Reward, or the Observation Stream? Auditing World Model Quality in RL on a Turbofan Substrate

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning agents in partially observable environments need internal models of the world to act well and to support downstream tasks. What signal should shape those models? One view holds that maximising reward in a loop of action and observation is sufficient: a good policy implies a good internal model as a byproduct Silver et al. (2021). An opposing view holds that reward shapes representations toward a minimum policy-sufficient statistic and discards the rest; faithful world models must emerge instead from self-supervised prediction on the observation stream LeCun et al. (2022). We arbitrate between the two on TurboSens, a turbofan engine simulator in which the true latent state is exposed as ground truth, so an encoder's learned representation can be probed directly against the state it should have recovered. The latent has two parts: an underlying wear state that drifts slowly, and a fouling layer (dust and deposits on engine components) that masks it and which the agent's own maintenance actions periodically clear. Together, Turbosens produces dependant noisy sensor observations across multiple operating contexts that the model observes. We find that the two objectives shape the encoder along orthogonal axes. PPO tracks the fouling, because the policy must react to it, and discards the underlying wear. Self-supervised pretraining with JEPA does the opposite: it recovers the wear better but is invariant to fouling. Both build internal models of the engine, but of different parts. We show that the two compose cleanly sequentially rather than jointly: pretrain the encoder once with JEPA, freeze it, then train PPO on top per task. The recipe matches end-to-end PPO on the default reward, improves wear recovery, and yields a reusable encoder. We discuss implications for representation learning in safety-critical domains.

1 Introduction

Many systems we want learning agents to operate are only partially observable. Predictive maintenance of an aircraft engine is a canonical instance Côme; Fan et al. (2024): internal wear is distributed across rotating components (fan, compressors, turbines) that operators would like to track for maintenance planning, anomaly detection, and remaining useful life forecasting De Pater and Mitici (2022), but none of it is directly observable. Operators rely on sensors whose readings reflect the underlying wear together with operating conditions Peng et al. (2021); De Pater and Mitici (2023), ambient regime, and fouling, the dust and pollution deposits that accumulate on engine surfaces and obscure the sensor signal until the next maintenance event clears them. A drop in apparent efficiency on a single reading could be real wear, or it could be dust; only the history of past sensor readings and maintenance actions, integrated into a coherent latent representation, resolves the ambiguity. To act in such environments, and to support downstream consumers of its perception (predictive maintenance pipelines, anomaly detectors, transfer to new tasks), a learning agent therefore needs

an internal model of that latent state. In modern deep learning practice this model is carried by an encoder: a neural network module that compresses raw observations into a low-dimensional latent on top of which a policy, value function, or downstream predictor can be trained. We call this latent the agent's world model when it faithfully captures the true state of the system, faithfully enough that downstream tasks the agent was not originally trained for can read the state off it. The question of what signal should shape such a representation has two competing answers.

The reward is enough hypothesis Silver et al. (2021) holds that an agent maximising reward in a continual loop of action and observation acquires a faithful internal model as a byproduct of learning a good policy: perception, on this view, is subsumed by the pressure to act well. An opposing view LeCun et al. (2022); Maes et al. (2026) holds that the reward gradient shapes the encoder toward a minimum sufficient statistic for the policy and discards the rest; faithful world models must instead emerge from self-supervised prediction on the observation stream itself. These two views correspond to two different ways of using the world's feedback. The reward is a scalar projection per step that compresses many channels of consequence (operating cost, fuel burn, maintenance overhead) into one. The observation stream is the same kind of world signal in its uncompressed form: many channels, each reflecting the underlying state through a different physical coupling. The question is not whether to use feedback from the world (both views do) but which form of it should shape the agent's perception.

We arbitrate between the two views on TurboSens Thil et al. (2026), an interactive turbofan degradation environment that exposes the engine's true latent state as ground truth at training time but never to the agent. The substrate makes the audit unambiguous: the simulator records paired (observation stream, ground truth state) trajectories, so we can probe a trained encoder's representation directly against the state it should have recovered. We use a linear probe Alain and Bengio (2016): an ordinary least squares regressor fit from encoder activations onto the ground truth on held-out trajectories. The probe is kept linear so that the comparison reflects what is directly recoverable from the representation, not what a deeper network could in principle reconstruct. Code at is released at `https://anonymous.4open.science/r/turbosens-CBCD`.

What this paper asks.

1. *What does PPO learn about the engine?* The encoder under reward gradient tracks the fouling layer, because the policy must react to it, and partially discards the underlying wear. The recurrent state that feeds the actor and critic then compresses further to a policy-sufficient statistic with $R^2(s, c) \approx 0.04$.
2. *What does SSL learn?* A JEPA model trained on the same backbone preserves the underlying wear at near-oracle quality and is structurally invariant to fouling: even a sixteen-frame window of JEPA encoder outputs cannot linearly recover the fouling state. The two losses shape the encoder along orthogonal axes.
3. *Can the two be combined?* Pretrain the encoder once with JEPA, freeze it, then train PPO on top per task. The aggregator under reward learns fouling tracking from any upstream encoder; the frozen JEPA encoder contributes wear preservation. The recipe matches end-to-end PPO on the default reward, transfers across reward functions, and gives a reusable encoder.

2 Background and related work

World models and what shapes them. The two views introduced in Section 1 correspond to two technical traditions. On the reward-is-enough side, model-free policy gradient methods such as PPO Schulman et al. (2017) commit fully to the position that representation learning is a byproduct of policy optimisation, with no explicit objective on the encoder. Model-based methods such as Dreamer and its successors Hafner et al. (2019, 2024) take a softer version: a world model is learned alongside the policy, but its objective (reconstruction of future observations) is still inserted into a reward-driven pipeline. On the observation-stream side, the JEPA family LeCun et al. (2022); Balestriero and LeCun (2025); Maes et al. (2026) argues that the right level for predictive modelling is the latent embedding rather than the pixel observation, and that a prediction loss on latents (with an anti-collapse regulariser such as SIGReg Balestriero and LeCun (2025); Epps and Pulley (1983)) suffices, with no reward signal at any point. The most recent member of that family, LeWorldModel Maes et al. (2026), achieves competitive representation quality with a streamlined latent MSE plus

SIGReg recipe in a purely offline, reward-free setting. We use this method as the SSL component in our experiments.

Auxiliary objectives in reinforcement learning. A third tradition occupies the middle ground: keep the reward gradient as the primary signal, but add a representation-learning loss alongside it. UNREAL Jaderberg et al. (2016), SPR Schwarzer et al. (2020), and CURL Laskin et al. (2020) are representative; they add prediction, contrastive, or auxiliary-control losses to the policy gradient. SPR in particular shares structure with the joint losses we evaluate in Section 6.1, differing in its use of a momentum target encoder and a stop-gradient design borrowed from BYOL Grill et al. (2020) rather than the explicit anti-collapse regulariser of the JEPA family. We find that this middle ground does not work cleanly on our substrate, for reasons explored in Section 6.1.

