rare actions last. Training that probe is future work (it also becomes an RL reward; see the companion briefs). For checkpoint-over-checkpoint tracking we built a behavioral proxy requiring no training.

Two probe designs were tried. *Sustained-hold probing fails.* Holding A for 3 s and measuring background pan is invalid under Rocket League's ball-cam (the camera tracks the ball, not the car's heading) and under sustained-turn orbiting (net pan ≈ 0). Both our 10k and 30k sustained-hold runs measured pan differentials indistinguishable from no-input drift: a metric-design failure, reported here because it is exactly the mistake a reproduction should warn about. *Pulsed impulse-response works.* We drive seat 0 with short signed steering pulses on a coasting baseline (5 frames W+A, then 5 frames W+D, period 30), segment the player's car by color, and cross-correlate the signed command trace with the car's lateral screen velocity. Authority is peak $|r|$; latency is peak lag; a no-input "coast" run under the same analysis gives the null.

At wm4p-30k (protocol v1: fixed 30-frame pulse period, 90-frame session, car tracked 90/90 frames) the overall reading is $r = +0.136$ (block-permutation $p = 0.204$; coast null $r = +0.003$): steering influence present but weak, consistent with the paper's ARR ≈ 0.65–0.70 zone at this depth. The per-window profile (Fig. 4) *rises* across the session ($0.00 \rightarrow +0.32 \rightarrow +0.57$), contradicting the intuitive hypothesis that control decays once the 2 s context window becomes fully self-generated; the felt "control fades" experience more likely reflects the motion prior taking over at speed (inputs fight believed momentum) than conditioning decay.

Protocol v2: aliasing fix.. A first 40k run under protocol v1 returned a polluted null (coast $r = +0.174$): the uncommanded car's orbit period aliased with the *fixed* 30-frame pulse schedule.

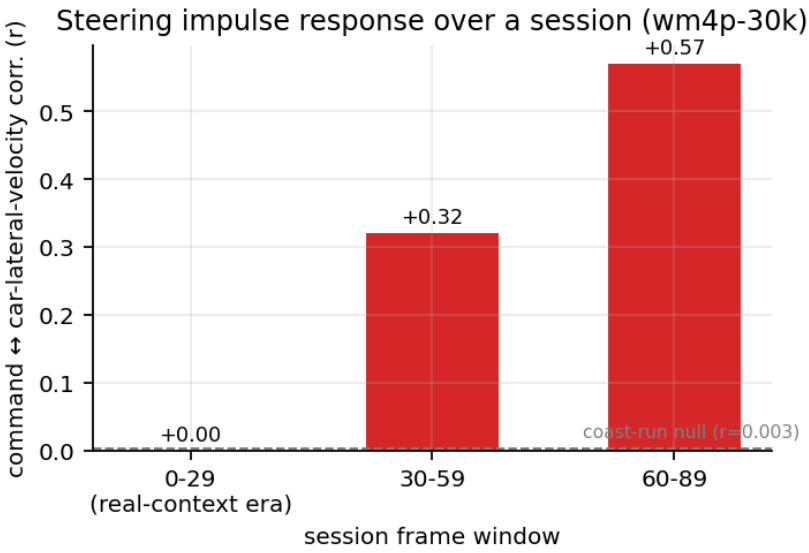

Figure 4: Command–response correlation by session window at 30k steps. The plotted curve is the v1 session-windowed authority run (the fixed-period run whose rise, $0.00 \rightarrow +0.32 \rightarrow +0.57$, is quoted in §10); the coast-run null (dashed) is clean. That v1 run is since superseded. `data/pulse_probe.json` archives the later v2 jittered-protocol re-run at the same checkpoint (authority $r = +0.462$ at lag 3, coast null 0.189, tracker valid on 21/120 pulse frames), a presence-only instrument demoted after SP calibration (§10).

v2 randomizes inter-pulse gaps (5-frame signed pulses, 10–22 coast frames between, alternating sign), extends sessions to 120 frames, matches throttle across conditions (coast holds W), and runs two independent replicates, each with its own coast null (Table 6).

Table 6: Pulsed impulse-response at wm4p-40k, protocol v2.

Replicate	authority r	lag (frames)	perm. p	coast null
A	+0.239	3	0.026	+0.062
B	+0.204	11	0.096	+0.071

Replicate A is the project's first statistically significant controllability reading ($p < 0.05$), at a lag (~ 3 frames ≈ 0.3–0.4 s) matching the known input-latency floor. Magnitude remains weak ($r \approx 0.2$ over nulls ≤ 0.07), the two replicates disagree on lag, and per-window profiles are noise-dominated: the instrument resolves *presence* and *coarse trend* (30k $\rightarrow$ 40k margin-over-null: $\approx 0.13 \rightarrow \approx 0.13$–$0.18$), not fine structure. This tracks the paper's mid-climb regime (their 1B ARR: 0.63 at 25k $\rightarrow$ 0.85 at 50k).

Discriminator: are model-driven opponents suppressing the commanded seat?. A plausible multiplayer-specific hypothesis: with three of four seats model-driven, the joint attention could resolve scene dynamics toward the prior's three-versus-one "vote," drowning the human seat's conditioning (the paper's ARR is measured on the *single-player* model, so this failure mode would be invisible to their curve). The test runs the identical v2 pulse probe with opponents `idle` (zero-key) instead of `model`. The result is $r = +0.204$ ($p = 0.072$): statistically indistinguishable from the autopilot condition. The hypothesis is not supported; steering weakness at this depth is training maturity, not multi-agent interference. Practically, this clears the model-driven-opponents feature for demo use at no controllability cost. (Caveat: single replicate; per-run null variance across all five 40k runs spans $|r| \approx 0.06$–0.22, so all readings here are coarse.)

A perceptual note reconciles instrument and player: $r \approx 0.2$ corresponds to steering explaining ~ 4–6% of the car's lateral-motion variance. It is *detectable* by correlation, *imperceptible* as agency. "It responds" reports should not be expected until the coupling rises severalfold; subjective reports of "zero control" at 30–40k are consistent with these measurements, not contradicted by them.

Running the identical v2 probe on the *single-player* 45k checkpoint (which human play rated clearly controllable) returned $|r| = 0.250$ ($p = 0.032$): statistically indistinguishable in magnitude from the "felt-dead" MP readings. The screen-space pulse probe therefore detects the *presence* of action coupling but cannot resolve the *degree* that separates a playable model from an unresponsive one (ball-locked camera geometry plus prior-driven motion dominate the pixel signal). Its cross-model comparisons are void; we retain it only as a cheap per-checkpoint presence check.

10.1 The instrument: seed-controlled action-divergence

The measurement that survives all the above confounds runs in the release's own inference path with no serving stack, camera proxy, or screen-space heuristic: two rollouts from an identical context and identical RNG seed, differing only in seat-0's held keys (hard-left-plus-throttle against hard-right-plus-throttle). Any trajectory difference is action-caused by construction. We measure relative L2 divergence between the two latent trajectories per 10 Hz step (Table 7, Fig. 5).

Table 7: Seed-controlled action divergence: single-player against multiplayer.

Model	@1 s	@2 s	@4 s
SP-45k (felt controllable)	0.51	0.95	1.22 (saturated)
MP-40k (felt dead)	0.19	0.48	0.90

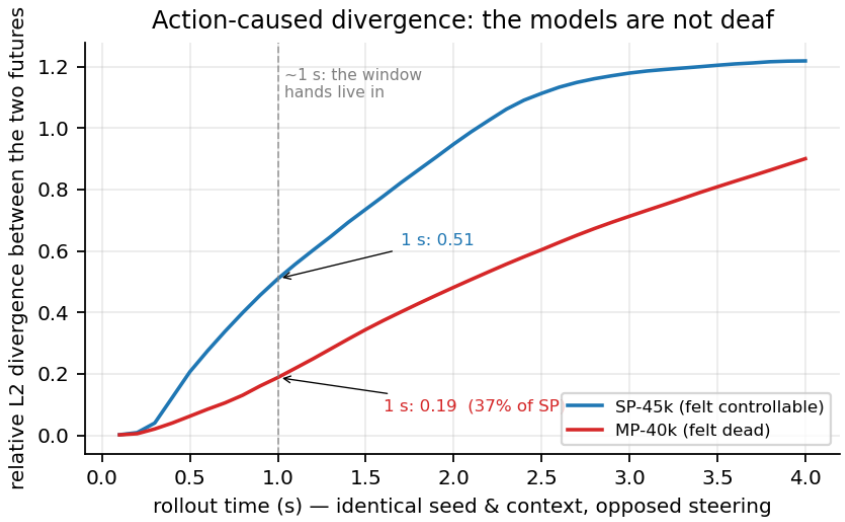

Figure 5: (Data: `data/earlywindow_divergence.json`.) Opposed steering bends both models' futures decisively. Neither is action-deaf. On this kickoff-context, single-seed pair, the multiplayer fine-tune at 40k retains $\approx 37\%$ of its single-player ancestor's early (1 s) action authority, the window that dominates the human feeling of control.

The verdict on the "training mistake" question is that no defect is found. The MP model obeys actions strongly in its native harness (90% divergence by 4 s). Its early-window coupling

is $\sim$ 2.7× weaker than the SP ancestor it warm-started from (consistent with the fine-tune temporarily suppressing conditioning while joint-view attribution re-forms), and that early window, compounded by the four-view latency floor ($\sim$ 0.5 s input loop) and a motion prior with strong opinions, fully explains "measurable coupling, imperceptible agency."

Re-running the protocol on every packaged MP checkpoint with context, seed, and inputs pinned gives the attestation ladder (Table 8, Fig. 6).

Table 8: Action-authority ladder across multiplayer training steps at three horizons. Rows 5k–40k are single-seed on the kickoff context; the 63k freeze row is three-seed (0.253/0.266/0.265) on the open-play context, so its @1 s step over 40k also reflects that context change (§10.2).

MP step	@1 s	@2 s	@4 s
5k	0.004	0.007	0.014
10k	0.046	0.160	0.452
20k	0.125	0.372	0.845
30k	0.204	0.438	0.865
40k	0.188	0.482	0.900
63k (freeze, open ctx)	0.261 ± 0.006	0.521	0.938
SP-45k ref	*0.51*	*0.95*	*1.22*

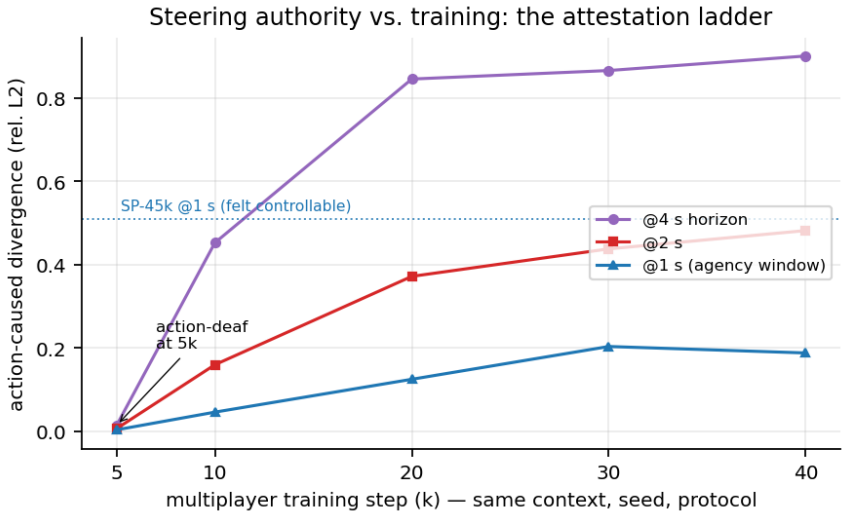

Figure 6: (Data: `data/earlywindow_divergence.json`; kickoff-context ladder.) Action authority against multiplayer training step at three horizons, single seed per point. The 5k checkpoint is essentially action-deaf (65× less total divergence than 40k); authority grows steeply to 20k, then the $\sim$ 1 s "agency window" plateaus at 30–40k (0.20 $\rightarrow$ 0.19; within single-seed noise) while later horizons continue to rise.

Two observations follow. First, the ladder independently reproduces the blinded human reports: 10k "can't control" (0.046), 20k "much better steering" (2.7× jump), 30k/40k "no difference, still feels dead" (early-window plateau at $\sim$ 0.19–0.20: $\approx$ 37% of the SP reference on the kickoff context; the open-context four-seed means give $\approx$ 45–47%, that is 0.239/0.509, §10.2; either way below the agency-perception threshold). The player's hands were a well-calibrated instrument all along; only their *binary* feel ("dead") hid the real monotone progress that the later horizons show. Second, the early-window plateau tempered freeze expectations honestly: if 40k→63k continued the 30k→40k trend rather than the 10k→20k trend, the frozen model would obey visibly on multi-second maneuvers but remain below the crisp-agency threshold in hand-feel. The demo-experience split (single-player for felt control; multiplayer for shared-

world spectacle) is planned accordingly. The freeze checkpoint has since resolved the conditional (§10.2): at 63k the open-context early window resumed climbing to 0.261 ± 0.006 (three seeds), above the 50k reading rather than continuing the 30k–40k flat, so the frozen model obeys visibly on multi-second maneuvers while the felt-agency band is now reachable through a clean guidance setting (§10.3).

