BEYOND STATIC GRAPH WORLD MODELS: LEARNING STOCHASTIC LATENT DYNAMICS OVER EVOLVING TOPOLOGIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph-based world models have recently emerged as a means of learning transitions over relational state representations. However, existing approaches are largely limited to fixed-topology graphs or deterministic, fully observable environments. We propose the Graph Dynamics Model (GDM), a world model for graph-structured observations that is designed to handle the more general setting of evolving topologies in stochastic and partially observable environments. The GDM uses a sparse recurrent adjacency matrix to model topology updates and perform message passing, together with a recurrent state-space architecture for modelling stochastic transitions. Furthermore, we identify a gap in the evaluation of graph-based world models, as existing methods do not provide a means of comparing predicted and true distributions over the joint graph state comprising the interdependent topology, node features, and graph features. We therefore introduce the Graph Distribution Distance (GDD) metric, which uses maximum mean discrepancy with a graph kernel to comprehensively compare joint next-state distributions. We evaluate the GDM across several environments, including stochastic and partially observable settings. We demonstrate that GDM outperforms baseline models and displays zero-shot generalisation on large graphs.

1 INTRODUCTION

World models have become popular as a means of learning environment dynamics for applications in reinforcement learning (RL) and planning, enabling reduced sample complexity, improved generalisation, and offline learning (Ha & Schmidhuber, 2018; Hafner et al., 2025). Most existing world model frameworks are designed for vector or pixel-based observations, which limits their applicability to environments with rich relational structures, such as graphs. Graph-based environments, where an agent interacts with a system comprising nodes, edges, and their respective features, are encountered in a number of contexts, including social networks (Meirom et al., 2021), navigation (Lu et al., 2021), and infrastructure (Amorosa et al., 2024). In many of these applications, the topology of the graph is variable, with edges being added or removed as the environment evolves. An advantage of graph-based models is the potential for zero-shot generalisation to larger environments, as a model trained on smaller graphs can then be applied to larger graphs where direct training is infeasible.

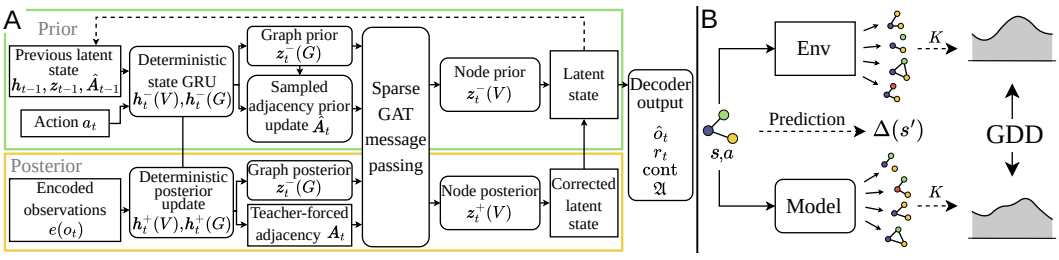


Figure 1: A) Architecture of the Graph Dynamics Model. B) Illustration of the GDD evaluation metric, which compares the predicted and true distributions of the next state.

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.

Despite the prevalence of graph-based environments, graph-based world models are still an emerging area of research. In environments with evolving topology, modelling becomes more difficult, as topology predictions and message passing are intertwined, and the model must learn to predict accurate topology in order to perform message passing effectively. As such, current graph-based world models are mostly limited to fixed topology (Feng et al., 2025; Chen et al., 2026), do not exploit the topology for message passing (Karacelebi et al., 2026), or are effective only in deterministic fully-observable settings (Song & Cai, 2026). We overcome these limitations by proposing the **Graph Dynamics Model (GDM)**, a world model for graph-structured observations with evolving topology, applicable to stochastic and partially observable environments. The GDM uses a sparse recurrent adjacency matrix, inspired by SparseGAT (Ye & Ji, 2021), with a recurrent state-space architecture (Hafner et al., 2019), in order to predict stochastic transitions in a latent space (Figure 1A). Topology prediction is one of the most important aspects of a graph-based world model, and the sharp adjacency updates facilitate accurate feature distribution predictions across the joint graph state.