Linear probes for representation evaluation. A linear probe Alain and Bengio (2016); Conneau et al. (2018) treats the quality of a learned representation as the answer to the question *how well does a linear function of the activations recover a target variable?* The probe is kept linear so the comparison reflects what is directly recoverable from the representation, not what a deeper network could in principle reconstruct. On reinforcement learning agents, the protocol has mostly been applied with pixel observations and hand-designed targets such as object positions Anand et al. (2019); Schwarzer et al. (2020); the substrate of the present work removes that proxy by exposing an explicit ground-truth latent.

Aerospace prognostics. Open-loop turbofan prognostics benchmarks (CMAPSS, N-CMAPSS) Saxena et al. (2008); Arias Chao et al. (2021) record trajectories to failure but are not interactive and do not expose latent state; TurboSens does both, and uses OpenDeckSMR Psaropoulos et al. (2025).

3 TurboSens: action conditioned observability in a turbofan

Setup. TurboSens models a turbofan engine as a partially observable Markov decision process (POMDP). The hidden state at each time step has two components: a slow-drifting wear state s_t describing the engine's underlying health, and a fast-changing fouling state c_t describing surface deposits that accumulate during operation. The agent selects a maintenance action a_t at the start of each operating window, observes noisy sensor readings o_t, and receives a reward r_t combining operating cost (fuel consumption, read directly from one of the sensors) and maintenance cost (a designer-specified scalar per action). The true latent (s_t, c_t) is recorded by the simulator at every step and exposed at training time as ground truth, but never to the agent itself.

The observability structure. What makes the substrate well-suited to representation auditing is the way the two latent components interact in the observation. The sensors are physically affected by both: a degraded module produces a sensor signature, and so does a fouled module, and the two signatures overlap. A drop in apparent efficiency on a single reading is consistent with many (s, c) pairs. Crucially, the fouling state is action-controllable: maintenance actions targeting a module clear its fouling, exposing the underlying wear in the next few sensor readings; fouling then rebuilds and masks the wear again. An encoder that ignores the agent's action history cannot tell these regimes apart, since the same observation can come from a clean engine that has worn or a dirty engine that has not. An encoder that integrates the action history can. This makes the substrate a natural test for whether a representation has captured the underlying dynamics or only the surface observation.

State. We denote the wear state $s_t \in \mathbb{R}^{10}$, with two indices (efficiency and corrected mass flow) for each of five rotating modules: fan, booster, high-pressure compressor, high-pressure turbine, and low-pressure turbine. The fouling state $c_t \in \mathbb{R}^5$ has one channel per module. The full latent is the concatenation $(s_t, c_t) \in \mathbb{R}^{15}$.

Dynamics. Wear and fouling evolve under disjoint subsets of actions: maintenance restores wear, cleaning removes fouling. Concretely,

$$s_{t+1} = s_t + R(a_t) \odot (s^\star - s_t) + \boldsymbol{\xi}_t^s, \tag{1}$$

$$c_{t+1} = \boldsymbol{\gamma}(a_t) \odot c_t + \boldsymbol{\xi}_t^c, \tag{2}$$

where $R(a_t) \in \{0,1\}^{10}$ is a binary mask selecting which components an action restores toward the nominal state $s^\star$; $\gamma(a_t) \in [0,1]^5$ is a per-module fouling decay vector (close to 1 under `do_nothing`, close to 0 under a cleaning action targeting that module); $\odot$ denotes elementwise multiplication; and ξ_t^s, ξ_t^c are stochastic process noise terms.

Observation. The sensor channel does not see s_t directly. It sees an apparent state $\tilde{s}_t$ in which the fouling layer is additively coupled into the apparent wear through a fixed coupling matrix $C \in \mathbb{R}^{10\times 5}$, together with a sign clamping $M \in \mathbb{R}^{10\times 10}$ that keeps measurable indices physically valid:

$$\tilde{s}_t = \mathrm{clip}\big(s_t + M \cdot \max(0, -s_t) + Cc_t,\ \tilde{s}^{\min}, \tilde{s}^{\max}\big)\,, \tag{3}$$

$$o_t = g(\tilde{s}_t, \phi_t) + \varepsilon_t. \tag{4}$$

Here ϕ_t is the flight context (operating regime and ambient region), g is the simulator's sensor model, ε_t is sensor noise, and $M \in \mathbb{R}^{10\times 10}$ is a sign-clamping correction keeping measurable indices physically valid. The observation o_t contains readings from multiple operating contexts at each step (either four or sixteen contexts depending on the tier; see below), so the agent's per-step observation is structured rather than a single vector.

Dataset. The version of TurboSens used in this paper Thil et al. (2026) contains on the order of 10^3 episodes per scenario tier, each spanning 1.4×10^4 timesteps. Each episode records, at every step, the latent (s_t, c_t), seven sensor channels per context across either four or sixteen contexts, the maintenance action, and the operational reward. We use the scenario T2 tier (multi-region, $C = 16$ contexts per step) for all experiments. The latent (s_t, c_t) is what we probe encoders against; it is never input to any agent. Observations generated with OpenDeckSMR turbofan simulator.

4 Experimental setup

Policy architecture. The recurrent policy stacks four modules (Figure 1a). A *SensorEncoder* (two transformer layers, four heads, embedding dimension $d = 64$) maps the $C \cdot 7$ sensor tokens of a frame onto a per-frame embedding enc_t; each frame is then fused with a 16-dimensional embedding of the action that triggered it and projected back to d. A *TemporalAggregator* (one transformer layer, four heads) sees a window of $H = 16$ fused frames and outputs the aggregator embedding $z_t^{\text{enc}} \in \mathbb{R}^d$. A single-layer *LSTM* on top produces a hidden state $z_t^{\text{rec}} \in \mathbb{R}^h$ ($h = 128$) that persists across steps and is reset on episode termination, and linear *actor* and *critic* heads consume it. We train with PPO Schulman et al. (2017) for 5×10^2 updates of 2×10^3 environment steps each. A non-recurrent MLP variant (skipping the TemporalAggregator and LSTM) serves as a control.

Receptive field. The TemporalAggregator's $H = 16$ frame window does not have to consist of consecutive simulator steps. A receptive-field stride S controls the spacing: $S = 1$ exposes the encoder to 16 consecutive simulator steps; $S = 20$ stretches the same buffer over 320 steps. At $S = 1$, the encoder sees only the fast fouling dynamics within a single operating window. At $S = 20$, the buffer spans long enough to bracket maintenance events, exactly the regime transitions that (per Section 3) make (s, c) disambiguable. We sweep $S \in \{1, 5, 20\}$ as a deliberate manipulation of how much action history the encoder is allowed to see.

