HRP: HIERARCHICAL RESIDUAL PREDICTION FOR LATENT WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce Hierarchical Residual Prediction (HRP), a latent world model where the upper level predicts the lower level's error instead of the future. This allows the upper level to focus only on the states where the lower level fails. The error is a vector and also informs how the lower level failed. This enables the automatic detection of decision points (important events). In addition, we also find that how the correction changes with the action, and not only its size, allows us to separate real decision points from random events that the agent cannot control. Decision points therefore emerge from offline, reward-free transitions with no labels or sparsity penalty, and restricting coarse search to them improves long-horizon success while reaching the best success of prior hierarchies at 2.4$\times$ fewer model rollout steps. The construction is recursive. A third level commits at timescales the second cannot reach.

1 INTRODUCTION

World models are attractive for control because a single model supports planning for any task in an environment, rather than a policy trained per goal. Classical planners require compact states and known dynamics, neither of which is available from raw observation. Latent prediction made planning from observations practical, since a model trained only to predict its own future latents keeps whatever the prediction task needs and discards the rest (LeCun et al., 2022; Zhou et al., 2024; Sobal et al., 2026). What it does not make practical is planning far ahead. Errors compound along a rollout, and the number of action sequences to search grows exponentially with the horizon. Hence a flat model is weakest at the horizons where planning matters the most.

Temporal abstraction is the usual response (Sutton et al., 1999). An upper level that predicts k steps at a time shortens the horizon the planner has to search. It can predict at a fixed rate (Saxena et al., 2021), over learned goals (Hafner et al., 2022), or over macro-actions, as in HWM, which uses a slow model's predictions as subgoals for a fast one (Zhang et al., 2026). In each of these works, the upper level predicts on a fixed stride, which assumes uncertainty accumulates with elapsed time. However, that is not the case for most environments. Wherever the outcome follows deterministically from the action taken, a one-step model carries the trajectory forward on its own. The uncertainty is high at the few states where the future is non-deterministic. A hierarchy tied to those states rather than to a fixed stride allots the planner's budget where it matters. THICK makes this argument, triggering its high level on learned context changes rather than on a schedule (Gumbsch et al., 2024).

How are those context changes learned? THICK learns them from a sparsity penalty, whose coefficient sets how often they occur. Such a penalty can make context changes infrequent, but it cannot make them the states where the agent decides the outcome. Stochasticity in the environment changes the context as much as a real decision does, and escalating there wastes the search, since no choice the agent makes changes the outcome. The lower level fails differently in the two cases, and no existing method looks at *how* it fails.

We introduce **H**ierarchical **R**esidual **P**rediction (HRP), a latent world model whose upper level predicts the lower level's error rather than the future. The lower level rolls forward k steps, its prediction is lifted into a coarse space, and the upper level predicts the difference between that prediction and the state actually reached. The correction vanishes wherever the lower level is already right, so it is sparse without a penalty and the sparsity comes from the environment instead of a coefficient. It is also a vector rather than a score, so it records *how* the lower level failed and not only that it failed.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The latter is what allows us to distinguish a genuine decision point from mere surprise. A large correction alone only signals surprise, and dynamics the agent does not control produce corrections as large as a decision does. We find that the size of the correction locates where the lower level fails but cannot reliably separate decision points from uncontrolled branching, while its dependence on the action separates them reliably. Neither a network trained to regress error size nor a learned sparse gate does better than size.

Decision points therefore emerge from offline, reward free transitions with no labels, rewards, or sparsity penalty, and they tell a planner where its search budget is worth spending. We evaluate on navigation, manipulation, and open ended survival tasks of varying difficulty. The gain is small where an episode contains few decisive states and grows with the number of decisions it contains, both in success and in the search needed to reach it. The construction also recurses, and a third level commits at timescales the second cannot reach.

Contributions (1) A latent world model whose upper level predicts the lower level's error instead of the future, giving a correction that is sparse without a penalty and records how the lower level failed. (2) The size of this correction locates where the lower level fails but cannot separate states the agent controls from states it does not, while its dependence on the action can. (3) Placing coarse search at the detected decision points improves long horizon success at reduced compute, and that the construction recurses, with each level predicting the residual of the level below.

2 Related Work

Latent world models and planning Latent predictive world models plan entirely in latent space (LeCun et al., 2022; Schwarzer et al., 2020; Guo et al., 2022; Zhou et al., 2024; Sobal et al., 2026; Bagatella et al., 2026), and temporal abstraction extends them to long horizons, at a fixed rate (Koutnik et al., 2014; Saxena et al., 2021; Vezhnevets et al., 2017; Nachum et al., 2018), over learned goals (Hafner et al., 2022), or over macro-actions whose slow predictions serve as subgoals for a fast model (Zhang et al., 2026). We hold the stride of Zhang et al. (2026) fixed, so the change we make is isolated to the prediction target. Adaptive schemes learn when to terminate (Bacon et al., 2017) or place boundaries where predictability is unusual (Neitz et al., 2018; Jayaraman et al., 2018; Zakharov et al., 2021), where prediction error spikes (Gumbsch et al., 2019), or where a learned detector fires under a segment bound or a sparsity penalty (Kim et al., 2019; Gumbsch et al., 2024). All of these read a scalar, which Sec. 4.3 shows cannot separate a state the agent decides from one the environment decides. Residual hierarchies exist in reconstruction based models (Xu et al., 2026), but there the residual improves prediction rather than locating where it fails. App. A discusses decision states and controllable dynamics.

3 Hierarchical Residual Prediction for Latent World Models

3.1 Setup

We consider offline, reward free learning from a dataset of trajectories collected by unknown behavior policies. The agent observes o_t and takes action a_t. For a model with L levels we sample segments of length k_L, the stride of the topmost level, together with every intermediate observation, so that each segment supplies targets at all L scales at once. With two levels a segment is $(o_t, a_{t:t+k_2-1}, o_{t+k_2})$. Nothing in the dataset marks task boundaries, events, or rewards.

The model is a stack of levels indexed by i, and every level has the same parts. Level i has a separate encoder ϕ_i, predictor f_i, latents z^i, action a^i, and a stride k_i. Level 1 encodes observations at the full frame rate of the dataset and advances one step at a time. Level $i > 1$ encodes the latents of level $i - 1$ and advances once per k_i steps. Level 1 predicts the next latent, and every predictor above it predicts the residual of the level below. Fig. 1 shows two levels, and indicates how the concept continues with more levels. In contrast to many two-stage world models that either use an off-the-shelf encoder for the latents or train it separately, the encoders and predictors in our model are learned end-to-end.

Following Zhang et al. (2026), level $i > 1$ is conditioned on a macro-action $a^i_t = \rho_i(a_{t:t+k_i-1})$, a continuous vector produced by their transformer action encoder, which we use without modification.

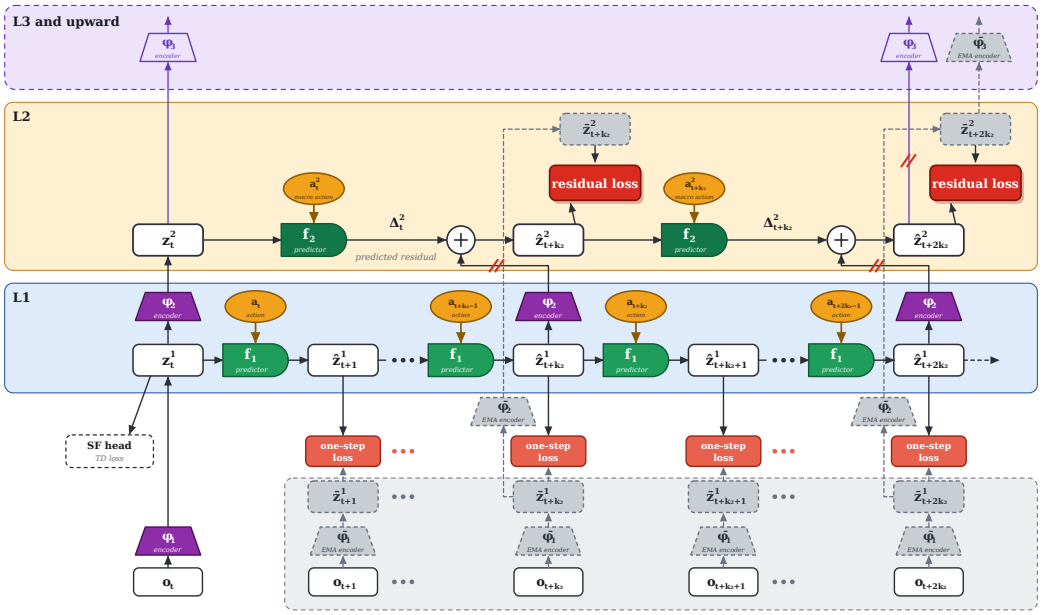

Figure 1: HRP. Level 1 encodes observations and rolls forward one step at a time under primitive actions. Level 2 lifts the same trajectory at stride k_2 and predicts the *residual* of the level-1 rollout rather than the future, conditioned on a macro-action a^2. Stop gradients keep the residual loss from reaching level 1, so the residual measures where level 1 fails on the environment's terms. Every level above repeats the construction on the level below, and the three quantities a third level consumes are shown leaving level 2. Dashed gray elements are EMA copies that receive no gradient.

The search space at each level therefore does not grow with k_i, and the construction applies to continuous and discrete primitive actions alike. App. O details the recursion for a third level.

3.2 Level 1

Level 1 consists of an encoder ϕ_1 and two prediction heads. The encoder maps observations to latents, $z_t^1 = \phi_1(o_t)$, and a copy $\bar{\phi}_1$ updated as an exponential moving average of ϕ_1 supplies prediction targets, which prevents the encoder from making its own targets trivially easy to predict.

The first head is a one step predictor f_1 that maps a latent and an action to the next latent, trained to regress the target encoding of the state actually reached,

$$\mathcal{L}_{\text{step}} = \left\| f_1(z_t^1, a_t) - \bar{\phi}_1(o_{t+1}) \right\|^2. \tag{1}$$

Multi step predictions are formed by applying f_1 repeatedly, so a k_2 step rollout under the chunk $a_{t:t+k_2-1}$ gives $\hat{z}^1_{t+k_2}$. We train on the rollout, applying Eq. 1 at each intermediate step against the corresponding target encoding, so the loss sees that predictions the planner will actually use. The planner rolls this head forward, and level 2 corrects where it fails.