10.2 Isolating the cause: single-player never plateaus

To test whether the ~ 1 s plateau is intrinsic to the handoff regime or specific to the multiplayer setup, we ran the identical instrument across the single-player ladder on one fixed SP context, and re-ran the multiplayer points on a fixed open-play context. The kickoff-heavy contexts of Fig. 6 suppress early authority ~ 30–43%, so isolating the mechanism required a context that does not (Table 9, Fig. 7).

Table 9: Early-window (@1 s) authority: single-player fixed context against multiplayer open context.

step	SP @1 s (fixed ctx)	MP @1 s (open ctx)
5k	0.008	–
10k	0.007	–
20k	0.284	–
30k	0.451	0.222
40k	–	0.230 ± 0.008 ($n = 4$ seeds)
45k	0.509	–
50k	–	0.239 ± 0.003 ($n = 4$ seeds)
63k (freeze)	–	0.261 ± 0.006 ($n = 3$ seeds)

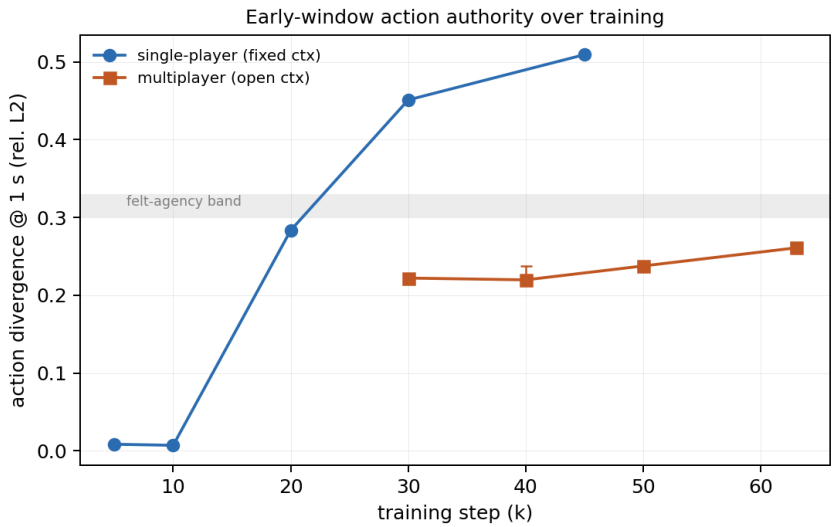

Figure 7: Early-window (@1 s) action authority against training step. The single-player ladder (fixed context) is action-deaf through 10k, inflects sharply between 10k and 20k, and climbs monotonically to 0.51. The multiplayer ladder (open context) is flat across 30k–50k. Data: `data/earlywindow_divergence.json`.

The 30k point is single-seed; 40k and 50k are seed-replicated after the single-seed 30k→40k comparison proved noise-sensitive (seed 1234 alone showed 40k *below* 30k, while the four-seed means show a slow climb). The 40k→50k gain holds across seeds (every 50k draw is $\geq$ the best 40k draw except one tie) at $\sim +4\%$ per 10k steps; @2 s means are flat ($0.501 \rightarrow 0.499$) and @4 s is

ceiling-noise. Interpretation: since 30k the steering profile is in a slow-grind regime, $\sim +0.01$ @1 s per 10k steps: moving, not capped, but no SP-style inflection through 50k. Boost remains at the divergence floor at 50k, consistent with the pre-registration below targeting first boost signal ≈ 63k.

The single-player model shows no plateau: early authority is action-deaf through 10k, inflects sharply between 10k and 20k, and climbs monotonically to 0.51, slowing only as it *saturates high*. The multiplayer flat spot is therefore multiplayer-specific. At matched 30k steps SP holds $\sim 2\times$ MP's early authority (0.451 against 0.222), and SP at *20k* (0.284) already exceeds MP at *40k* (0.230 ± 0.008, the four-seed mean; the single-seed 1234 draw is 0.220): multiplayer runs roughly $2\times$ behind single-player in step-efficiency on the early-window axis.

The mechanism is the recipe, working as designed, amplified by data frequency.. Figure 13 of [8] (restated in the lab's blog post [4]) gives the per-action decomposition: final ARR correlates with an action's frequency in the dataset at $r = 0.82$. Forward (the most common key) is near-ideal (≈ 0.99) already early in training, while rare actions (air-rolls, reverse, ≈ 5–7% frequency) converge last and lowest (≈ 0.80–0.85), and their own caption states that "controllability keeps improving well after image quality plateaus." Frequency is thus the paper-lab-certified first-order driver of per-key controllability. Stacked on it, the dropout recipe dilutes the signal further: per-player action dropout ($p = 0.1$) plus subset dropout (0.5, biased to the binary keys Q/E/Space/Shift/Ctrl; our config matches the release default exactly, §3) deliberately trains the model to predict each car *from the scene* when its actions are withheld: the theory-of-mind that makes the autopilot seats convincing (§9), paid for by rewarding the model for *ignoring* inputs. Together these predict, and we observe: (i) multiplayer needs $\sim 2\times$ the steps of single-player for equal steering authority (four-way dilution), and (ii) the binaries (boost, jump) lag steering and remain inert at 40k while steering is measurable: they are both rarer in the data and dropped twice as often by construction.

Pre-registered prediction for the freeze ladder. (written 2026-07-12, before the 50k and freeze measurements): mapping the blog's per-action SP curves through our measured $\sim 2\times$ MP step-lag, the $\sim$ 63k freeze corresponds to $\approx$ SP-30k per-action ordering: steering (W/A/S/D) clearly measurable and still climbing; boost (turbo, $\sim 20\%$ frequency) showing its first non-null divergence signal but well below steering; jump weaker; air-roll negligible. If the freeze measurement contradicts this ordering (for example, boost still at the divergence floor), the frequency-plus-dilution account is wrong or incomplete for MP.

Freeze verdict (measured 2026-07-13, after the prediction): confirmed.. At 63k the steering point is 0.261 ± 0.006 @1 s (three seeds): clearly measurable and still climbing, with the 2 s horizon resuming its climb ($0.499 \rightarrow 0.521$) after the 40–50k stall. Boost shows exactly the predicted first non-null signal: its mid-horizon divergence more than doubles (@2 s $0.085 \rightarrow 0.214$; @4 s $0.556 \rightarrow 0.701$) while the earliest window stays at the floor (@1 s 0.020). Air-roll remains untested (predicted negligible). Data: `data/earlywindow_divergence.json` (freeze_63k block).

So the early-window weakness is undertrained action-attribution, over-determined by three stacking factors and no defect: (a) the dropout recipe makes attribution the last-emerging capability, especially for binaries (intrinsic to MIRA, improves with steps); (b) we are 40–63k MP steps in against the demo's 100k and the paper's 150k baseline (improves with steps); (c) we are a 1B model against the demo's 5B (a lower ceiling that steps do not raise). Ruled out as causes, each explicitly: the 52k warm start (§6, did not cost MP steps), a dropout misconfiguration (matches the release 0.1/0.5), the codec (single-player drives cleanly through the same frozen codec), and the serving pipeline (§10 confounds eliminated end-to-end). The freeze

checkpoint and the 50k ladder point test whether contributor (b) is still actively closing at our step budget.

10.3 A serving-time control amplifier: action guidance

The same action dropout that dilutes control (§10.2) leaves a usable lever at inference. Because the model is trained with a learned "actions-absent" token, both the action-conditioned and action-absent velocity fields are in-distribution, so we can apply classifier-free-guidance-style extrapolation with no retraining: per diffusion step, run a two-row batch (conditioned; absent) and integrate the guided velocity $v = v_{\text{absent}} + w\,(v_{\text{cond}} - v_{\text{absent}})$. Here $w = 1$ is exactly the conditioned rollout: the sanity gate, which reproduces baseline (@1 s 0.236 against 0.238). On wm4p-50000 (open context, W,A against W,D, seeds 1234/1235) the sweep is monotone (Table 10, Fig. 8; the sweep curve is Fig. 9).

Table 10: Action-guidance sweep on wm4p-50000.

w	@1 s	@2 s	@4 s	visual (Figs 8, 10)
1.0	0.238	0.502	0.914	baseline, coherent
2.0	0.271–0.285	0.54	0.94	clean: stronger, coherent steering through 3.9 s
3.0	0.303–0.304	0.56	0.95	overdriven: artifacts by $\sim$ 4 s (sky mesh, edge smear, garbled HUD)

Figure 8: Seat-0 view at 2.5 s, rows = left (W+A) / right (W+D) steer, columns $w = 1/2/3$. The right-turn arc tightens monotonically with w while the arena stays coherent.

The gain is monotone, replicated across seeds, and, crucially, the relative lift is largest where the model is weakest (@1 s +15% at $w = 2$ against @4 s +3%), the signature of true action amplification rather than uniform noise. Visual inspection is the arbiter that divergence magnitude alone cannot be: $w = 2$ amplifies steering cleanly, whereas $w = 3$'s larger number is partly distortion (its extra divergence comes with visible artifacts). So $w = 2$ buys a clean $\approx +15\%$ early-window authority for free (roughly four training checkpoints' worth, §10.2 grind rate) at $\sim 2\times$ inference compute (two WM forwards per step; $\sim 11 \rightarrow \sim 6$–7 fps). It does not reach the SP reference (0.271 against 0.51), but it stacks with training and is a per-session toggle. To our knowledge this is the first application of action-CFG to amplify controllability in a multiplayer world model; the mechanism generalizes to any action-dropout-trained WM.

Figure 9: Steering divergence at three horizons (1 s / 2 s / 4 s) against guidance weight w, on the 50k checkpoint. The gain is monotone and largest where the model is weakest (the early window); $w = 2$ sits below and $w = 4$ clears the felt-agency threshold. Data: `data/earlywindow_divergence.json`.

The unlock ($w \approx 4$) and a felt-agency threshold.. Wired into the serving path (`?cfg=<w>`, shown in the HUD), the human playtest verdict was immediate and unprompted: $w = 4$ is where multiplayer becomes *steerable*, the first "I can drive it" report for the MP model. This reconciles with the metric: @1 s divergence climbs $0.239 \rightarrow 0.271 \rightarrow 0.304 \rightarrow 0.330$ for $w = 1/2/3/4$, implying a felt-agency threshold ≈ 0.30–0.33 that $w = 2$ sits under and $w = 4$ clears. Two objective checks confirm it is real steering, not amplified noise or expectation: (1) the divergence is *command-aligned* by construction (left-key and right-key futures diverge: the number *is* the response-to-direction); (2) a scripted identical-input probe (seat-1 driven L/R/L/R on the real serving path, Fig. 11) shows the $w = 4$ car banking visibly harder than $w = 1$ while the arena stays coherent. Crucially, interactive play dodges the sustained-hold artifacts the divergence frames showed at $w \geq 3$ (Fig. 10): real driving is short varied inputs that continuously re-anchor the scene, so the artifact ceiling that appears under a 4 s constant hold is rarely reached.

Figure 10: Seat-0 view at 3.9 s under sustained hold. At $w = 2$ the steering amplification stays coherent; at $w = 3$ the extra divergence arrives with distortion (sky mesh, edge smear, garbled HUD), which interactive play with varied inputs largely avoids.

Costs and limits, stated honestly: $w = 4$ runs $\sim$ 4–5 fps (against $\sim$ 11 baseline) and shows mild arcade-style drift and softening (heavy artifacts only at $w \geq 6$, where @4 s divergence

Figure 11: Seat-1 under identical L/R/L/R commands: top $w = 1$ (car barely deviates), bottom $w = 4$ (car banks and carves, coherent). Objective, scripted: no human in the loop.

regresses: guidance breaking the rollout, not steering it). Guidance also *helps* the binaries but does not unlock them like steering: boost @1 s triples ($0.029 \rightarrow 0.087$ at $w = 4$) and its mid-horizon grows ($0.12 \rightarrow 0.31$), jump lifts $\sim 1.5\times$, both still well below steering. Net: action-CFG at $w \approx 4$ converts the 50k multiplayer model from "obeys but feels dead" to "steerable" at a real framerate cost. It is a shippable inference-time knob that stacks on top of the training-depth gains, and reframes the multiplayer demo from spectacle-only to drivable.

