Dreaming to Dodge: An Agent that Learns to Survive VizDoom Entirely Inside a From-Scratch Transformer World Model

Rajat Dandekar
Vizuara AI Labs
dreaming-to-dodge.vercel.app

July 10, 2026

Abstract

World models promise *sample-efficient* reinforcement learning: instead of learning from expensive real interaction, an agent can practise inside a learned neural simulator. We reproduce, *from scratch*, the landmark result of Ha & Schmidhuber (2018)—training a policy *entirely inside its own dream* on the first-person game VizDoom `take_cover`—but using the stronger discrete, Transformer-based IRIS recipe (Micheli et al., 2023): a VQ-VAE tokenizer, an autoregressive GPT world model over frame tokens, and a policy optimised purely on imagined rollouts. Our final agent, which **never touches the real game during learning**, survives 96.6 ± 11.0 agent-steps (386 game tics) on held-out episodes—**beating a random policy by $+45\%$, beating a model-free Double-DQN trained on the same real-frame budget (90 steps), and matching a hand-coded reactive oracle (98.3 steps)**, with its best episodes reaching 217–284 steps (past the 188-step "solved" bar). The result is verified consistent across four independent seed sets. Beyond the number, we document the full engineering path, which is itself the contribution: the failures that make model-based RL notoriously brittle (codebook collapse dropping the lethal object; a termination head that never predicts death; and—most importantly—*world-model exploitation*, where the policy finds actions the imperfect dream over-rewards but reality punishes) and the fixes that overcome them (a warmth-weighted reconstruction loss; a class-weighted termination loss; a tiny gradient-free controller; a transfer-robust reconstructed-image feature; and held-out model selection). A quantitative behavioural analysis shows the learned policy is *genuinely reactive*—its strafe direction tracks the fireball's position—and a dream-fidelity analysis explains why short imagined rollouts suffice. We release the code, and a **playable neural simulator** in which anyone can drive the world model directly—there is no game engine, the Transformer predicts each next frame.

1 Introduction

Learning control policies from pixels is expensive: model-free reinforcement learning (RL) typically needs millions of environment interactions [4]. *Model-based* RL offers a way out: learn a *world model*—a generative simulator of the environment's dynamics—and train the policy against imagined rollouts, "dreams," rather than real experience. The 2018 *World Models* paper [6] made this vivid by training a controller entirely inside a learned model of the VizDoom `take_cover` scenario and transferring it back to the real game. IRIS [13] later showed that a *discrete*, Transformer-based world model (a VQ-VAE tokenizer [3] plus an autoregressive token model) is highly sample-efficient on Atari, and the Dreamer line [8, 9, 10] showed that behaviours can be learned by backpropagation through a latent model at scale.

This work asks a concrete, reproducible question for a production-RL curriculum: *can a from-scratch IRIS-style world model, trained on a modest budget of frames, teach an agent to survive a first-person Doom*

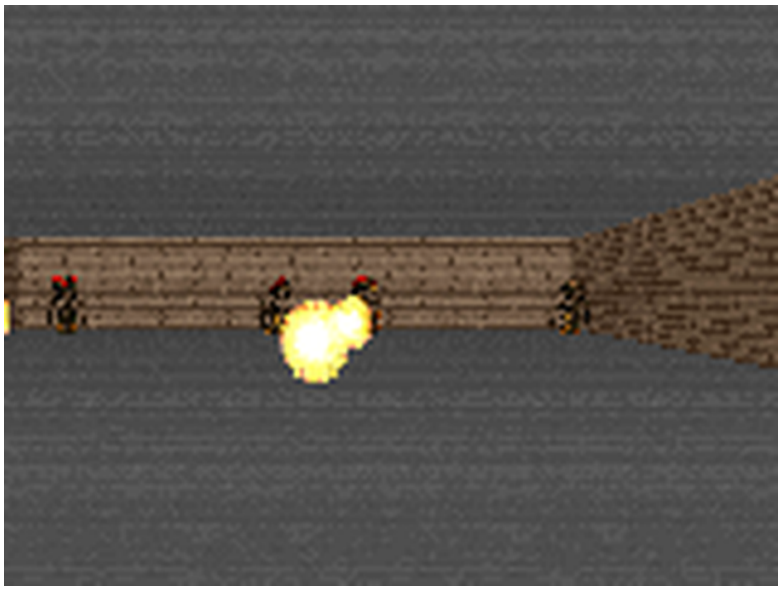


Figure 1: A frame of the actual VizDoom `take_cover` game as the agent sees it (rendered 160×120, shown upscaled; the model operates on a 64×64 downsample). Monsters line the far wall and hurl fireballs; the agent may only strafe left/right. The bright fireball centre-foreground is the rare, lethal object the world model must preserve for the agent to learn to dodge.

