EVOGRAPHWM: A STRUCTURE-EVOLVING WORLD MODEL WITH ADAPTIVE OBJECT GRAPH FOR LONG-HORIZON TASKS IN DYNAMIC ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon tasks are challenging due to their sparse rewards, where useful behaviour takes many steps to pay off. World models address such tasks by learning environment dynamics and optimising behaviour through imagined rollouts, which therefore need to stay accurate over long horizons. In dynamic environments, new objects continually appear as the agent explores, changing the structure required to represent the environment state. Existing world models fix the structure of their state representation as a holistic vector or a predefined set of object slots, leaving new objects uncaptured and their relations unmodelled, and the resulting errors compound over long-horizon imagination. To solve this problem, in this paper we propose a world model that evolves its state structure with the objects encountered during training. However, realising such structural evolution raises three challenges: 1) how to discover and prioritise new objects during training; 2) how to model the relations between newly discovered objects and those already known; 3) how to learn dynamics and behaviour over a structure-evolving state. To address them, we present **EvoGraphWM**, a **structure-evolving world model** centred on an adaptive object graph, whose nodes, focus, and relations evolve as the agent learns. EvoGraphWM consists of: (i) a *Self-evolving Node Extractor* that grows the node set, discovering new objects through the reconstruction error and prioritising the reward-relevant ones as dedicated nodes while the rest stay in a shared background; (ii) a *Relational Dynamics Modeller* that learns the graph's transition through collaborative node prediction, where object relations emerge; (iii) an *Evolved Graph Integrator* that anchors the adaptive object graph to a steady holistic state through two complementary routes, giving dynamics and behaviour a stable footing to learn from. Experiments demonstrate that EvoGraphWM outperforms strong baselines on both Crafter and Procgen Heist, and further keeps the imagined rollouts accurate over remarkably longer horizons.

1 INTRODUCTION

Solving long-horizon tasks is a crucial yet challenging problem in reinforcement learning. The challenge lies in their sparse rewards, where useful behaviour pays off only after a series of intermediate steps. World models address such tasks by learning environment dynamics from experience and optimising agents' behaviour in imagined rollouts (Ha & Schmidhuber, 2018; Hafner et al., 2018; 2024; Micheli et al., 2023; Hansen et al., 2024), which requires the imagination to be accurate over a long horizon. Dynamic environments make this harder. New objects continually appear as the agent explores, a new material, a new creature, or a new tool, and each gates the next stage of progress. The imagined rollouts therefore run through objects the model has just met. To keep its rollouts accurate, a world model must *discover* new objects as they appear, *prioritise* the ones that matter, and *relate* them to the objects it already knows.

Existing world models, however, fix their representation structure before training. Holistic models compress the scene into a single latent vector, in which every object is entangled with the rest. Such a vector can barely tell objects apart, let alone take in a new one. Object-centric models do separate objects into slots, but the object set is decided in advance (Delfosse et al., 2024; Zhang et al., 2026; Feng et al., 2025), and exploration inevitably exceeds it. In both cases the model cannot capture

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 objects it newly meets, nor model how they relate to the ones it already knows. The imagined rollouts then fail on such objects, and the errors compound over the long horizon.

Meeting these demands calls for a state representation whose structure grows by itself rather than being given in advance. We therefore propose a novel structure-evolving world model that adapts its state structure with the objects it encounters.

Nevertheless, building such a structure-evolving world model faces three challenges:

- It is challenging to discover and prioritise new objects during training. New objects arrive unlabelled, show their importance many steps later, and are often much rarer than the objects already known.
- It is challenging to relate newly discovered objects to those already known. A new object appears with few observed interactions, and no label tells which of them it acts on.
- It is challenging to learn dynamics and behaviour over a structure-evolving state, as every structural change breaks what has already been learned.

To solve these challenges, we propose **EvoGraphWM**, a structure-evolving world model centred on an *adaptive object graph*, where i) the node set expands as new objects are discovered, ii) the focus shifts as reward reveals which objects matter, and iii) the relations emerge as the nodes learn to predict one another. To the best of our knowledge, EvoGraphWM is the first world model to evolve its own state structure inside the reinforcement-learning loop. Specifically, EvoGraphWM consists of three key components: *Self-evolving Node Extractor*, *Relational Dynamics Modeller*, and *Evolved Graph Integrator*. The *Self-evolving Node Extractor* discovers new objects from the model's own reconstruction error. Where the residual reveals an unexplained region, a new node is born to represent it, and reward prioritises the relevant ones as dedicated nodes while the rest stay in a shared background. The *Relational Dynamics Modeller* learns the graph's transition by predicting each node's dynamics from the others. Relations between the new nodes and the known emerge rather than being declared. The *Evolved Graph Integrator* maintains a holistic state as a steady footing and integrates the adaptive object graph along two routes. One trains the holistic state to predict the graph's future, folding the graph's dynamics into it, and the other lets the policy read the graph as it acts.

On Crafter (Hafner, 2021) and Procgen Heist (Cobbe et al., 2020), EvoGraphWM outperforms strong baselines by a clear margin, and further keeps its imagined rollouts accurate over remarkably longer horizons. In summary, our contributions are as follows.

- We propose **EvoGraphWM**, to the best of our knowledge the first world model to evolve its own state structure inside the reinforcement-learning loop, in the form of an *adaptive object graph*.
- We realise this with three components: a *Self-evolving Node Extractor* that grows and prioritises the node set, a *Relational Dynamics Modeller* that learns the graph's transition and lets the relations emerge, and an *Evolved Graph Integrator* that anchors the graph to a steady holistic state, keeping structural changes from breaking what has been learned.
- EvoGraphWM outperforms strong baselines on Crafter and Procgen Heist. Analyses demonstrate that the adaptive object graph keeps imagined rollouts accurate on the decisive objects over long horizons.

2 Related Work

2.1 Holistic World Models

Latent world models compress observations into a single recurrent state and train a policy inside rollouts imagined from that state, reusing each real transition many times. PlaNet introduced latent planning with a recurrent state-space model (Hafner et al., 2018); Dreamer and DreamerV2 moved policy optimisation into the imagination itself (Hafner et al., 2020; 2022); and DreamerV3 showed that one configuration transfers across continuous control, Atari and Minecraft, where it is the first method to obtain diamonds without human data (Hafner et al., 2024). A parallel line replaces the recurrent state with an autoregressive sequence model over discrete tokens (Micheli et al., 2023;

2024; Zhang et al., 2023), and a third abandons reconstruction altogether and predicts directly in representation space, through decoder-free latent models in reinforcement learning (Hansen et al., 2024) and joint-embedding predictive architectures at scale (Assran et al., 2023; Bardes et al., 2024). Despite their different objectives, all of these models fix the structure of the representation before training. The scene is compressed into one entangled state, and a newly appearing object is absorbed rather than represented on its own.