Freeze re-test of the threshold (measured 2026-07-13).. On the frozen 63k model the guidance sweep gives @1 s $0.253/0.303/0.320/0.329$ at $w = 1/2/3/4$: the higher baseline (0.253 against 0.239 at 50k) lifts $w = 2$, the artifact-free setting, across the ≈ 0.30 felt-agency band: what took $w = 4$ at 50k now comes clean at $w = 2$, a gentler default for the demo. [pending: blind human A/B of the guidance toggle before "steerable" enters launch copy]

10.4 Accelerating the guided path: what the framerate costs

The framerate cost is only partly recoverable (measured, not projected). The FlashDreams CUDA-graph engine (denoise loop and KV shift graphed, noise drawn eagerly) was ported to the four-view model (milestone M4) and verified bit-exact (PSNR = ∞, all four players, eager *and* fast, on wm4p-50000). But the multiplayer speedup is only $\sim 1.24\times$ (reference $8.6 \rightarrow$ fd-fast 10.8 fps at 8 steps on B200), far below the single-player $2.75\times$ ($9.3 \rightarrow 25.7$ fps). The reason is structural: at four tiled views the codec decode dominates (~ 89 ms, $\sim 40\%$ of the per-step budget) and decode is *not* graphed in the reference engine. Cutting diffusion steps 8→4 barely moves fps ($10.7 \rightarrow 11.4$), confirming decode as the floor (a floor the kernel work below subsequently removed). An earlier projection that acceleration would carry guidance-$w = 4$ to $\sim$ 13–14 fps was therefore wrong (it extrapolated the SP factor); the honest figures are un-guided MP fast 11.1 fps (a free, lossless smoothness gain worth shipping), and guided w=2/w=4 on the fast path 8.1 fps (measured; the 2-row batch costs $\sim 1.4\times$ and decode stays single-row), against ~ 5.6 fps unaccelerated. The port is gate-verified bit-exact: stock parity, the w=1 bypass, and guided w=2/4 all reach PSNR = ∞ against a layout-normalized reference implementation of the engine's guided step (batch-2 bf16 kernels on B200 are memory-layout-sensitive at the 1–2 ULP-per-forward level, which guidance amplifies over autoregressive steps; the reference was contiguity-aligned to the served layout: a normalization, not a math change). Composed with the kernel work below, the guided path projects ~ 11.5 fps.

Decode overlap, built and tested, does not help on B200 (negative result).. The obvious lever was to run the per-view decode on a side CUDA stream overlapping the next step's denoise (the "free ~ 50 fps headroom" the SP design assumed but never measured). We implemented it, bit-exact (PSNR = ∞, real weights, fast path; clone-before-graph-overwrite plus a one-step flush), and measured no throughput gain (11.2 fps with and without overlap, clean on-GPU probe). The measured result stands regardless: stream overlap only helps latency-bound or under-utilizing kernels, and on B200 neither stage leaves the device idle: denoise is compute-bound, and the stock decode, as the profiling below shows, saturated the GPU with mask-materialization and

launch overhead rather than math. This corrects the SP-era assumption, and matches the paper, which reaches 20 fps ($\sim$ 70 ms per step end-to-end, denoise-through-decode, serial; §5 of [8]) with *no* decode overlap. *Postscript (2026-07-13, post-kernels): the result flipped.* Re-tested against the 15.7 ms kernel decode (below), whose SM profile no longer saturates the device, the same overlap now hides the decode entirely: serve latent $116.8 \rightarrow 97.3$ ms, room-path delivered $16.4 \rightarrow 18.2$ fps un-guided (11.8 at w=4), for one latent step (~100 ms) of added input latency. The zero-gain conclusion was correct for the 89 ms decode it was measured against, and stopped applying once the kernels changed that decode; we re-test a negative result whenever its premise changes.

Their 70 ms against our 178 ms decomposes into the two levers the paper actually names: (1) few-step diffusion distillation (PSD, applied on $\sim 10\%$ of updates) so that the denoise runs in $\sim$ 1–2 flow-matching steps instead of our 8 (the largest saving); and (2) custom Triton decode kernels (they use CuDNN for spatial/prefill attention and hand-written Triton for the decode stage) so that decode is far cheaper than our stock ViT decode ($\sim$ 89 ms). Both stages are made cheap and run *serially*, plus TorchInductor and CUDA graphs for host overhead (which we already have via FlashDreams). The implication is that a PSD-distilled, kernel-optimised 1B model should reach, even exceed, 20 fps, since it is smaller than their 5B; the gap is entirely recipe (distillation plus decode kernels), not scale or scheduling.

Custom decode kernels, delivered (measured 2026-07-13).. Profiling the stock four-view decode (88.7 ms) showed it was never compute-bound: only 6.5 ms is real matrix math, the rest a materialized causal attention mask forcing the slow cutlass path (28 ms) plus an elementwise/launch storm ($\sim$ 40 ms: 12k copy and 6.8k multiply launches). Each rung of the fix ladder was parity-gated at ≥ 40 dB against the stock decode (achieved ≥ 57.4 dB): `is_causal` in place of the materialized mask (76.9 ms), then `torch.compile` with CUDA-graph capture (30.4 ms), then a custom fused Triton kernel for the sequence-length-3 temporal attention (`fd_mira/triton_attn.py::causal_attn_short`): 15.7 ms, a $5.65\times$ decode speedup. End-to-end the four-view fast path moves $11.2 \rightarrow 18.0$ fps (step $179 \rightarrow 111$ ms) on the full-quality decoder, reproducing the paper's "custom Triton kernels for the decode stage." The one caveat is cold start: `torch.compile` costs 110–193 s, so production needs a persisted Inductor cache (or the no-compile fallback: the Triton kernel with cuDNN and bf16 RoPE, $\approx 1.5\times$ at instant startup).

The small decoder (§16) is an independent second decode lever, shipped as a live toggle: encoder bit-identical to the big codec; decode $89 \rightarrow 22$ ms, $\approx 4\times$; PSNR 29.8 / LPIPS 0.12 against the big decoder; $+40\%$ end-to-end at 8 steps on the reference engine. The two levers do not stack naively (the compiled graph is per-decoder architecture), so the serving stack picks one. The one remaining recipe lever is post-launch: PSD few-step distillation of the MP model.

Is the multiplayer model's weak felt control a training defect?

No. A seed-controlled divergence instrument (immune to the camera and screen-space confounds) shows the MP model obeys actions strongly (90% divergence by 4 s). Its early-window coupling is $\sim 2.7\times$ weaker than its single-player ancestor and plateaus at 30–40k; single-player, run through the same instrument, never plateaus, so the flat spot is multiplayer-specific and traces to the action-dropout recipe and action frequency, not a bug. At the 63k freeze the early window resumed climbing to 0.261 ± 0.006 and the pre-registered boost signal appeared exactly as predicted. Inference-time action guidance, exploiting the recipe's own absent-action token, crosses the felt-agency band with no retraining ($w \approx 4$ at 50k, and a clean $w = 2$ on the frozen model), at a measured framerate cost.

11 Limitations and honest deltas

1. *Behind the paper where it counts visually:* -1.1 dB codec PSNR; gFID $1.25\times$ at 45% budget. The deltas are consistent with the release-versus-flagship data gap; we have not demonstrated parity, only reproduction of the recipe's trajectory.
2. *1B, not 5B.* All results are at the paper's ablation scale.
3. *Multiplayer binding immature at 30k* (§9), the last checkpoint at which cross-view agreement was assessed. The demo-honesty line for launch: coherence *emerging*, not achieved.
4. *Action attribution is trained, not architectural, and at this budget it is visibly incomplete.* The wrapper mean-pools all four players' tagged action embeddings into one conditioning vector (`_combine_player_actions`: player embedding + projection, then `mean(dim=1)`) injected via AdaLN, spatially uniform over the tile; binding seat i's actions to view i must be learned, with per-player dropout as the curriculum. A seeded discriminator (all seats human, non-driven idle, so imagination cannot diverge; the only run-to-run difference is the commanded seat's keys) shows the leak at w=1: the commanded actions move *other* views (0.014–0.021 mean $|\Delta\text{RGB}|$ at 1 s) as much as the commanded view itself (0.017), and the leak is prior-shaped (it tracks scene coupling: a weakly-coupled view leaks 0.003), i.e. the mid-training model knows *that* an action happened better than *whose* it was. Live consequence: under a scalar guidance weight all four cars visibly obeyed the one human at w=4; serving now applies guidance per view-row (commanded views w, autopilot views 1.0; measured amplification 3.36× driven vs ≤1.93× undriven by a w=1-control protocol). The residual natural-strength leak is a property of the checkpoint, expected to sharpen with training, consistent with the ladder's monotone growth and the paper's rare-actions-recovered-last. Every instrument in this section watches the commanded seat; this class of defect is invisible to all of them and was caught by hands-on play.
5. *Controllability proxy ≠ ARR.* Our pulse probe correlates with neither the paper's probe nor human Elo yet; it is a trend instrument, not a comparable number.
6. *No distillation.* The paper's 20 fps demo uses few-step distillation (§5 of [8]); our 8-step inference is undistilled: ~ 7.5 fps at four views (≈ 10.4 fps single-view). (Companion work: the FlashDreams runtime reaches 25.7 fps at 8 steps on the single-player model and ~ 10.8 fps on the four-view multiplayer model, both bit-exact; see §14 and §13.)
7. *License scope.* The dataset is CC BY-NC-SA with Rocket League content by Epic's permission; our artifacts and demo are non-commercial research. A weights release (NC-SA) is pending a courtesy exchange with the original authors.

How close to the paper is this reproduction, honestly?

It reproduces the recipe's trajectory at 1B, not parity: -1.1 dB codec PSNR, gFID $1.25\times$ at 45% budget, controllability tracked by a trend proxy rather than ARR, multiplayer binding still emerging at 30k, and 8-step undistilled inference. Every result is non-commercial research on NC-SA data, at the paper's ablation scale.

12 Roadmap

The roadmap below is living; the freeze-dependent items are marked as pending.

- Freeze evaluations: the pulse-probe trend and the seed-controlled divergence ladder are in (§10, §10.2), and the game-state cross-view agreement probe is planned. MP gFID, measured at the freeze at the paper's mandated 10 sampling steps: **24.7** at the 4 s horizon (27.3 at 50k, improving with training; real-vs-real floor ≈6.0); at our 8-step serving setting the same protocol reads 28.7 / 29.0. gFDD: 0.262 at the freeze vs 0.313 at 50k (floor 0.032), measured in DINOv3-*Large* space and labeled as such. The paper's gFDD uses DINOv3-B (gated weights), so direction is comparable, scale is not. Both by the repo's paired-Fréchet protocol (generated vs the real continuation frames, feature 2048, 5,120 frames per side; a literal small-N "512-sample" FID is rank-deficient at this feature size; our implementation is a global paired Fréchet rather than the repo's per-window SlicedFrechetMetric: same family, not bit-identical). Alongside: the first 10-minute rollout on the multiplayer model (baseline drift 0.425, a 30 s all-cars-parked OOD melts it 4.1×, recovery to within 1.25× baseline in **10 seconds**, zero frozen frames end-to-end), the paper's hold-melt-snap-back signature reproduced at 52% budget. The paper's gFID references are 9.4–9.9 at 5B / 100k / ∼5× data
- A two-human playtest protocol and blinded controllability comparisons.
- Distillation (PSD knobs are config-present in the release) and the consumer-hardware path: companion report (§14, §16).
- Agents in the world model (the paper's first named application): behavior-cloned policies deployed as seats, then RL-in-imagination with a probe-based reward (project brief exists; not started).

What did the freeze settle, and what is still open?

The freeze resolved the controllability scorecard: the divergence ladder's early window resumed climbing to 0.261 ± 0.006 at 63k (§10.2), the pre-registered boost signal appeared as predicted, and the guidance sweep re-centered so a clean $w = 2$ crosses the felt-agency band. Still to land from the freeze are the 512-sample multiplayer gFID against the paper's 9.4–9.9 references (eval running) and the game-state cross-view agreement probe; the blinded two-human playtest, distillation to consumer hardware, and the agents-in-the-world-model program follow from it.

13 Serving runtime: the FlashDreams port

Per-frame cost factors into sampler steps, world-model FLOPs, decoder FLOPs, and hardware throughput; this half of the report attacks each with its own technique and its own gate, starting from the runtime that serves the trained 1.18B single-player checkpoint (§5). FlashDreams is NVIDIA's open inference framework for diffusion world models: CUDA-graph capture, static-shape fast paths, streaming KV management [10]. The paper's own serving stack is in-house ([8], §5); reproducing its throughput on a public framework makes the result portable, but only if the port provably computes the same function.