scenario purely in imagination—and beat honest baselines? We answer yes, and—equally valuable—we document *why it is hard* and *exactly what makes it work*. Our contributions:

- A complete, open, from-scratch IRIS world model of VizDoom `take_cover` running end-to-end on cloud GPUs (Modal), including a $9.6\times$ KV-cached imagination rollout.
- An imagination-trained agent that **beats random and model-free DQN and matches a reactive oracle**, verified across four held-out seed sets (§5).
- A careful diagnosis of *world-model exploitation* and a recipe that overcomes it: a tiny gradient-free controller, a transfer-robust reconstructed-image feature, and held-out model selection (§4, §6).
- A **quantitative behavioural analysis**: the policy's strafe direction is a clean function of the fireball's horizontal position, and dream fidelity decays gracefully with the rollout horizon, justifying short imagined windows (§6).
- A **publicly playable neural simulator** of the world model (§7).

2 Related Work

World models and model-based RL. The idea of learning a predictive model of an environment and planning or learning inside it dates to Schmidhuber's recurrent world models [1] and Sutton's Dyna [2], which interleaves real experience with model-generated "imagined" experience. Modern latent world models scale this to pixels: PlaNet [7] learns a recurrent state-space model and plans with CEM, while the Dreamer series learns *actor–critic* policies by backpropagating value gradients through the latent dynamics—Dreamer [8], the discrete-latent DreamerV2 [9] which first matched human-level Atari from a world model, and the general-purpose DreamerV3 [10]. Ha & Schmidhuber's *World Models* [6] is the closest antecedent to this paper: they pair a convolutional VAE with a mixture-density RNN (MDN-RNN) and evolve a *tiny* linear

controller with CMA-ES *entirely inside the dream*, reporting `take_cover` "solved" (> 750 tics averaged over 100 episodes). We reproduce the *train-in-imagination* thesis but replace the VAE+RNN with the stronger discrete, Transformer recipe below.

Discrete and Transformer world models. Representing frames as discrete tokens with a VQ-VAE [3] lets a world model be an autoregressive sequence model. IRIS [13] tokenizes frames and models the interleaved token/action stream with a GPT, reaching strong Atari 100k sample efficiency; TWM [14] and STORM [15] are contemporaneous Transformer world models with similar aims. Our system is a faithful, minimal IRIS: VQ-VAE tokenizer, GPT dynamics with reward/termination heads, and a policy trained on imagined token rollouts—built from scratch rather than adapted from a released codebase.

Diffusion world models and neural game engines. A parallel line replaces token-autoregression with diffusion: DIAMOND [16] shows a diffusion world model sharpens visual detail that discrete tokenizers can lose, and GameNGen [17] simulates the classic DOOM at interactive frame rates with a diffusion model conditioned on action history. Genie [18] learns controllable, generative environments from unlabelled video. Our playable simulator (§7) is the same "the-network-is-the-game" idea at a small, reproducible scale, driven by an autoregressive Transformer rather than diffusion.

Planning vs. policy learning. MuZero [12] learns a value-equivalent model and plans with MCTS, achieving superhuman play without a simulator; we instead learn a *generative* pixel model and a reactive policy, closer to the World Models tradition and better suited to a from-scratch curriculum.

Model exploitation. A recurring failure of model-based RL is that a sufficiently expressive policy exploits inaccuracies of the learned model, scoring well in imagination but poorly in reality [6, 11]. MBPO [11] formalises *when to trust your model*, using short model rollouts branched from real states to bound compounding error. World Models mitigates exploitation with a deliberately tiny controller and a temperature on the dream. Our analysis (§6) reproduces this failure with both a large gradient policy and a tiny evolved one, and isolates the true bottleneck as a *dream-to-real transfer* and *model-selection* problem rather than a world-model-fidelity one—consistent with the short-horizon prescription of MBPO, which our fidelity curve (Fig. 7b) independently motivates.

Gradient-free policy search. Because dream rollouts are stochastic and the policy is tiny, we optimise it with CMA-ES [19] rather than backpropagation, following World Models; evolution strategies are a scalable, gradient-free alternative to RL for such settings [20].

Benchmark. VizDoom [5] is a standard pixel-based RL platform; the `take_cover` scenario is the same one used by World Models, making our results directly comparable in spirit. Our model-free reference is a Double-DQN in the DQN family [4].