The second head follows TD-JEPA (Bagatella et al., 2026) and predicts long term latent occupancy through a Bellman relation in latent space, recovering the successor features of the policies in the data. A one step objective supervises the representation only about the next state, whereas this head forces ϕ_1 to retain whatever separates states with different long term futures. That matters here because a decision point is exactly such a state, and if ϕ_1 discards what separates the futures, the residual there cannot say which branch was taken. The head is not on the planning path and is used otherwise only for zero shot evaluation and Sec. 4.4 reports detection without it.

The one step head is conditioned on the action. Where the successor changes smoothly with the action, level 1 predicts it and error accumulates only slowly through a rollout. That is most of an episode, and it is what lets level 2 and the coarse search stay idle there. The argument fails where

the successor changes abruptly with the action, as at a junction, and where it does not depend on the action at all, as under uncontrolled dynamics. A level 1 that ignored the action would be wrong everywhere and leave nothing sparse to correct.

3.3 LEVEL 2

Where several futures are possible, the target in Eq. 1 becomes a distribution, and the minimizer of a squared error is its mean. Level 1 therefore predicts the average of the states that can follow, which need not resemble any of them and can correspond to no state the environment visits. Level 2 addresses this failure directly rather than by predicting the future again at a coarser rate. The coarse encoder ϕ_2 maps level-1 latents into the level-2 space, $z_t^2 = \phi_2(z_t^1)$, and the same network is applied to level 1's k_2 step rollout to give the lifted prediction $\phi_2(\hat{z}_{t+k_2}^1)$. The target is formed by an exponential moving average copy $\bar{\phi}_2$ applied to the encoded observation actually reached,

$$z_{t+k_2}^2 = \bar{\phi}_2\big(\bar{\phi}_1(o_{t+k_2})\big), \tag{2}$$

and the residual that level 2 is asked to supply is the difference between the two,

$$\Delta_t^2 = z_{t+k_2}^2 - \mathrm{sg}\big[\,\phi_2(\hat{z}_{t+k_2}^1)\,\big], \tag{3}$$

where sg denotes a stop gradient. The predictor f_2 estimates the residual from the current coarse state and the macro-action, and the corrected prediction is the lifted rollout plus that estimate,

$$\hat{\Delta}_t^2 = f_2\big(z_t^2,\, a_t^2\big), \qquad \hat{z}_{t+k_2}^2 = \mathrm{sg}\big[\,\phi_2(\hat{z}_{t+k_2}^1)\,\big] + \hat{\Delta}_t^2. \tag{4}$$

We write the loss on the corrected prediction,

$$\mathcal{L}_{\text{res}} = \big\|\hat{z}_{t+k_2}^2 - z_{t+k_2}^2\big\|^2, \tag{5}$$

which is identical to $\|\hat{\Delta}_t^2 - \Delta_t^2\|^2$ since the lifted prediction appears on both sides and cancels. We state it in the form of Eq. 5 because $\hat{z}_{t+k_2}^2$ is the quantity the planner scores.

f_2 predicts the error from the current coarse state and the intended macro-action alone, never inspecting level 1's prediction, so the estimate is available at planning time before any future has been observed. Wherever level 1 is already correct, Δ_t^2 is near zero and level 2 has nothing to supply, so level 2 is active on a small fraction of states without any penalty driving it there. The residual is also a vector rather than a scalar, so it records *how* level 1 failed and not only that it failed, which Sec. 3.5 builds on.

Without the stop gradients in Eqs. 3 and 4, the residual loss reaches level 1 through the lifted prediction. The joint objective could then be reduced by making level 1's errors easy for level 2 to predict rather than by making them small, and the residual would become an artifact of joint optimization rather than a measurement of where level 1 fails. With them, level 1's predictions receive no gradient from the residual loss, so level 1 cannot reduce Eq. 5 by reshaping its own errors. The coarse encoder ϕ_2 recieves gradients through z_t^2 and passes them to ϕ_1 through z_t^1 so the level-1 encoder is also shaped by the level-2 prediction problem, while f1 and the rollout it produces are not. App. N verifies that adding level 2 leaves the level-1 representation intact.

3.4 TRAINING

Regularization ϕ_2 can reduce Eq. 5 to zero by mapping every latent to the same point, which makes every residual vanish. We therefore regularize the outputs of every coarse encoder with SIGReg (Maes et al., 2026), which pushes the distribution of coarse embeddings towards an isotropic reference and keeps the space from losing dimensions or concentrating on a point. SIGReg acts only on that distribution, not on which state produced which embedding, so it prevents collapse but cannot relate the residual to the decision structure of the environment. That relationship comes from Eq. 5, and App. K confirms that SIGReg alone does not produce it. We also constrain the latents at every level to unit norm, targets included, which keeps residual magnitudes comparable across states.

Objective All components are trained jointly on the same batches of k_L step segments. For two levels the objective is

$$\mathcal{L} = \mathcal{L}_{\text{step}} + \lambda_{\text{td}}\,\mathcal{L}_{\text{td}} + \lambda_{\text{res}}\,\mathcal{L}_{\text{res}} + \lambda_{\text{sig}}\,\mathcal{L}_{\text{sig}}, \tag{6}$$

with ϕ_1, f_1, and the successor feature head updated by $\mathcal{L}_{\text{step}}$ and $\mathcal{L}_{\text{td}}$, each higher level's encoder and predictor updated by its own residual and SIGReg terms, every EMA copy tracking its encoder, and the unit norm constraint applied to every encoder and EMA copy at each forward pass. We train jointly rather than fitting level 2 to a frozen level 1, so that the residual measures the error further training does not remove rather than the idiosyncratic failures of one checkpoint, and the stop gradients of Sec. 3.3 keep the levels from co-adapting.

Since the residual is meaningless while level 1 is untrained, λ_{res} is held at zero for N_{warm} updates and ramped linearly over the next N_{ramp}, with the same values in every environment (App. G).

3.5 Reading the Residual

The residual tells the planner where level 1 fails, but failing is not the same as deciding. Fix a state s and consider the residuals $\{\Delta_t^2(a^2)\}$ obtained by resetting the simulator at s and executing each candidate macro-action. The first reading is the *magnitude*, the mean norm of that set, which is what a scalar error signal reports. The second is the *clusterability*, the fraction of the set's variance explained by the macro-action, debiased by subtracting the same fraction under permuted labels,

$$c(s) \;=\; \mathrm{EVF}\big(\Delta^2 \mid a^2\big) \;-\; \mathbb{E}_\pi\big[\mathrm{EVF}\big(\Delta^2 \mid \pi(a^2)\big)\big]. \tag{7}$$

Two situations produce a large residual, and magnitude cannot tell them apart. At a decision point several futures are possible and the action selects among them. Under dynamics the agent does not control, several futures are also possible but nothing the agent does changes which occurs. Both spread the target of Eq. 1, so a threshold on magnitude escalates in both and spends coarse search where nothing is being decided. The law of total variance separates them. Writing Y for the reached level-2 state,

$$\mathrm{Var}(Y \mid s) \;=\; \underbrace{\mathbb{E}_a\big[\,\mathrm{Var}(Y \mid s,a)\big]}_{\text{not resolved by choice}} \;+\; \underbrace{\mathrm{Var}_a\,\big[\mathbb{E}(Y \mid s,a)\big]}_{\text{resolved by choice}}. \tag{8}$$

Eq. 7 estimates the share of the second term. At a decision point the conditional mean of Y varies with the action, so residuals fall into tight groups indexed by the branch the action selects. Under uncontrolled dynamics it does not, and they stay diffuse. App. E gives the derivation.

Eq. 8 concerns the reached state, whereas Δ_t^2 also contains the error level 1 accumulates along its rollout, so our claim is about the residual as measured. Sec. 4.3 and App. D test it in that form. At planning time the future is unobserved, so we sweep a_t^2 at a fixed z_t^2 and read clusterability off the predicted residuals of Eq. 4. Sec. 4.3 reports both, since f_2 could be smooth across actions where the true residuals separate.

3.6 Planning

At test time the agent receives a goal latent and plans using MPC, replanning after every executed step, in one of two modes. Level 1 plans alone by sampling primitive action sequences, rolling them forward with f_1, and scoring the endpoints against the goal. This is cheap and correct wherever its rollouts are accurate. Where they are not, the planner samples macro-actions, rolls level 1 forward under each, applies $\hat{\Delta}^2$ as in Eq. 4, and scores the corrected coarse states. The selected macro-action fixes an implicit waypoint that level 1 then plans towards until the next escalation. The residual decides between the modes. Sweeping a^2 at the current state gives both readings of Sec. 3.5 without observing the future. We escalate only when the predicted residuals are large and clusterable, which is when level 1 is about to be wrong and the agent's choice settles the outcome. App. H gives the optimizers, the rule and its thresholds, and App. I reports sensitivity to both.

4 Experiments

4.1 Setup

Environments We test our method across multiple environments. GraphMaze is a Minigrid-style navigation task over procedurally generated graphs, whose layouts can be resampled and whose junctions can be built to reconverge after a pre-specified number of steps. DiverseMaze-Slip is a

stochastic variant of DiverseMaze (Wang et al., 2026) in which a fraction of corridor cells apply a random action with probability p, so junctions and uncontrolled branching coexist. Crafter (Hafner, 2021) has decision points at several timescales and tests depth. Push-T (Chi et al., 2025) branches continuously through contact. App. C describes these in more detail.

Baselines For planning we compare against HWM (Zhang et al., 2026), which invokes its upper level at every stride boundary, THICK (Gumbsch et al., 2024), which invokes it when a learned context gate fires, and a flat model sharing our level 1. For detection we compare against THICK's gate and a scalar predictor of level 1's error size matched to f_2 in capacity and data. App. M adds goal conditioned RL baselines and three further error signals from level 1, and App. J gives our adaptation of THICK.