Parity as a hard gate.. We treat parity as a gate, not a similarity score. The reference implementation and the port run side by side from identical seeds, with RNG call order mirrored (the port must consume random numbers in exactly the reference's order: the subtle part of any such port, and where silent divergence hides), identical context frames, and identical action streams. The gate is per-frame PSNR between the two rollouts: ∞ (bit-exact) on all 60 frames of the parity rollout (a separate run from the 120-frame throughput bench), minimum/mean/last, for

eager mode at 8 and 4 steps and fused mode at 8 steps. Bit-exactness removes the confound entirely: every downstream measurement in this half is then attributable to the technique under test, not to the port. The same instrument later certifies the MLX port (§16, maxdiff 0.00000).

Throughput.. With parity locked, we fuse the sampling loop into two CUDA graphs (context-dependent and context-free segments) and overlap host work.

Configuration	fps @8 steps (B200)	Per-step breakdown
Reference implementation	9.33	latent 174.2 ms + decode 39.5 ms per latent step
FlashDreams, eager	9.25	generate 195.8 ms (incl. decode) + finalize 20.1 ms
FlashDreams, fused	**25.69** (2.75×)	generate 71.0 ms (~31 ms latent + ~39 ms serial decode) + finalize 6.2 ms

Conditions: 60 latent steps (120 frames), real context, warm pipeline, checkpoint `wm1b-sp-45000`, FlashDreams commit `862be4c8`. The fused profile makes the next bottleneck legible: at 8 steps the DiT still dominates (~31 ms), but at 2 steps (§14) the serial ~39 ms decode becomes the wall, which is what motivates the decoder compression of §15. (Scheduling the decode on a side CUDA stream is a measured negative: bit-exact but no throughput gain, because denoise and decode are both SM-saturated; §10.4.)

A production probe through the full serving path (a WebSocket producer paced at the model's native 10 Hz latent rate) sustains **20.53 fps delivered** (60/60 JPEG frames, mean frame interval 48.7 ms, p95 109.8 ms; the p95 sits at the latent-step boundary since each latent step emits 2 frames).

Serving and capacity.. Throughput by hardware tier is in Figure 12 (measured cells only). The 2-step distilled model turns the $1.95/hr L40S into a ~20 fps tier (the live WebSocket-paced serving probe reads 19.4 fps sustained), which is the launch tier; the B200 (~$11/hr) remains the premium/multiplayer tier. Seat latency has three measured regimes: a **warm seat** reaches first frame in 1.1 s (a pooled container accepts the session); **overflow self-seating** takes 62–70 s (at capacity, the client's auto-retry loop spawns and claims a fresh container; the platform does not queue, so the retry loop is the queue, shown in the UI as a countdown); and a **cold start** is 108 s on a fresh container after our fast-load work (down from 300+ s, the largest single win being removal of a DINO compile from the serving path, worth 210–260 s). For transport, WebRTC replaced WebSocket delivery after a measured 4 freezes/min on WS versus 0 on WebRTC under the same conditions; the production relay probe then sustained 19.9 fps for 2 minutes with zero dropped frames.

Does a public inference framework reproduce the paper's serving throughput?

Yes. A bit-exact FlashDreams port (PSNR = ∞ on all 60 parity frames) reaches 25.69 fps at 8 steps on B200 (2.75× the reference implementation) via two fused CUDA graphs, and sustains 20.53 fps through the full WebSocket serving path. Bit-exact parity is the load-bearing discipline: it makes every later compression result attributable to its technique rather than to the port.

14 Few-step distillation

The first cost factor is the number of sampler steps. The paper defines PSD ([8], §4.3): the model learns its own average velocities over integration windows via a midpoint two-hop,

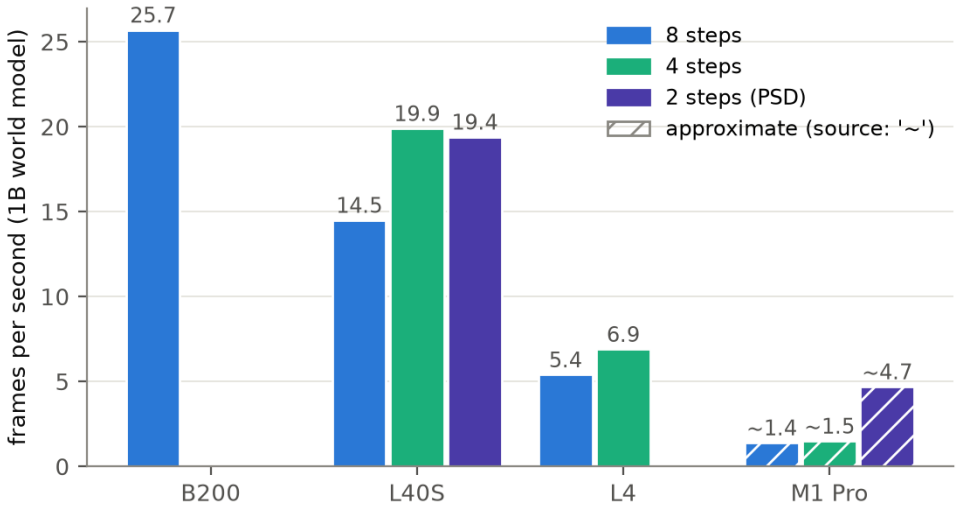

Figure 12: 1B world-model throughput by hardware and sampler steps (measured cells only; hatched = approximate in source; estimates excluded). Instruments differ by cell: the L40S 2-step bar is the live WebSocket-paced serving probe (conservative, since the pacing caps it), while the other bars are engine benches. The 2-step distilled model turns the $1.95/hr L40S into a ~20 fps tier (the launch configuration); the B200 (~$11/hr) remains the premium/multiplayer tier.

stop-gradient self-teacher, conditioned on a step size Δ alongside the noise level, with Δ=0 recovering standard flow matching. The paper applies it by *distilling its pretrained model* ("we distill the pretrained model for few-step sampling"), but its description ends where the engineering begins: a checkpoint trained with vanilla flow matching has no Δ-conditioning pathway, and the paper does not say how one is added without disturbing the trained function, nor at what learning rate the distillation runs. Our 45k checkpoint was trained without any Δ pathway. This section contributes the explicit warm-start construction and the finetune recipe; the objective and the distill-the-pretrained-model setting are the paper's (concurrent self-distillation/shortcut work: [2]).

14.1 Method

Zero-init Δ-embedding injection.. PSD conditions on the step size Δ via an embedding MLP added to the AdaLN conditioning stream. We inject this pathway into the trained checkpoint with its **output projection zero-initialized**: at load, the model is bit-identical to the base checkpoint (the new pathway contributes exactly zero), and the pathway fades in through training. Two instruments verify one structural claim (Figure 13): at the first backward pass, gradients *upstream* of the zero projection are exactly 0.0 while the projection itself receives gradient (first-backward grads 269.9 vs 0.0), so the zero-init gates the pathway's *output*, not its *learning*. The projection's per-step grad-norm then spikes as it moves off zero (step 2) and settles within ~10 steps.

Objective and recipe.. As in the paper: sample $s < u < t$ and teach the $s \to t$ jump to match the stop-gradient two-hop $s \to u \to t$; Δ=0 recovers flow matching; the term is mixed stochastically into ~10% of steps. The finetune recipe is the hard-won part. The from-scratch learning rate (1e-4) **diverges on warm-start**: validation 0.35 rises to 2.7-scale spikes within the first thousand steps, which we attribute to optimizer shock (fresh second-moment estimates meeting a converged loss surface). The recipe that works: **lr 3e-5, warmup 100,** `torch.compile` **off,** `activation_checkpointing=psd-only` (the PSD student pass otherwise OOMs an 80 GB

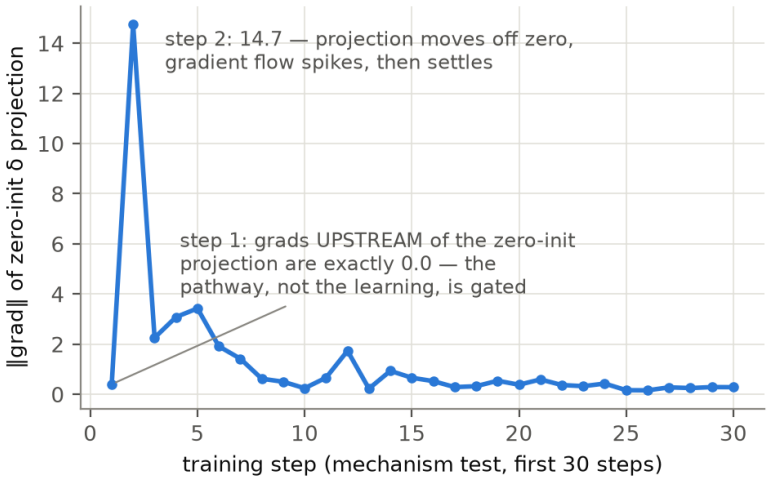

Figure 13: Mechanism-test signature of the zero-init Δ injection (first 30 training steps, per-step grad-norm of the δ output projection). At the first backward pass, gradients upstream of the zero projection are exactly 0.0: the projection's zeros gate the pathway's output and its upstream gradient flow, not its own learning; as the projection moves off zero the gradient flow spikes (step 2) and settles within ~10 steps. This is gate A3 of the mechanism test.

H100).

14.2 Evaluation instruments, and two we rejected

The positive gates for this work are three. *Do-no-harm validation parity*: the model's ordinary diagonal validation loss, measured before and after, gated at "no regression." *Teacher-forced LPIPS at k steps* [15]: teacher-forcing on held-out clips means context is always ground truth, so errors do not compound and frames are comparable pixel-to-pixel; this is the primary few-step gate, and the decision variable is the *delta* at matched step count. *Free-running instruments* are descriptive only: side-by-side and 2×3-grid rollout videos, finiteness/step-ordering checks, and a *bitrate proxy* for texture retention (JPEG-compressed size of generated frames, which collapses when a model smooths detail away).

Two instruments were rejected, and the rejections shaped the recipe. *Free-running per-pixel PSNR, as a gate*: two rollouts from the same model with infinitesimally different sampling are macroscopically different videos seconds later, so per-pixel PSNR measures chaos, not quality. In the first mechanism-test run it mis-scored the PSD variant (psd@2 vs base@2: 16.9 dB, indistinguishable from noise across reruns), failing a model whose teacher-forced quality was healthy; it was demoted to descriptive. *PSNR of any kind for few-step ordering*: PSNR rewards blur; a 1-step sample is smoother than an 8-step one and wins PSNR while losing LPIPS (base@1 has the best psnr_gt 24.55 and the worst lpips_gt 0.1242), so ordering few-step variants by PSNR selects for exactly the failure mode distillation must avoid. A third gate was recalibrated rather than rejected: a fixed "head drift $< 5\%$" threshold actually encoded "the finetune shouldn't train" (the healthy drift at 300 steps was 8.4%; a random head sits at ~140%), so it was re-scoped to "drift $\ll$ random."

14.3 Mechanism test ($4, 301 steps, one H100)

Every training technique climbed a spend ladder: CPU smoke test → local real-weights run → $4 rented-GPU mechanism test → $110 8×H100 pilot → fleet adoption. The $4 rung ran 301 real steps against seven named gates in three families (A: fade-in mechanics; B: trainer inte-

gration; C: sampling behavior). Its first run **failed B and C**: the EMA snapshot was taken after Δ injection rather than before (fixed), and the C-gate was the free-running PSNR instrument later rejected. The v2 run passed all seven in 20.2 wall-minutes (val-loss ratio 0.999). Only then did the pilot launch. The rung is load-bearing. For $4 it caught three launch-killers: the PSD second forward pass OOMs an 80 GB H100 without `psd-only` activation checkpointing, dataset access needed authenticated pulls, and the free-running PSNR gate defect above.

14.4 Pilot: 10k finetune steps on 8×H100 (spot)

Training health (Figure 14).. Diagonal validation loss went $0.3953 \rightarrow$ **0.3741**: the do-no-harm gate held and in fact improved. (Whether the PSD term genuinely regularizes the base objective or the improvement is simply continued training at the lower finetune LR is not something our experiments separate; we note the observation without claiming the mechanism.) The PSD component itself is noisy-small throughout ($\leq$0.012), as expected for a term mixed into ~10% of steps.

Few-step quality (Figure 15).. The decision variable is teacher-forced LPIPS at 2 steps, PSD minus base. It improves monotonically through the pilot (-0.0001 at 2.5k, -0.0005 at 5k, -0.0021 at 10k), so the distilled model at 2 steps ends *better* than the base at 2 steps, and within 0.0012 LPIPS of the base at its native 8 steps. At 1 step the delta is -0.0053. The PSNR column ordering inverts the LPIPS ordering (base@1 wins PSNR with the worst LPIPS): the rejected instrument, visible in the data.