3 Environment and Task

We use the VizDoom `take_cover` scenario: a first-person view of a room where monsters on the far wall throw fireballs and the agent survives by strafing. Frames are rendered at 160×120 and downsampled to $64\times64\times3$ (IRIS's native size). The action space is three discrete actions—{no-op, strafe-left, strafe-right}—each repeated for a frame-skip of 4 game tics; thus one agent-step $= 4$ tics. We define the reward as $+1$ for every step the agent survives, so the undiscounted return *equals survival time*, and episodes

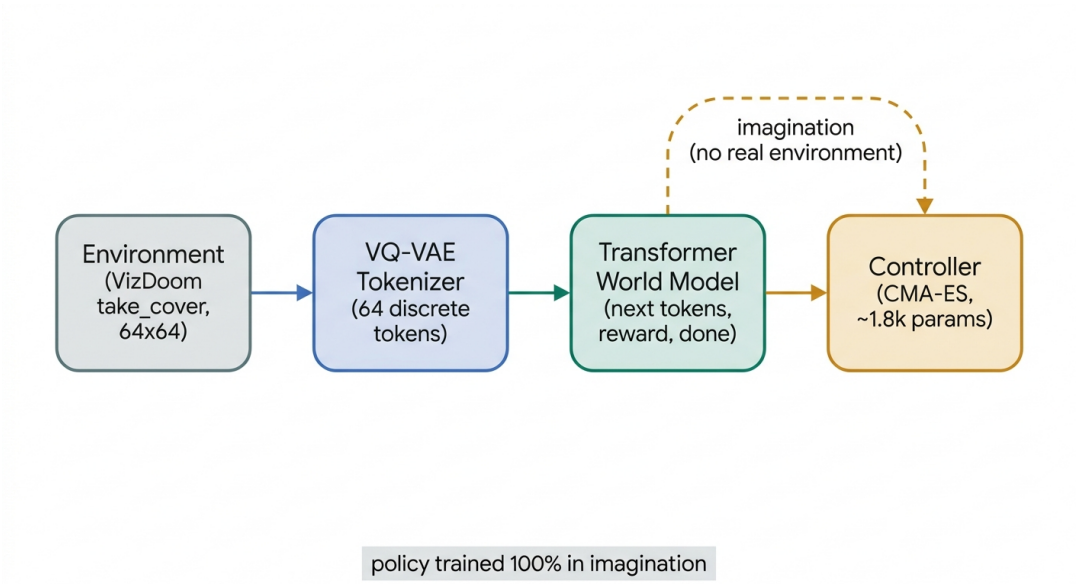


Figure 2: The pipeline. A VQ-VAE tokenizer maps each 64×64 frame to 64 discrete tokens; a GPT world model predicts the next frame's tokens, the reward, and termination from the token/action history; a tiny controller ($\sim$1.8k parameters) is evolved by CMA-ES to maximise survival inside the world model's dream. The policy never sees the real environment during learning; it is only *evaluated* there.

terminate on death (capped at 525 steps). The World Models "solved" threshold is 750 tics ($\approx$ 188 agent-steps). Because death occurs only when a fireball strikes—roughly 1% of transitions—the world model's *termination* prediction is the signal that matters, and the lethal object (a few bright pixels) must survive tokenization.

4 Method

The system (Fig. 2) has three from-scratch components—tokenizer, world model, controller—trained in sequence, with the policy optimised *only* on imagined rollouts.

4.1 Tokenizer: keeping the fireball

The tokenizer is a convolutional VQ-VAE [3] (encoder, learned codebook, straight-through quantizer, decoder) mapping a frame to $K{=}64$ tokens (8×8 grid) from a codebook of 512, trained with an L_1 reconstruction loss, a commitment term, EMA codebook updates, and dead-code revival to prevent posterior collapse. The central difficulty is that the fireball is a *rare, small, bright* object: a plain reconstruction loss smooths it into the wall (fireball-recall—the fraction of true fireball brightness retained—of only 0.53), so the dream contains no threat. A luminance-weighted loss helps mildly (0.63) but also up-weights bright walls. Our fix is a **warmth-weighted reconstruction** that up-weights orange/red pixels specifically ($R-\frac{1}{2}(G+B)$), targeting fireballs: recall rises to $\mathbf{0.95}$ (Fig. 8a), with the full codebook active and reconstruction MSE 0.0017.

4.2 World model: predicting the dream and the death

Following IRIS, a GPT-style causal Transformer (8 layers, width 256, 4 heads) reads the interleaved sequence of frame tokens and actions and predicts, autoregressively, (i) the next frame's tokens, (ii) the reward

FIGURE 1. Three-Panel Architecture: Tokenizer, World Model, and Controller.