There does not currently exist a standardised protocol for evaluating world models. In the absence of a downstream task, world models are typically evaluated based on the accuracy of their predictions compared to the environment. However, this is insufficient for stochastic environments, where the model must learn to capture the full distribution of state predictions rather than simply the mean. Furthermore, when the state consists of multiple factors, such as a graph environment with different node features, graph features, and topology, the evaluation must consider the full joint state rather than comparing factors independently. In this work, we propose a novel evaluation metric for next-step prediction of stochastic graph states based on maximum mean discrepancy (MMD) (Gretton et al., 2012). The evaluation metric, which we call the **Graph Distribution Distance (GDD)** metric (Figure 1B), captures the joint distribution of the graph topology, node features, and graph features, and is zero if and only if the predicted distribution matches the true distribution. This provides a principled way to compare predictions of graph-based world models in stochastic environments.

Our contributions can be summarised as follows. 1. We propose the Graph Dynamics Model (GDM), a world model for graph-structured observations with evolving topology, applicable to stochastic and partially observable environments. 2. We introduce a novel evaluation metric, GDD, for measuring world model quality using the full joint next-state distribution of graph topology, node features, and graph features, and validate it using per-factor distribution metrics. 3. We conduct a study of the proposed model and metric on a set of environments with varying characteristics at different graph sizes not encountered during training, demonstrating the GDM's ability to generalise to unseen graph sizes and outperform state-of-the-art baselines on multi-step rollouts. 4. We perform an ablation of the proposed model, demonstrating the importance of the recurrent adjacency sampling.

2 Related Work

World Models. In model-based RL, a world model is a model of the environment's transition function, which predicts the next state given the current state and action, commonly also including a reward and continuation signal prediction. The RSSM framework (Hafner et al., 2019; 2025) models environment dynamics in latent space through recurrent updates of deterministic and stochastic states. Earlier latent state-space approaches include the Deep Variational Bayes Filter (Karl et al., 2017), based on structured variational inference, and the Probabilistic Recurrent State-Space Model (Doerr et al., 2018), using Gaussian process dynamics. More recent approaches employ alternative representations for latent dynamics, including autoregressive transformer world models (Micheli et al., 2024; Agarwal et al., 2024), structured state-space sequence models (Mattes et al., 2024), and object-centric latent dynamics (Mosbach et al., 2025).

Graph-Based World Models. Graph-based world models apply the world model paradigm to graph-structured observations. Building on the action-node representation of Feng et al. (2025), Song & Cai (2026) study graph world models operating in state space, formulating GNN-based world models as $\sigma(\tilde{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}_1 + \boldsymbol{U}a\mathbf{1}^\top)\boldsymbol{W}_2$, with $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$, where $\boldsymbol{X}$ is the node feature matrix, $\boldsymbol{U}$, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are learned parameters, and $\boldsymbol{A}$ is predicted by a learned function of the pairwise node feature embeddings. Direct state space transitions can accurately capture deterministic dynamics, but are less effective in stochastic settings, where the model must capture the full distribution of next states rather than just the mean. Karacelebi et al. (2026) propose the G-RSSM model for learning ad-hoc network dynamics, which generalises the latent-space RSSM to graphs by

including a recurrent state vector for each node, being applicable to stochastic and partially observable environments. G-RSSM uses all-pairs message passing to update the latent state, rather than exploiting the underlying topology of the graph, only using a GNN to encode observations of the true state. Several other works have proposed graph-based world models for specific applications, such as structured origami folding (Huang et al., 2026), LEO satellite networks (Liu et al., 2026b), and structural dynamics in graphs (Wang et al., 2026). These approaches rely on domain-specific knowledge and are not applicable to general graph-based environments.

World Models on Learned Graphs. Several works integrate graph structures into world models where the observation space is not a graph, but structure can be inferred. Feng & Magliacane (2023) build a world model for object interactions, modelling object relationships using dynamic Bayesian networks. Dynamic Neural Relational Inference (Graber & Schwing, 2020) learns relations for each entity pair at each time step. In their survey paper, Liu et al. (2026a) highlight that existing graph world models do not adequately address topological plasticity and probabilistic modelling.