2.2 Object-Centric World Models

Object-centric world models represent a scene as a set of slots and model dynamics as interaction among them, which yields localised transitions and compositional generalisation in structured domains such as robotic manipulation, where the objects are few, bounded, and present from the first frame (Kipf et al., 2020; Veerapaneni et al., 2020; Kossen et al., 2020; Mosbach et al., 2025; Feng et al., 2025). The slots descend from Slot Attention and its video successors (Locatello et al., 2020; Kipf et al., 2022; Elsayed et al., 2022), but fully unsupervised decomposition collapses once objects are textured (Singh et al., 2022b). The field's remedies reconstruct discrete tokens or frozen self-supervised features instead (Singh et al., 2022a; Seitzer et al., 2023), at the price of a pretrained visual model. The object set comes from outside the agent and is read from emulator internals or trained detectors (Delfosse et al., 2024), propagated from per-game annotation by frozen segmentation backbones (Zhang et al., 2026), or built on pretrained embeddings (Feng et al., 2025). However the object set is fixed before learning, and the representation cannot adapt to the new objects that continually appear as an agent explores.

2.3 Self-Evolving Models

Self-evolving models have many precedents in machine learning (Mitchell et al., 2018). In continual learning, networks expand their capacity as new tasks arrive (Rusu et al., 2022; Yoon et al., 2018; Li et al., 2019). Open-ended learning grows an agent's repertoire over its lifetime, from co-evolved tasks to libraries of skills (Wang et al., 2019; 2023). These precedents each grow on an external cue such as a task boundary, which a world model does not receive.

3 EvoGraphWM

In this paper, we propose EvoGraphWM, a structure-evolving world model built around an *adaptive object graph* $\mathcal{G}_t = (s_t^{1:K}, R_t)$, with K node states $s_t^k \in \mathbb{R}^{d_s}$ and a relation matrix $R_t \in [0,1]^{K\times K}$. Its nodes carry object types drawn from a growing set of prototypes $C \in \mathbb{R}^{P\times d}$ through a seat map σ. Three parts of the graph evolve during training. The prototype set grows as new objects are discovered, the seat map σ reallocates capacity as reward reveals which types matter, and its relations sharpen as the graph learns to predict its own dynamics.

EvoGraphWM is realised by three components, as shown in Figure 1. The *Self-evolving Node Extractor* (Section 3.1) reads each observation and emits a per-region node assignment $u_{t,i}$, growing the prototype set and choosing which types hold dedicated nodes. The *Relational Dynamics Modeller* (Section 3.2) binds the regions into node states $s_t^{1:K}$ under $u_{t,i}$ and models their transition, from which the relation matrix R_t emerges. The *Evolved Graph Integrator* (Section 3.3) adopts two routes to combine the graph $\mathcal{G}_t$ with a holistic latent (h_t, z_t) to learn environment dynamics, forming the state the actor π and critic v act from. One folds the graph's dynamics into the holistic transition, and the other lets the policy read the graph as it acts.

3.1 Self-evolving Node Extractor

As the agent explores, new objects keep surfacing, and only a few of them matter to the task. The Self-evolving Node Extractor discovers them with a parameter-free process, opening a fresh type where reconstruction breaks down, and raises those proven to matter into dedicated nodes.

The Extractor adopts a frozen front end ϕ that segments each observation into regions and embeds each as $e_{t,i} = \phi(x_{t,i}) \in \mathbb{R}^d$ (Appendix F). Every region is assigned to its nearest active prototype,

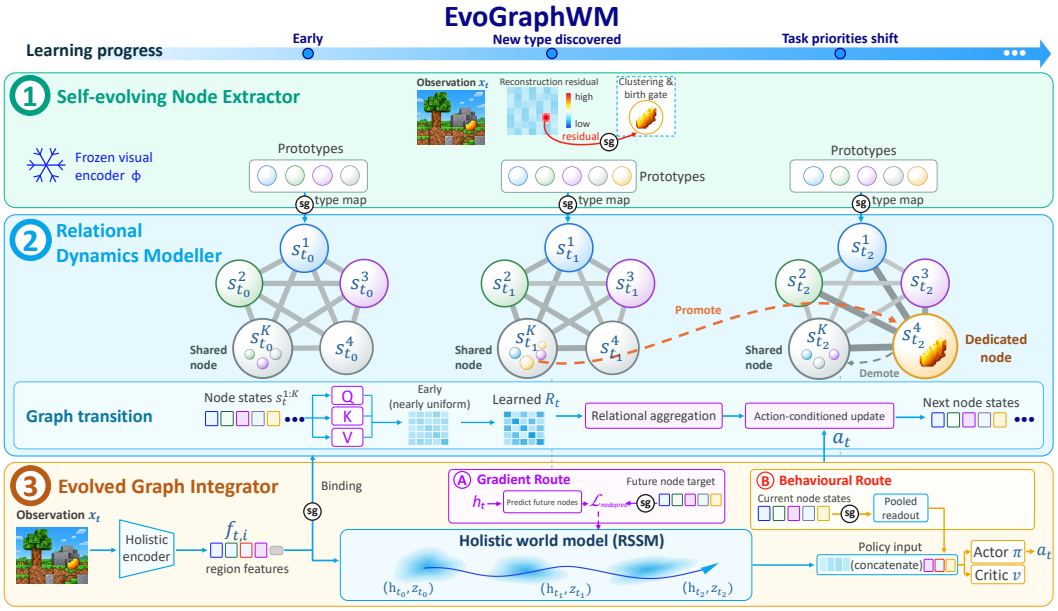


Figure 1: EvoGraphWM and its three modules. **(1) The Self-evolving Node Extractor** embeds each region with a frozen encoder ϕ, where the reconstruction residual $r_{t,i}$ marks an unexplained region, clustering and a novelty gate generate a new object type, and reward reallocates seats between dedicated and shared nodes, emitting the type map $u_{t,i}$. **(2) The Relational Dynamics Modeller** binds the region features $f_{t,i}$ into node states $s_t^{1:K}$ under $u_{t,i}$, then predicts the next states with a multi-head attention, from which the relation matrix R_t emerges. **(3) The Evolved Graph Integrator** keeps a holistic state and integrates the graph through two separate routes, a gradient route that trains h_t to predict future node states and a behavioural route that appends nodes' readout to the policy input, from which the actor π and critic v learn in imagination.

and from there to a graph node,

$$j^\star(t,i) = \arg\min_{j:\,\alpha_j=1} \|e_{t,i} - c_j\|^2, \qquad u_{t,i} = \sigma\big(j^\star(t,i)\big), \tag{1}$$

where each prototype c_j is an object type, stored as a row of a growing matrix $C \in \mathbb{R}^{P\times d}$; the mask $\alpha \in \{0,1\}^P$ marks which rows are active, and the seat map $\sigma : \{1,\dots,P\} \to \{1,\dots,K\}$ routes each type to one of the K graph nodes, of which the first $K-1$ are dedicated to a single type and the last is shared by all the rest.

Growth of the prototype set. A birth proceeds in two steps. The model first proposes a candidate from the regions it fails to reconstruct, then keeps it only if it is genuinely new.

What deserves attention emphasises what the model still fails to reconstruct. The slot decoder of Section 3.2 reconstructs each region, and the N regions with the largest reconstruction error $r_{t,i}$ in the batch form the candidate pool $\mathcal{T}$. Weighted by the reconstruction error normalised per frame, each candidate a is clustered through a weighted k-means in embedding space, and the centre of the heaviest cluster becomes the proposed prototype $\hat{c}$.

$$w_a = \frac{r_a}{m_a\,\bar{r}_{\mathcal{T}}}, \tag{2}$$

where m_a is the number of candidates from the same frame as a, and $\bar{r}_{\mathcal{T}}$ the pool's mean error.