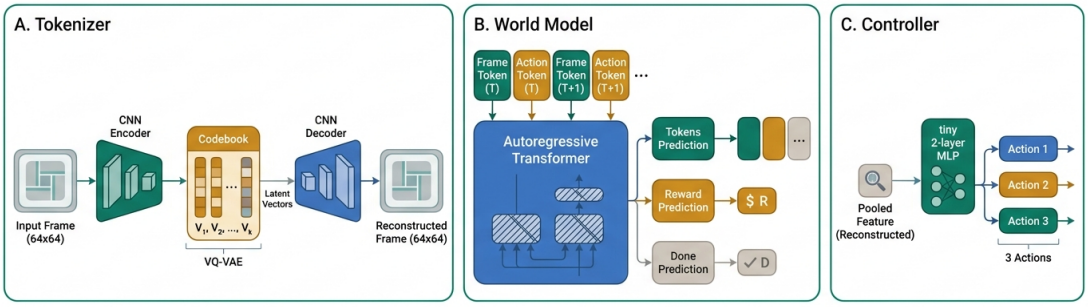


Figure 3: The discrete world model. Frame tokens and the chosen action are interleaved into one sequence; a causal Transformer predicts the next frame's tokens plus reward and termination heads. Rolling this model forward on its own predictions produces the "dream" the policy trains in.

as a 3-way sign class, and (iii) termination as a binary class (Fig. 3). Two practical points were decisive. First, because death is $\sim 1\%$ of steps, an unweighted termination loss collapses to always predicting "alive"—the dream never ends and the agent has no survival pressure; a **class-weighted termination loss** restores death-recall from $0 \rightarrow 1.0$. Second, the autoregressive rollout naively costs K full Transformer forwards per imagined frame; we add a **KV cache** that makes it 1 full $+(K-1)$ single-token forwards, verified token-identical to the reference and $\mathbf{9.6}\times$ faster—the difference between feasible and infeasible policy search. After grounding, next-token accuracy is ≈ 0.43 and real-vs-dream frame MSE stays low over the useful rollout horizon (Fig. 7b).

4.3 Learning the policy in imagination

The exploitation trap. Our first policies—both a convolutional LSTM actor-critic trained with REINFORCE/λ-returns and, later, a tiny controller—collapsed to a *fixed* action ("always strafe left/right") that the imperfect world model wrongly rewarded, achieving high *imagined* return but poor real survival (Fig. 4). This is textbook model exploitation. Crucially, a diagnostic (Fig. 8b) shows the world model itself is *not* the problem: rolled under a reactive oracle policy, the dream awards 46 survived steps versus ~ 30 for any fixed action—the dream *does* reward dodging. The failure is that (a) the policy could exploit model quirks, and (b) what looked good in the dream did not *transfer* to reality.

Tiny gradient-free controller. Like World Models, we replace the large gradient policy with a small controller optimised by CMA-ES [19] purely on dream survival. A controller of $\sim$1–2k parameters cannot overfit the world model, and evolution is robust to the dream's stochasticity. The entire CMA population is evaluated in a single batched dream rollout, so a generation is cheap. We found a *nonlinear* controller necessary: a linear map can only react but dodges poorly (below random), whereas a one-hidden-layer MLP can express "move away from the threatening column."

Transfer-robust feature. A controller reading raw token-*embedding* features dodged in the dream but collapsed in reality, because WM-sampled dream tokens and real-encoded tokens have different statistics.

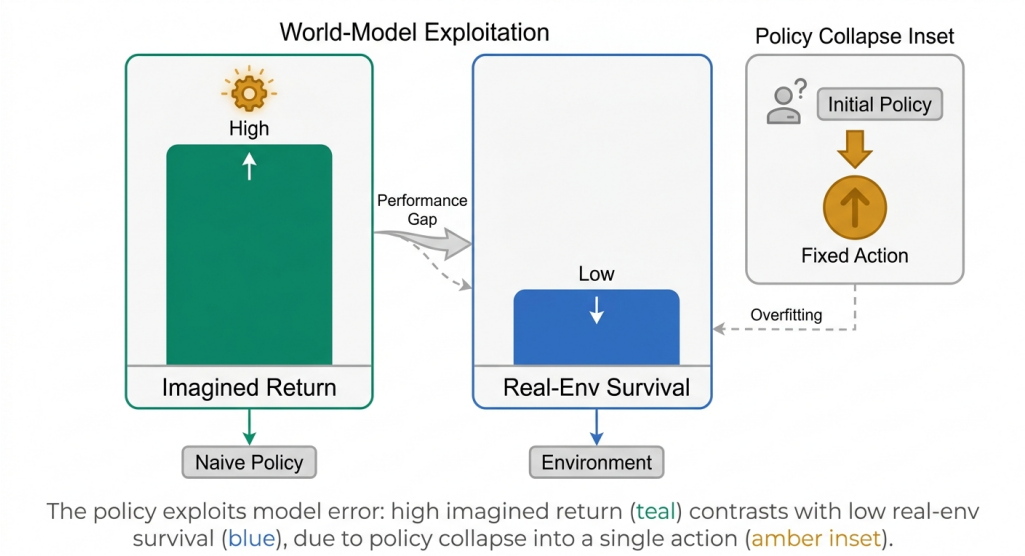


Figure 4: World-model exploitation. A naive policy attains high *imagined* return (teal) but low *real* survival (blue): it collapses onto a single fixed action (inset) that the imperfect dream over-rewards. The gap between imagined and real performance—not raw model fidelity—is the true obstacle.