Probe protocol. For each trained checkpoint we roll out N deterministic episodes on held-out random seeds, log $(o_t, a_t, s_t, c_t, z_t^{\text{enc}}, z_t^{\text{rec}})$ per step, and fit an ordinary least squares regressor from a candidate activation onto $y_t = (s_t, c_t) \in \mathbb{R}^{15}$ on an 80/20 train/test split. We report test R^2 overall and per dimension. When we audit the JEPA encoder in Section 6, the analogous probe target is the per-frame JEPA embedding z_t^{JEPA} (Figure 1b). The probe is linear by design **?**: the comparison measures what is directly recoverable from the representation, not what a deeper network could in principle reconstruct.

Training matrix. We train 36 PPO checkpoints on scenario T2: receptive-field stride $S \in \{1, 5, 20\}$, encoder embedding dimension $d \in \{32, 64, 128, 256\}$, and LSTM hidden $h \in \{64, 128, 256, 512\}$, plus the MLP control. Each run is 10^6 environment steps. Operational return saturates at $\approx 11{,}000$ across recurrent variants and $\approx 11{,}135$ for matched MLP variants (full curves in Appendix A); the operational metric is essentially a single plateau across architecturally diverse runs, and the variation

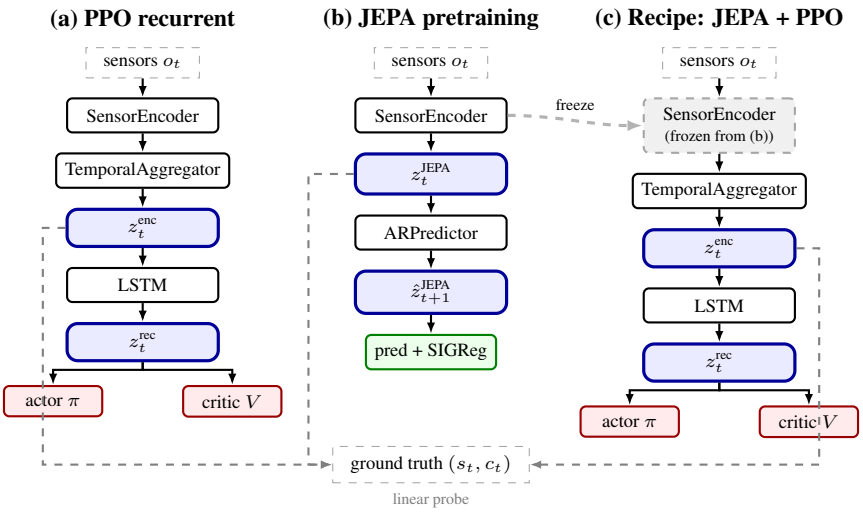

Figure 1: Architectures we compare. (a) End to end PPO trains all modules with the policy gradient. (b) JEPA pretrains the SensorEncoder alone on next embedding prediction with SIGReg, without any reward signal. (c) The recipe: load the SensorEncoder weights from (b), freeze them, train the rest of the PPO stack (TemporalAggregator + LSTM + heads) on top per task. The dashed arrow marks weight transfer from (b) to (c). All three pipelines expose the same probe targets (z_t^{enc}, z_t^{rec} where they exist, and z_t^{JEPA}), which we audit against the simulator's ground truth latent (s_t, c_t) in Section 5 and Section 6.

Table 1: Reward and representation are decoupled across the receptive-field stride. Operational return is unchanged; the aggregator probe R^2 moves by $+0.38$. The LSTM hidden state is collapsed at all three settings. Of the five seeds at $S = 20$, four reach $R^2 \in [0.81, 0.83]$; one underconverges and pulls the mean to 0.72.

Metric	$S = 1$	$S = 5$	$S = 20$
ep_return	$\approx 11{,}000$	$\approx 11{,}000$	$\approx 11{,}000$
R^2 on z_t^{enc}	0.34 ± 0.10	0.61 ± 0.21	$\mathbf{0.72 \pm 0.21}$
R^2 on z_t^{rec}	-0.00 ± 0.04	$+0.01 \pm 0.01$	$+0.01 \pm 0.05$

worth investigating lives elsewhere. The reward is the sum of a designer-specified maintenance cost per action and the fuel consumption read directly from one of the sensors. Because the fuel sensor is also present in o_t, the reward is not an external signal disjoint from the observation stream but a scalar projection of one channel within it, plus an action cost, a point we return to in the discussion.

5 Results

5.1 Receptive field decouples reward from representation

We sweep the receptive-field stride $S \in \{1, 5, 20\}$ at fixed encoder and policy capacity ($d = 64$, $h = 128$, the matrix default), with $N = 5$ seeds per cell. Operational return is statistically indistinguishable across S; the linear probe R^2 on the aggregator embedding z_t^{enc} moves from 0.34 to 0.72 (Table 1). Reward and representation are decoupled.

What varies between $S = 1$ and $S = 20$ is not what the encoder sees per step but how far back in agent time it is allowed to look. At $S = 20$ the encoder's 16-frame buffer spans 320 simulator steps, enough to bracket a maintenance event and observe the regime transition that Equations (2)–(3) predict: fouling drops sharply after a cleaning action and the underlying wear becomes briefly visible in the sensor signature. At $S = 1$ the buffer covers 16 steps, the fouling layer is saturated throughout, and the agent has no view of any transition. The encoder at $S = 1$ is therefore structurally blind to the disambiguation cue that the substrate's dynamics provide.

The operational reward does not surface this difference because the optimal policy on this benchmark is dominated by `do_nothing` once the operating plateau is hit; a policy that performs near-optimally needs a much smaller sufficient statistic than the one a probe asks about. This is the first piece of evidence for the central claim of this paper: a representation that supports a near-optimal policy is not the same object as a representation that recovers the true latent state.

5.2 The collapse of the recurrent state is sharp and tracks policy entropy

Probing every saved PPO checkpoint of the $S = 20$ run ($u \in \{25, 50, 75, \ldots, 475\}$) locates the collapse of the recurrent state to a 50-update window in early training. At $u = 25$ the LSTM hidden state has $R^2 = 0.13$ on (s, c); by $u = 75$ it has dropped to near zero and stays there for the rest of training. The aggregator one layer earlier moves in the opposite direction over the same interval: $R^2 = 0.12$ at $u = 25$, rising to 0.75 by $u = 75$ and plateauing.

The two transitions are sharp rather than gradual, and they coincide with policy entropy saturation: the optimal policy locks in around $u = 50$ to 75, and the recurrent state collapses at the same moment. Information about the latent state that the LSTM *had* early in training is gone at convergence, on the same checkpoints whose operational return is already saturated. This is the cleanest temporal evidence we have that the compression is driven by the policy gradient itself, not by an unrelated capacity bottleneck.