Protocol All models are trained on offline, reward free transitions with no event labels, decision point annotations, subgoals, or rewards. Ground truth decision points are used only for evaluation, and measuring the commitment ratio and the true residual requires resetting the simulator at labeled states, which is never used for training or planning. Planning compute is counted as model rollout steps per environment step across every level. We train two and three level models on GraphMaze and Crafter and state the depth with every result. Every number is a mean over five seeds with standard errors, and every baseline is retrained under identical data, seeds, and compute.

4.2 Level 1 Averages, and the Residual Is Concentrated at Branch Points

The premise of this work is that a predictor trained to regress the next latent returns the average of the states that can follow wherever several are possible. We measure that failure rather than inferring it from the objective. At a labeled decision point s we reset the simulator and roll out K macro actions to obtain the true branch endpoints $z_1 \dots z_K$, and compare them against the model's prediction $\hat{z}$,

$$\rho(s) \;=\; \frac{\|\hat{z} - \frac{1}{K}\sum_k z_k\|}{\min_k \|\hat{z} - z_k\|}. \tag{9}$$

A model that averages sits at the centroid and has $\rho \to 0$, while one that commits sits near some z_k and has $\rho > 1$. The ratio is scale free, so it compares models that do not share a representation. For a hierarchical model, $\hat{z}$ is the prediction the planner consumes at that scale. For HRP this would be the corrected rollout, while for HWM it would be the level 1 rollout under its upper level's subgoal. Level 1 alone uses its uncorrected rollout, so the differences between models reflect what each upper level contributes. A prediction far from every branch also gives ρ slightly above one, so we calibrate against a null control that displaces the prediction from the centroid in a random direction. This gives 1.02 ± 0.03, and we report the distance to the nearest branch alongside ρ in App. F.

Fig. 2(a) shows that Level 1 sits at 0.29, well below the null control, on training and held out layouts alike, so the averaging is not a failure to generalize. Adding the residual level moves the prediction onto a specific branch, which reduces the distance to the nearest branch. This causes the ratio to rise because the prediction is close to an outcome the environment would actually reach. A third level leaves the ratio at this scale essentially unchanged, which is as expected when junctions are resolved within the second level's stride. HWM sits between the two, committing on training layouts by a smaller margin than the residual and not resolvably above the null control on held out ones.

The correction is concentrated The correction is also concentrated where level 1 fails, with the top decile of states carrying 58% of the total residual mass on GraphMaze and 21% on Push-T. Across GraphMaze sizes the concentration tracks the true fraction of junction states, so it follows from the environment rather than from a coefficient we set (App. I).

4.3 Size Does Not Identify Decision Points, but Action Dependence Does

A large residual has two possible causes. Either the agent's action settles which future occurs, or dynamics it does not control do. Only the first is worth coarse search, so a planner that escalates on size alone spends its budget where nothing is being decided. This section asks whether action dependence tells the two apart where size cannot.

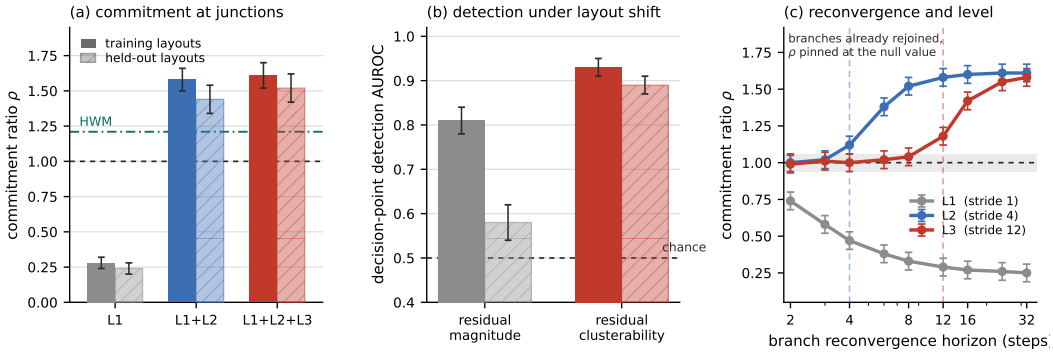

Figure 2: GraphMaze under layout shift and branch reconvergence. **(a)** Commitment ratio at junctions on training and held out layouts. The dashed line is the null control and the dash-dotted line is HWM. **(b)** Detection AUROC on training and held out layouts. **(c)** Commitment ratio against the horizon R at which branches reconverge, with each level's stride marked.

Table 1: Detection AUROC on DiverseMaze-Slip. The middle row is the comparison the paper is about. The lower row is a control where clusterability should fail.

State pair	Clusterability	Residual size	Scalar predictor	THICK gate
Junction vs. ordinary corridor	0.90 ±0.02	0.89 ±0.02	0.87 ±0.03	0.84 ±0.03
Junction vs. slip cell	**0.86** ±0.03	0.53 ±0.03	0.55 ±0.03	0.57 ±0.04
Slip cell vs. ordinary corridor	0.54 ±0.03	0.88 ±0.02	0.86 ±0.03	0.81 ±0.04

Held out layouts Size and controllability are confounded whenever branching is the only source of prediction error, which is why Sec. 4.2 cannot settle the question. An unfamiliar layout raises prediction error almost everywhere without adding decision points, so size should degrade. Clusterability should not, because novelty error does not depend on the macro action and cancels in Eq. 7. Both readings clear the criterion on training layouts, with size at 0.81 and clusterability at 0.93 (Fig. 2b). Under layout shift size falls to 0.57, close to chance, while clusterability holds at 0.89. App. D reports the same argument in an environment where size is matched by construction.

The same separation under stochastic dynamics DiverseMaze-Slip is a harder test, since junctions and slip cells are present at once and neither is novel. Table 1 scores each signal on three pairs of state types. Every signal separates junctions from ordinary corridors. On junctions against slip cells, size falls to 0.53 while clusterability reaches 0.86. On slip cells against corridors, where the agent's choice settles the outcome at neither, clusterability fails as it should at 0.54.

Neither a scalar nor a learned gate is sufficient A scalar predictor trained to regress error size correlates with it at 0.94, yet falls to 0.55 on junction against slip. THICK's gate, trained with a bias towards firing at transitions, reaches 0.57. An L_0 penalty on gate activity favors any state at which something changes, so both kinds of branching qualify. This is consistent with the 88% to 97% recall that Gumbsch et al. (2024) report on MiniHack, where there is no uncontrolled branching.

Predicted and measured residuals agree At planning time only $\hat{\Delta}^2$ is available, and its clusterability reaches 0.83 on junction against slip, close to the 0.86 of the true residual. Detection is therefore usable at planning time and not only offline.

4.4 The Residual Is Responsible, Not the Coarser Timescale

We train five models on DiverseMaze-Slip under identical data, seeds, and compute. Beside HRP, we have the *full prediction variant* that replaces the residual target with the coarse future at the same stride, which is HWM's design in our architecture, and detects with its own coarse error. The *stride one variant* keeps the residual and removes temporal abstraction, so its macro-action is a single primitive and its clusterability sweeps primitives rather than macro-actions. The fourth

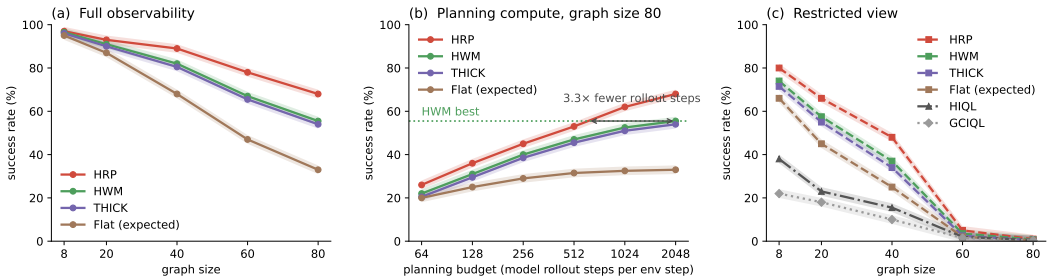

Figure 3: GraphMaze, where a single size parameter varies difficulty continuously. **(a)** Success against graph size under full observability. HRP, HWM, and THICK are indistinguishable at size 8 and separate from size 40, with the margin widening monotonically. **(b)** Success against planning compute at size 80, measured as model rollout steps per environment step across every level. **(c)** Under a restricted view all methods degrade sharply beyond size 40, which Sec. 5 attributes to prediction error from unobserved state rather than from branching. Five seeds, shaded bands are standard errors. All results are on held-out layouts.

model *removes the successor feature head*, and the fifth is *THICK retrained under our protocol*. Planning is compared at matched rollout steps per environment step. The comparison is stochastic by design, since on a deterministic environment the full prediction variant's own error would locate the branches and the ablation would credit the residual with nothing.

Table 2 shows a double dissociation, in which the full prediction variant loses detection (0.55) but keeps most of its planning. The stride one variant keeps detection (0.84) but loses planning, so the residual produces the decision point signal and the stride produces tractability. THICK, detects no better than the full prediction variant (0.57) so what matters is what the upper level predicts rather than when it is invoked. Removing the successor feature head costs 0.14 AUROC, as Sec. 3.2 predicts, with little effect on planning, and SIGReg is necessary but not sufficient (App. K).

Table 2: Ablation on DiverseMaze-Slip at matched compute. Detection is junction against slip cell.

Variant	Detection	Success (%)
HRP	0.86±.03	78.4±1.5
Full prediction	0.55±.04	71.2±1.8
Residual, stride 1	0.84±.03	63.8±2.0
No SF head	0.72±.04	77.1±1.6
THICK	0.57±.04	69.5±2.1

4.5 Each Level Commits at Its Own Timescale

Whether a junction is a decision depends on the timescale it is viewed from. A junction whose branches rejoin after R steps is a decision for a predictor whose stride is shorter than R, and not for one whose stride is longer. GraphMaze lets us set R directly, and the metric supplies its own control once we do. When the branches have rejoined inside a level's stride every endpoint is the same state, so the centroid coincides with the nearest endpoint and ρ equals one for any prediction at all.

Fig. 2(c) varies R. Each level sits at that null value while R is inside its stride and departs from it near its own stride rather than at a fixed R. The corrector levels rise above one, which is commitment to a single branch, while level 1 falls below one wherever the branches stay apart, which is the averaging the method removes.