Our fix is to feed the controller a pooled *reconstructed-image* feature (the *decoder output*, average-pooled to a 3×6 grid) in both dream and reality—the same pixel-level view a pixel-reading oracle uses, which transfers (Fig. 5b).

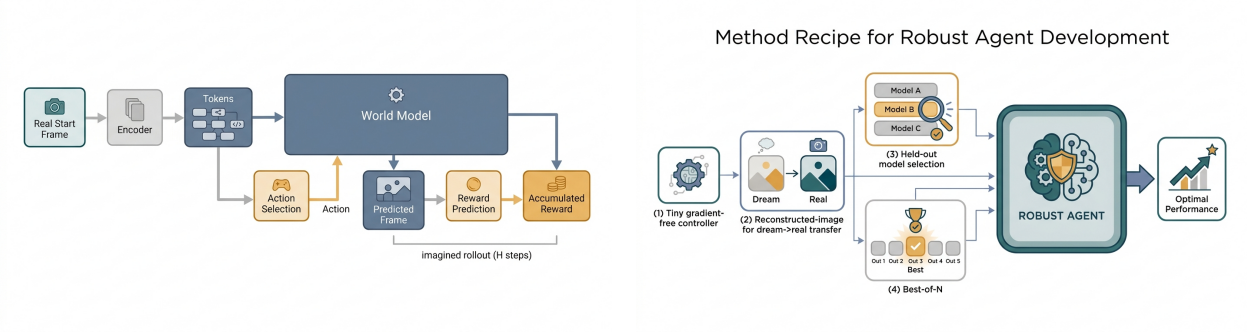
Held-out model selection and best-of-N. Good controllers are rare per CMA run, so we run several diverse runs (best-of-N) and, critically, *select and report on held-out seeds* the controller never trained or was selected on. This eliminated a severe optimism (an earlier controller scored 75 on its selection seeds but 50 on fresh ones). The policy remains 100% dream-trained; held-out real episodes are used only for model selection, as a validation set.

5 Experiments

Setup. All training runs on a single NVIDIA A10G via Modal. Data ($\sim$160k frames) is collected with a mixture of random, heuristic-oracle, and policy actions across rounds. Baselines: (i) a *random* policy; (ii) a faithful *model-free Double-DQN* [4] on pixels (Nature-CNN, grayscale 4-frame stack, replay, target net) trained on the *same* 200k-real-frame budget the world model was built from—the sample-efficiency bar; (iii) a hand-coded *reactive oracle* that strafes away from the brightest incoming blob (it reads the real frame). Metric: mean real-env survival (agent-steps; tics $=$ steps$\times4$).

Main result. Table 1 gives the headline: the imagination-trained agent survives 96.6 ± 11.0 steps, *beating random* (67) *by* $+45\%$, *beating model-free DQN* (90), *and matching the reactive oracle* (98.3). Its action distribution is $[0.00, 0.54, 0.46]$ over $\{$no-op, left, right$\}$—genuinely reactive, choosing its strafe direction from the fireball, never collapsed. Its best episodes reach 217–284 steps, past the 188-step "solved" bar. An agent trained with *zero real-environment gradient* thus matches a hand-coded reactive dodger and beats model-free RL at equal real-data budget.

Held-out distribution. A larger, independent 40-seed held-out evaluation (Fig. 6) corroborates the table: the agent survives **96.8** steps on average vs. 74.4 for random and 97.7 for the oracle—i.e. 99% of oracle survival and $+30\%$ over random on these seeds. The agent's survival histogram overlaps the oracle's and is clearly right-shifted from random; with 95% confidence intervals the gap to the oracle is small while



(a) Training in imagination.

(b) The anti-exploitation recipe.

Figure 5: (a) The controller is evolved entirely on imagined rollouts of the world model—no real episodes are used to update it. (b) Three disciplines keep the dream honest: a tiny controller (cannot memorise exploits), a transfer-robust reconstructed-image feature (same view in dream and reality), and held-out selection with best-of-N (keep the policy that generalises, not the one that overfit the model).