A nonlinear MLP probe (one hidden layer, 64 units) does not lift z_t^{rec} above $R^2 = -0.08$ either, so the compression is information loss rather than nonlinear obfuscation. The collapse is also soft rather than structural with respect to capacity: at moderately larger widths a fraction of seeds escape to $R^2 \approx 0.48$ on the recurrent state, but the qualitative picture (collapse at the matrix default, partial stochastic escape at larger capacity) is robust; full ablation in Appendix A.

5.3 The reward gradient specialises the encoder one layer at a time

The three layers play three different roles, traceable in a single layer-by-layer probe. The per-frame encoder enc_t preserves a substantial fraction of the latent: $R^2(s, c) = 0.27$, with $R^2(s) = 0.55$ on wear and $R^2(c) = 0.19$ on fouling. The TemporalAggregator one layer up gives up some wear and acquires near-perfect fouling tracking: $R^2(s) = 0.34$, $R^2(c) = 0.97$. The LSTM then compresses to a policy-sufficient statistic with $R^2(s, c) = 0.04$ on the full latent.

The story this tells is consistent across our experiments. Under the reward gradient, the encoder preserves whatever its per-frame view permits; the aggregator specialises toward the dimensions the policy must react to (the fouling, which determines maintenance timing); and the recurrent state compresses everything else away. Each layer narrows the representation toward the action-relevant slice, and information not needed for action is discarded along the way.

The practical implication is direct. A deployment pipeline that re-uses the policy network as its representation is reading from the wrong layer: the LSTM hidden state has been compressed to a policy-sufficient representation, and the richer aggregator already specialised toward fouling, is the best the policy stack. To recover an encoder that captures the underlying state and not only the action-relevant projection of it, the encoder needs a different objective. We turn to that in Section 6. A full per-dimension breakdown across all audited layers appears in Figure 2; an out-of-distribution probe on held-out flight regions (Appendix) shows the same layer-wise pattern generalises.

6 JEPA and the recipe

We train the same SensorEncoder backbone with the LeWorldModel Maes et al. (2026) objective on the TurboSens T2 trajectory dataset. No reward gradient enters the encoder at any point. We then probe the trained encoder against (s, c) with the same linear probe used for PPO in Section 5.

Standalone JEPA: wear-preserving, fouling-invariant. The JEPA encoder learns the opposite of what PPO's encoder learns. It preserves better the underlying wear state and is structurally invariant to fouling. Quantitatively, on (s, c) at the per-frame layer it reaches $R^2 = 0.49$, with $R^2(s) = 0.51$ on wear (against a supervised oracle ceiling of ≈ 0.55) and $R^2(c) = 0.10$ on fouling. Concatenating 16 frames of JEPA encoder outputs at stride $S = 20$, a temporal window matching PPO's aggregator,

Table 2: The recipe matches or exceeds end-to-end PPO across three reward functions. Frozen JEPA encoder paired with a trainable PPO downstream (aggregator + LSTM + heads) gives equivalent operational return on the default reward, slightly higher wear recovery $R^2(s)$ at the aggregator, and matches on the fouling tracking the policy needs to time maintenance. The wash-penalty row shows lower return but identical aggregator $R^2(s, c)$ (0.81 vs 0.83), suggesting the gap is policy-side rather than representation-side. Numbers under "frozen JEPA + PPO" are the mean of two converged seeds for the default reward and a single seed for the cross-task variants.

reward	encoder	ep_return	$R^2(s)$ on z_t^{enc}	$R^2(c)$ on z_t^{enc}
default	end-to-end PPO	11132	0.34	0.97
default	**frozen JEPA + PPO**	11097	**0.40**	0.97
uptime max	end-to-end PPO	13791	0.33	0.97
uptime max	**frozen JEPA + PPO**	**13890**	0.34	0.97
wash penalty	end-to-end PPO	10583	0.34	0.97
wash penalty	frozen JEPA + PPO	9300	0.34	0.96

gives essentially the same numbers: $R^2 = 0.46$ overall, $R^2(c) = 0.04$. The fouling information is not present in any temporal window of JEPA encoder outputs we tested, not just absent from a single frame. We attribute this to the structure of the SSL objective itself: predicting future latent embeddings from past ones does not require the encoder to track fouling, because the fouling dynamics that matter for prediction are simple enough to be captured by the autoregressive predictor on top of a wear-aware latent. The encoder is free to specialise toward what is hardest to predict, and that is the underlying state.

The recipe. The two objectives shape the encoder along orthogonal axes; they compose by sequence rather than jointly. We pretrain the SensorEncoder once with JEPA, freeze it, and train the rest of the PPO stack (TemporalAggregator, LSTM, actor, critic) on top per task. The recipe matches end-to-end PPO on operational return and improves wear recovery at the aggregator, on the default reward and on a second reward function we test (uptime-maximisation). The resulting encoder is reusable: a single frozen JEPA encoder serves multiple downstream policies trained for different objectives.

Why the recipe works. The aggregator under reward gradient learns fouling tracking from *any* upstream encoder: end-to-end PPO, frozen random, frozen JEPA all give the aggregator $R^2(c) \approx 0.97$. The encoder's job in the policy stack is therefore not fouling tracking; it is preserving the per-frame information the aggregator and LSTM will build on top of. JEPA's encoder happens to do this better than PPO's reward-shaped encoder on the wear dimensions, because the SSL loss has no reason to give up wear in exchange for action-relevant signal. Pretraining JEPA and freezing it upstream of a trainable PPO downstream therefore gives the aggregator a strictly better starting representation to specialise from, and yields a reusable encoder as a side effect.

Cross-task caveat. On wash-penalty the recipe underperforms end-to-end PPO on operational return (9300 vs 10583) despite identical aggregator-level R^2 on (s, c) (0.81 vs 0.83): the representation transfers, the policy does not. Reward functions that need fine-grained action timing may require encoder flexibility that freezing precludes. The claim is therefore narrower: the recipe yields a reusable encoder matching PPO on representation quality, and on operational return for the default and uptime-max rewards.

Per-dimension complementarity. Beyond the aggregate R^2, the two encoders preserve overlapping but distinct subspaces of the latent. A dimension-wise linear probe (Figure 2) shows that JEPA wins on five of the ten wear dimensions; PPO wins on the other five wear dimensions and on every fouling channel. Canonical correlation analysis confirms a shared 5-dimensional subspace (top correlations > 0.9), and a linear probe on the concatenated encoder outputs reaches $R^2 = 0.87$, $+0.04$ over PPO alone. The combination is materially helpful on the few dimensions one method has captured and the other has not. The two objectives encode different things about the engine: JEPA encodes what the world is, PPO encodes what the policy needs to know to act on it.