World Model Evaluation. World model evaluation procedures vary depending upon both the environment being modelled (e.g., deterministic versus stochastic) and the downstream task for which the model is intended. Many works rely on downstream planning performance as a measure of world model quality (Hafner et al., 2019; 2021; Yildiz et al., 2021; Chung et al., 2023), which can be computationally expensive. In the absence of a downstream task, world models are often evaluated using non-distributional metrics such as accuracy and MSE (Schiewer et al., 2024; Hao et al., 2025). The stochastic world model benchmark proposed in (Barsainyan et al., 2025) finds that "world models dramatically underestimate environmental variance". Distributional metrics for world model evaluation are common in video-generative applications (Gao et al., 2024; Bar et al., 2025), being much less common in model-based RL applications (Sedlmeier et al., 2022).

3 BACKGROUND

Graph-based POMDP Formulation. Let an attributed graph be $G = (V, E, \{x(v)\}_{v \in V}, x(G))$, comprising a set of nodes V, edges $E \subseteq V \times V$, node feature vectors $x(v) \in \mathcal{X}_n$, and a graph feature vector $x(G) \in \mathcal{X}_g$. Consider a tuple $\mathcal{P} = \langle S, \mathcal{A}, T, O, \Omega, R, H, \Delta s_0 \rangle$ defining a partially observable Markov decision process (POMDP) with state space S, action space $\mathcal{A}$, transition probability function $T : S \times \mathcal{A} \to \Delta S$, observation space O, observation probability function $\Omega : S \times \mathcal{A} \to \Delta O$, reward function $R : S \times \mathcal{A} \to \mathbb{R}$, horizon H, and initial state distribution $\Delta s_0 \in \Delta S$. We say that $\mathcal{P}$ is fully observable if $S = O$ and $\Omega(o \mid s') = \mathbb{I}[o = s']$, and partially observable otherwise. When O is a set of feature-equipped graphs, i.e., each $o \in O$ corresponds to some G, we call $\mathcal{P}$ a graph-based environment. We consider dynamic graphs with evolving topology, where the edge set E is gradually updated, in contrast to temporal graphs, where the topology is highly dynamic and instantaneous snapshots do not represent the overall structure (Skarding et al., 2021). Here we will assume that (V, E) is always fully observable, but the node and graph features may be partially observable. We define $\mathcal{A}$ to be a set of node selections. Complex actions such as edge additions or rewiring can be composed from multiple node selections. For a state $s \in S$, the action mask returns the set of valid nodes: $\mathfrak{A} : S \to 2^V$. The set of valid actions in state s is then $\mathcal{A}(s) = \{a \subseteq \mathfrak{A}(s) \mid |a| = k\}$, for some problem-defined k.

Maximum Mean Discrepancy. As we have argued, comparing distributions predicted by graph-based world models is essential in stochastic environments. However, this has not been studied in prior work. We build on the versatile MMD for this purpose, which is a kernel-based measure of the distance between two probability distributions P and Q defined on a common space $\mathcal{X}$ (Gretton et al., 2012). Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel, with associated reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$. Assuming the required expectations are finite, the MMD is defined as:

$$\mathrm{MMD}_K(P, Q) = \sup_{f \in \mathcal{H}_K : \|f\|_{\mathcal{H}_K} \leq 1} \left(\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{y \sim Q}[f(y)] \right). \tag{1}$$

Each distribution can be represented in $\mathcal{H}_K$ by its kernel mean embedding: $\mu_P = \mathbb{E}_{x \sim P}[K(x, \cdot)]$ and $\mu_Q = \mathbb{E}_{y \sim Q}[K(y, \cdot)]$, in which case $\mathrm{MMD}_K(P, Q) = \|\mu_P - \mu_Q\|_{\mathcal{H}_K}$. Using the reproducing property, its squared value can equivalently be written as:

$$\mathrm{MMD}_K^2(P, Q) = \mathbb{E}_{x, x' \sim P}[K(x, x')] + \mathbb{E}_{y, y' \sim Q}[K(y, y')] - 2\mathbb{E}_{\substack{x \sim P \\ y \sim Q}}[K(x, y)],$$

where x, x' are independent draws from P, y, y' are independent draws from Q, and the draws from P and Q are mutually independent. For i.i.d. samples $X = \{x_i\}_{i=1}^n \sim P^n$ and