What is genuinely new is settled by a birth gate. The proposal enters the prototype set only if it lies beyond a radius τ_{twin} of every active prototype and its cluster carries enough mass $M^\star$. Growth is thereby self-limiting, since once a type holds a row, its novelty vanishes and the gate filters it out.

$$\text{birth} = \mathbf{1}\big[\delta > \tau_{\text{twin}}\big] \cdot \mathbf{1}\big[M^\star > M_{\min}\big], \qquad \delta = \min_{j:\,\alpha_j=1} \|\hat{c} - c_j\|^2. \tag{3}$$

Node allocation. The seat map σ routes each prototype to a graph node, giving a dedicated node to the $K-1$ types with the highest seat score crit_j and pooling the rest onto the shared node K. The score balances task relevance and stability. Relevance shifts as the agent's competence does, while stability matters since σ supplies the binding targets $u_{t,i}$ (Section 3.2), and every reshuffle changes what is trained against. Formally, with $b_{t,j} \in \{0,1\}$ indicating whether any region of frame t is assigned to type j, denote $\bar{b}_j$ as its exponential average, the presence rate of prototype j, and $\bar{e}_j$ the average conditioned on reward. Relevance is their contrast $\rho_j = \alpha_j \max(\bar{e}_j - \bar{b}_j, 0)$, and the score mixes it with the presence rate. Hyperparameters are given in Appendix G.

$$\mathrm{crit}_j \;=\; (1-m)\,\frac{\rho_j}{\sum_{j'}\rho_{j'}} \;+\; m\,\frac{\bar{b}_j}{\sum_{j'}\bar{b}_{j'}}, \qquad m \in [0,1]. \tag{4}$$

3.2 Relational Dynamics Modeller

The discovered graph is static, a snapshot that names the objects but freezes them. However, the agent learns from imagining the dynamics of each object and the evolving relations among them. We therefore set the graph in motion with the Relational Dynamics Modeller, binding regions into node states and learning their transition.

Binding. At each frame, the node states are initialised from a prior over the previous frame and refined by competitive attention over the region features $f_{t,i}$,

$$A = \mathrm{softmax}_k\Big(\tfrac{q(s)\,k(f)^\top}{\sqrt{d_s}}\Big), \qquad s^k \leftarrow s^k + \mathrm{MLP}\big(\big[\,s^k,\ \textstyle\sum_i \tilde{A}_{ik}\,v(f_{t,i})\,\big]\big), \tag{5}$$

where q, k, v are learned linear maps, A is normalised over nodes and $\tilde{A}$ over regions, yielding the node states $s_t^{1:K}$. The attention is trained against the prototype map $u_{t,i}$ by an inverse-frequency weighted cross-entropy

$$\mathcal{L}_{\mathrm{bind}} = -\sum_i \omega_i \log A_{i,\,u_{t,i}}, \qquad \omega_i \propto \mathrm{freq}(u_{t,i})^{-1/2}\,\kappa^{\mathbf{1}[u_{t,i}\neq K]}, \tag{6}$$

which up-weights rare prototypes and boosts the dedicated nodes by κ. A reconstruction loss decodes each node back to its region features with a spatial-broadcast decoder (Singh et al., 2022b; Seitzer et al., 2023) and sums the per-region error,

$$\mathcal{L}_{\mathrm{rec}} = \sum_i r_{t,i}, \qquad r_{t,i} = \big\|f_{t,i} - \hat{f}_{t,i}\big\|^2, \tag{7}$$

where $\hat{f}_{t,i}$ is the softmax-over-slots composite. This per-region residual $r_{t,i}$ also serves for node evolution (Section 3.1).

Transition. The transition predicts the next node states $\hat{s}_{t+1}$ from the current states $s_t^{1:K}$ and the action, each conditioned on the others that act on it. The key challenge is that relational structure does not have a reliable label. For this reason, we let it emerge as a relation matrix $R_t \in [0,1]^{K\times K}$, read from the node states and shaped by prediction alone.

At each step, R_t is a multi-head attention over the node states, and every node is updated by the aggregated message before an action-conditioned recurrence,

$$R_t = \frac{1}{N_h}\sum_{n=1}^{N_h} \mathrm{softmax}\left(\frac{q_n(s_t)\,k_n(s_t)^\top}{\sqrt{d_s/N_h}}\right), \qquad \hat{s}_{t+1} = \tilde{s}_t + \mathrm{MLP}(\tilde{s}_t + W_a\,e(a_t))\,, \tag{8}$$

where N_h is the number of attention heads and q_n, k_n their per-head projections, $\tilde{s}_t = s_t + W_o\big(R_t\,v(s_t)\big)$ aggregates related nodes through a learned map W_o, and W_a projects the action embedding $e(a_t)$.

The predicted states $\hat{s}_{t+1}$ are aligned with the actual states s_{t+1} bound from the next frame by a two-sided squared error,

$$\mathcal{L}_{\mathrm{sdyn}} = \big\|\mathrm{sg}(s_{t+1}) - \hat{s}_{t+1}\big\|_2^2, \qquad \mathcal{L}_{\mathrm{srep}} = \big\|s_{t+1} - \mathrm{sg}(\hat{s}_{t+1})\big\|_2^2, \tag{9}$$

where $\mathcal{L}_{\mathrm{sdyn}}$ moves the prediction toward the bound states, for an accurate transition, and $\mathcal{L}_{\mathrm{srep}}$ moves the bound states toward the prediction, for a representation that stays predictable.

3.3 Evolved Graph Integrator

Policy requires a consistent state to act from, while the graph instead evolves, its node semantics shifting as training proceeds. We therefore maintain a holistic latent as the steady part of the state, integrating the adaptive object graph for structural evolution. The holistic latent stays a standard recurrent state-space model, and the graph reaches it through two separate routes, gradient and behavioural.

Holistic latent state. Adapted from Hafner et al. (2024), the holistic state (h_t, z_t) pairs a deterministic recurrent part h_t with a stochastic part z_t. A posterior $q(z_t \mid h_t, x_t)$ infers the stochastic state from the current observation x_t, and a prior $p(\hat{z}_t \mid h_t)$ predicts it from the recurrent state alone, aligned by a two-sided KL,

$$\mathcal{L}^h_{\text{dyn}} = \mathrm{KL}\big[\mathrm{sg}\big(q(z_t \mid h_t, x_t)\big) \,\big\|\, p(\hat{z}_t \mid h_t)\big], \qquad \mathcal{L}^h_{\text{rep}} = \mathrm{KL}\big[q(z_t \mid h_t, x_t) \,\big\|\, \mathrm{sg}\big(p(\hat{z}_t \mid h_t)\big)\big]. \tag{10}$$

Together with the reconstruction, reward, and continue terms, these form the holistic state's objective, which we denote $\mathcal{L}_{\text{base}}$.