Checkpoint	Steps	psnr_gt (dB)	lpips_gt ↓
base (45k)	8	23.88	0.1195
base	4	24.11	0.1186
base	2	24.43	0.1228
base	1	24.55	0.1242
psd10000	8	23.66	0.1222
psd10000	4	23.79	0.1202
psd10000	2	23.86	**0.1207**
psd10000	1	24.17	0.1189

Eval shard: one held-out match, 1.81 GB; val_loss_diffusion on the same shard is base 0.4086, psd10000 0.3902. Free-running 2-step rollouts visibly retain texture the 2-step base smooths away; the bitrate proxy puts the effect at +5–27% across clips. A human playtest on the serving tier passed at 19.4 fps sustained on the L40S ("looks damn fine").

14.5 What this section claims, and what it does not

Claimed: a checkpoint trained with no Δ-conditioning pathway can acquire the paper's few-step property in 10k finetune steps (≈4 h on 8×H100, ~$110 all-in including the diverged attempt) with no measured damage to base capability, via a warm-start construction verifiable at every stage. Not claimed: parity with the paper's own distilled model (we have no such baseline); transfer of the exact LPIPS deltas beyond this model/data scale; and the deltas themselves are measured on a single held-out match shard (1.81 GB), so wider eval coverage precedes any external use of the -0.0021 figure. Free-running quality evidence at 2 steps is descriptive (videos + bitrate proxy + one passed playtest); the paper's distributional metrics (gFID/gFDD) [7] and per-action ARR on the distilled variants (deferred at the time of this pilot) are now run across the full compression ladder in §17 (FVD excepted; the harness implements none).

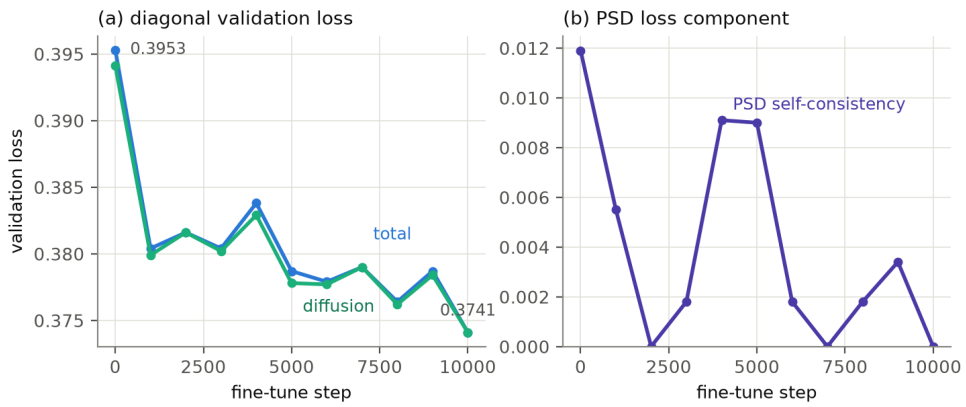

Figure 14: PSD warm-start pilot, 10k finetune steps of the 1.18B model on 8×H100. (a) Diagonal validation loss (total and diffusion components): the base capability improves during distillation, 0.3953 → 0.3741. (b) The PSD self-consistency component, noisy-small as expected for a stochastically mixed term. The diverged lr=1e-4 first attempt's logs did not survive the node relaunch and are reported from the campaign brief (secondary source).

Can a checkpoint trained without few-step sampling acquire it after the fact?

Yes. A zero-initialized Δ-pathway injection makes the warm-started model bit-identical to the base at step 0, and a low-LR (3e-5) 10k-step finetune yields a 2-step model that is better at 2 steps than the base at 2 steps (teacher-forced LPIPS −0.0021) while its ordinary validation loss improves (0.3953 → 0.3741). The injection construction and finetune recipe are ours; the PSD objective and the distill-the-pretrained-model setting are the paper's. The evidence is single-shard, and wider eval precedes any public claim.

15 Cross-scale distillation: the 364M student

The second cost factor is the DiT's own FLOPs. Rather than train a smaller model on data, we distill the now-PSD-capable 1.18B teacher into a **364M student**: hidden 1280, 12 layers, 10 heads with head_dim kept at 128 so RoPE and attention conventions match the teacher exactly; 3.2× fewer FLOPs.

Method.. The student regresses the teacher's velocity predictions on shared (z_t, τ, clean-past) draws (both models see identical noised latents and diffusion times) with an anchoring ground-truth flow-matching term at weight 0.25. The teacher rides along frozen, attached via `object.__setattr__` so it is invisible to DDP wrapping, EMA, and `state_dict` (checkpoints stay student-only; a checkpoint cannot accidentally ship 1.5B parameters). Distilling from the PSD teacher hands the student the best available velocity field and a head start on its own stage-2 PSD finetune.

Stage-1 (40k) and stage-2 PSD (10k).. Stage-1 (Figure 16): the distillation loss (teacher-velocity MSE) falls 2.53 → 0.054, and the ground-truth flow loss falls 2.90 → **0.4251**, against the 1.18B teacher's own converged 0.3741. A model 3.2× smaller finishes 0.051 above its teacher. One spot preemption at step ~12.5k; the run resumed from checkpoint with a re-validation at the resume point (both recorded; the pre-restart value is plotted). Stage-2 then applies the §14 recipe verbatim (warm-start injection, lr 3e-5): step-0 validation is 0.4231, and the model is the stage-1 40k checkpoint bit-identically, so the 0.002 offset from stage-1's own 0.4251 is eval-batch noise between runs, not drift. The diagonal then holds ~0.41 (do-no-harm) and the PSD

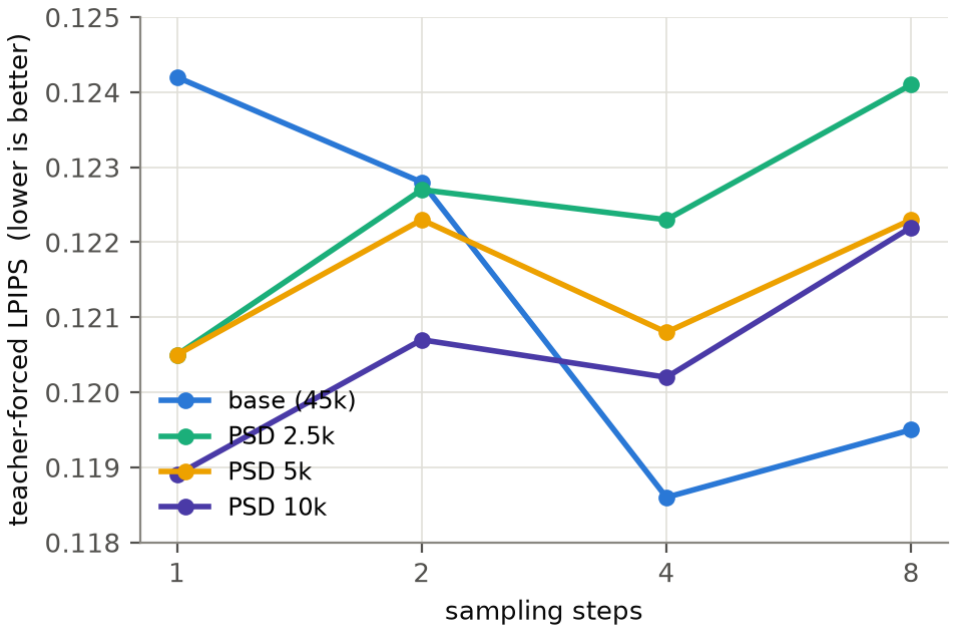

Figure 15: Teacher-forced LPIPS across sampling steps for the base checkpoint and three PSD pilot checkpoints. The base degrades sharply below 4 steps; PSD checkpoints flatten the few-step end of the curve, crossing below the base at 2 steps by 10k finetune steps. The base retains its advantage at its native 8 steps, as expected: distillation trades peak many-step fidelity for few-step usability.

self-consistency term distills to 0.0000, i.e. the student is few-step-capable. This stage-2 student is the DiT shipped in the on-device stack (§16), which measures its 1- and 2-step fidelity and throughput directly.

15.1 Decoder-only compression

At 2 sampling steps the decoder becomes the wall (§13): ~39 ms of the ~70 ms latent step on B200, ~53 ms on L40S, and 1,337 ms per latent on M1 (ViT-XL at MPS fp16). The codec's encoder defines the latent space every world model was trained in; the decoder is only a renderer. So we retrain **only the decoder**, at 576 width × 14 depth (from 1152×28, ~8× fewer FLOPs), against the **frozen** encoder.

Swap-safety by construction.. The encoder's weights load strictly and bit-identically; the DINO-latent consistency loss is ≤0.001 from step 0 and settles to 0.0002 (it measures a frozen pathway). The latent stream is therefore unchanged, and *no world model can distinguish the decoders at its input*: the cheap decoder retrofits to every checkpoint we have, including the multiplayer model still training and any future one, with zero retraining and zero dynamics risk. This is the reconstruction-gate instrument of the evaluation set (recon MAE/LPIPS curves plus the two structural gates), and it is what lets decoder-only distillation compose with the previous two levers rather than multiply their risk.

Final (60k, Figure 17): reconstruction LPIPS 0.891 → **0.179** and MAE 0.542 → 0.057; the curve flattened past ~40k (0.228 at 40k → 0.203 at 47.5k → 0.179 at 60k). Decode effect, now measured on M1 via Core ML: 116.3 ms per 3-latent chunk (§16), or ~38.8 ms per latent versus 1,337 ms for the ViT-XL under MPS-torch (with the caveat that runtime and model size change together in that comparison). The final decoder is exported and shipped in §16's on-device stack (bundle-v2); cloud-tier per-latent decode (L40S 53 → ~8 ms *est*, L4 95 → ~15 ms

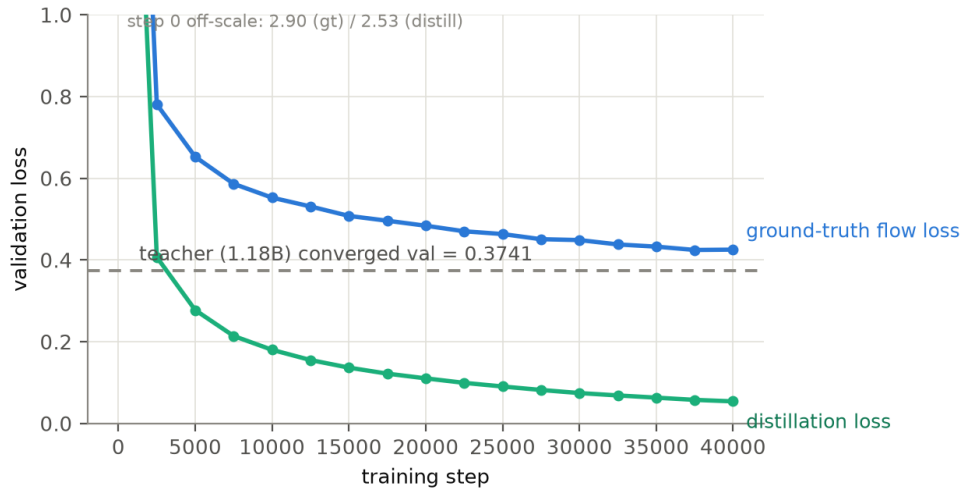

Figure 16: Student stage-1 validation losses to the final 40k. The distillation loss (teacher-velocity MSE) falls 2.53 → 0.054; the ground-truth flow loss falls 2.90 → 0.425, closing on the 1.18B teacher's own converged 0.3741 (dashed): a 3.2×-smaller model within 0.051 of its teacher. One spot preemption at step ~12.5k; the dashed teacher line is the teacher's converged validation loss from its own PSD-pilot run (same held-out protocol, but a cross-run comparison, not a same-batch eval).

est) remains projected pending per-tier benches. The L40S 2-step serving stack as a whole is measured (19.4 fps live probe), but its per-stage decode split is not.

Can the DiT and decoder be shrunk without retraining the world model or moving its latent space?

Yes to both, separately. A 364M student regressing the PSD teacher's velocities reaches ground-truth flow loss 0.4251 at 40k (0.051 above the 3.2×-larger teacher), then distills a 2-step variant (PSD term → 0.0000). A 576×14 decoder retrained against the frozen encoder reaches reconstruction LPIPS 0.179 at 60k; because the encoder is bit-identical, the cheap decoder is provably swap-safe under any world model, current or in-flight.

16 On-device deployment

The M1 MacBook (M1 Pro, 16 GB, 2021) is the forcing function: old, thermally limited, and owned by millions. Every lever that gets a world model onto it pays for itself on the cloud tiers first.