4.6 Placing Coarse Search at Decision Points Improves Long Horizon Planning

Fig. 3(a) reports success against graph size. These are held out layouts, where error size no longer locates junctions (Fig. 2(b)), and HWM's upper level runs on every stride boundary regardless. HRP, HWM, and THICK are indistinguishable at size 8, separate from size 40, and the margin widens to 68.0% against 55.4% for HWM and 54.1% for THICK at size 80. Small graphs contain few decisive states, so the advantage grows with the number of decisions an episode contains. At size 80 HRP reaches HWM's best success at 3.3× fewer rollout steps (Fig. 3b).

Table 3: Escalation rule on DiverseMaze-Slip at matched compute, with the HRP model fixed. Wasted escalations are those at states not labelled junctions.

Gate	Success (%)	Esc./ep.	Wasted (%)
Size and clust.	**78.4**$\pm$1.5	6.1	11
Clust. only	74.9$\pm$1.8	11.7	38
Size only	72.6$\pm$1.9	14.3	49
Every boundary	70.8$\pm$1.9	22.0	67

Table 4: Commitment ratio ρ on Crafter at strides k_1, k_2, k_3. The null control is 1.02. L1+L2 and L1+L2+L3 are HRP. Mean over N decision points and five seeds.

Model	k_1	k_2	k_3
L1 only	0.31$\pm$.04	0.22$\pm$.03	0.18$\pm$.03
HWM	1.24$\pm$.11	0.94$\pm$.09	0.21$\pm$.04
L1+L2	1.62$\pm$.09	1.41$\pm$.08	0.26$\pm$.04
L1+L2+L3	1.60$\pm$.09	1.55$\pm$.07	**1.38**$\pm$.10

DiverseMaze On the benchmark Zhang et al. (2026) use (Wang et al., 2026), HRP and HWM are within standard errors at the two shorter horizons. At the longest, HRP reaches 87.2% against 83.1% for HWM and 81.4% for THICK while the flat model falls to 43.8%, and HRP reaches HWM's best success at $2.4\times$ fewer rollout steps (App. C.2).

Where escalation happens, and on what Table 3 holds the model fixed and varies only the escalation rule. Both single gates escalate about twice as often as the conjunction and reach lower success, and they waste search in different places, as Table 1 predicts. Size alone escalates at slip cells, where nothing is being decided, and clusterability alone at states whose outcome depends on the action but which level 1 already predicts, where search is correct but unnecessary. Without any gate, HRP reaches 70.8%, indistinguishable from the full prediction (71.2%), so the residual's planning value arrives entirely through where the search is placed. App. I reports sensitivity to both thresholds jointly. Seeding the macro-action proposals from the grouping of residuals adds a further 5.5 points on GraphMaze at size 80 and 2.9 at DiverseMaze's longest horizon at matched proposal count (App. H.1), the one place where the vector structure of the residual does planning work.

4.7 Extending to Three Levels

Sec. 4.2 and Sec. 4.5 report a three level model on GraphMaze, where each level begins to commit at its own stride. Crafter tests whether this extends to a task whose decision structure spans several timescales at once. Table 4 is a lower triangle, each variant committing at every scale up to its own depth and averaging beyond it. HWM commits at the fine scale and not at its own mid scale, since its upper level regresses a coarse future, which causes it to return an average at the stride it operates on. This still narrows the fine branches below it. App. O shows that each added level extends the range of reachable goals from its own stride, that three levels reach the two level model's success with $4.3\times$ fewer model calls, and that trained without task reward HRP reaches a Crafter score of 9.2% against 2.1% for the best prior agent that uses none.

5 Conclusion

We introduced HRP, a latent world model whose upper level predicts the lower level's error instead of the future. Since the lower level is already correct on most states, this correction is sparse without any penalty. It is also a vector, so it records *how* the lower level failed and not only that it failed. Its size locates where the lower level fails, and its dependence on the action tells whether the agent can resolve that failure. A planner that escalates only where both hold improves long horizon success at reduced compute. The construction recurses, and a third level commits at timescales the second cannot reach.

These gains rest on an assumption about the environment, that prediction fails at a small set of states and that the agent's action resolves them. When this does not hold, the residual has nothing sparse to report. On Push-T, contact makes the dynamics branch continuously, mean clusterability is 0.04 against 0.29 at GraphMaze junctions, and HWM stays ahead at every horizon. Under partial observability, most of the prediction error comes from state the agent cannot see rather than from branching, and every method collapses on the restricted view variant. Conditioning the residual on a belief state rather than a single observation is a natural way to extend it to such settings.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods. We have not used generative AI tools for developing theoretical models or conceptual frameworks, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning and reformatting datasets, or interpreting results, and generating synthetic datasets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation, and supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for editing the paper to improve readability. We have reviewed all AI-assisted work. All AI-assisted code was reviewed and validated by the authors. All AI-edited text was checked against the underlying experimental results by the authors. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies world models for planning tasks and does not involve human subjects or user studies. All benchmarks used for evaluation are established public research datasets, used for their intended research purpose. At this stage, the work remains an early research contribution, and any practical societal effects will depend on substantial future progress in model design, scaling, validation, and system integration.

REFERENCES

P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

M. Bagatella, M. Pirotta, A. Touati, A. Lazaric, and A. Tirinzoni. Td-jepa: Latent-predictive representations for zero-shot reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pages 36211–36244, 2026.

C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

C. Gumbsch, M. V. Butz, and G. Martius. Autonomous identification and goal-directed invocation of event-predictive behavioral primitives. *IEEE Transactions on Cognitive and Developmental Systems*, 13(2):298–311, 2019.

C. Gumbsch, N. Sajid, G. Martius, and M. V. Butz. Learning hierarchical world models with adaptive temporal abstractions from discrete latent dynamics. In *International Conference on Learning Representations*, volume 2024, pages 4756–4792, 2024.

Z. Guo, S. Thakoor, M. Pîslar, B. Avila Pires, F. Altché, C. Tallec, A. Saade, D. Calandriello, J.-B. Grill, Y. Tang, et al. Byol-explore: Exploration by bootstrapped prediction. *Advances in neural information processing systems*, 35:31855–31870, 2022.

D. Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

D. Hafner, K.-H. Lee, I. Fischer, and P. Abbeel. Deep hierarchical planning from pixels. *Advances in Neural Information Processing Systems*, 35:26091–26104, 2022.

D. Jayaraman, F. Ebert, A. A. Efros, and S. Levine. Time-agnostic prediction: Predicting predictable video frames. *arXiv preprint arXiv:1808.07784*, 2018.

T. Kim, S. Ahn, and Y. Bengio. Variational temporal abstraction. *Advances in neural information processing systems*, 32, 2019.

I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber. A clockwork rnn. In *International conference on machine learning*, pages 1863–1871. PMLR, 2014.

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

L. Maes, Q. L. Lidec, D. Scieur, Y. LeCun, and R. Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

A. Neitz, G. Parascandolo, S. Bauer, and B. Schölkopf. Adaptive skip intervals: Temporal abstraction for recurrent dynamical models. *Advances in Neural Information Processing Systems*, 31, 2018.

S. Park, D. Ghosh, B. Eysenbach, and S. Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023.

S. Park, K. Frans, B. Eysenbach, and S. Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pages 94937–94982, 2025.

R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

V. Saxena, J. Ba, and D. Hafner. Clockwork variational autoencoders. *Advances in neural information processing systems*, 34:29246–29257, 2021.

M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

U. Sobal, W. Zhang, K. Cho, R. Balestriero, T. G. Rudner, and Y. LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *Advances in Neural Information Processing Systems*, 38:43905–43941, 2026.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pages 3540–3549. PMLR, 2017.

Y. Wang, O. Bounou, Y. LeCun, and M. Ren. Adajepa: An adaptive latent world model. *arXiv preprint arXiv:2606.32026*, 2026.

Y. Xu, L. Liu, W. Zhou, M. Feng, and H. Li. Self-supervised hierarchical visual reasoning with world model. *arXiv preprint arXiv:2605.17537*, 2026.

A. Zakharov, Q. Guo, and Z. Fountas. Variational predictive routing with nested subjective timescales. *arXiv preprint arXiv:2110.11236*, 2021.

W. Zhang, B. Terver, A. Zholus, S. Chitnis, H. Sutaria, M. Assran, R. Balestriero, A. Bar, A. Bardes, Y. LeCun, et al. Hierarchical planning with latent world models. *arXiv preprint arXiv:2604.03208*, 2026.

G. Zhou, H. Pan, Y. LeCun, and L. Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A Related Work

Latent World Models and Planning Latent predictive methods train representations by predicting encoded futures rather than observations, so planning can run entirely in latent space (LeCun et al., 2022; Schwarzer et al., 2020; Guo et al., 2022). DINO-WM and PLDM optimize action sequences against latent rollouts, reaching image goals without rewards or task specific policies (Zhou et al., 2024; Sobal et al., 2026), and TD-JEPA extends the target from one step to long term dynamics with a temporal difference objective (Bagatella et al., 2026). Long horizons are where these models are hardest to use, and temporal abstraction is the usual response. Director plans over learned goals at a coarser scale (Hafner et al., 2022), and HWM plans over macro-actions with slow predictions serving as subgoals for a fast model (Zhang et al., 2026). Our lower level follows this family, and we hold the stride of Zhang et al. (2026) fixed so that the change we make is isolated to the prediction target itself, which turns out to be the part that carries the signal.

Temporal Abstraction and Event Segmentation Given a hierarchy, the remaining question is which states the coarse level should predict across. The simplest answer is a fixed rate (Koutnik et al., 2014; Saxena et al., 2021), which hierarchical RL also adopts when a manager sets goals at fixed intervals (Vezhnevets et al., 2017; Nachum et al., 2018), while option-critic instead learns when to terminate (Bacon et al., 2017). Adaptive answers in sequence models place boundaries at states that are unusually predictable or unpredictable (Neitz et al., 2018; Jayaraman et al., 2018; Zakharov et al., 2021), at spikes in a forward model's prediction error (Gumbsch et al., 2019), or where a learned detector fires, with VTA bounding the number of segments directly (Kim et al., 2019) and THICK driving its detector to fire rarely through a sparsity penalty (Gumbsch et al., 2024). All of these read a scalar, either the size of an error or the activity of a gate, and Sec. 4.3 shows that a scalar cannot separate a state the agent decides from one the environment decides. Residual formulations of hierarchy do exist in reconstruction based models (Xu et al., 2026), but there the residual improves prediction quality rather than saying where prediction fails.