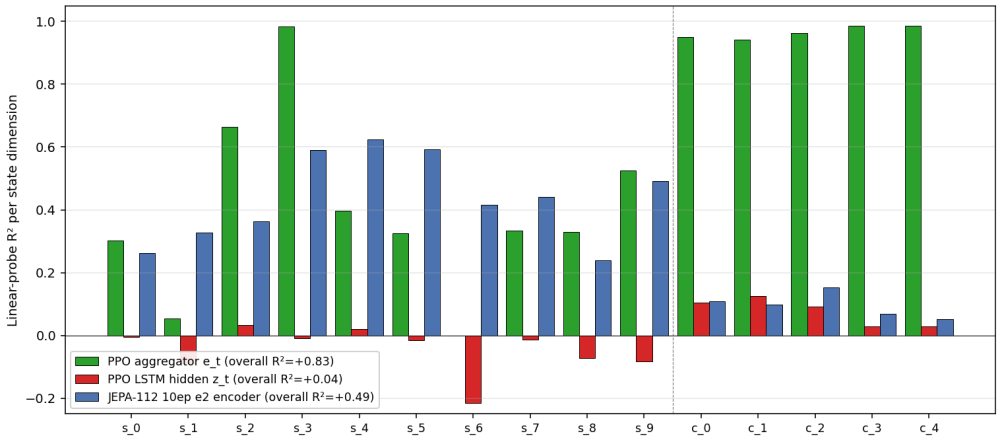

Figure 2: Per-dimension linear-probe R^2 for the PPO aggregator z_t^{enc} (green), the PPO LSTM hidden z_t^{rec} (red), and the JEPA encoder z_t^{JEPA} (blue), against the ten wear dimensions s and the five fouling dimensions c. PPO saturates the fouling channels and the recurrent state collapses across every dimension; JEPA covers the wear dimensions PPO gives up, with little overlap on fouling. The two encoders carry complementary information about the latent.

Table 3: Per-action marginal forecast $R^2(s)$, grouped by whether the post-branch action preserves the wear state (`do_nothing`, `engine_wash`) or discretely changes it (overhauls, targeted patch). Recipe is the frozen-JEPA + PPO encoder, 3-seed mean. PPO values are mean across five seeds with underlying forecasts at $R^2 \approx -25$ to -800 (the policy-shaped encoder does not generalise off the deployed action distribution). Per-action breakdown in Appendix A.

	state-preserving			state-changing		
horizon k	JEPA	Recipe	PPO	JEPA	Recipe	PPO
25	$+\mathbf{0.33}$	$+\mathbf{0.30}$	$\ll 0$	-0.03	-0.17	$\ll 0$
100	$+\mathbf{0.32}$	$+\mathbf{0.29}$	$\ll 0$	$+0.01$	-0.10	$\ll 0$
250	$+\mathbf{0.30}$	$+0.19$	$\ll 0$	$+0.07$	-0.14	$\ll 0$

6.1 Forecasting future wear

The operationally relevant test for predictive maintenance is whether an encoder can flag, ahead of time, that an intervention will be needed; the maintenance itself is carried out by ground crews with direct inspection of the engine. We therefore evaluate each encoder's ability to predict future wear s_{t+k} from its activation at time t. We sample $M = 8$ branch points per held out episode, replay the simulator forward $k \in \{25, 100, 250\}$ steps under the deployed policy's actions, and fit a linear probe from z_t^{JEPA} onto s_{t+k} for JEPA and an action-conditioned MLP forecasting head on top of the frozen PPO aggregator z_t^{enc} for PPO (training setup in Appendix B).

Splitting post-branch action sequences into *state-preserving* (`do_nothing`, `engine_wash`; leave s untouched) and *state-changing* (overhauls and targeted patch; discretely restore s), JEPA holds $R^2(s) \approx 0.30$ on the former across horizons but falls to near zero on the latter (Table 3); PPO is uniformly large negative regardless of action. The frozen-JEPA + PPO recipe encoder inherits JEPA's forecasting capability at essentially no cost ($R^2(s) \approx 0.29$ on state-preserving actions, 3 seeds). The fouling axis reverses: a counterfactual sign probe on c_{t+k} shows PPO above chance (≈ 0.7 at $k = 100$) and JEPA at chance (≈ 0.58). Each encoder forecasts the half of the latent that its training objective rewarded it for representing.

JEPA's drop on state-changing actions is what the SSL objective predicts: the encoder learns the slow drift of s but has no reason to model action-triggered resets. The state-preserving regime is the operationally relevant one for predictive maintenance, and the recipe inherits it. A single reusable encoder therefore supports both an operating policy with end-to-end PPO's return and a downstream wear forecaster with JEPA-level $R^2(s)$.

Why not co-train instead? A natural alternative is to add the JEPA prediction loss to the PPO objective and update the encoder end-to-end with both gradients. We tested this across three attach layers $\ell \in \{\mathrm{enc}_t, z_t^{\mathrm{enc}}, z_t^{\mathrm{rec}}\}$ and three loss weightings at each ($N = 6$ runs per cell). Attached at enc_t the prediction loss matches end-to-end PPO on return but does not lift $R^2(z_t^{\mathrm{rec}})$ above baseline (0.02 to 0.04): the aggregator and LSTM compress its shaping away before the gradient reaches the heads. Attached at z_t^{rec} itself it breaks training, with prediction loss stalled at 0.88, $\mathrm{emb_std}$ below the SIGReg collapse threshold, and return oscillating; raising SIGReg from 0.10 to 0.30 produced identical curves, so the instability is structural. The reason is that reward and prediction shape the encoder toward different summaries of the latent: when both gradients reach the same parameter, it lands at the pull with the stronger signal at convergence, which is PPO's. Pretraining and freezing decouples the two phases.

7 Discussion: open questions for the workshop

Q1. What is the abstract structure of substrates where this recipe should work? TurboSens has a specific latent structure: a slow state of interest, a faster nuisance variable that masks it, and agent actions that move the nuisance. We expect the recipe (SSL pretraining for state preservation, frozen upstream of reward training for action-relevant specialisation) to generalise to substrates with the same abstract shape. Candidate domains: HVAC systems with filter fouling and scheduled cleaning; battery cells with surface films masking underlying capacity loss; agricultural yields under intervention, where soil and weather state are masked by recent management actions.

Q2. What auditability bar must a representation clear for industrial deployment? The recipe yields a probe R^2 of 0.84 on (s, c) at the aggregator and 0.49 at the encoder, with the encoder reusable across reward functions. Whether these numbers clear an operator's deployability bar is not an RL question; it depends on the consequence model the operator works within. Some dimensions of the latent matter more than others (turbine wear is safety-critical; booster mass flow is mostly economic), and the probe-quality bar plausibly varies per dimension. The bar is also temporal: a representation that recovers the state with $R^2 = 0.5$ at the per-frame layer is more or less useful depending on whether the operator needs it for shift-by-shift maintenance scheduling or for long-horizon fleet planning. SOTA operational return alone does not answer any of these questions, and the workshop community, closer to deployment than the RL community alone, is better placed to set the bar.