Gradient route. The gradient route trains the holistic state to predict the graph with a node prediction loss, whose two terms carry what each node holds and where it sits. The holistic transition is unrolled for H steps along the actual actions, and at every depth a shared head f_θ predicts the node states, while another g_θ decodes each region's node assignment,

$$\mathcal{L}_{\text{nodepred}} = \sum_{l=0}^{H} \Big[\frac{1}{|\mathcal{A}|} \sum_{j \in \mathcal{A}} \big\| f_\theta(\hat{h}_{t+l})_j - \mathrm{sg}\big(s^{\sigma(j)}_{t+l}\big) \big\|_2^2 - \lambda_{\text{lay}} \sum_i \omega_i \, \log g_\theta\big(u_{t+l,i} \mid \hat{h}_{t+l}\big) \Big], \tag{11}$$

where $\hat{h}_{t+l}$ is the unrolled holistic state, $\mathcal{A}$ are the active prototypes and $\sigma(j)$ the node prototype j occupies. ω_i are the inverse-frequency weights, to avoid the common types dominating the loss. Detaching the targets sends the gradient into the holistic state alone, drawing its modelling toward the graph while leaving the graph's own dynamics untouched.

Behavioural route. The behavioural route allows the actor and critic to read the graph during learning. The node states are pooled by a permutation-invariant sum and appended to the policy input,

$$\psi_t = \Big[h_t,\ z_t,\ W_o \textstyle\sum_{k=1}^{K} g(\mathrm{sg}(s^k_t)) \Big], \qquad W_o \equiv 0 \text{ at initialisation}, \tag{12}$$

where g is a shared per-node encoder and W_o a zero-initialised projection. The policy thus starts at the holistic state and departs from it only as the graph proves useful.

3.4 Objective

The model is trained end-to-end under

$$\mathcal{L} = \underbrace{\lambda_{\text{bind}}\mathcal{L}_{\text{bind}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{dyn}}\mathcal{L}_{\text{sdyn}} + \lambda_{\text{rep}}\mathcal{L}_{\text{srep}}}_{\text{Modeller}} + \underbrace{\mathcal{L}_{\text{base}} + \lambda_{\text{pred}}\mathcal{L}_{\text{nodepred}}}_{\text{Integrator}}. \tag{13}$$

The two groups collect the losses of each component. All coefficients are listed in Table 4 (Appendix G).

4 Experiments on Crafter

4.1 Setup

Environment. Crafter (Hafner, 2021) is an open-world survival game whose progress forms a long-horizon tree of 22 achievements. We follow the benchmark's published protocol. The budget is one million environment steps, and we report the mean over three seeds. For each achievement the success rate is the fraction of *all* training episodes in which it was unlocked, and the score is the geometric mean of those rates with a one-percent offset,

$$S \;=\; \exp\Big(\tfrac{1}{22} \textstyle\sum_i \ln\big(1 + 100\, p_i\big) \Big) - 1. \tag{14}$$

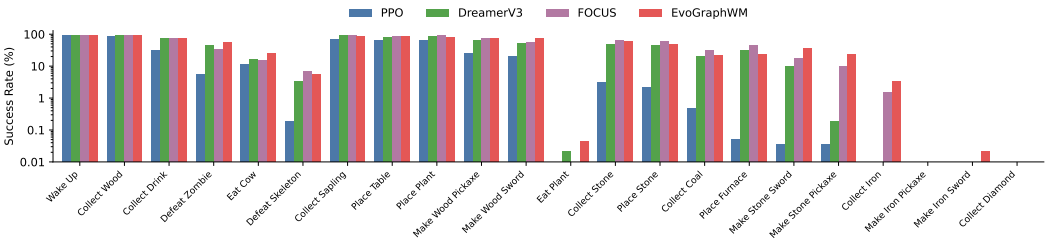


Figure 2: Per-achievement success rates on Crafter, ordered by depth in the achievement tree.

The reward is the mean episode return over the final 100k steps of the budget.

Implementation. Our model has 172M parameters. Following DreamerV3 (Hafner et al., 2024), the holistic state adopts a recurrent state-space model. Detailed model configuration can be found in Appendix G.

Baselines. The baselines fall into three families. We compare with a representative object-centric world model, FOCUS (Ferraro et al., 2023); two holistic model-based agents, DreamerV3 (Hafner et al., 2024) and its predecessor DreamerV2 (Hafner et al., 2022); and two model-free agents, the policy-gradient PPO (Schulman et al., 2017) and the value-based Rainbow (Hessel et al., 2017). More details are given in Appendix B.

4.2 Results

On Crafter, EvoGraphWM reaches a score of 19.9 ± 1.5 with a reward of 12.6 ± 1.2, above every baseline in Table 1 and 40% above the DreamerV3 backbone. FOCUS, given ground-truth segmentation masks for its objects, scores above the holistic baselines as expected, while EvoGraphWM, capturing the evolving objects, priorities, and the corresponding relations, passes it by a further 2.7 points. Figure 2 locates the gain. The world models sit well above the model-free PPO and Rainbow. On the common early achievements the three world models are comparable, and the gap opens where the later objects enter. EvoGraphWM crafts the stone sword and stone pickaxe roughly twice as often as FOCUS, far beyond DreamerV3, and collects iron more than twice as often as any baseline. These late objects are exactly the ones the graph discovers and promotes during training.

Table 1: Results on Crafter at one million steps. Mean and std are reported over three seeds.

Method	Score	Reward
PPO	4.6 ± 0.3	4.4 ± 0.4
Rainbow	4.3 ± 0.2	5.6 ± 0.2
DreamerV2	10.0 ± 1.2	9.3 ± 0.6
DreamerV3	14.2 ± 1.7	11.6 ± 0.5
FOCUS	17.2 ± 1.8	12.2 ± 1.0
EvoGraphWM	$\mathbf{19.9 \pm 1.5}$	$\mathbf{12.6 \pm 1.2}$

4.3 Ablation Studies

To verify the effectiveness of each module, we remove one mechanism at a time from the full model (Table 2). Each variant isolates one of the three modules: node births in the Extractor, the relational prediction in the Modeller, and the coupling routes in the Integrator. Visualisation of the learned relation matrix is given in Appendix D.

Every ablation lands below the full model. Removing births costs 2.3 points, showing that the born types carry real progress. Removing the relations costs 4.0, showing that progress rests on the dependencies between objects rather than the objects alone. The two injection routes are not symmetric, the gradient route alone still scores 16.7, while the behavioural route alone drops to 13.9, the largest fall in the table. The

Table 2: Ablations on Crafter. Each row removes one mechanism from the full model. Mean and std are reported over three seeds.

Variant	Score	Reward
EvoGraphWM (full)	$\mathbf{19.9 \pm 1.5}$	$\mathbf{12.6 \pm 1.2}$
no births	17.6 ± 1.1	12.0 ± 0.8
no relations	15.9 ± 2.1	12.1 ± 0.7
gradient route only	16.7 ± 1.9	11.9 ± 0.9
behavioural route only	13.9 ± 2.2	11.0 ± 0.7

gap follows from where each route places the graph. The gradient route folds the graph into the holistic transition, and the policy, trained in imagination, inherits the gain; the behavioural route lets the policy read the graph as it acts, while imagination stays blind to the objects it sees.

4.4 Results of Node Evolution

Node evolution adds prototypes for the objects the initial set misses, and we census what evolved in Figure 3. The number for each type is the purity of its best prototype. The evaluation assigns 2.5M ground-truth-labelled regions to their nearest prototype, in the initial set and in the evolved set separately, giving the two columns in Figure 3. Ground truth is used for evaluation only and never enters the model.