B Hyperparameters

Table 5: Hyperparameters. Warmup and ramp are in App. G and planner settings in App. H. Values are shared across environments unless a column differs.

	GraphMaze	DiverseMaze(-Slip)	Crafter	Push-T	antmaze
Stride k_2	4	5	8	5	10
Stride k_3	12	–	64	–	–
Latent dim. $d_1/d_2/d_3$	256 / 128 / 64				
Encoder i_1	4-layer CNN, 64×64 input				
Coarse encoders i_i, $i > 1$	3-layer MLP, width 512				
Predictors f_1, f_i	4-layer MLP, width 1024, action concatenated				
Macro-action encoder ρ_i	as in Zhang et al. (2026)				
EMA rate τ	0.996				
λ_{td} / λ_{res} / λ_{sig}	1.0 / 1.0 / 0.1				
Optimizer, learning rate	AdamW, 3×10^{-4}				
Batch size (segments)	256				
Training updates	200k	300k	500k	300k	1M
Seeds	5				

C Environments and Datasets

C.1 GraphMaze

GraphMaze embeds a procedurally generated graph in a Minigrid-style grid. Nodes are rooms, edges are corridors of 3 cells, and the size of a layout is its number of nodes, from 8 to 80 in Fig. 3. A junction is a node with three or more incident corridors. For Sec. 4.5 we generate layouts in which two corridors leaving a junction meet again after $R \in \{2, 4, 8, 12, 16, 32\}$ steps, and elsewhere branches do not reconverge.

The agent observes a 64×64 top-down image of the full layout and acts with four discrete moves to the neighboring cells. The restricted view variant of Fig. 3(c) replaces the observation with an egocentric 5×5 window. We train on 1,000 layouts per graph size and evaluate on 100 held out layouts per size drawn from the same generator and never seen in training. Every planning result on GraphMaze uses held out layouts (Fig. 3). A goal is the observation at a target node at least 10 steps away, and an episode succeeds if the agent reaches it within three times the shortest path length. Junction labels come from the graph and are used only for evaluation.

C.2 DIVERSEMAZE AND DIVERSEMAZE-SLIP

DiverseMaze follows the setup of Neitz et al. (2018), with a point agent, 64×64 top-down images, and two dimensional continuous actions and the three evaluation horizons used there (App. L). DiverseMaze-Slip divides each maze into the cells of its layout grid and marks a fraction $q = 0.2$ of corridor cells as slip cells, at which the executed action is replaced by a uniformly random action with probability $p = 0.5$. Slip cells are present in the training data, so neither they nor junctions are novel at evaluation. Every cell is labelled for evaluation as a junction (three or more open neighbors), a slip cell, or an ordinary corridor cell.

C.3 CRAFTER

We use Crafter (Hafner, 2021) with its standard 64×64 observations, 17 actions, and 22 achievements. The dataset is 5M reward free transitions collected by Plan2Explore run without task reward. For Table 4 we label decision points at three scales from the achievement dependency graph. A state is a decision point at scale k_i when two available primitive choices lead to outcomes that stay distinct for more than k_i steps, for example facing different materials at k_1, heading to different resources at k_2, and starting different achievement chains, such as a table versus saplings, at k_3. Table 4 averages over $N = 600$ decision points. Goals are observations taken from held out trajectories at the step where an achievement is first unlocked, and the Crafter score protocol is in App. O.3.

C.4 PUSH-T

We use the Push-T simulator and demonstrations of Chi et al. (2025). Contact makes the outcome depend continuously on the action, so the environment has no labelled decision points. Concentration and clusterability on Push-T are computed over 5,000 states sampled uniformly from held out episodes (App. P.1).

C.5 OGBENCH ANTMAZE

Used only for App. N, following the protocol of Bagatella et al. (2026) on the medium, large, and giant navigate settings of OGBench (Park et al., 2025).

C.6 DATASETS

Table 6: Datasets. No dataset contains rewards, event labels, decision point annotations, or subgoals. State labels are used only for evaluation.

Environment	Behavior policy	Transitions	Episodes
GraphMaze	random walks over the graph	2M	20k
DiverseMaze(-Slip)	Neitz et al. (2018)	1M	5k
Crafter	reward free Plan2Explore	5M	16k
Push-T	Chi et al. (2025)	25.6k	206
antmaze	Park et al. (2025)	1M per setting	1k per setting

D THE FORK CHAMBER

The fork chamber is a controlled environment we introduce to separate two properties that are confounded in natural environments. It contains fork states, at which the agent selects among several

corridors and the selection determines the outcome, and scatter states, at which the same set of outcomes occurs with the action having no influence. The two are constructed so that level 1 incurs the same distribution of prediction error at both. We verify this before evaluating any model, and report the verification in Sec. D.1.

Each chamber has 40 fork and 40 scatter states, each leading into $C = 3$ corridors. At a scatter state the corridor is drawn uniformly at random whatever the action. Level 1 errs at both kinds of state, at scatter states because the outcome is random and at fork states because it underuses the action where its effect is large (App. F.1). We set the corridor separation at scatter states so that the two error distributions match, choosing it on a separate set of states before any evaluation.

D.1 Matched error verification

The mean k_2 step level-1 error is 0.412 ± 0.011 at fork states and 0.418 ± 0.012 at scatter states, a ratio of 0.99, and a two sample Kolmogorov–Smirnov test does not reject equality of the two distributions ($p = 0.47$).

D.2 Result

Table 7: Detection AUROC for fork against scatter states. Level-1 error is matched between the two by construction (Sec. D.1).

Signal	AUROC
Residual size	0.51 ± 0.03
Scalar predictor	0.53 ± 0.03
THICK gate	0.55 ± 0.04
Clusterability, true residual	0.93 ± 0.02
Clusterability, predicted residual	0.90 ± 0.03

With level-1 error matched, every scalar signal is at chance, while clusterability separates forks from scatter states as reliably as it separates junctions from corridors on training layouts (Fig. 2(b)).

E Clusterability: Derivation and Estimator

Throughout, the variance of a vector is the trace of its covariance, $\mathrm{Var}(X) = \mathbb{E}\|X - \mathbb{E}X\|^2$.

E.1 Law of total variance for the reached state

Let Y be the level-2 target reached from s under macro-action a, $Y = \bar{\phi}_2(\bar{\phi}_1(o_{t+k_2}))$. Conditioning on a and expanding $\|Y - \mathbb{E}(Y \mid s)\|^2$ around $\mathbb{E}(Y \mid s, a)$, the cross term vanishes in expectation and

$$\mathrm{Var}(Y \mid s) = \mathbb{E}_a\big[\mathrm{Var}(Y \mid s, a)\big] + \mathrm{Var}_a\big[\mathbb{E}(Y \mid s, a)\big], \tag{10}$$

which is Eq. 8. The first term is the spread that remains once the action is fixed, and the second is the spread the choice of action accounts for.

E.2 What the measured residual contains

The residual is $\Delta = Y - g(a)$ with $g(a) = \mathrm{sg}[i_2(\hat{z}^1_{t+k_2})]$, the lifted level-1 rollout, which is fixed once s and the primitive chunk behind a are. Subtracting a quantity fixed by the action leaves the spread under that action unchanged, and shifts its mean, so

$$\mathrm{Var}(\Delta \mid s) = \underbrace{\mathbb{E}_a\big[\mathrm{Var}(Y \mid s, a)\big]}_{\text{not resolved by choice}} + \underbrace{\mathrm{Var}_a\big[\mathbb{E}(Y \mid s, a) - g(a)\big]}_{\text{resolved by choice, missed by level 1}}. \tag{11}$$

The first term is identical to that of Eq. 8. The second is the part of the action's effect that level 1 fails to predict, and it is this share that Eq. 7 estimates, which is why Sec. 3.5 states the claim for the residual as measured. Four cases follow.

At a decision point, $\mathbb{E}(Y \mid s, a)$ moves with the action and level 1 underuses the action there (App. F.1), so $g(a)$ does not follow it, the second term is large, and residuals separate by branch. Under uncontrolled dynamics the action changes little of the outcome, and what it does change level 1 fits, so the second term is small while the first is large. Error shared by all macro-actions at s, such as the error from an unfamiliar layout, enters g as a common offset $b(s)$, shifts every $\mathbb{E}(Y \mid s, a) - g(a)$ by the same amount, and leaves the second term unchanged, which is the argument of Sec. 4.3 for held out layouts. Where level 1 already resolves the action, both terms are small but their ratio need not be, so clusterability alone can fire where magnitude does not. This is the waste of the clusterability-only gate in Table 3 and the reason the planner requires both readings.

E.3 Estimator

At a state s we sample $M = 32$ primitive chunks from the planner's initial proposal distribution, encode each with ρ_2, and group the M macro-actions into $G = 4$ clusters by k-means on a^2. For the true residual we reset the simulator at s and execute each chunk n times, with $n = 1$ on deterministic environments and $n = 8$ on DiverseMaze-Slip, giving residuals $\{\Delta_j\}$ with group labels ℓ_j. The explained variance fraction is the between group sum of squares over the total,

$$\mathrm{EVF}(\Delta \mid a^2) = \frac{\sum_g n_g \|\bar{\Delta}_g - \bar{\Delta}\|^2}{\sum_j \|\Delta_j - \bar{\Delta}\|^2}, \tag{12}$$

and the permutation term of Eq. 7 recomputes it after shuffling the residuals across candidates with the groups held fixed, averaged over $P = 20$ permutations. For the predicted residual the same estimator is applied to $\hat{\Delta}^2 = f_2(z_t^2, a^2)$, one per candidate, which needs no simulator and no level-1 rollout.

E.4 Estimator behaviour

Because each group contains several distinct macro-actions, the within group variance is not zero even with $n = 1$, so the estimator does not degenerate on deterministic environments or on predicted residuals. Its permutation mean is zero, so $c(s) \approx 0$ wherever residuals do not depend on the action, and the debiased value can be slightly negative. It is scale free, so it measures how the residual depends on the action and not how large the residual is (Sec. E.2).