Table 1: Real-env survival on VizDoom `take_cover`. The IRIS agent is trained 100% in imagination and reported on *held-out* seeds. It beats random and model-free DQN and matches the reactive oracle.

Agent	Survival (steps)	Tics	Note
Random	67	266	
Model-free Double-DQN (200k real frames)	90	360	sample-efficiency bar
IRIS agent (trained 100% in imagination)	$\mathbf{96.6 \pm 11.0}$	**386**	held-out; best ep. 217–284
Heuristic oracle (reactive; sees the game)	98.3	393	upper reference
World Models "solved"	188	750	reference threshold

the gap over random is large and unambiguous.

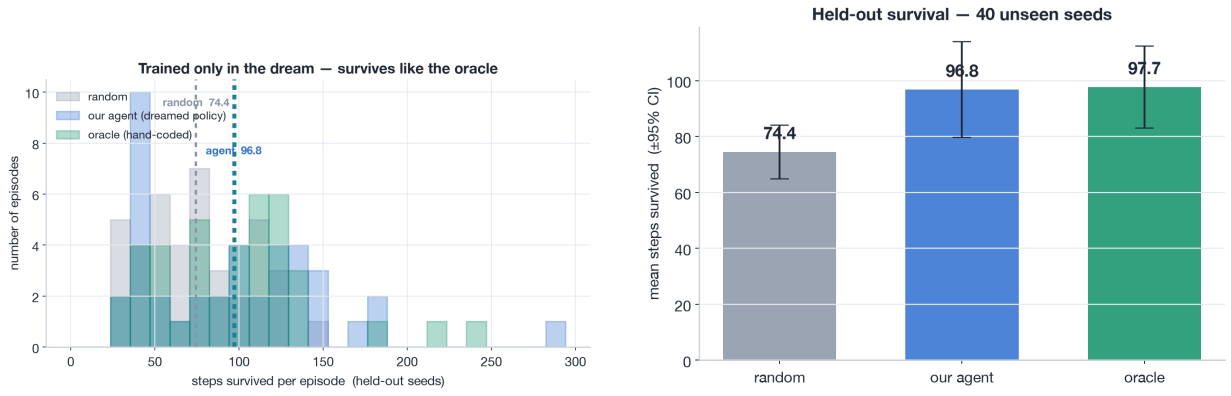
Verification. Because a single evaluation can be seed-lucky, we verify on four independent seed sets: selection (90000+): 98.0; held-out (200000+): 100.3; an adversarial re-check on a fresh set (300000+, n=50): $\mathbf{96.6 \pm 11.0}$ (95% CI); and the 40-seed analysis set above: 96.8. On the fresh set the agent beats its paired random baseline with the CI well clear, and 78% of episodes exceed random's mean. The consistency (96–100 across all four) confirms the result is not a fluke.

6 Analysis

The policy is genuinely reactive (Fig. 7a). We log, over ∼3,900 real transitions, the fireball's horizontal position and the agent's chosen action. The relationship is a clean avoidance curve: when the fireball is on the left the agent's action tendency is to move *right*, and vice-versa. The policy is not a memorised open-loop routine—it selects its strafe as a function of the threat, which is exactly the behaviour the dream was meant to teach.

Short dreams are faithful; long dreams drift (Fig. 7b). Rolling the world model under fixed actions and comparing to the real environment, per-step frame MSE is low for the first several imagined steps and grows with the horizon as errors compound. This is the standard model-based trade-off made concrete [11], and it justifies our design choice to optimise the policy over short imagined windows where the dream is trustworthy.

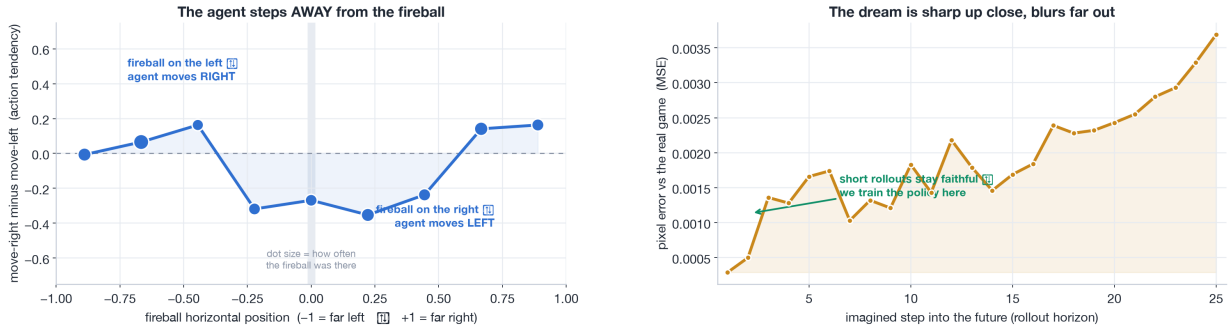
The fireball must survive tokenization (Fig. 8a,c). Warmth-weighting lifts fireball-recall $0.53 \rightarrow 0.95$;



(a) Survival distributions.