Q3. Which other world signals could shape representations? Reward and the observation stream are two world signals; they are not the only ones. Industrial assets also produce safety events (alarms, threshold crossings, near-failures), operator interventions outside the scope of nominal actions (unscheduled maintenance, parts replacements), and economic outcomes that are observable but lagged (fuel-cost reports, fleet-availability metrics). Each is a potential supervisory signal that could shape an encoder differently from either reward or prediction. A safety event, for instance, is a sparse but highly informative label about latent state: an alarm at time t tells the agent the state was in a specific region of latent space. Whether such signals are best integrated as additional auxiliary losses, as targets for separate probe heads, or as anchors for self-supervised contrastive objectives is, to our knowledge, open. The workshop's framing of world feedback as a first-class training signal suggests this is the right community to investigate.

Conclusion

Reward maximisation Silver et al. (2021) and self-supervised prediction on observations LeCun et al. (2022) are not in conflict so much as orthogonal on TurboSens. PPO and JEPA both learn to represent the engine but capture different parts of it: reward training tracks the fouling layer the policy must react to, self-supervised prediction recovers the underlying wear because seeing through the mask is the only way to predict future observations. The two compose by sequence rather than jointly: pretrain the encoder once with JEPA, freeze it, train PPO on top per task. We find the joint recipe to match end-to-end PPO on the default reward, while matching the vanilla JEPA underlying state prediction. It yields a reusable representation, and a question motivating world models more generally: what supervisory signal teaches an agent the mechanisms beneath its observations.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari. *Advances in neural information processing systems*, 32, 2019.

Manuel Arias Chao, Chetan Kulkarni, Kai Goebel, and Olga Fink. Aircraft engine run-to-failure dataset under real flight conditions for prognostics and diagnostics. *Data*, 6(1):5, January 2021. ISSN 2306-5729. doi: 10.3390/data6010005. URL https://www.mdpi.com/2306-5729/6/1/5.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. (arXiv:2511.08544), November 2025. doi: 10.48550/arXiv.2511.08544. URL http://arxiv.org/abs/2511.08544. arXiv:2511.08544 [cs].

Alexis Conneau, German Kruszewski, Guillaume Lample, Loic Barrault, and Marco Baroni. What you can cram into a single &!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, 2018.

Côme. Aircraft engine health monitoring using self-organizing maps. URL https://hal.science/hal-00458298v1.

Ingeborg De Pater and Mihaela Mitici. Novel metrics to evaluate probabilistic remaining useful life prognostics with applications to turbofan engines. *PHM Society European Conference*, 7 (1):96–109, 2022. ISSN 2325-016X, 2325-016X. doi: 10.36001/phme.2022.v7i1.3320. URL https://papers.phmsociety.org/index.php/phme/article/view/3320.

Ingeborg De Pater and Mihaela Mitici. Developing health indicators and rul prognostics for systems with few failure instances and varying operating conditions using a lstm autoencoder. *Engineering Applications of Artificial Intelligence*, 117:105582, January 2023. ISSN 09521976. doi: 10.1016/j.engappai.2022.105582. URL https://linkinghub.elsevier.com/retrieve/pii/S0952197622005723.

Thomas W Epps and Lawrence B Pulley. A test for normality based on the empirical characteristic function. *Biometrika*, 70(3):723–726, 1983.

Yalun Fan, Weilin Li, Liang Guo, Yaoxiang Yu, Tao Luo, and Hongli Gao. A metric learning-based health indicator for early degradation point detection of machines. In *2024 Global Reliability and Prognostics and Health Management Conference (PHM-Beijing)*, page 1–8, Beijing, China, October 2024. IEEE. ISBN 979-8-3503-5401-0. doi: 10.1109/PHM-Beijing63284.2024.10874536. URL https://ieeexplore.ieee.org/document/10874536/.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. (arXiv:2006.07733), 2020. doi: 10.48550/arXiv.2006.07733. URL http://arxiv.org/abs/2006.07733. arXiv:2006.07733 [cs].

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. (arXiv:2301.04104), April 2024. doi: 10.48550/arXiv.2301.04104. URL http://arxiv.org/abs/2301.04104. arXiv:2301.04104 [cs].

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, pages 5639–5650. PMLR, 2020.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. (arXiv:2603.19312), March 2026. doi: 10.48550/arXiv.2603.19312. URL `http://arxiv.org/abs/2603.19312`. arXiv:2603.19312 [cs].

Cheng Peng, Yufeng Chen, Qing Chen, Zhaohui Tang, Lingling Li, and Weihua Gui. A remaining useful life prognosis of turbofan engine using temporal and spatial feature fusion. *Sensors*, 21(2): 418, January 2021. ISSN 1424-8220. doi: 10.3390/s21020418. URL `https://www.mdpi.com/1424-8220/21/2/418`.

M. Psaropoulos, V. Gkoutzamanis, A. I. Kalfas, P. Giannakakis, S. Razakarivony, S. Thepaut, and D. Q. Vu. OpenDeckSMR, 2025. URL `https://github.com/OpenDeckLab/OpenDeckSMR`.

Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 International Conference on Prognostics and Health Management*, page 1–9, Denver, CO, USA, October 2008. IEEE. ISBN 978-1-4244-1935-7. doi: 10.1109/PHM.2008.4711414. URL `http://ieeexplore.ieee.org/document/4711414/`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. Reward is enough. *Artificial intelligence*, 299:103535, 2021.

Lucas Thil et al. TurboSens: An interactive turbofan environment for probing world model representations against ground truth. In preparation, 2026.

A Supplementary material

Per checkpoint probe trajectory. Probing every saved PPO checkpoint of the $S = 20$ run at $u \in \{25, 50, \ldots, 475\}$ locates the destruction in time. The aggregator z_t^{enc} climbs from $R^2 = 0.12$ at $u = 25$ to 0.75 by $u = 75$ and plateaus; the LSTM hidden z_t^{rec} briefly carries $R^2 = 0.13$ at $u = 25$ and crashes to near zero by $u = 75$, where it remains. Both transitions coincide with policy entropy saturation around $u = 50$. The receptive field variance bars (mean $\pm$ std over $N = 5$ seeds per cell at $S \in \{1, 5, 20\}$) appear in the released figure set.

Operational return across the run matrix. Figure 3 reports operational return across the 36 trained runs that underpin the analysis. ep_return saturates at $\approx 11{,}000$ across recurrent variants (and at $\approx 11{,}135$ for the matched MLP variants); the operational metric is essentially a single plateau across architecturally diverse runs. All probe analyses use these checkpoints.

Capacity ablation full table. At $(d, h) = (32, 64)$ the aggregator alone reaches $R^2 = 0.47 \pm 0.26$ and the recurrent state collapses universally. At $(64, 128)$ the aggregator is 0.83 ± 0.00 and the recurrent state is 0.01 ± 0.00. At $(128, 256)$ the aggregator is 0.74 ± 0.20, the recurrent state mean is 0.12 ± 0.18, and the max recurrent state across six seeds is 0.48. At $(256, 512)$ the aggregator recovers to 0.83 ± 0.01 but the recurrent state returns to 0.03 ± 0.01. Figure 4 visualises the sweep across both layers.