Absolute values are moderate. The mean of c is 0.29 at GraphMaze junctions, 0.06 at corridor states, and 0.04 on Push-T. Values stay well below one because level 1 resolves part of the action's effect even at junctions (App. F.1), because a group can contain macro-actions that lead to different branches, and because the residual carries error shared across actions. Detection depends on the ordering of states and not on the absolute level, which is why an AUROC of 0.93 coexists with a mean of 0.29.

Table 8: Sensitivity of junction against slip detection on DiverseMaze-Slip to the number of action groups G and candidates M. Standard errors are 0.02 to 0.03.

	$G = 2$	$G = 4$ (default)	$G = 8$
$M = 16$	0.82	0.83	0.80
$M = 32$ (default)	0.84	0.86	0.85

F The Commitment Ratio

At a labelled decision point we reset the simulator and execute $K = 3$ macro-actions whose endpoints are pairwise distinct at the scale being measured. For models that condition on the macro-action, each prediction $\hat{z}_k$ is formed under the macro-action that produced z_k, ρ of Eq. 9 is computed for each, and the result is averaged over the K branches. When the branches have rejoined within the measured scale all endpoints coincide, the centroid equals the nearest endpoint, and $\rho = 1$ identically, which is the reference value of Sec. 4.5.

Null control The null control replaces $\hat{z}$ by the centroid displaced by $r = 1.5$ times the mean distance of the branch endpoints from the centroid, in a direction drawn uniformly from the subspace spanned by the ten leading principal components of the level's latents at junctions, the directions in which those latents actually vary. A prediction displaced this far lies slightly closer to its nearest branch than to the centroid, which gives ρ slightly above one (Sec. 4.2). Averaged over junctions and draws, the null control gives 1.02 ± 0.03, with a nearest branch distance of 1.41 (Table 9).

Table 9: Commitment ratio ρ from Eq. 9 at GraphMaze junctions, on layouts seen in training and on held out layouts. Values above the null control indicate the model commits to a single branch. Mean over $N = 480$ junctions and five seeds.

Model	Train layouts	Held out layouts	Nearest branch distance
Null control	1.02±0.03	1.02±0.03	1.41
Level 1 only	0.29±0.04	0.25±0.05	0.96
HWM (Zhang et al., 2026)	1.21±0.10	1.12±0.11	0.71
HRP (L1+L2)	1.58±0.09	1.44±0.10	0.42

F.1 Why level 1 averages at junctions

Level 1 is conditioned on the action, and at a GraphMaze junction the action fixes which branch is taken, so the averaging reported in Sec. 4.2 does not follow from the squared error objective alone. Two mechanisms could produce it. Level 1 could underuse the action at junctions, so that its one step prediction already lies between the successors, or it could select the correct successor at the first step and drift between branches as the rollout continues. The two predict different results for the diagnostics below, and the distinction matters beyond this appendix, since the stride one variant of Sec. 4.4 keeps detection at 0.84 on DiverseMaze-Slip with a single step rollout, which only the first mechanism allows.

Action sensitivity For a state s with observation o and latent $z = i_1(o)$, let $\mathcal{A}_s$ be the primitive actions available at s and o'_a the observation reached by resetting the simulator at s and executing a. We compare how far apart level 1 places the successors with how far apart they are,

$$S(s) = \frac{\mathrm{mean}_{a \neq a'} \, \| f_1(z, a) - f_1(z, a') \|}{\mathrm{mean}_{a \neq a'} \, \| \bar{\phi}_1(o'_a) - \bar{\phi}_1(o'_{a'}) \|}, \tag{13}$$

with both means over the same pairs in $\mathcal{A}_s$. $S = 1$ means level 1 separates the successors as much as the environment does, and $S \to 0$ means it ignores the action. We compute S at all 480 labelled junctions and at 2,000 corridor states sampled uniformly from the same trajectories, on training and held out layouts.

Commitment along the rollout At each junction we take the K macro-actions of Sec. 4.2, roll level 1 forward h steps under the first h primitives of each, and compute ρ (Eq. 9) against the true endpoints after h steps, forming each prediction under the same macro-action as the endpoint it is compared with and averaging over the K branches. At $h = k_2$ this is the level-1 entry of Table 9.

Result Fig. 4(a) shows that level 1 uses the action where its effect is small and recurring and underuses it where the effect is large. In corridors S is 0.94 on training and 0.91 on held out layouts, while at junctions it falls to 0.34 and 0.31. Fig. 4(b) shows that the averaging is complete at the first step, with ρ at 0.27 for $h = 1$ on training and 0.24 on held out layouts, far below the null control and within standard error of the 0.29 and 0.25 of Table 9 at $h = k_2$. The rollout neither creates the averaging nor deepens it, which agrees with the level-1 curve of Fig. 2(c) at its one step stride. Fig. 4(c) links the two measurements per junction, with Spearman correlation 0.76 between S and ρ at k_2, so the junctions at which level 1 separates the successors least are the ones at which its k_2 step prediction lies closest to the centroid.

In a corridor the action moves the latent by a small displacement that recurs throughout the data, and level 1 fits it. At a junction the action selects between successors that are far apart in latent space, and junction states are a small fraction of the data (App. I.2). Level 1 fits the common case and places the rare, large action effect closer to the average of the successors than to any of them. The

failure is the same on held out layouts, so it is not a generalization failure, and it appears at the first step, so it does not depend on rollout error. This is also why the level-2 residual is concentrated at junctions and why the residual of the stride one variant still separates them (Sec. 4.4).

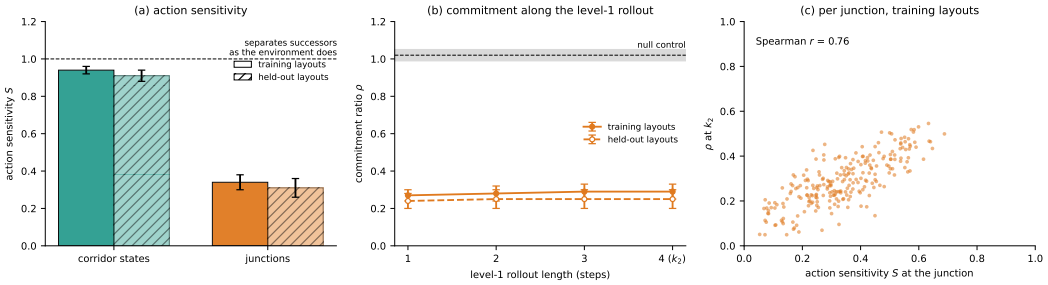

Figure 4: Why level 1 averages at GraphMaze junctions. (a) Action sensitivity S, the spread level 1 assigns to successors under different actions relative to their true spread, at corridor states and junctions. The dashed line is $S = 1$. (b) Level-1 commitment ratio ρ at junctions against rollout length. The values at k_2 are those of Table 9. The dashed line and band are the null control. (c) S against ρ at k_2 for individual junctions on training layouts, pooled over seeds. Five seeds, bars are standard errors.

G WARMUP AND RAMP

λ_{res} is zero for the first $N_{\text{warm}} = 30$k updates and rises linearly to its final value over the next $N_{\text{ramp}} = 20$k, with the same values in every environment. SIGReg is active from the start, so the coarse encoder cannot collapse while it receives no residual gradient. We chose N_{warm} as the point at which the level-1 one step loss is within 10% of its final value in every environment, so that the residual is measured against a level 1 that has largely converged. In three level models, level 3 follows the same schedule starting when level 2's ramp ends. Halving or doubling N_{warm} changes junction against slip detection on DiverseMaze-Slip to 0.84 and 0.86, against 0.86 at the default, with standard errors of 0.03.

H PLANNING PROTOCOL

Compute accounting Every method receives the same budget of B model rollout steps per environment step, where one application of any predictor (f_1, f_2, f_3, or a baseline's dynamics model) counts as one step. The gate's sweep of M macro-actions through f_2 is counted within B, as are the calls to the macro-action encoder ρ_i. Fig. 3(b) and App. L vary B, and every other result uses $B = 2048$, the largest budget in those sweeps. "Matched compute" in Tables 2 and 3 means equal B, so a rule that escalates more often spends the same budget, divided differently.

Level-1 mode The planner runs CEM (Rubinstein, 1999) over primitive sequences of length $H_1 = 2k_2$, rolls each forward with f_1, and scores its endpoint by its distance to the goal latent $i_1(o_g)$. Sample counts are set so that every step spends at most B, which on GraphMaze at $B = 2048$ is 124 samples, 16 elites, and 2 iterations after the gate's 64 calls. Only the first action is executed before replanning.

Escalated mode The planner runs CEM over sequences of 2 primitive chunks of length k_2, encodes each chunk with ρ_2, rolls level 1 forward under it, applies $\hat{\Delta}^2$ as in Eq. 4, and scores the corrected coarse state by its distance to the lifted goal $i_2(i_1(o_g))$. On GraphMaze at $B = 2048$ this is 82 samples and 2 iterations after the gate. The corrected coarse state after the selected first macro-action is the waypoint, and until the next escalation level 1 plans in level-1 mode with its cost replaced by the level-2 distance of its lifted endpoint to the waypoint.

Gate At every environment step we sample $M = \min(32, B/8)$ macro-actions, so that the gate, with its M calls to f_2 and M to ρ_2, never takes more than a quarter of the budget, compute $\hat{\Delta}^2$

for each, and read the magnitude $m(s)$ and clusterability $c(s)$ with the estimator of App. E.3. The planner escalates when $m(s) > \tau_m$ and $c(s) > \tau_c$. τ_m is the 80th percentile of m over training states and $\tau_c = 0.15$, the same values in every environment. Expressing τ_m as a percentile makes it independent of the scale of the residual. The size only and clusterability only rules of Table 3 drop one condition. With neither condition the planner falls back to the every boundary rule, which escalates at every k_2th step as HWM does. App. I.1 reports sensitivity to both thresholds.