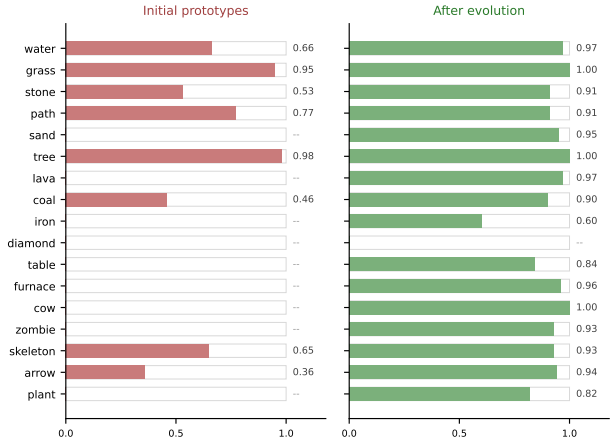


Figure 3: The coverage of the prototype set before and after evolution. Numbers indicate the purity of each type's best prototype.

The initial set is fitted on a short rollout before training (Appendix F), and involves eight prototypes, several of them impure. Evolution doubles the set to sixteen prototypes. Eight new prototypes are born, covering the entities and crafted stations the initial set never named, and the initial prototypes are purified in turn, e.g., water rising to 0.97 and stone to 0.91. Nevertheless, iron is still separated imperfectly and diamond has yet to appear, as exploration seldom reaches them, matching the low completion rates on the two achievements. The prototype set is visualised in Appendix C.

4.5 Long-horizon imagination

To measure how far imagination stays accurate, we warm the models up on 50 observed steps and let them imagine the next 50 open-loop, along the actions the agent actually took. Each imagined step is decoded to pixels and scored against the ground-truth frame over 105 held-out sequences. We compare EvoGraphWM, the no-births ablation (the initial prototype set is fixed during training), and

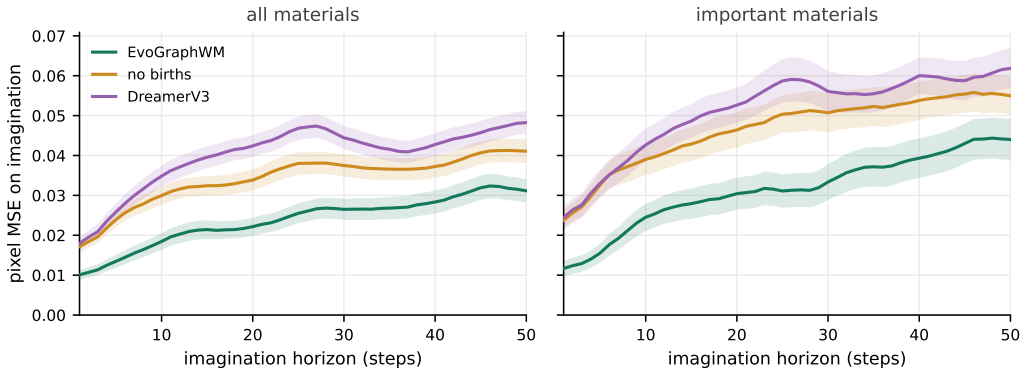


Figure 4: Imagination error over the horizon, mean over 105 sequences with the standard error shaded. Left: all materials. Right: the important materials (stone, coal, iron, lava, table, furnace, diamond). Lower is better.

DreamerV3 in Figure 4, over all materials and the important ones (namely stone, coal, iron, lava, table, furnace, and diamond) separately.

EvoGraphWM keeps the lowest error at every horizon, and the margin opens within the first ten steps and holds to fifty. On the important materials the separation is widest, DreamerV3 approaching double our error by the middle of the horizon. The no-births ablation sits between the two, splitting the gain into a part the object stream itself carries and a part the born prototypes add. Concrete rollout cases are given in Appendix E.

5 More Experiments on Procgen Heist

Procgen Heist (Cobbe et al., 2020) is a procedurally generated maze in which the agent collects colour-coded keys to open a sequence of locks guarding a gem. We use it to verify that EvoGraphWM adapts to a different environment. Training and evaluation both use the hard difficulty over the full, unbounded level distribution. The budget is 40M environment steps. We report raw episode return and compare against DreamerV3 and the model-free PPO and PPG (Cobbe et al., 2021).

Figure 5 shows the result. EvoGraphWM pulls ahead of every baseline within the first eight million steps and holds the lead for the rest of the run, reaching a return of 9.6 at 40M against DreamerV3's 7.0, PPO's 5.8, and PPG's 2.4, an improvement of 37% over the strongest baseline.

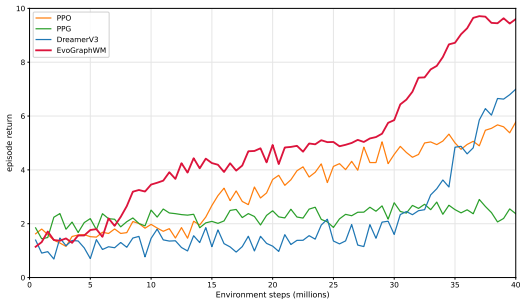


Figure 5: Episode return on Procgen Heist over 40M environment steps.

6 Limitations

Our proposed method has three limitations. First, it relies on a frozen front end that embeds the regions in a metric space, which could be difficult in an entirely new environment. Second, discovery runs on appearance reconstruction, and a type that appears rarely or late is hard to find, as with iron and diamond. Third, the evolving focus only credits the types present as the reward is given. Temporally extended credit is left to future work.

7 Conclusion

Existing world models struggle in dynamic environments, where a fixed state representation cannot take in the objects that keep appearing, and the errors compound over long imagined rollouts. In this paper, we propose **EvoGraphWM**, a structure-evolving world model that adapts its state structure as the agent learns. By introducing an *adaptive object graph* as a relational state and evolving it with a *Self-evolving Node Extractor*, a *Relational Dynamics Modeller*, and an *Evolved Graph Integrator*, EvoGraphWM captures relational dynamics that evolve with the task and keeps its long-horizon imagination accurate on crucial objects that decide progress. Experiments on Crafter and Procgen Heist demonstrate that EvoGraphWM improves over strong baselines and mitigates the drift accumulated over long rollouts. These findings highlight the value of an adaptive object graph for extending world models beyond a fixed state. EvoGraphWM offers a practical step toward more reliable planning for long-horizon tasks in dynamic environments.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish the writing, find some of the related works, and for research execution, including assistance with code implementation and debugging. We have not used generative AI tools for the other tasks with required disclosure, and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. AI-assisted text was revised and checked by the authors, AI-retrieved related works were read and confirmed, and AI-assisted code was verified and tested for correctness 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

All authors of this work follow the ICLR Code of Ethics. No human subjects or personal data are involved, and all experiments run on publicly available data. We have considered the potential biases and fairness of the method, and foresee no applications or insights that could cause harm. All experiments comply with applicable legal and ethical standards in machine learning research.

REPRODUCIBILITY STATEMENT

We take several steps to make the results reproducible. The evaluation protocol is specified in Section 4.1, the implementation details of every module are given in Appendix F, and the full hyperparameter configuration in Appendix G. The source code will be released publicly upon acceptance.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture, 2023. URL https://arxiv.org/abs/2301.08243.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video, 2024. URL https://arxiv.org/abs/2404.08471.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/cobbe20a.html.

Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2020–2027. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/cobbe21a.html.

Quentin Delfosse, Jannis Blüml, Bjarne Gregori, Sebastian Sztwiertnia, and Kristian Kersting. Ocatari: Object-centric atari 2600 reinforcement learning environments, 2024. URL https://arxiv.org/abs/2306.08649.

Gamaleldin F. Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael C. Mozer, and Thomas Kipf. Savi++: Towards end-to-end object-centric learning from real-world videos, 2022. URL https://arxiv.org/abs/2206.07764.

Fan Feng, Phillip Lippe, and Sara Magliacane. Learning interactive world model for object-centric reinforcement learning, 2025. URL https://arxiv.org/abs/2511.02225.

Stefano Ferraro, Pietro Mazzaglia, Tim Verbelen, and Bart Dhoedt. Focus: Object-centric world models for robotics manipulation, 2023. URL https://arxiv.org/abs/2307.02427.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

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

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2020. URL https://arxiv.org/abs/1912.01603.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models, 2022. URL https://arxiv.org/abs/2010.02193.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/cf73d57b6dcda32b293df7c2d5341f49-Paper-Conference.pdf.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning, 2017. URL https://arxiv.org/abs/1710.02298.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models, 2020. URL https://arxiv.org/abs/1911.12247.

Thomas Kipf, Gamaleldin F. Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video, 2022. URL https://arxiv.org/abs/2111.12594.

Jannik Kossen, Karl Stelzner, Marcel Hussing, Claas Voelcker, and Kristian Kersting. Structured object-aware physics prediction for video modeling and planning, 2020. URL https://arxiv.org/abs/1910.02425.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting, 2019. URL https://arxiv.org/abs/1904.00310.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention, 2020. URL https://arxiv.org/abs/2006.15055.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models, 2023. URL https://arxiv.org/abs/2209.00588.

Vincent Micheli, Eloi Alonso, and François Fleuret. Efficient world models with context-aware tokenization. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35623–35638. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/micheli24a.html.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. *Commun. ACM*, 61(5):103–115, April 2018. ISSN 0001-0782. doi: 10.1145/3191513. URL https://doi.org/10.1145/3191513.

Malte Mosbach, Jan Niklas Ewertz, Angel Villar-Corrales, and Sven Behnke. SOLD: Slot object-centric latent dynamics models for relational manipulation learning from pixels. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 44911–44935. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/mosbach25a.html.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks, 2022. URL https://arxiv.org/abs/1606.04671.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, and Francesco Locatello. Bridging the gap to real-world object-centric learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=b9tUk-f_aG.

Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate dall-e learns to compose, 2022a. URL https://arxiv.org/abs/2110.11405.

Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022b. Curran Associates Inc. ISBN 9781713871088.

Rishi Veerapaneni, John D. Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua B. Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning, 2020. URL https://arxiv.org/abs/1910.12827.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions, 2019. URL https://arxiv.org/abs/1901.01753.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks, 2018. URL https://arxiv.org/abs/1708.01547.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning, 2023. URL https://arxiv.org/abs/2310.09615.

Weipu Zhang, Adam Jelley, Trevor McInroe, Amos Storkey, and Gang Wang. Object-centric world models from few-shot annotations for sample-efficient reinforcement learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 91366–91394, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/93a4fc360b66fd23d6276156ca69eb50-Paper-Conference.pdf.

A NOTATION

Table 3: Notation used throughout the paper.

Symbol	Meaning
Regions and front end	
$x_{t,i}$	raw region i at step t
ϕ	frozen metric front end
$e_{t,i}$	region embedding
$f_{t,i}$	region feature, used for binding and reconstruction
Object types (Extractor)	
C	prototype matrix. Row c_j is object type j
P	prototype capacity
α	active-prototype mask
$\hat{c}$	proposed prototype at a birth
$r_{t,i}$	per-region reconstruction residual
τ_{twin}	novelty radius for a birth
$\bar{b}_j,\ \bar{e}_j$	presence rate, presence conditioned on reward
ρ_j	reward relevance
crit_j	seat score
m	frequency mixture weight
σ	seat map from type to graph node
$u_{t,i}$	per-region node assignment
$\mathcal{A}$	active prototypes
Graph and dynamics (Modeller)	
K	number of graph nodes
s_t^k	state of node k
R_t	relation matrix
$\hat{s}_{t+1}$	predicted next node states
$a_t,\ e(a_t)$	action and its embedding
Holistic state and policy (Integrator)	
(h_t, z_t)	holistic latent state (recurrent, stochastic)
$\pi,\ v$	actor and critic
ψ_t	policy input
g	per-node encoder
W_o	readout projection
Objective	
$\mathcal{L}_{\text{bind}}, \mathcal{L}_{\text{rec}}$	binding and reconstruction loss
$\mathcal{L}_{\text{sdyn}}, \mathcal{L}_{\text{srep}}$	node dynamics and representation loss
$\mathcal{L}_{\text{base}}$	holistic state loss
$\mathcal{L}_{\text{nodepred}}$	gradient-route node prediction loss

B BASELINES

Our five baselines span three families.

FOCUS (Ferraro et al., 2023) is an object-centric world model that decomposes each observation into a set of object slots and learns their dynamics, optimising behaviour in this slot-structured latent space. Its slots are tied to objects through segmentation masks, which we supply from the environment's ground truth, and the number of slots is fixed in advance. It is our representative object-centric comparison.

DreamerV3 (Hafner et al., 2024) learns a recurrent state-space world model from pixels and trains an actor and critic entirely in imagined rollouts of that model. It is the strongest general-purpose

holistic world model, running a single configuration across domains. It is our representative holistic comparison.

DreamerV2 (Hafner et al., 2022) is DreamerV3's predecessor, a holistic latent world model that encodes the scene as a discrete stochastic state and, like DreamerV3, optimises behaviour purely in imagination, with hyperparameters tuned per benchmark. It is another representative holistic comparison.

PPO (Schulman et al., 2017) is a classic and representative model-free, on-policy policy-gradient agent that optimises a clipped surrogate objective directly from environment interaction.

Rainbow (Hessel et al., 2017) is another classic model-free, value-based agent that combines several improvements to DQN, including double Q-learning, prioritised replay, duelling networks, multi-step returns, distributional value estimation, and noisy exploration.

C The Evolved Prototype Set

Figure 6 shows a visualisation of the prototype set in Figure 3. For each prototype, five random regions assigned to it are visualised, with the number indicating the purity of the assignment. The impurity of the initial set is visible directly, foreign regions mixed into the strips of water, stone, and coal, and the eight types it cannot name at all appear only on the evolved side.

D The Emergent Relation Matrix

The relation matrix of Section 3.2 is trained by prediction alone. We average it over the final 100k steps, name each node by the census of its assigned regions, and show it as a heatmap (Figure 7).

The relation that emerges is a dependency map centred on stone. Nearly every node attends to the stone node above the uniform rate, water, path, coal, and the crafting table most strongly, while grass, the most frequent type on screen, draws attention from no node but itself. Stone is the resource the tool tier runs through at this stage of the game, and the attention gathers on it rather than on what is merely common. This indicates that the learned relations track what progress depends on, rather than what appears most often.