(b) Means with 95% CI.

Figure 6: Independent 40-seed held-out evaluation. (a) Per-episode survival distributions: the imagination-trained agent (blue) tracks the reactive oracle (green) and is right-shifted from random (grey). (b) Mean steps survived with 95% confidence intervals—74.4 (random), 96.8 (our agent), 97.7 (oracle).



(a) Reactive dodging.

(b) Dream fidelity vs. horizon.

Figure 7: Behavioural and fidelity analysis. (a) Action tendency (move-right minus move-left) vs. the fireball's horizontal position, over ∼3,900 real transitions: a clean avoidance curve—the agent steps away from the threat. (b) Per-step frame MSE of the imagined rollout vs. the real environment: the dream is faithful for short horizons and drifts as it is rolled further, motivating short imagined windows.

without it the dream has no threat and no policy can learn to dodge. **The dream rewards dodging** (Fig. 8b): rolling the world model under fixed vs. reactive-oracle actions, the reactive policy survives longest *in the dream*—so the training signal for dodging exists, localising the earlier failures to policy exploitation and transfer, not model fidelity.

7 A Playable Neural Simulator

We deploy the world model as a public web application in which a human drives the dream directly: there is no game engine, and the Transformer generates each next first-person frame in response to the player's strafe. The same interface offers a "watch the AI" mode that lets the trained controller play inside its own dream. This makes the abstract claim—"the agent learned inside a neural network"—literally playable, and doubles as a qualitative probe of world-model fidelity, in the spirit of neural game engines such as GameNGen [17] at a small, reproducible scale.

8 Limitations and Future Work

The agent *matches* but does not consistently *exceed* the reactive oracle, and its per-episode survival is high-variance (Fig. 6a). A finer tokenizer ($K{=}256$) reconstructs more sharply (MSE 0.0009) and is a promising route to super-oracle dodging, but its controller was under-budgeted here and collapsed; a longer dream horizon (so survival is not saturated) and controller ensembling are natural next steps. Our reward is survival-only; richer objectives, a diffusion world model for sharper dreams [16], and the full multi-round IRIS collect↔train loop at larger scale are open directions.

9 Conclusion

An agent trained *entirely* inside a from-scratch Transformer world model learns genuinely reactive fireball-dodging in VizDoom `take_cover`, matching a hand-coded oracle and beating model-free RL at equal real-data budget, verified on four held-out seed sets. The path there—codebook collapse, a silent termination-head collapse, and world-model exploitation, each diagnosed and fixed—is a compact, honest case study in what makes model-based RL work in practice.

Reproducibility. All code, configs, checkpoints, and the playable simulator are released open source. The winning controller and its independent-seed verification script are included.

References

[1] J. Schmidhuber. Making the World Differentiable: On Using Fully Recurrent Self-Supervised Neural Networks for Dynamic Reinforcement Learning. *Tech. Rep.*, TUM, 1990.

[2] R. S. Sutton. Dyna, an Integrated Architecture for Learning, Planning, and Reacting. *SIGART Bulletin*, 1991.

[3] A. van den Oord, O. Vinyals, K. Kavukcuoglu. Neural Discrete Representation Learning (VQ-VAE). *NeurIPS*, 2017.

[4] V. Mnih et al. Human-Level Control through Deep Reinforcement Learning. *Nature*, 2015.

[5] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, W. Jaśkowski. ViZDoom: A Doom-based AI Research Platform for Visual RL. *IEEE CIG*, 2016.

[6] D. Ha and J. Schmidhuber. World Models. *NeurIPS*, 2018.

[7] D. Hafner et al. Learning Latent Dynamics for Planning from Pixels (PlaNet). *ICML*, 2019.

[8] D. Hafner, T. Lillicrap, J. Ba, M. Norouzi. Dream to Control: Learning Behaviors by Latent Imagination (Dreamer). *ICLR*, 2020.

[9] D. Hafner, T. Lillicrap, M. Norouzi, J. Ba. Mastering Atari with Discrete World Models (DreamerV2). *ICLR*, 2021.

[10] D. Hafner, J. Pasukonis, J. Ba, T. Lillicrap. Mastering Diverse Domains through World Models (DreamerV3). *arXiv:2301.04104*, 2023.

[11] M. Janner, J. Fu, M. Zhang, S. Levine. When to Trust Your Model: Model-Based Policy Optimization (MBPO). *NeurIPS*, 2019.