16.1 Runtime ports

MPS-torch (correctness first). The full serving stack (engine, relay, production web UI) runs locally with device fallback cuda → mps → cpu and fp16 autocast. The 1B model at 8 steps runs at 0.13 fps: a loop proof, not a product; its purpose was to validate the plumbing end-to-end before optimizing anything.

MLX (speed). An MLX port of the DiT: maxdiff 0.00000 against torch (at the parity harness's five-decimal print precision, so "exact to display precision" rather than proven bit-exact) on both the prefix and KV-cached step paths. Per-forward at real serving shapes: 137–141.8 ms fp16 (two sessions' measurements, both recorded) versus 145 ms for torch-MPS eager. int4 quantization is *not* faster (149 ms): at 144-token KV-cached shapes the DiT is compute-bound,

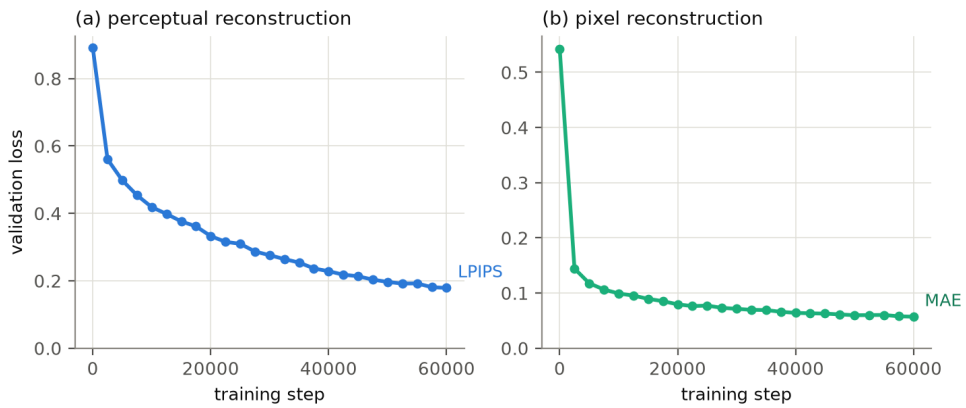

Figure 17: Small-decoder (576×14) reconstruction losses against the frozen encoder's latents, to the final 60k. LPIPS 0.891 → **0.179** and MAE 0.542 → 0.057; the curve flattened past ~40k. The DINO-latent consistency term (not plotted) stays ≤0.001 throughout, settling to 0.0002: the frozen-encoder pathway, i.e. the swap-safety invariant.

so FLOP reduction (the student, §15) is the only lever on this axis. One diffusion step (2 forwards) ≈ 275 ms, a ~7 fps ceiling for the 1B at 2 steps latent-only; hence the student.

Local package. `pip install alakazam-mira-mini && mira-mini play`: weights auto-download, engine + relay + web UI in one process. The Mac rehearsal passed end-to-end from a fresh venv; five packaging defects were found and fixed (dependency pins, a `torch.compile`-on-MPS crash fixed by monkeypatching compile to identity, a router double-prefix 404, a stale-wheel footgun, and a config-localization path miss). Linux and Windows rehearsals are scheduled with the same harness.

16.2 The ANE compiler investigation

Core ML is the route to Apple's GPU/ANE. Conversion itself is easy (`torch.export` → `run_decompositions({}` → strip `aten.alias` nodes coremltools cannot lower → convert). The problem: on first load, Apple's `ANECompilerService` pegged a core at 100% for longer than every deadline we allowed (>15 min observed), and that on the unmodified decoder. We bisected under a guarded runner (every attempt in its own process group with a hard deadline; on expiry, kill the group, kill any hot `ANECompilerService` by PID, clear the compile cache, so the machine is never left wedged). The probe ladder, all at the decoder's real shapes:

Subgraph	Result (ComputeUnit.ALL, macOS14)
Linear / conv-lift / SwiGLU / unembed	compile ≤1 s, parity 54–87 dB: clean
Space attention (batch 3, seq 576)	2.6 s convert: clean (more FLOPs than the trigger, tiny batch)
Time attention (batch 576, seq 3)	**compile-cost blowup**: the trigger
Same, batch 32	148 s convert: passes a 180 s deadline, but ~2 orders over normal
Same, causality removed	296.0 s ANE compile vs 284.5 s causal: mask-independent
Space attention pattern at batch 576	296.0 s: the blowup follows batch width, not the pattern
einops rearranges (in surgical rungs)	retained verbatim and compile fine: acquitted

The pathological construct is SDPA with a wide batch dimension in the ANE partitioner/compiler, mask-independent and pattern-independent (the space pattern at batch 576 blows up identi-

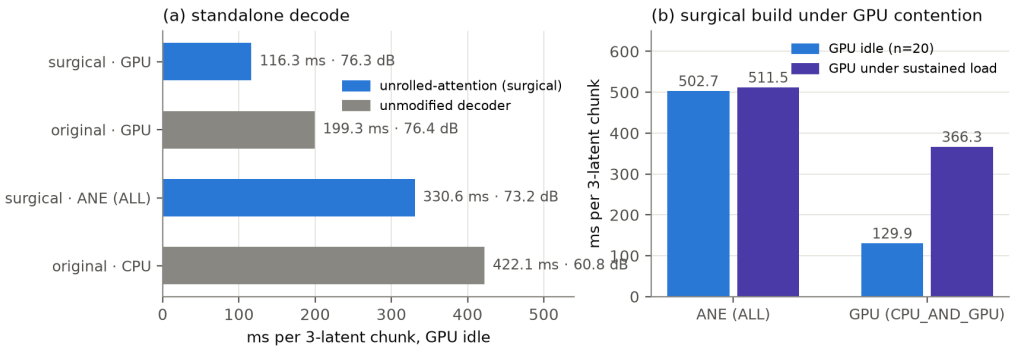

Figure 18: Full-decoder Core ML latency on M1 Pro (fp16, one 3-latent chunk → 6 frames at 288×512). (a) Standalone decode (GPU idle), by build and compute unit: the surgical build on CPU+GPU is fastest (116.3 ms) and avoids the ANE compile (2.5 s to first prediction vs ~4 min). (b) The surgical build under sustained GPU load: the ANE path degrades only 1.7% while the GPU path degrades 2.8×. Parity is against torch fp32. Caveat: this contention-resistance does not make the ANE the better choice. It is ~4× slower in absolute latency even idle, and the fuller 60k bench selects CPU_AND_GPU as the deployment path.

cally while compiling in 2.6 s at its native batch 3). The decoder hits it through its temporal attention, which runs $(b \cdot h \cdot w) = 576$ sequences of length 3.

16.3 The surgery

`TimeAttnUnrolled` re-expresses the T=3 attention as elementwise ops: per-step RoPE, dot products as multiply-sum, softmax over ≤3 scalars, weighted sums, with no `sdpa`, no batched matmul anywhere in the subgraph. Critically it is a **graph rewrite, not a reparameterization**: it consumes the original module's `wqkv/q_ln/k_ln/wo` tensors by reference, so weight fidelity holds by construction and the rewrite applies verbatim to any future checkpoint of this architecture. Parity against the original module: **1.83e-06 maxdiff** (fp32, full decoder), which is reassociation noise, five orders of magnitude below feature scale.

One methodological caveat, sharpened by the adversarial review, scopes the parity-in-dB ladder: on *unbounded* intermediate tensors, range-based PSNR flatters the number (the range of a $\sim 10^6$-element Gaussian tensor is $\approx 10\sigma$, inflating range-based PSNR by $20 \cdot \log_{10}(\approx 10) \approx 20$ dB over σ-based SNR, which is a derivation, not a measurement); on the tanh-bounded decoder output it is honest. Isolated-subgraph parities are therefore treated as diagnostics; only end-to-end parity gates.

Build	Compute units	Convert	First predict	ms/decode	Parity
original	ALL (implicit)	blowup (killed ≥180–420 s)	–	–	–
original	CPU_ONLY	12.6 s	1.8 s	422.1	60.8 dB
original	CPU_AND_GPU	14.0 s	2.2 s	199.3	76.4 dB
surgical	ALL	254.8 s	236.2 s (one-time)	330.6	73.2 dB
surgical	**CPU_AND_GPU**	**20.6 s**	**2.5 s**	**116.3**	**76.3 dB**

The deliverable (`ane_export.py`) is a weight-agnostic export recipe with all gates inline (torch parity hard-fails at 1e-3; Core ML parity gated ≥30 dB), verified end-to-end on the real checkpoint in 46 s under the guarded runner. At 116.3 ms per 3-latent chunk (6 frames, Figure 18a), decode ceases to be the standalone-Mac bottleneck: ~38.8 ms per latent, 34× under the ViT-XL MPS baseline.

Is the ANE actually used, and does it matter?. Two follow-ups answer what the idle benches could not. First, an `MLComputePlan` per-op census of the surgical build at `ALL`: 2,663 of 2,766 ops (96%) are ANE-resident, 103 on GPU, zero on CPU. The `ALL` numbers measure genuine ANE execution, not partitioner fallback. Second, a contention probe (the question that matters on-device, where the GPU is never idle because it runs the DiT), benching each config while a sustained MPS fp16 matmul load occupies the GPU:

Config	idle (n=20)	under GPU load	degradation
surgical @ ALL (ANE)	502.7 ms	511.5 ms	+1.7%
surgical @ CPU_AND_GPU	129.9 ms	366.3 ms	2.8×

(The 502.7 ms idle figure vs the ladder's 330.6 ms reflects n=20 vs n=5 and session state; both records kept.) The ANE path is contention-resistant (+1.7% vs the GPU path's 2.8×), but it is also ~4× slower in absolute terms even idle.

Correction (final-checkpoint re-measurement). With the 60k decoder and the finished PSD student, a fuller bench changes this subsection's original conclusion (M1 Pro, 60k decoder):

Config	idle	under GPU load	parity
CPU_AND_GPU	115 ms	276 ms	73.2 dB
ANE (`ALL`)	341 ms	434 ms	62.0 dB

CPU_AND_GPU decode is faster in **absolute** milliseconds *both* idle and under contention: the ANE's smaller percentage-degradation never overcomes its ~3× higher baseline. An earlier draft leaned on "contention-immune ⇒ ANE is the deployment path"; that over-weighted the degradation *ratio* and under-weighted raw latency. The corrected verdict: for the concurrent stack today, CPU_AND_GPU decode is the deployment choice, not the ANE. The ANE's one remaining theoretical advantage (running off-GPU so it does not slow the DiT) was not cleanly isolated (the integrated pipelined A/B hit a flaky decode-child bug), and the component data favours CPU_AND_GPU.

Op-cost breakdown (where does the 341 ms go?). An `MLComputePlan` op profile of the 60k decoder at `ALL` is dominated by 742 `mul` + 402 `reshape` + 252 `add` (the unrolled attention's elementwise dot-products, softmax, weighted sums, RoPE, and einops reshapes) against only 99 `linear`. The bisection ladder confirms the attribution: one unrolled time-attention benches 9.2 ms and a full block 13.3 ms, so attention is ~69% of each block's ANE cost and the Linears (all that `linear`→`conv2d-1×1` shaping touches) sit in the other ~31%. Even a 2× on the Linears is ~15% overall (341 → ~290 ms), not the ~200 ms an earlier draft projected. The ANE decode is attention-bound, not Linear-bound; conv-shaping cannot deliver the target, so we did not build it. The root tension is structural: the batched-SDPA attention that is *fast* is exactly what wedged Apple's compiler; the unrolled form that *compiles* is elementwise-heavy, the ANE's worst case. A fast ANE decode needs an attention that is *both* compiler-safe *and* conv/matmul-shaped, a research problem, not a few-hour transform. CPU_AND_GPU decode (115 ms) is the deployment choice; the ANE path is set aside pending that research. A 10-minute measurement retired a multi-hour rewrite that would not have paid off.

1-step viability (final stack).. The distilled student at 1 diffusion step is 97.5% as sharp as at 8 steps (Laplacian variance on aligned frames), for a modest perceptual cost (LPIPS-vs-8-step 0.112 at 1 step vs 0.079 at 2); the perceived softness is the 364M model + distilled decoder, not the step count. The DiT is 1.5× faster (112 ms vs 168 ms latent), so 1-step is viable for interactive play (~12 fps live vs ~8 at 2 steps); 2+ steps are retained for hero renders. At 2 steps the 168

ms DiT hides the 115 ms decode trivially, but at 1 step (112 ms) decode and DiT are the same magnitude, which is what makes decode the co-bottleneck.