E Imagined Rollouts

Figure 8 shows the visualised evaluation sequences of Section 4.5, the models warmed up on the same observations and rolled forward along the same actions. The boxes mark the ground-truth tiles of some important objects. In (a), EvoGraphWM keeps the coal and furnace in place at a long horizon, both types born during training, while the no-births ablation and DreamerV3 imagine bare stone, showing that births supply what the initial set lacks. In (b), EvoGraphWM holds the crafting table, another type born during training, to the end of the horizon while DreamerV3 and the no-births ablation erase it within five steps. In (c), EvoGraphWM and the no-births ablation both consistently predict the stone correctly during a long-horizon imagination. Stone is a type given at the beginning, while DreamerV3 replaces it with grass and water. Object nodes thus keep long-horizon imagination accurate, and the evolving graph does better still, staying closer to the ground truth than the no-births ablation beyond 40 steps.

F Implementation Details

Regions. A fixed grid cuts each frame into regions. On Crafter a 9×9 grid of 7×7-pixel cells coincides with the tile layout ($S = 81$ regions, of which $S_{\text{scene}} = 63$ show the world and 18 the inventory display). On Procgen a 16×16 grid of 6×6-pixel cells covers the 96×96 frame. The presence statistics, binding loss and prototype assignment run over the scene regions in each case. Two views of each region are used: the raw cell $x_{t,i}$ feeds the metric front end ϕ, and a 96-dimensional feature $f_{t,i}$, produced by the holistic encoder (a two-layer SiLU MLP with a learned position embedding) and read detached, feeds binding and reconstruction.

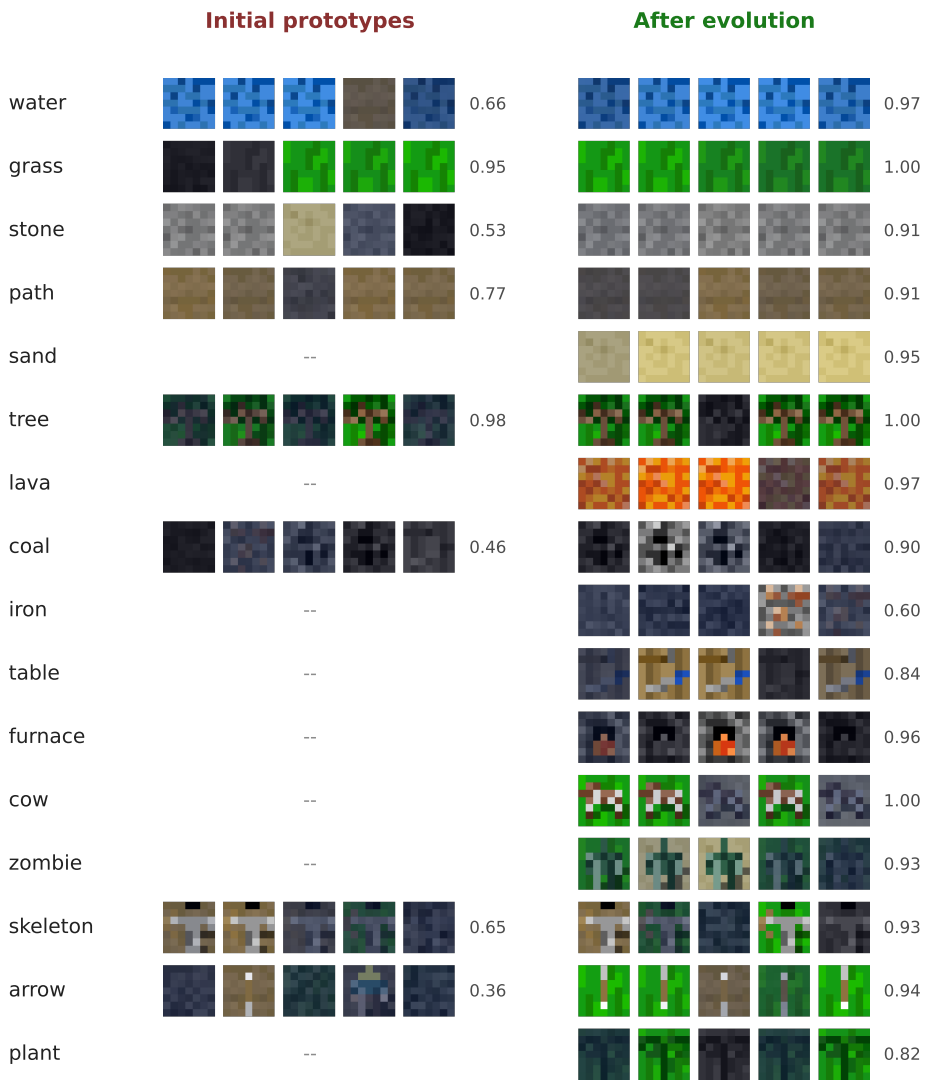


Figure 6: A visualisation of the prototype set. For each prototype, five random regions assigned to it are visualised, with the number indicating the purity of the assignment.

Metric front end. ϕ has three linear layers ($147 \to 256 \to 256 \to 64$) with SiLU after the first two, and the embedding $e_{t,i} = \phi(x_{t,i})$ is ℓ_2-normalised. It is trained by supervised contrastive learning on tile identities and then frozen. To verify the generalisation, we hold out two object types (iron and diamond) from its training entirely. The prototype set is initialised by farthest-point selection on a short rollout, which also sets $\tau_{\text{twin}} = Q_{0.90}$ of the nearest-prototype distance. It has 48 rows on Crafter (8 fitted, 40 reserved for births) and 25 on Procgen Heist (5 fitted, 20 reserved). All other hyperparameters, including the node budget K, are shared across domains.

Births. The k-means uses $k = 3$ clusters and $n_{\text{iter}} = 8$ iterations, and the winning centre is normalised before entering the prototype matrix.

Relevance statistics. With $w_t = \max(\text{reward}_{t+1}, 0)$ marking rewarding moments and $b_{t,j} \in \{0, 1\}$ indicating whether any region of frame t is assigned to type j, the per-batch statistics are $e_j = \sum_t w_t b_{t,j} / \sum_t w_t$ and $b_j = \frac{1}{T} \sum_t b_{t,j}$, with T the number of frames in the batch. Both are tracked as exponential averages,

$$\bar{e}_j \leftarrow (1 - \alpha)\, \bar{e}_j + \alpha\, e_j, \qquad \bar{b}_j \leftarrow (1 - \beta)\, \bar{b}_j + \beta\, b_j, \tag{15}$$

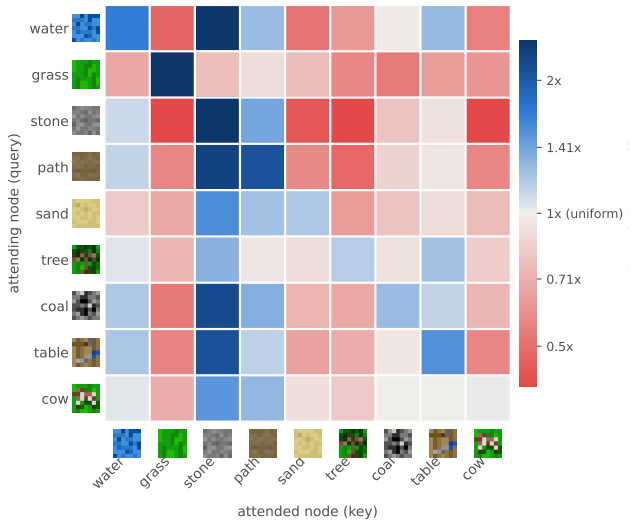


Figure 7: The emergent relation matrix. Colour is the attention a node pays to another, on a normalised log scale.

with $\bar{e}_j$ updated only on batches that carry reward mass, and relevance is $\rho_j = \max(\bar{e}_j - \bar{b}_j, 0)$. The subtraction of $\bar{b}_j$ inside ρ_j removes ubiquity from the estimate of relevance. The time constants $\alpha = 0.02$ and $\beta = 0.001$ give effective windows of roughly 50 and 1000 batches.