$Y = \{y_j\}_{j=1}^m \sim Q^m$, the biased empirical estimator of the squared MMD is

$$\widehat{\mathrm{MMD}}_K^2(X,Y) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i'=1}^{n} K(x_i, x_{i'}) + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{m} K(y_j, y_{j'}) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} K(x_i, y_j). \quad (2)$$

A kernel K is characteristic if its kernel mean embedding $P \mapsto \mu_P$ is injective over the class of distributions under consideration, meaning that $\mu_P = \mu_Q$ implies $P = Q$. Thus, when K is characteristic, $\mathrm{MMD}_K(P, Q) = 0 \Leftrightarrow P = Q$ (Sriperumbudur et al., 2010).

Recurrent State-Space Models. The recurrent state-space model (RSSM) framework (Hafner et al., 2019) represents the environment's state in a latent space, composed of a *deterministic state* $\boldsymbol{h}_t$ and a *stochastic state* $\boldsymbol{z}_t$. In a forward pass, also called *imagination*, the deterministic state is updated recurrently, while the stochastic state is sampled from a learned *prior distribution* $p(\boldsymbol{z}_t \mid \boldsymbol{h}_t)$. During training, the *posterior update* $q(\boldsymbol{z}_t \mid \boldsymbol{h}_t, o_t)$ is used to correct the latent state based on the observation o_t, and the model is trained to minimise the KL divergence between p and q, as well as the reconstruction loss between the decoded latent state $\mathcal{D}(\boldsymbol{h}_t, \boldsymbol{z}_t)$ and o_t.

4 Graph Dynamics Model

4.1 Architecture

The GDM uses an RSSM-style architecture designed for dynamic graph-structured observations, as shown in Figure 1A. Rather than relying on a single global latent state, the model assigns a latent vector $\boldsymbol{h}(v)$ and $\boldsymbol{z}(v)$ for each node $v \in V$, as well as a latent vector $\boldsymbol{h}(G)$ and $\boldsymbol{z}(G)$ for the graph as a whole. Key to the model is the sampled recurrent adjacency matrix $\hat{\boldsymbol{A}}_t$, enabling message passing on the predicted topology and including it in the state, rather than only reconstructing it in the decoder as per G-RSSM (Karacelebi et al., 2026). While the stochastic components of the latent state $\boldsymbol{z}$ and $\hat{\boldsymbol{A}}$ are factorised, they are not independent, and the model captures their dependencies through conditional distributions. During imagination, the model predicts states using the prior dynamics, denoted $^-$, while during training, the model uses the posterior dynamics, denoted $^+$. Hereafter, the latent states $\boldsymbol{h}$ and $\boldsymbol{z}$ refer to the posterior state during posterior inference, and the prior state otherwise.

Encoder. The encoder produces embeddings from the observation o, factorised into node embeddings $\boldsymbol{e}_v(o)$ for each node v and a graph embedding $\boldsymbol{e}_g(o)$. A feature embedding is produced by passing each feature through a feature-specific learned linear transformation and taking the mean of the resulting embeddings.

Recurrent Dynamics and Posterior-Informed Deterministic State. For node v, let $\boldsymbol{a}_t(v)$ be its action embedding, produced by a linear transformation of the action set. The deterministic prior states are then updated recurrently:

$$\boldsymbol{h}_t^-(v) = \mathrm{LayerNorm}\left(\mathrm{GRU}_n\left(\boldsymbol{z}_{t-1}(v), \boldsymbol{a}_t(v) \mid \boldsymbol{h}_{t-1}(v)\right)\right),$$
$$\boldsymbol{h}_t^-(G) = \mathrm{LayerNorm}\left(\mathrm{GRU}_g\left(\boldsymbol{z}_{t-1}(G), \mathrm{Pool}_{v\in V}(\boldsymbol{a}_t(v)) \mid \boldsymbol{h}_{t-1}(G)\right)\right).$$