16.4 The assembled stack (end-to-end, running today)

All components are assembled inside the production engine and measured end-to-end: the student DiT on Metal via a parity-gated MLX streaming sampler (max drift 4.1e-05 vs the production torch sampler over 8 streaming steps, identical injected noise), the small decoder on the ANE in a dedicated *decode child process* (a worker thread is not enough: coremltools' predict holds the GIL through ANE execution; a "hidden" decode re-appeared as ~700 ms of main-thread stall until the decode moved out of process), and torch retained only for the one-time context encode.

Configuration (M1 Pro, 2 diffusion steps)	latent	decode	step	fps E2E
overlapped decode (per-step overlapping windows)	206.5 ms	269.7 ms	483.6 ms	4.14
+ amortized non-overlapping 3-latent chunks	194.4 ms	**1.4 ms**	202.9 ms	**9.86**

Amortization removes the 3× redundant decode work of per-step overlapping windows (cost: up to 3 latent steps of added visual latency); decode then runs entirely in the DiT's shadow and the stack is DiT-bound. Same-day ladder: 0.13 fps (1B, torch-MPS, ViT-XL decoder) → 9.86 fps, a 76× gain in one day. On the final checkpoints (stage-2 PSD student + 60k decoder) the live server sustains ~**8 fps at 2 steps and ~12 fps at 1 step** (below the 9.86 bench because the live path adds WebSocket transport and the sampler's cross-thread executor round-trip). A human playtest confirmed 1-step "runs well" but "less stable" than 2-step, the felt corroboration of its higher per-step LPIPS drift.

16.5 What adversarial review corrected

Our conclusions got the same adversarial scrutiny as our code. We ran independent review agents prompted to *refute*: one attacking the ANE-surgery semantics (is the rewrite exactly the original attention?), one attacking the bisection methodology (do the artifacts support the stated conclusions?). The semantics review confirmed the rewrite exact but demanded three robustness guards (now in the code); the methodology review found our original conclusions ("the wedge occurs on every compute unit" and "it hangs indefinitely") both wrong, from the artifacts:

- No `.mlpackage` existed for any wedged case: every wedged run died *inside* `ct.convert`, which (undocumented in our probe design) performs an implicit ComputeUnit.ALL model load unless `skip_model_load=True`. Our "CPU_ONLY wedge" observations had never exercised a CPU-only compile. Re-run with `skip_model_load` + explicit loads, CPU_ONLY and CPU_AND_GPU compile the *unmodified* attention in ~1 s and the unmodified full decoder in ~14 s. **The blowup is ANE-path-only. Claim falsified.**
- With a longer leash, the ANE compile of one time-attention subgraph *terminates* (284.5 s), so "hang" was an artifact of our deadlines meeting a finite but enormous compile: 148 s at batch 32 and ~284 s at batch 576 per block (vs ≤1 s for every other decoder subgraph), ×14 blocks ⇒ on the order of an hour for the full decoder. Restated as a pattern-specific compile-cost blowup, unusable at our shape but finite.
- The RUNLOG had survivorship bias: timed-out runs never emitted records (the kill preceded the write). The guarded runner now tees all output, so negative results are artifacts.

- One planned control (macOS15 target, whose newer SDPA lowering might have avoided the blowup) turned out **untestable on this host**: it emits opset `CoreML8`, which macOS 14.5 cannot parse. Recorded as untested, not skipped.

The corrected picture is more useful than the original: the fastest standalone configuration (surgical + CPU_AND_GPU) was invisible under the old conclusion, which had written off non-ANE compute units. The ANE verdict then went through three further self-corrections, kept here deliberately: round 3, a census + contention probe found the ANE 96% resident and contention-resistant (read by an early draft as "ANE is the deployment path"); round 4, a fuller bench on the final decoder showed CPU_AND_GPU decode faster in absolute milliseconds both idle and loaded, retiring that claim; round 5, the op-cost breakdown showed the ANE decode attention-bound, not Linear-bound, so the projected "~1.7× via conv-shaping" could not exist, retiring the last reason to build it. Four revisions to one subsystem's verdict is not indecision: each was forced by a measurement the previous round had not taken, and the final answer (CPU_AND_GPU decode; ANE set aside) is one an unmeasured intuition would have gotten backwards twice.

16.6 Deployment configurations (the menu)

There is no single "best" configuration: sampler-step count, decode compute unit, and hardware tier are independent dials, and the right setting depends on whether the goal is a *smooth* interactive session, a *stable* one, or an *offline showcase* render. Everything below is measured on the final stack (stage-2 PSD student + 60k decoder, M1 Pro). "Ceiling" is the decode-hidden rate $2000/\text{latent}$; "live" is what the interactive server realizes after WebSocket transport and the cross-thread executor round-trip.

Steps	DiT latent	fps ceiling	fps live	Sharpness vs 8-step	LPIPS vs 8-step	Character
1	112 ms	17.8	~12	97.5%	0.112	smoothest; playtested 'runs well, less stable'
2	168 ms	11.9	~8	99.1%	0.079	balanced; playtested 'stable', shipped default
4	278 ms	7.2	–	99.6%	0.042	higher fidelity, sub-real-time
8	495 ms	4.0	–	100% (ref)	–	offline hero renders

Sharpness barely moves across the ladder (97.5 → 100%; the softness is the 364M model and the distilled decoder, not the step count), so the real 1-vs-2 choice is fps vs stability (per-step drift), on which the LPIPS column and the human playtest agree: 1 step is smoother to move in but wobbles more; 2 steps is steadier. We ship 2-step as the default and expose 1-step as a "smooth mode." The decode-path and hardware-tier dials:

Path (60k decoder)	ms/chunk idle	under GPU load	parity	verdict
CPU_AND_GPU	115 ms	276 ms	73 dB	deployment choice
ANE (`ALL`)	341 ms	434 ms	62 dB	set aside; attention-bound

Hardware tier (cross-ref §13): the M1 Pro is the deliberate worst case; the same artifacts scale up: an M3/M4-class laptop or RTX 3060+ runs ~2–3× (≈20–30 fps *est*), a cloud L40S serves the 2-step model at ~20 fps (the launch tier), and B200 is the premium/multiplayer tier. The M1's ~8–12 fps is a mid-range machine's comfortable real-time. The compression work is what turns "needs a B200" into "runs on hardware people own."

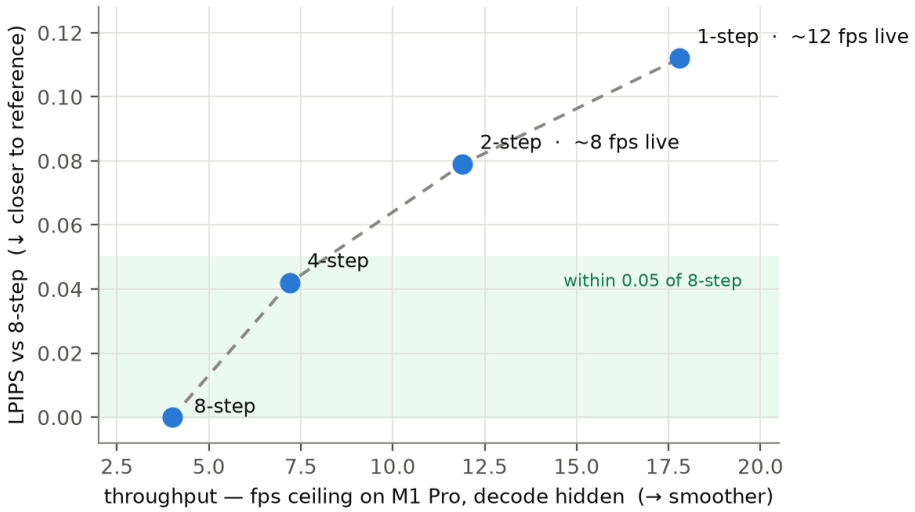

Figure 19: The speed↔fidelity frontier on M1 Pro. Each point is a sampler-step count; x is the decode-hidden fps ceiling, y is teacher-forced-style LPIPS vs the 8-step reference on aligned frames. The knee is shallow (1-step buys ~1.5× the throughput of 2-step for +0.033 LPIPS) because sharpness barely changes across the ladder (97.5–100%). The band marks configurations within 0.051 of the 8-step reference.

Open items.. We left the ANE decode out of the menu; its remaining research question (a compiler-safe *and* conv-shaped attention) is out of scope; a macOS 15 host re-test of the export is still open, as is the one throughput comparison we could not cleanly isolate: decode-on-ANE vs decode-on-GPU at the *system* level (the integrated pipelined A/B hit the flaky decode-child bug; the component data favours CPU_AND_GPU and the deployment call does not depend on resolving it). Wider multi-shard eval beyond the single-config distributional ladder of §17, cloud per-tier decode benches for the small decoder, and Linux/Windows package rehearsals remain; the shared roadmap is §12 and §11.

Does the compressed stack run the full game on a 2021 laptop, and which configuration ships?

Yes. The stage-2 PSD student on MLX plus the 60k decoder on Core ML runs live on an M1 Pro at ~8 fps (2-step, the shipped default) to ~12 fps (1-step "smooth mode"), 60–90× the naïve torch-MPS baseline, with 1-step measured at 97.5% of 8-step sharpness. The decode ships on CPU_AND_GPU (115 ms/chunk), not the ANE: an attention-bound Core ML graph makes ANE decode ~3× slower, and four measurement-forced revisions retired the earlier ANE-as-deployment-path conclusion.

17 Physics verification on the compressed ladder

Every gate to this point certifies pixels or parity: LPIPS against ground truth, PSNR against a reference implementation, validation loss against the base model. None asks whether the *world* survived: whether the compressed models still carry the game's state, answer its controls, and hold its dynamics far past the training horizon. A student can hold LPIPS flat and still have compressed the world away, the failure the pixel instruments are structurally blind to. This campaign (run 2026-07-13 on 8×H100, protocols frozen before results) turns the paper's own evaluation instruments (its §6.2, [8]), plus a longer-horizon stability protocol, on the compression ladder: **L0** 1B base (45k, 10 steps, ViT-XL decoder); **L1** + PSD warm-start (§14, 2 steps); **L2** 364M student (§15, 2 steps, ViT-XL); **L3** student + 576×14 small decoder (2 steps). The whole distilled subtree descends from the 45k single-player checkpoint, so 45k is the base

rung.

17.1 Game-state probing: is the state still decodable?

The paper reads position, rotation, and velocity for ball and cars out of the world model's pre-head activations: a probe trained on activations from *real, encoded* latent sequences, then applied to the model's *own generated rollouts* and scored against ground-truth physics in Unreal units. We run the identical protocol single-player (20-dim state) on the 1.18B teacher and the 364M student, with two probes per hook point, the paper's 3-layer MLP (hidden 1024) and a stricter closed-form ridge readout, at identical budgets (20,000 probe-train and 2,016 generated-rollout eval frames per model). The decision variable is the teacher-vs-student gap under an equal probe budget, not the absolute error.

Model	Probe	Real floor (ball-pos, uu)	Generated (uu)	gen ÷ floor
1.18B teacher	MLP (paper)	1,400	1,768	1.26
364M student	MLP (paper)	1,562	2,016	**1.29**
1.18B teacher	ridge (linear)	1,846	2,234	1.21
364M student	ridge (linear)	2,000	2,507	1.25

At this probe's resolution, the physics the probe can read survived distillation, and what it cannot read, it cannot read for either model. The probe resolves *position* well (normalized MSE 0.22–0.49), resolves *velocity* only weakly (nMSE ≈0.9–1.1, near the predict-the-mean line), and does not resolve *rotation* at all (quaternion nMSE ≈28–62 for the MLP on *both* models), so rotation neither passes nor fails this test. On the resolvable channels the decision variable is the last column (how much readability degrades under rollout versus real perception), and the student matches the teacher (1.29 vs 1.26 under the paper's MLP; the strict linear probe agrees). The student's higher absolute error (+14% on generated rollouts) is already present on *real* sequences (+12% floor gap): a capacity effect on the representation, echoing the paper's own capacity curve, not a rollout-specific collapse. The LPIPS-blind failure this instrument exists to catch (position preserved in pixels but dropped from the world model's state) did not occur.

17.2 Per-action recoverability (ARR)

ARR (paper §6.2) is $\mathrm{AP}_{\mathrm{gen}}(a)/\mathrm{AP}_{\mathrm{recon}}(a)$ per action a: a frozen DINOv3-B action probe [12] is slid over a rung's generated rollouts and over the same clips' codec reconstructions, and each action's average precision against the commanded inputs is compared. The reconstruction denominator cancels the probe's own imperfection and the codec's rendering quality, isolating whether the *world model* renders the action. We report per action, not the aggregate, because the paper shows ARR rises with an action's training frequency (log-linear $r = 0.82$) and compression should be expected to degrade the tail first. An aggregate hides exactly that.