Revision safeguards. At a revision, a challenger takes a dedicated seat only when its score exceeds the incumbent's by a factor γ, which means that a smaller lead leaves the seat unchanged. A prototype within the duplicate radius τ_{twin} of a strictly more relevant one is barred from a dedicated node to avoid duplication.

Transition. The relational attention and the action-conditioned update of Section 3.2 run over the $K - 1$ dedicated nodes, with the shared node held out.

Reconstruction decoder. The spatial-broadcast decoder adds a learned position embedding to each slot and decodes a per-slot prediction with an alpha logit before the softmax composite.

Node prediction head. The gradient route maps the unrolled state $\hat{h}_{t+l}$ through a linear head sized $P \times d_s$, one output row per prototype. The sum runs over the active prototypes on dedicated seats, with the shared node excluded.

Behavioural readout. The per-node encoder g is a two-layer MLP (128-unit hidden, SiLU, RMS-normalised). Excluding the shared node, the pooled sum is projected by a bias-free, zero-initialised linear map W_o to 128 dimensions and appended to $[h_t, z_t]$. Notably, the measured magnitude ratio of the readout to the holistic contribution lies in $[0.87, 1.04]$, confirming the route is not silent.

G HYPERPARAMETERS

Table 4 lists the hyperparameters of the graph. Everything not listed, including the recurrent state size, the replay ratio, and the optimiser, follows the published 200M DreamerV3 configuration.

Table 4: Hyperparameters of the graph, shared across Crafter and Procgen except where a Procgen value is given in brackets.

Symbol	Meaning	Value
S / S_{scene}	regions per frame / scene regions	81 / 63 (256 / 256)
d_p	region dimension	147 (108)
P	prototype rows, fitted + dormant	8 + 40 (5 + 20)
K	number of nodes	12
d_s	node width	64
N_h	relation attention heads	4
H	prediction horizon	10
N	birth candidate pool	64
k	birth clusters	3
n_{iter}	iterations	8
m	seat mixture weight	0.5
γ	incumbent hysteresis	1.2
κ	dedicated-class CE boost	2.5
τ_{twin}	twin radius	$Q_{0.90}$ of init-fit distance
α, β	presence / reward EMA rates	0.02, 0.001
$M_{\min}$	birth mass floor	4.0
λ_*	bind / rec / sdyn / srep / nodepred / lay	1.0, 1.0, 0.5, 0.1, 1.0, 0.3

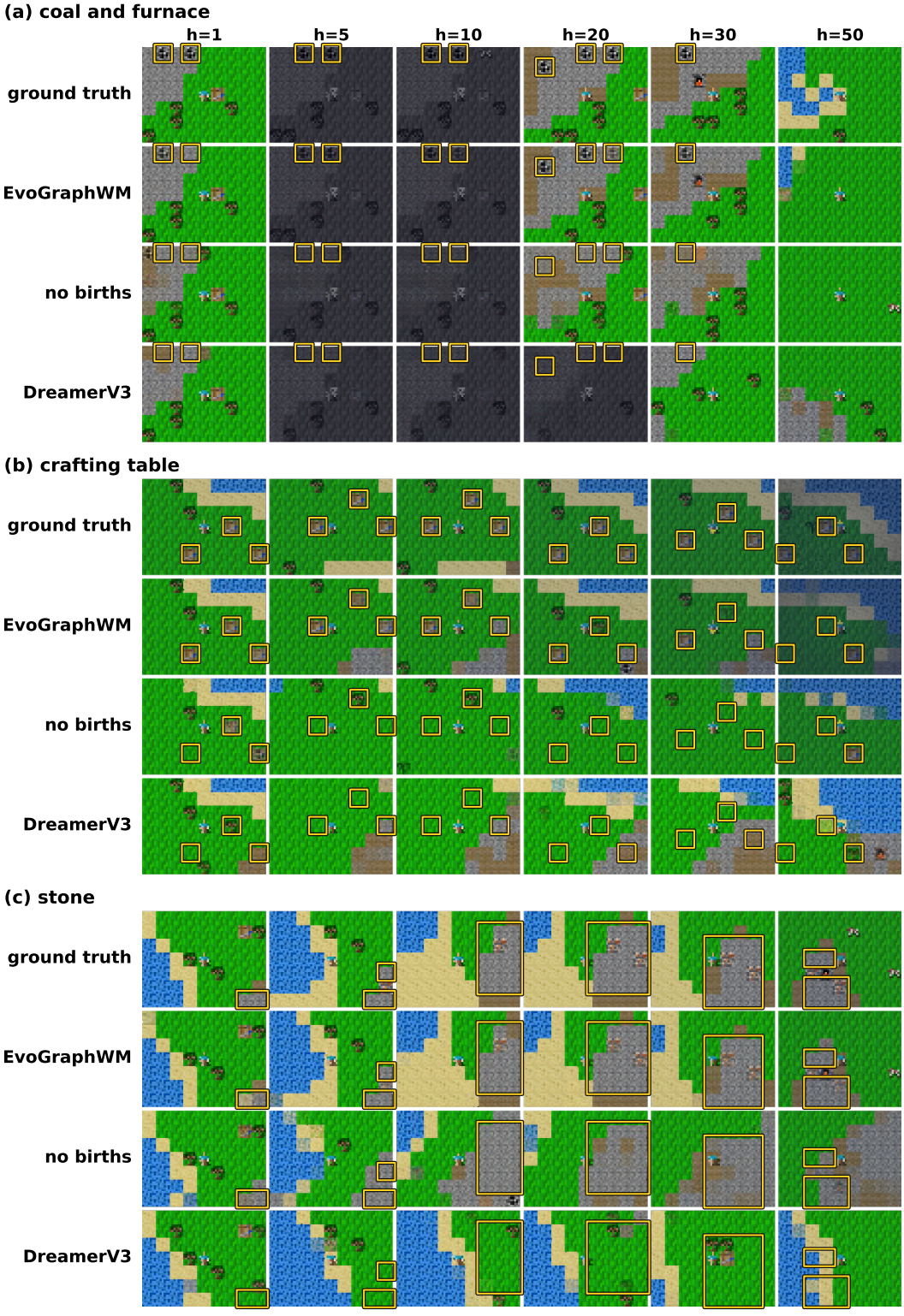


Figure 8: Imagined rollouts on three held-out sequences, decoded to pixels. Columns: imagination horizons $h \in \{1, 5, 10, 20, 30, 50\}$, labelled on the top block. Rows: the ground truth and the three models. Boxes mark the ground-truth tiles of some important objects, at the same positions in every row.