In an ordinary RSSM model, the observation is used to update only the stochastic component of the latent state, while the deterministic component is updated solely based on the previous state and the current input. While this model is effective for pixel-based observations, where there can be much redundancy in the observation, we found it to be less effective for dense observation features. As such, during training, the deterministic states are updated: $\boldsymbol{h}_t^+ = \boldsymbol{h}_t^- + \mathrm{sigmoid}\left(\boldsymbol{g}_t\right) \odot \Delta\boldsymbol{h}_t$, where

$$\Delta\boldsymbol{h}_t(v) \| \boldsymbol{g}_t(v) = \mathrm{MLP}_n^+\left(\boldsymbol{h}_t^-(v), \boldsymbol{e}_v(o_t), \boldsymbol{e}_g(o_t)\right), \quad \Delta\boldsymbol{h}_t(G) \| \boldsymbol{g}_t(G) = \mathrm{MLP}_g^+\left(\boldsymbol{h}_t^-(G), \boldsymbol{e}_g(o_t), \mathrm{Pool}_{v\in V}(\boldsymbol{e}_v(o_t))\right).$$

Adjacency Prediction and Message Passing. The graph stochastic state is sampled from a categorical distribution conditioned on the graph latent state:

$$\boldsymbol{z}_t^-(G) \sim p_g(\boldsymbol{z}_t(G) \mid \boldsymbol{h}_t^-(G), \mathrm{Pool}_{v\in V}(\boldsymbol{h}_t^-(v))), \quad \boldsymbol{z}_t^+(G) \sim q_g(\boldsymbol{z}_t(G) \mid \boldsymbol{h}_t^+(G), \mathrm{Pool}_{v\in V}(\boldsymbol{h}_t^+(v)), \boldsymbol{e}_g(o_t)).$$

We apply Unimix of 1% to all categorical sampling as per Hafner et al. (2025). The adjacency matrix $\hat{\boldsymbol{A}}_t$ is predicted using an update network, predicting edge additions and removals under the assumption of incremental changes. For each ordered node pair (i, j), we calculate the edge update

logit $\ell_{ij}^{\Delta\text{adj}} = \text{MLP}_{\text{adj}}\Big(\boldsymbol{h}_t^-(i), \boldsymbol{h}_t^-(j), \boldsymbol{h}_t^-(G), \boldsymbol{z}_t^-(G), \hat{\boldsymbol{A}}_{t-1}(i,j)\Big)$. We then apply Hard Concrete sampling (Louizos et al., 2018) to obtain a differentiable sample:

$$\tilde{\ell}_{ij} = \ell_{ij}^{\Delta\text{adj}} + \beta \log\left(\frac{-\gamma}{\zeta}\right), \qquad s_{ij} = \text{sigmoid}\left(\frac{\log u_{ij} - \log(1-u_{ij}) + \tilde{\ell}_{ij}}{\beta}\right), \qquad u_{ij} \sim \text{U}(0,1),$$

with Hard Concrete gates given by $c_{ij} = \text{clip}_{[0,1]}\left(s_{ij}(\zeta-\gamma)+\gamma\right)$. The adjacency matrix is updated as $\hat{\boldsymbol{A}}_t(i,j) = \hat{\boldsymbol{A}}_{t-1}(i,j) \operatorname{xor} \mathbf{1}[c_{ij} > 0]$. The corresponding differentiable next-edge weight is given by $b_{ij} = (1 - \hat{\boldsymbol{A}}_{t-1}(i,j))c_{ij} + \hat{\boldsymbol{A}}_{t-1}(i,j)(1-c_{ij})$. To preserve the hard topology in the forward pass while allowing downstream losses to differentiate through every edge update, we define the straight-through next-edge weight $\bar{b}_{ij} = b_{ij} + \text{sg}\Big(\hat{\boldsymbol{A}}_t(i,j) - b_{ij}\Big)$, where sg denotes stop-gradient. We then perform SparseGAT-style message passing on $\hat{\boldsymbol{A}}_t$, with normalised weights derived from $\bar{b}_{ij}$ being shared across all attention heads and layers per Ye & Ji (2021). We inject $[\boldsymbol{h}_t(G), \boldsymbol{z}_t(G)]$ into the first layer, and apply a learned root connection followed by layer normalisation:

$$\boldsymbol{m}_t^{(\ell+1)} = \text{LayerNorm}\left(\boldsymbol{W}_{\text{root}}^{(\ell)} \boldsymbol{m}_t^{(\ell)} + \text{SparseGAT}^{(\ell)}\right), \qquad \boldsymbol{m}_t^{(0)} = \boldsymbol{h}_t.$$