H.1 Proposal seeding

When level 2 is invoked, the gate has already swept a^2 over M candidate macro-actions and computed the predicted residuals $\hat{\Delta}^2$ for each. Seeding reuses that sweep. We group the candidates by their residuals, unlike the estimator of App. E.3, which groups them by the macro-action, using k-means with 3 groups, and initialize the proposal distribution at the macro-actions nearest the group centers rather than sampling uniformly, holding the number of proposals fixed, so the search starts on the branches the residual separates.

Table 10: Seeding macro-action proposals from the residual structure, at matched proposal count. Uniform proposals are the setting used for every other number in this paper.

Setting	Uniform proposals	Residual seeded
GraphMaze, size 80	68.0±2.2	73.5±1.9
DiverseMaze, longest horizon	87.2±1.1	90.1±1.0

I Escalation thresholds and concentration

I.1 Threshold sensitivity

Table 11: Success (%) on DiverseMaze-Slip at matched compute as the two gate thresholds vary, with the HRP model fixed. "None" removes that condition, and removing both gives the every boundary rule. Black cells are the rules of Table 3. Standard errors are 1.4 to 2.1.

τ_m percentile \ τ_c	None	0.10	0.15 (default)	0.20	0.30
None	70.8	73.2	74.9	74.6	73.1
70	71.9	76.1	77.6	77.2	75.0
80 (default)	72.6	77.0	78.4	77.9	75.4
90	72.3	76.4	77.3	76.8	74.2

Success varies by at most 1.4 points over the thresholds adjacent to the default, and falls towards the single condition rules of Table 3 as either threshold is removed.

I.2 Concentration

The residual mass carried by the top decile of states is 58% on GraphMaze (size 80), 47% on DiverseMaze-Slip, 39% on Crafter, and 21% on Push-T, against 10% for a uniform spread (Fig. 5(b)). On DiverseMaze-Slip the top decile contains junctions and slip cells alike, since both carry large residuals and magnitude does not distinguish them (Sec. 4.3). Across GraphMaze sizes the concentration rises monotonically with the fraction of junction states (Table 12), so it follows the environment and not a coefficient we set.

Table 12: Junction fraction and residual concentration across GraphMaze sizes. Standard errors are below 2 points.

Graph size	8	20	40	60	80
Junction states (% of data)	3.1	4.4	5.6	6.3	6.9
Top decile residual mass (%)	41	48	53	56	58

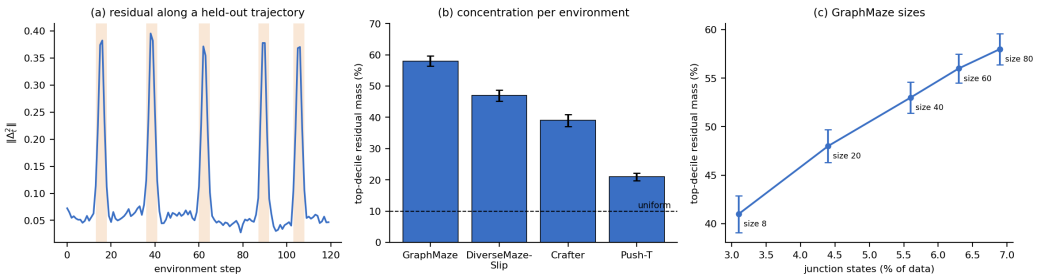

Figure 5: Concentration of the correction. (a) Residual norm along a held out GraphMaze trajectory, with ground truth junctions shaded. (b) Fraction of total residual mass carried by the top decile of states, per environment, with the uniform value of 10% marked. (c) Top decile mass against the fraction of junction states across GraphMaze sizes. Five seeds, bars are standard errors.

J ADAPTING THICK

We use THICK (Gumbsch et al., 2024) with the authors' public implementation. Its low level is a recurrent state space model with discrete latents trained by reconstruction, with a context that changes only when a gate opens and an L_0 penalty on how often it opens. The gate opens where updating the context improves the low level prediction, which biases it towards transitions, while the penalty keeps it closed elsewhere. The high level predicts the state and action at the next context change. We train it offline on the same reward free data as every other method, with its reward and value heads removed. The sparsity coefficient is the value the authors use for their navigation environments, kept fixed across ours, so it is never chosen by detection or planning results. For planning, the high level's prediction serves as a subgoal for the low level in the same CEM planner as HRP, with goals scored in THICK's latent space, and the high level is queried whenever the gate opens. Its detection score at s is the gate's opening probability. Planning compute counts applications of both its low and high level models (App. H).

K REGULARIZATION ABLATIONS

Table 13: Regularization ablations on DiverseMaze-Slip at matched compute. Detection is junction against slip cell. Effective rank is that of the level-2 embeddings, out of $d_2 = 128$.

Variant	Detection	Success (%)	Effective rank
HRP	0.86±.03	78.4±1.5	97
No SIGReg	0.50±.01	57.9±2.2	1.3
SIGReg only, i_2 without residual gradient	0.61±.04	71.5±1.9	94
No unit norm	0.81±.03	76.2±1.7	90

Without SIGReg the coarse encoder collapses, every residual vanishes, detection falls to chance, and the gate escalates at the wrong states, which costs more success than having no gate at all. With SIGReg but no residual gradient into i_2, the level-2 space keeps its dimensions but detection reaches only 0.61, so SIGReg prevents collapse without producing the relation between the residual and the decision structure, which must come from Eq. 5.

L DIVERSEMAZE

HWM's best success over the budgets we test is 83.1%, at $B = 2048$. Interpolating log-linearly between the budgets we evaluate, HRP reaches it at $B \approx 837$, a budget 2.4× smaller, and the same interpolation gives the 3.3× of Fig. 3(b).

Table 14: Success (%) on DiverseMaze at the three horizons of Zhang et al. (2026).

Horizon	HRP	HWM	THICK	Flat
Short (40 steps)	97.1±0.6	96.8±0.7	96.0±0.9	92.4±1.2
Medium (80 steps)	93.5±0.9	92.6±1.0	91.2±1.2	71.5±2.1
Longest (160 steps)	87.2±1.1	83.1±1.3	81.4±1.4	43.8±2.3

Table 15: Success (%) at the longest DiverseMaze horizon against the planning budget B (App. H). Standard errors are 1.0 to 2.4.

B	64	128	256	512	1024	2048
HRP	56.1	67.8	75.9	80.9	84.0	87.2
HWM	49.0	60.3	70.1	77.2	81.0	83.1

M Additional baselines

M.1 Goal conditioned RL

HIQL (Park et al., 2023) and GCIQL (Kostrikov et al., 2021) are trained on the same reward free data, with goals relabeled in hindsight from the trajectories and no task reward, using the implementations of Park et al. (2025). Table 16 reports both on GraphMaze, and Fig. 3(c) shows them under the restricted view.

Table 16: Success (%) of goal conditioned RL baselines on held out GraphMaze layouts. Restricted view values are those plotted in Fig. 3(c). Under full observability both stay below every world model method in Fig. 3(a) at every size.

View	Method	8	20	40	60	80
Full	HIQL	71.0	52.4	31.2	17.3	9.1
Full	GCIQL	58.3	39.0	21.1	10.2	5.0
Restricted	HIQL	38	22	15	3	1
Restricted	GCIQL	21	18	9	2	1

M.2 Error signals from level 1

We add three signals computed from level 1 alone: the one step error $\|f_1(z,a) - \bar{\phi}_1(o')\|$, the k_2 step rollout error in the level-1 space, and the disagreement of an ensemble of five one step predictors. Each is a scalar, like residual size, and each is scored on the state pairs of Table 1.

None of the three separates junctions from slip cells better than 0.58, the same failure as residual size, the scalar predictor, and THICK's gate in Table 1. Ensemble disagreement is the only one that fails on slip cells against corridors as well, since slip cells are common in the data and the ensemble agrees on their average outcome, while it is somewhat higher at the rarer junctions.

N Level 2 does not degrade level 1

Level 2's objective reaches i_1 through the coarse encoder's input (Sec. 3.3), so it could damage the representation level 1 relies on. The expected outcome is no difference, so the test is informative only if it can exclude a degradation worth caring about. We treat a drop of more than 0.05 in zero shot normalized return as material, and with five seeds the standard errors below are small enough to resolve one. On OGBench antmaze in the protocol of Bagatella et al. (2026), zero shot evaluation through the successor feature head of HRP's level 1 matches TD-JEPA trained without an upper level on all three settings, with no difference in either direction outside the standard errors.

We do not read the small positive difference on the large setting as an improvement, since no mechanism in the model would produce one.

Table 17: Detection AUROC on DiverseMaze-Slip for signals from level 1 alone, on the state pairs of Table 1.

State pair	One step error	Rollout error	Ensemble disagreement
Junction vs. ordinary corridor	0.84±0.03	0.88±0.02	0.74±0.04
Junction vs. slip cell	0.52±0.03	0.54±0.03	0.58±0.04
Slip cell vs. ordinary corridor	0.86±0.02	0.87±0.02	0.55±0.04

Table 18: Zero shot normalized return on OGBench antmaze.

Method	medium	large	giant
TD-JEPA	0.71±0.02	0.52±0.02	0.18±0.02
HRP level 1	0.70±0.02	0.54±0.02	0.18±0.01

O THREE LEVELS

O.1 RECURSION

Level 3 repeats the construction of Sec. 3.3 on level 2. Level 2's k_3 step prediction is formed by chaining Eq. 4 over k_3/k_2 strides, each applying f_2 to the previous corrected coarse state and adding the result to the lift of the uncorrected level-1 rollout at that stride,

$$\hat{z}^2_{t+jk_2} = \mathrm{sg}\big[i_2(\hat{z}^1_{t+jk_2})\big] + f_2\big(\hat{z}^2_{t+(j-1)k_2},\, a^2_{t+(j-1)k_2}\big), \qquad \hat{z}^2_t = z^2_t, \tag{14}$$

as drawn for $j = 2$ in Fig. 1. Level 3 encodes $z^3_t = i_3(z^2_t)$ and receives the lifted prediction $i_3(\hat{z}^2_{t+k_3})$ and the target $\bar{\phi}_3(\bar{\phi}_2(\bar{\phi}_1(o_{t+k_3})))$, the three quantities leaving level 2 in Fig. 1, which draws $k_3 = 2k_2$ for illustration while we use $k_3 = 3k_2$ on GraphMaze and $k_3 = 8k_2$ on Crafter. Its residual and loss are those of Eqs. 3 to 5 with every index raised by one,

$$\Delta^3_t = z^3_{t+k_3} - \mathrm{sg}\big[i_3(\hat{z}^2_{t+k_3})\big], \qquad \hat{z}^3_{t+k_3} = \mathrm{sg}\big[i_3(\hat{z}^2_{t+k_3})\big] + f_3(z^3_t, a^3_t), \tag{15}$$

with $a^3_t = \rho_3(a_{t:t+k_3-1})$. The stop gradient keeps level 3's loss out of level 2's predictions, while i_3 passes gradients to i_2 through z^2_t, as i_2 does to i_1. i_3 is regularized with SIGReg and its outputs normalized, and λ_{res} for level 3 follows App. G.