Action (by freq.)	base 1B @10	PSD 1B @2	student @2	student+small @2
Forward	0.980	0.984	0.973	0.975
Powerslide	0.936	0.964	0.954	0.958
Right	0.956	0.970	0.960	0.964
Left	0.970	1.015	0.960	0.940
Boost	0.907	0.937	0.930	0.944
Jump	0.913	0.930	0.931	0.906
Air-roll L	0.962	0.907	0.900	**0.874**
Reverse	0.909	0.949	0.860	**0.856**
Air-roll R	0.959	0.976	0.958	0.920
aggregate	**0.943**	**0.959**	**0.936**	**0.927**

The tail degrades first, and the aggregate hides it as predicted: down the full ladder the aggregate loses 0.017 (0.943 → 0.927, which reads as noise), while the two rarest controls lose 5–9 points (air-roll-left 0.962 → 0.874, reverse 0.909 → 0.856) and forward/steer stay flat. Air-roll-left declines *monotonically* down the ladder (0.962 → 0.907 → 0.900 → 0.874; it is also PSD's one regression); reverse improves under PSD (0.949) then drops at the student step (0.860); the largest single drop falls on the small-*decoder* rung, since rare actions are subtle visual events the 8× decoder renders less legibly (the probe reads video, so decoder fidelity is part of recoverability by construction). Two honesty notes bound this. Our absolute ratios run above the paper's aggregate anchor (0.91) because our probe is weaker: 0.752 held-out mAP (the eval gate is ≥0.75, passed on the second probe after a first at 0.733 was rejected) versus the paper's 0.838. A weaker probe compresses $\mathrm{AP}_{\mathrm{gen}}$ and $\mathrm{AP}_{\mathrm{recon}}$ toward each other, inflating the ratio; so cross-paper absolutes carry that caveat while ladder-internal comparisons (same probe, same clips) are the valid instrument. And the bootstrap CIs at 200 clips overlap, including at the ladder's endpoints (air-roll-left base $[0.91, 1.02]$ vs student+small $[0.80, 0.95]$), so no single cell separates statistically; the signal is the directional consistency of the monotone air-roll-left decline plus the student-step reverse drop, read at that strength, not as separated intervals.

17.3 Long-rollout stability and snap-back

MIRA's signature property is stability far past the training horizon plus untrained recovery from far-out-of-distribution states, attributed to the codec's smoothness (nearby states map to nearby latents, so a wrong prediction stays valid and is absorbed). We compressed the DiT (§15), the sampler (§14), and half of the codec, and §16's playtest note that 1-step "wobbles more" is the loose thread; nothing else in this report would detect a broken snap-back. Protocol: **ten-minute rollouts** (6,000 latent steps, 2× the paper's measured horizon) per rung at its shipped settings, five fixed real contexts each, scored over time by windowed distributional distance (Inception and DINOv3 feature spaces) against a frozen real-clip reference, plus two scripted OOD holds from the paper's own cases (a 60-second all-still hold and a 30-second still-in-goal hold), each followed by resumed driving.

An instrument note is part of the result. The pre-registered melt rule (fdd $> 4\times$ own band, or latent-z magnitude > 6) stamped MELTS on every student rung through the latent-z channel, whose denominator (the std of window-mean RMS over real clips) is degenerately tight, so a few percent of drift reads as $z > 6$. The uncompressed control passes both pre-registered thresholds (its drift is genuinely small), so the control did not expose the rule; the students' flat-but-elevated traces did. A control-anchored recalibration (melt = sustained 1.5× the control's envelope) *still* labels the student rungs "melts" because their distance to the reference is a level shift present from the first minute, not a progressive trend. The table therefore uses trajectory language: *elevated-flat* for a bounded, flat level shift, reserving *melt* for progressive collapse, which no rung exhibits.

Rung	band fdd (60–120 s)	end fdd (t=592 s)	max $\lvert z \rvert$	verdict	snap-back
base 1B @10 (control)	4.3	4.2	3.7	**stable**	recovers 2 s after resume
PSD 1B @2	6.2	13.6	5.3	elevated after OOD	**stuck** (~2.2× band, final 7 min)
student @2	51.1	63.5	9.4	elevated-flat (~12× control)	unmeasurable above own band
student + small dec @2	44.5	51.3	9.4	elevated-flat	unmeasurable above own band
student @1	56.7	59.3	10.4	elevated-flat, fastest drift-in	unmeasurable above own band
student + small, still-OOD	39.8	49.0	9.4	elevated-flat	**recovers 2 s** after still hold

Four findings, in decreasing confidence. (1) *No rung shows runaway collapse*: every trace is flat-to-slowly-rising over ten minutes, latents stay bounded, and the base control is textbook-stable, which validates the instruments. (2) *The 3.2× student pays a long-horizon distributional price every short-horizon instrument missed*: at the 2 s eval unroll it matches its teacher (12.72 vs 12.67 gFID, below), but by the one-minute mark of a free rollout its windowed distance runs ~12× the control's and stays there, the catch the do-no-harm gates and the 2 s paper-axis protocol structurally could not see. (3) *Snap-back survives where it is measurable*: the control recovers in 2 s and the student recovers from the still-scene hold in 2 s; for the student's goal-OOD runs the hold never registers above the already-elevated band, so recovery there is unanswerable, and the one measured deficiency is the **PSD 1B**, left ~2.2× elevated for the remaining seven minutes after the goal hold: the closest thing to a broken snap-back in the family, on the rung where it *is* measurable (with the caveat that each trace pools n=5 contexts, which cannot resolve a single hard-stuck context). (4) *1-step drifts faster than 2-step* (threshold-crossing 22 s vs 42 s; max $\lvert z \rvert$ 10.4 vs 9.4): the playtest's "runs well, less stable" (§16), now a measured ordering.

Decoder-free latent throughput down the ladder (latent-only, no decode, H100, streaming per-frame, its own fps family; not comparable to the decode-hidden ceilings of §16): base@10 4.3 → PSD@2 12.0 → student@2 16.2 → student@1 **20.6** latent-fps, 4.8× end to end: the configuration a control-oriented reader wants, since a policy consuming the game-state probe's readout renders no pixel.

17.4 The paper's axis: distributional ladder

Finally we place each rung on the paper's own tables. The offline evaluation harness (the configuration whose 45k reading reproduced our published gFID 13.46) runs across the ladder and a sampler-step matrix ($n = 512$ clips per cell), reporting gFID and gFDD (the harness implements no FVD, which the paper also leaves unspecified).

Rung	1 step	2 steps	8 steps	10 steps
base 1B (no PSD)	17.56 / 0.763	15.39 / 0.614	–	13.46 / 0.463
PSD 1B (teacher)	13.57 / 0.517	12.67 / 0.431	–	**11.61 / 0.393**
364M student (PSD)	13.93 / 0.536	12.72 / 0.467	12.31 / 0.435	–

(gFID / gFDD, lower is better.) The base@10 cell reads 13.46, reproducing our published canonical eval to the second decimal through this harness. Three findings follow. *PSD improves the model at every step count*, including the many-step regime: the warm-started PSD 1B beats the base by 1.85 gFID at the base's own 10-step setting, and its 2-step beats the base's 10-step

outright (12.67 vs 13.46), the paper's Fig-11 shape reproduced under our *post-hoc* construction (distinct from the paper's from-scratch PSD ablation; ours retrofits PSD onto a finished checkpoint, §14, the paper's distill-the-pretrained-model direction). *The 3.2× student shrink costs* +0.05 *gFID* / +0.036 *gFDD at matched 2-step sampling* (+0.4% / +8% relative; no CIs for this protocol, so "free" would overstate an un-CI'd $n = 512$ reading). *The deployed configurations sit on the paper's axis*: the shipped 2-step student reads 12.72 / 0.467 and the 1-step "smooth mode" 13.93 / 0.536, approaching the un-distilled base's 10-step gFID; the paper's anchor at this scale is 1B @10 = 10.7 / 0.55, reported at a 4 s horizon against our harness's 2 s.

Did compression keep the world, or only the pixels?

On the channels its instruments can read, it kept the world. Activation probes recover game-state from the 364M student's own generated rollouts as well as from the 1.18B teacher's (gen÷floor 1.29 vs 1.26); per-action recoverability holds except for a rare-action tail that degrades first (air-roll-left 0.962 → 0.874 down the ladder); and over ten-minute rollouts no rung shows runaway collapse: the base control is textbook-stable and snap-back recovers in 2 s where it is measurable. The honest cost is a long-horizon distributional level shift the 3.2× student carries (~12× the control's windowed distance by one minute) that every short-horizon gate missed, plus a probe weaker than the paper's (0.752 vs 0.838 mAP) that bounds cross-paper absolutes to ladder-internal comparisons. On the paper's own axis the shipped 2-step student reads gFID 12.72, within +0.05 of its teacher.

18 Compression cost ledger

The compression program's entire training spend is small against the serving-cost curve it moves. Every fleet run here was preemptible; the observability failures that cost time (stale relaunch log directories, a boot service rewriting the run spec, lexicographic checkpoint ordering) and their mechanical fixes (census logs by creation time, disable the boot service, `sort -V`) are recorded with the operations discipline in §8.

Item	Cost (USD)
Mechanism test ($4 Modal H100)	4
PSD pilot (8×H100 spot, incl. one diverged attempt)	~110
Pilot evals	~3
Decoder (60k)	~180
Student stage-1 (40k)	~170
Student stage-2 PSD (10k)	~50
Total	~**520**

The ~$520 moves the per-session serving cost down a full curve: B200 ~$11/hr/session today → L40S $1.95/hr at 2 steps (the launch tier) → L4-class tiers with the student + small decoder (~$0.70/hr *est*) → $0 marginal on player-owned hardware, where the finished student + decoder run at ~8–12 fps (§16).

The program is model-agnostic within the MIRA family: the same four levers (few-step post-hoc distillation, cross-scale velocity distillation, renderer-only compression, and verified runtime ports) apply to any latent world model with a frozen-encoder codec. For embodied/simulation uses the eval discipline is arguably the more transferable artifact: a world model on edge compute is only useful with a certificate that compression did not break its dynamics, which is what the instrument set of §13–§16 produces.

What did the compression program cost, and what does it buy?

The full training spend was ~$520 of preemptible GPU time ($4 mechanism test, ~$110 PSD pilot, ~$3 evals, ~$180 decoder, ~$170 student, ~$50 stage-2). Against it, per-session serving cost moves from B200 (~$11/hr) to an L40S ($1.95/hr at 2 steps) toward L4-class tiers (~$0.70/hr *est*) and to $0 marginal on player-owned hardware. The four levers are model-agnostic within the MIRA family; the evaluation discipline is the more transferable artifact.

Acknowledgements

The architecture, training recipe, and dataset are General Intuition's and Kyutai's, released openly, in collaboration with Epic Games. This is an independent reproduction; we are not affiliated with, or endorsed by, General Intuition, Kyutai, or Epic Games. Errors and shortcuts in the reproduction are ours alone.

References

[1] Eloi Alonso, Vincent Micheli, et al. Diffusion for world modeling: Visual details matter in atari and cs:go. In *NeurIPS*, 2024.

[2] Nicholas Boffi, Kevin Frans, et al. Progressive self-distillation / shortcut models for few-step flow matching. 2025. Concurrent works; see the MIRA report §6.4.

[3] Decart. Oasis: a playable world model. https://oasis.decart.ai, 2024.

[4] General Intuition. Mira blog post. https://mira-wm.com/blog-post/, 2026.

[5] General Intuition and Kyutai. mira-wm/mira (release repository, commit 6aae8d3). https://github.com/mira-wm/mira, 2026. Apache-2.0.

[6] David Ha and Jürgen Schmidhuber. World models. 2018.

[7] Martin Heusel et al. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

[8] Anthony Hu, Václav Volhejn, Adrien Ramanana Rahary, Chris Mulder, Aditya Makkar, et al. Mira: Multiplayer interactive world models with representation autoencoders. 2026. arXiv:2607.05352. General Intuition and Kyutai, in collaboration with Epic Games.

[9] Kyutai. Rocket science dataset. https://huggingface.co/datasets/kyutai/rocket-science, 2026. CC BY-NC-SA 4.0; Rocket League content used with permission of Epic Games.

[10] NVIDIA. Flashdreams. https://github.com/NVIDIA/flashdreams, 2026. commit 862be4c8.

[11] Odyssey. Agora-1. https://odyssey.ml, 2026. Public multiplayer world model, research preview.

[12] Oriane Siméoni et al. Dinov3. 2025.

[13] TechCrunch. General intuition's $2.3b bet that video games can train ai agents for the real world, 2026. 2026-06-25.

[14] Dani Valevski et al. Diffusion models are real-time game engines. 2024. arXiv:2408.14837.

[15] Richard Zhang et al. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.