On the posterior path, $\hat{\boldsymbol{A}}_t$ is teacher-forced to the ground truth adjacency matrix from the observation, with $b_{ij} = \boldsymbol{A}_t(i,j)$. Finally, the node stochastic state is sampled from a categorical distribution:

$$\boldsymbol{z}_t^-(v) \sim p_n(\boldsymbol{z}_t(v) \mid \boldsymbol{m}_t^{(L)-}(v), \boldsymbol{z}_t^-(G)), \qquad \boldsymbol{z}_t^+(v) \sim q_n(\boldsymbol{z}_t(v) \mid \boldsymbol{m}_t^{(L)+}(v), \boldsymbol{z}_t^+(G), \boldsymbol{e}_v(o_t)).$$

Decoder. The decoder is used to predict the observation from the current latent state. The decoder is factorised into a mean component θ and a stochastic innovation component I:

$$\begin{aligned}
\mathcal{D}_g(\boldsymbol{h}_t^-(G), \boldsymbol{z}_t(G)) &= \theta(\boldsymbol{h}_t^-(G)) + I\big(\boldsymbol{h}_t^-(G), \boldsymbol{z}_t(G) - \mathbb{E}_{p_g(\boldsymbol{z}_t(G)\mid \boldsymbol{h}_t^-(G))}[\boldsymbol{z}_t(G)]\big), \\
\mathcal{D}_v(\boldsymbol{h}_t^-(v), \boldsymbol{z}_t(v)) &= \theta(\boldsymbol{h}_t^-(v)) + I\big(\boldsymbol{h}_t^-(v), \boldsymbol{z}_t(v) - \mathbb{E}_{p_n(\boldsymbol{z}_t(v)\mid \boldsymbol{m}_t^{(L)-}(v), \boldsymbol{z}_t^-(G))}[\boldsymbol{z}_t(v)]\big).
\end{aligned} \tag{3}$$

Here, the prior deterministic state $\boldsymbol{h}^-$ is always used, ensuring that the decoder does not bypass the stochastic state through the observation-informed $\boldsymbol{h}^+$. This formulation provides a clear separation between the deterministic and stochastic components of the model's predictions. The shared state produced by $\mathcal{D}$ is then passed to feature-specific heads to predict the observed node and graph features, and the action mask, reward, and continuation signal.

4.2 Training

The GDM is trained to minimise the standard RSSM loss, with an additional $\boldsymbol{h}$ alignment term:

$$\mathcal{L} = \beta_{\text{pred}}\mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{KL}} + \beta_h \mathcal{L}_h. \tag{4}$$

The prediction loss is the average over all feature prediction terms:

$$\mathcal{L}_{\text{pred}} = \frac{1}{|V||\mathcal{F}_n|}\sum_{v\in V}\sum_{f\in\mathcal{F}_n}\ell_f^{(n)} + \frac{1}{|\mathcal{F}_g|}\sum_{f\in\mathcal{F}_g}\ell_f^{(g)} + \ell_{\text{adj}} + \ell_{\text{action}} + \ell_{\text{reward}} + \ell_{\text{continue}},$$