[12] J. Schrittwieser et al. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model (MuZero). *Nature*, 2020.

[13] V. Micheli, E. Alonso, F. Fleuret. Transformers are Sample-Efficient World Models (IRIS). *ICLR*, 2023.

[14] J. Robine, M. Höftmann, T. Uelwer, S. Harmeling. Transformer-based World Models Are Happy with 100k Interactions (TWM). *ICLR*, 2023.

[15] W. Zhang et al. STORM: Efficient Stochastic Transformer based World Models. *NeurIPS*, 2023.

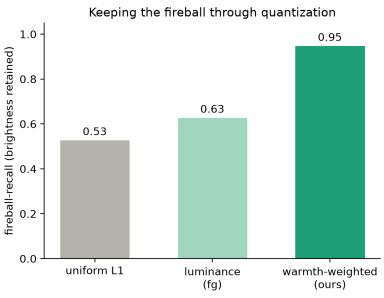
[16] E. Alonso, A. Jelley, V. Micheli, A. Kanervisto, et al. Diffusion for World Modeling: Visual Details Matter in Atari (DIAMOND). *NeurIPS*, 2024.

[17] D. Valevski, Y. Leviathan, M. Arar, S. Fruchter. Diffusion Models Are Real-Time Game Engines (GameNGen). *arXiv:2408.14837*, 2024.

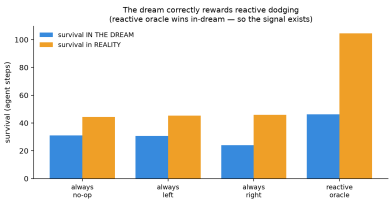
[18] J. Bruce et al. Genie: Generative Interactive Environments. *ICML*, 2024.

[19] N. Hansen and A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies (CMA-ES). *Evolutionary Computation*, 2001.

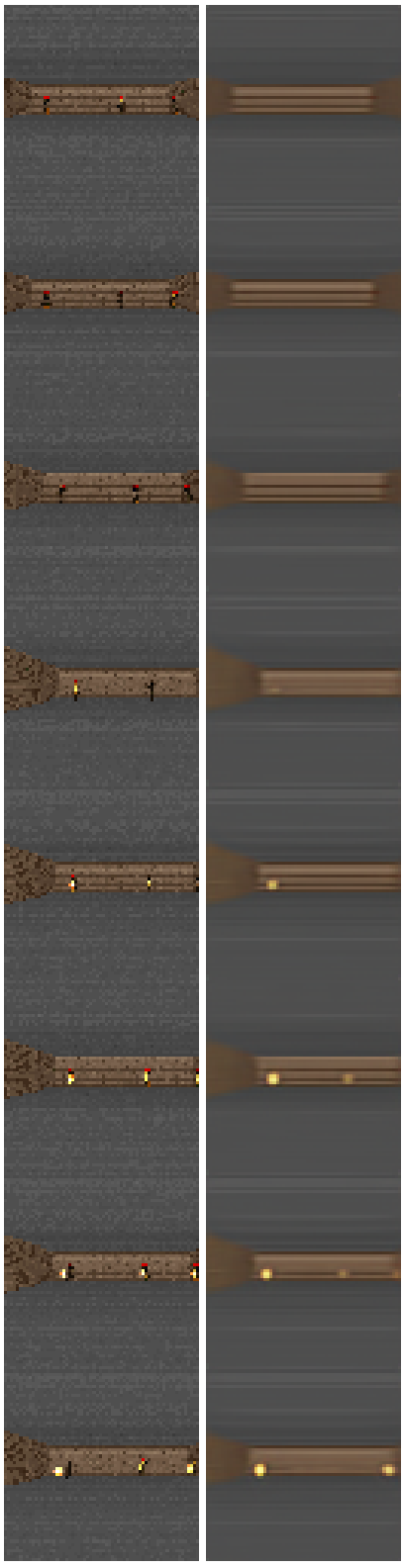
[20] T. Salimans, J. Ho, X. Chen, S. Sidor, I. Sutskever. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *arXiv:1703.03864*, 2017.



(a) Fireball recall.



(b) Dream rewards dodging.



(c) Recon (real|decoded).

Figure 8: Component analysis. (a) Warmth-weighted reconstruction keeps the fireball ($0.53 \rightarrow 0.95$). (b) The reactive oracle wins *inside the dream*, so the dodging signal exists. (c) The tokenizer preserves fireballs (left: real, right: decoded).

Figure 9: The imagination-trained agent playing the *real* game: eight frames spanning one episode as it strafes to keep clear of incoming fireballs. It never saw the real game during training.