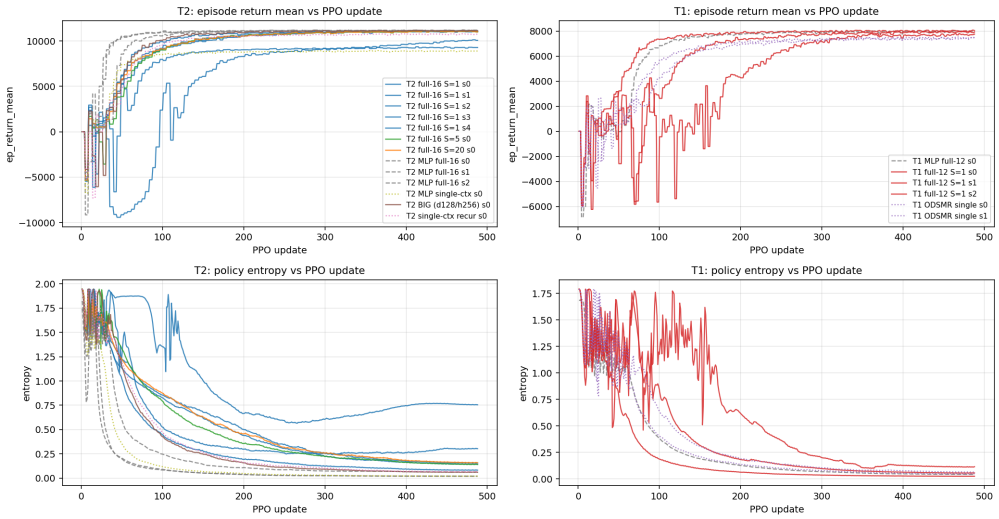

Figure 3: Operational return for the 36 trained runs that underpin the analysis (T2 scenario, 10^6 environment steps each).

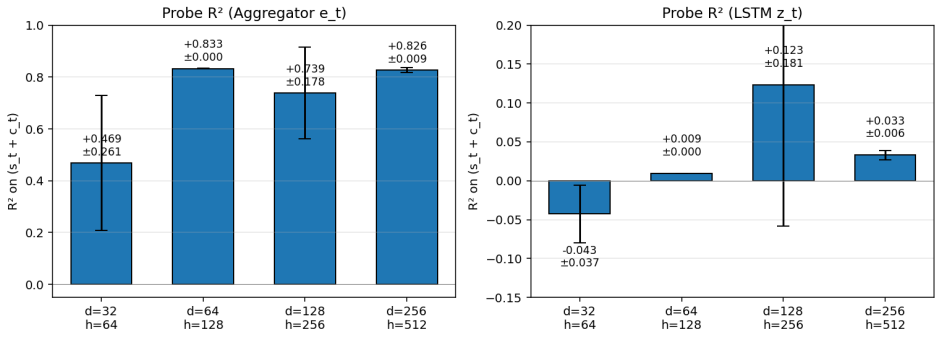

Figure 4: Capacity ablation at $S = 20$ ($N = 3$ seeds per cell): aggregator probe R^2 (left) and LSTM hidden probe R^2 (right) as a function of encoder embedding dimension d and LSTM width h. The recurrent state collapses to near zero at the matrix default and at the largest capacity tested; a stochastic minority of seeds at $(d, h) = (128, 256)$ lifts the recurrent state to $R^2 \approx 0.48$.

Co-training sweep. Figure 5 reports the layer-wise probe R^2 for the co-training experiments referenced in Section 6. A frozen-encoder PPO baseline (left, grey) and a trainable-encoder PPO baseline (blue) bracket the joint configurations. Adding the JEPA prediction loss at the per-frame embedding (orange, green, red, purple, brown) leaves the recurrent state at $R^2 \approx 0$ regardless of the loss weighting; only the SIGReg-only variant (rightmost) lifts the encoder probe slightly, and at the cost of breaking the aggregator and recurrent state.

Out of distribution probing. Training on region Continental and probing on held out regions Arid and Littoral at $S = 20$ drops the aggregator from $R^2 = 0.87$ in distribution to 0.72 to 0.75 out of distribution; the recurrent state remains collapsed at $R^2 \approx 0$ in every region.

Per-action wear forecast. Figure 6 reports the wear forecast $R^2(s)$ broken down by the maintenance action forced at the branch point, complementing the grouped view in Section 6.1. JEPA's degradation tracks the magnitude of the action's effect on s: `do_nothing` and `engine_wash` (which leave s untouched) produce positive $R^2(s)$ across all horizons; targeted single-module overhauls (`fan_overhaul`, `targeted_patch`) fall to near zero; multi-module overhauls (`hpc_overhaul`, `turbine_overhaul`) go mildly negative; `full_overhaul`, the action that resets every mechanical component at once, produces the largest miss ($R^2 = -10.5$ at horizon 25, recovering toward -3.9 by horizon 250 as the post-reset trajectory becomes more predictable). The PPO aggregator + fore-

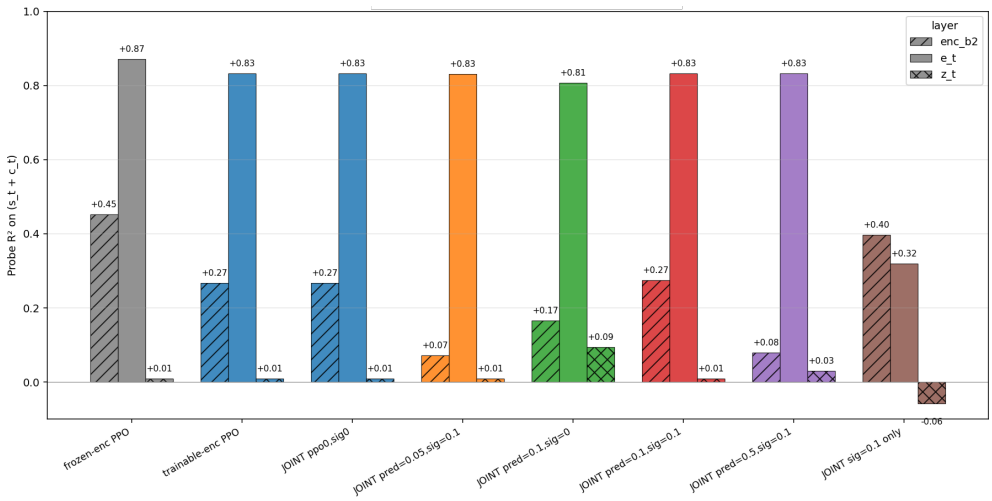

Figure 5: Joint loss probe comparison at T2 full-16, $S = 20$, seed 0, 10^6 environment steps. Bars show probe R^2 on (s, c) at the encoder block 2 (`enc_b2`, hatched), aggregator (z_t^{enc}, solid), and LSTM hidden (z_t^{rec}, cross-hatched) for each loss configuration. Co-training fails to preserve state in the recurrent embedding under any setting we tested.