Planning with three levels The gate of App. H is applied top down. The planner escalates to level 3 when level-3 predicted residuals are large and clusterable, otherwise to level 2 under the level-2 rule, and otherwise plans with level 1. The waypoint of a level-3 escalation is the corrected level-3 state, which level 2 then plans towards.

O.2 GOAL HORIZON AND PLANNING BUDGET

Fig. 6(a) bins goals by the horizon they require rather than reporting one success rate. All variants are close at the shortest horizon, which lies at k_2, the same pattern as Sec. 4.5 measured with goals instead of branches. Each added level then extends the range of goals that remain reachable, with the point of separation tracking that level's own stride. The two level model separates from level 1 beyond k_2, and the three level model from the two level model beyond k_3. A level that helped before its stride, or uniformly across horizons, would indicate that the gain comes from added capacity rather than from operating at a coarser timescale.

Fig. 6(b) fixes goals requiring 64 to 128 steps and varies the planning budget. A decision is one environment step and a model call is one predictor application, so the axis is the unit of Sec. 4.1. The three level model reaches 0.40, the success the two level model attains at a budget of 512, with 4.3× fewer model calls, and the ratio is larger at the two level model's plateau of 0.44. The curves separate horizontally rather than only vertically, which is the separation a flat model cannot obtain by scaling, since additional parameters reduce per call error but not the number of calls a primitive search needs.

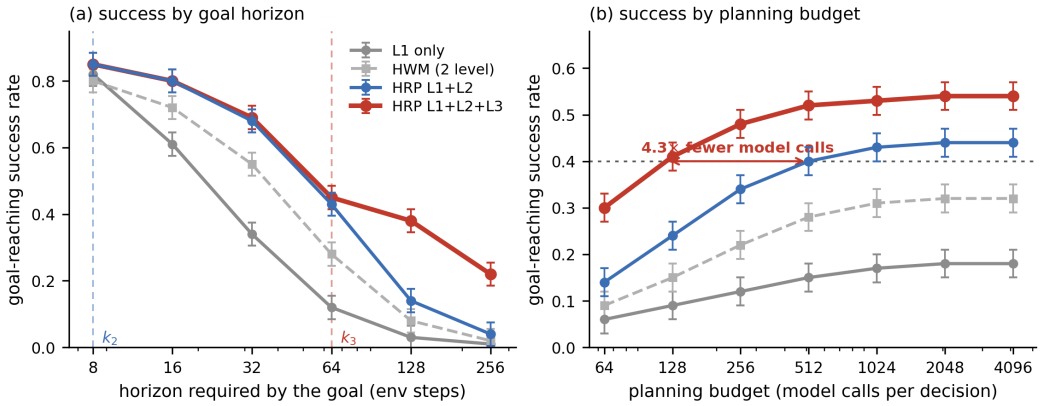

Figure 6: Crafter, trained without task reward. (a) Zero shot goal conditioned success against the horizon a goal requires, with each level's stride marked. The variants are close at k_2 and separate beyond the stride of each added level. (b) Success against planning budget on goals requiring 64 to 128 steps. Five seeds, bars are standard errors.

O.3 Crafter score

Aggregate benchmark scores measure an agent rather than a world model, since between HRP and a Crafter score sit exploration, a goal specification, and a planner, and any of them can account for a change in the number. We report it for comparability. HRP receives no task reward. During evaluation it is given goal observations for the achievements in a topological order of the achievement dependency graph, taken from held out trajectories as in App. C.3, and these goals are the only task information it receives. The score is computed over 500 evaluation episodes as defined by Hafner (2021).

Trained without task reward, HRP reaches 9.2% at three levels against 2.1% for the best prior agent that uses no task reward. The progression across depth is 2.0% at one level, 3.8% at two, and 9.2% at three, so most of the gain arrives with the third level. This is consistent with Table 4, where the second level does not reach the coarse scale at which Crafter's achievement tree branches. At one level HRP is at the level of the unsupervised baselines rather than above them, which is what a flat latent model should achieve here. We make no comparison against agents trained with task reward, since they optimize a different objective.

Two levels improve mid horizon goal while moving the aggregate score less than the third level does, because Crafter's score is dominated by achievements that require committing at the coarse scale, which Table 4 shows the second level does not reach. The score is a downstream measurement, and it responds only once the hierarchy is deep enough to contain the branching the task actually has.

P Scope: full numbers

P.1 Push-T

On Push-T the mean clusterability over sampled states is 0.04, against 0.29 at GraphMaze junctions, and the top decile of states carries 21% of the residual mass, close to the 10% of a uniform spread. With residuals that are neither concentrated nor action dependent, the gate escalates 0.4 times per episode, so HRP plans almost entirely with level 1, while HWM searches at the coarse scale at every stride. HWM stays ahead at every horizon.

P.2 State abstraction

We test one specific form of state abstraction, whether the coarse encoder maps states with identical futures close together. On a branching graph with 64 pairs of distinct states whose futures are identical under every action sequence, the level-2 distance does not separate these pairs from pairs

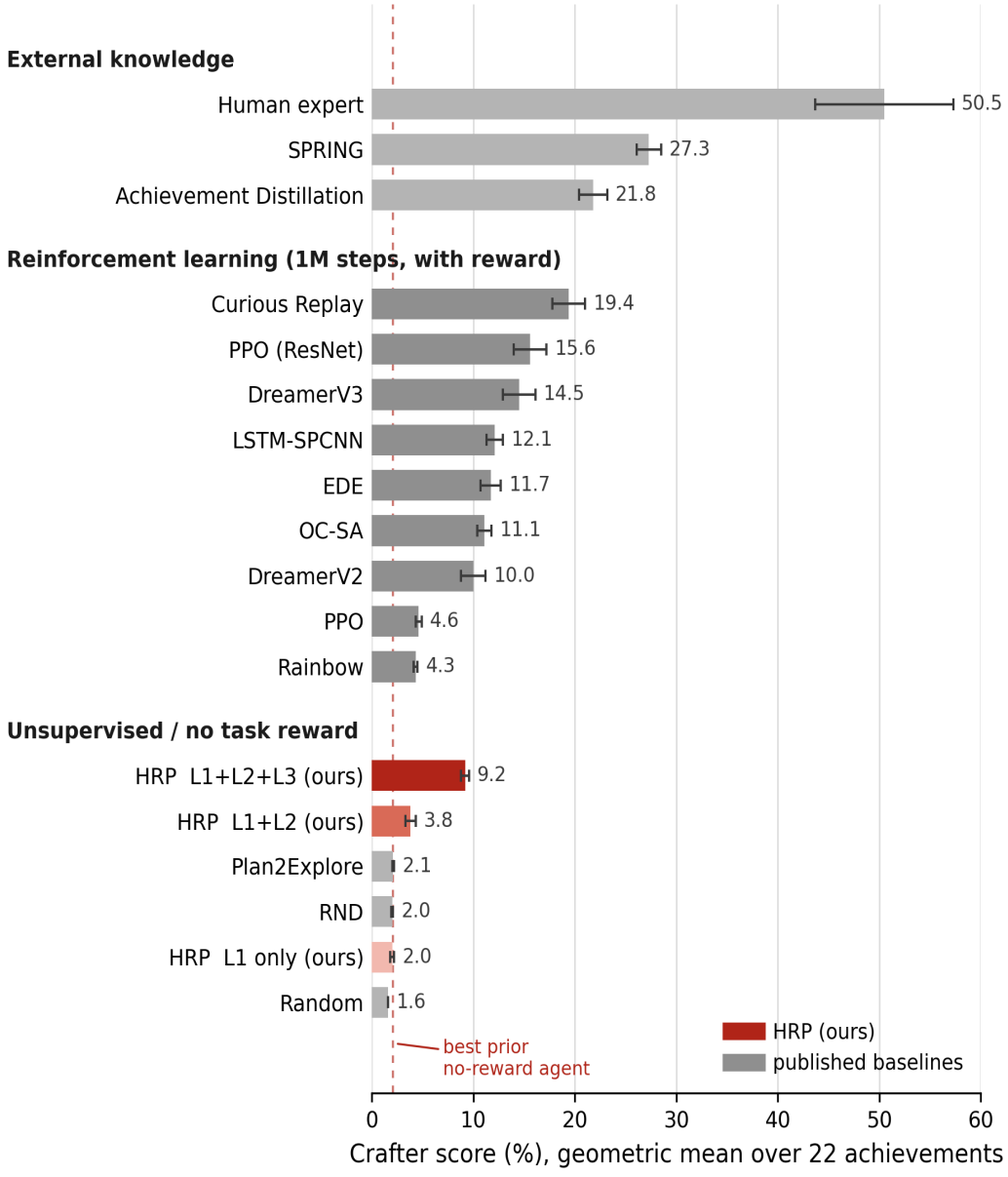

Figure 7: Crafter score (%) of current state of the art agents

Table 19: Success (%) on Push-T.

Horizon (steps)	HRP	HWM	Flat
25	70.1±1.8	74.8±1.6	69.5±1.9
50	43.2±2.1	55.3±2.0	41.0±2.2
100	16.9±1.7	30.6±2.0	14.2±1.6

with different futures at the same level-1 distance (AUROC 0.52±0.03), and merge pairs lie at 0.97 of the distance of matched non-merge pairs. Under this test the level-2 space does not merge equivalent states. This does not rule out other forms of abstraction, which we do not evaluate, and our claims rest only on the decision point signal.