465 casting head is uniformly large negative (≈ -25 to -800 depending on action and horizon) and
466 shows no comparable structure: the PPO encoder does not generalise off the deployed policy's action
467 distribution.

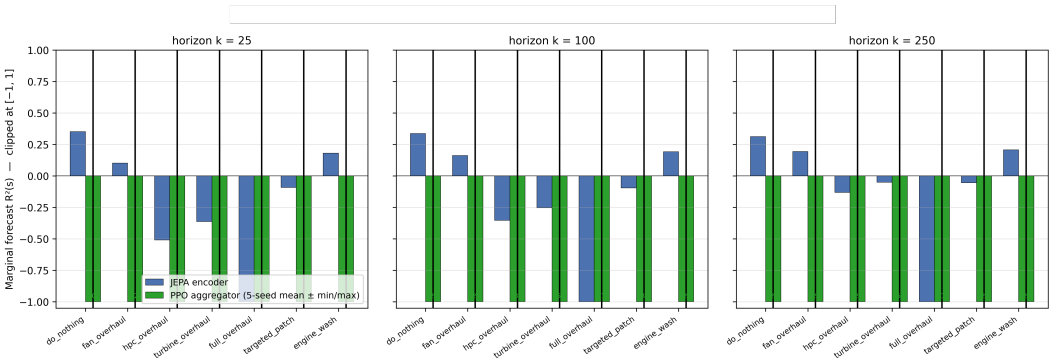

Figure 6: Per-action marginal forecast $R^2(s)$, JEPA encoder versus PPO aggregator + action-conditioned forecasting head, for each of the seven maintenance actions and for horizons $k \in \{25, 100, 250\}$. Bars clipped at $[-1, 1]$; PPO bars are all at -1 in display because the underlying values are large negative.

468
B Architectures and hyperparameters

469 **PPO recurrent policy.** A *SensorEncoder* (two transformer encoder layers, four heads, embedding
470 dimension $d = 64$) maps the $C \cdot 7$ sensor tokens of a single frame onto a per-frame embedding enc_t.
471 Each frame is fused with a 16-dimensional embedding of the action that triggered it (a `nn.Embedding`
472 table over the discrete action set) and projected back to d. A *TemporalAggregator* (one transformer
473 encoder layer, four heads, embedding dimension d) sees a window of $H = 16$ fused frames at
474 receptive-field stride S and outputs the aggregator embedding $z_t^{\text{enc}} \in \mathbb{R}^d$. A single-layer *LSTM* of
475 hidden width $h = 128$ (default; sweep over $\{64, 128, 256, 512\}$) takes z_t^{enc}, persists across steps, and
476 resets on episode termination. Linear actor and critic heads consume the LSTM hidden state z_t^{rec}.
477 The MLP control swaps the TemporalAggregator and LSTM for a two-layer MLP of width d over a
478 flattened length-H window.

PPO training. Adam, learning rate 3×10^{-4}, clip $\epsilon = 0.2$, generalised advantage estimation $\gamma = 0.99$, $\lambda = 0.95$, value-loss coefficient 0.5, entropy coefficient 0.01, max gradient norm 0.5. Mini-batch size 256, four epochs per update. We train for 5×10^2 updates of 2×10^3 environment steps per parallel worker (16 vector workers), totalling $\approx 1.6 \times 10^7$ environment interactions per run. Five seeds per (d, h, S) cell of the matrix in Section 4.

JEPA pretraining. Same SensorEncoder backbone as the PPO policy. An autoregressive predictor (ARPredictor: two-layer MLP, SiLU, hidden 128, action embedding via concatenation) maps $(z_t^{\text{JEPA}}, a_t) \to \hat{z}_{t+1}^{\text{JEPA}}$. Training loss is mean-squared error on the next-latent target plus SIGReg **?** with weight $\beta = 0.1$ on $M = 1024$ random projections. AdamW, learning rate 1×10^{-4}, weight decay 0.05, cosine annealing across 10 epochs, mixed-precision `bfloat16`. Predictor history $H = 16$, stride $S = 20$, `obs_window_size` $= 1$, matching the E_7 configuration of the TurboSens benchmark paper Thil et al. (2026). No reward gradient enters the encoder at any point.

Forecasting head for PPO. For the wear forecast probe of Section 6.1, we attach a small action-conditioned MLP on top of the frozen PPO aggregator: a two-layer MLP with SiLU, hidden 256, taking a 16-step strided history $(z_{t-15S}^{\text{enc}}, \ldots, z_t^{\text{enc}})$ concatenated with a one-hot embedding of the post-branch first action and producing $\hat{z}_{t+k}^{\text{enc}}$. The head is trained on trajectories drawn from a uniform action distribution over the seven maintenance actions (the dataset-generation policy, not the PPO policy), so that the forecaster sees varied action sequences during training. The aggregator weights stay frozen throughout.

Co-training (Section 6.1). Joint loss $L = L_{\text{PPO}} + \alpha\, L_{\text{pred}} + \beta\, \text{SIGReg}(z)$. L_{PPO} is the standard clipped PPO advantage loss plus value loss plus entropy bonus. L_{pred} is mean-squared error between the encoder's z_{t+1} and a same-architecture LatentPredictor's $\hat{z}_{t+1}$ produced from (z_t, a_t); target is not stop-gradiented (per LeWorldModel, SIGReg prevents collapse rather than a stop-gradient). SIGReg uses the Epps-Pulley statistic on 1024 random projections at $\lambda = 1.0$. Sweep: attach layer $\ell \in \{\text{enc}_t, z_t^{\text{enc}}, z_t^{\text{rec}}\}$; $(\alpha, \beta) \in \{(0.05, 0.1), (0.1, 0.1), (0.5, 0.1)\}$; $N = 6$ runs per cell.

Probe protocol. `sklearn.linear_model.LinearRegression` fitted on 80% of (z_t, y_t) pairs from held-out episodes, evaluated on the remaining 20%. The same protocol is used for present-state probes (Section 5, $y_t = (s_t, c_t)$) and forecast probes (Section 6.1, $y_t = (s_{t+k}, c_{t+k})$). Reported R^2 is the standard sklearn coefficient of determination on the test split, averaged across seeds where applicable.

Dataset. We use the T2 multi-region tier of TurboSens Thil et al. (2026): six regions (Continental, Littoral, Arid, Equatorial, Highland, Industrial), ≈ 500 training episodes plus 60 test plus 60 test_hard, mean episode length $14{,}000$ steps, range $8{,}000$ to $20{,}000$. Each step records the latent $(s_t, c_t) \in \mathbb{R}^{15}$, seven sensor channels per context across $C = 16$ contexts, the maintenance action, and the operational reward.