where ℓ_f is MSE for scalar features and CE for categorical features, ℓ_{adj} is BCE on $\hat{\boldsymbol{A}}$, ℓ_{action} is BCE on the action mask, ℓ_{reward} is MSE on the reward, and ℓ_{continue} is BCE on the continuation signal. The KL loss is the weighted sum of the node KL, graph KL, and adjacency KL: $\mathcal{L}_{\text{KL}} = \beta_n\mathcal{L}_{\text{KL}}^n + \beta_g\mathcal{L}_{\text{KL}}^g + \beta_{\text{adj}}\mathcal{L}_{\text{KL}}^{\text{adj}}$. For both $\mathcal{L}_{\text{KL}}^n$ and $\mathcal{L}_{\text{KL}}^g$, we separate the loss into a weighted sum of dynamics KL and a representation KL (Hafner et al., 2025): $\beta_{\text{dyn}}\left[D_{\text{KL}}\left(\text{sg}(q)\,\|\,p\right)\right]_{\text{FB}} + \beta_{\text{rep}}\left[D_{\text{KL}}\left(q\,\|\,\text{sg}(p)\right)\right]_{\text{FB}}$, where $[x]_{\text{FB}} = \max(x, \lambda_{\text{FB}})$ is the free-bits floor. For adjacency, using posterior edge probability q_{ij} (teacher-forced) and prior edge probability p_{ij},

$$\mathcal{L}_{\text{KL}}^{\text{adj}} = \frac{1}{|V|^2}\sum_{i,j\in V}\left(q_{ij}\log\frac{q_{ij}}{p_{ij}} + (1-q_{ij})\log\frac{1-q_{ij}}{1-p_{ij}}\right), \text{ where } p_{ij} = \begin{cases} \text{sigmoid}\Big(\ell_{ij}^{\Delta\text{adj}}\Big), & \hat{\boldsymbol{A}}_{t-1}(i,j) = 0, \\ 1 - \text{sigmoid}\Big(\ell_{ij}^{\Delta\text{adj}}\Big), & \hat{\boldsymbol{A}}_{t-1}(i,j) = 1. \end{cases}$$

Finally, we encourage the deterministic prior and posterior states to be aligned:

$$2\mathcal{L}_h = \frac{1}{|V|}\sum_{v\in V}\left(\left\|\boldsymbol{h}^-(v) - \text{sg}(\boldsymbol{h}^+(v))\right\|^2 + \left\|\boldsymbol{h}_t^+(v) - \boldsymbol{h}_t^-(v)\right\|^2\right) + \left\|\boldsymbol{h}^-(G) - \text{sg}(\boldsymbol{h}^+(G))\right\|^2 + \left\|\boldsymbol{h}_t^+(G) - \boldsymbol{h}_t^-(G)\right\|^2.$$

5 GRAPH DISTRIBUTION DISTANCE

We propose using MMD in order to evaluate the quality of a world model in a stochastic environment with joint distributions over topology, node features, and graph features. A whole-graph metric such as MMD is preferable to evaluating the distribution of individual factors separately, as it captures dependencies between the factors and provides a more complete measure of the model's performance. Using a characteristic kernel, the MMD is 0 only when the predicted distribution matches the true distribution, so we can determine when a model is truly capturing the full joint dynamics of the transition function. While previous works have defined graph kernels for use in MMD (Borgwardt et al., 2005; 2007), these kernels are not characteristic, and do not consider all feature types. Therefore, we construct a new characteristic kernel for attributed graphs and use it to define GDD.

Suppose that the node features can be partitioned into a set of categorical features indexed by $\mathcal{F}_n^\delta$ and a set of continuous features indexed by $\mathcal{F}_n^\mathcal{C}$, and similarly for the graph features with index sets $\mathcal{F}_g^\delta$ and $\mathcal{F}_g^\mathcal{C}$. For a finite index set I of continuous features, define

$$D_I^\mathcal{C}(x, x') = \frac{1}{|I|} \sum_{f \in I} \frac{|x_f - x'_f|^2}{\sigma_f^2}, \qquad \sigma_f > 0,$$

where x_f is the value of feature f and σ_f is a fitted bandwidth. For continuous node and graph features, we use the multi-bandwidth Radial Basis Function (RBF) kernel:

$$K_n^\mathcal{C}(G, G') = \frac{1}{|\mathcal{M}|} \sum_{r \in \mathcal{M}} \exp\left(-\frac{\sum_{v \in V} D^\mathcal{C}_{\mathcal{F}_n^\mathcal{C}}(x(v), x'(v))}{2|V|r^2}\right), \;\; K_g^\mathcal{C}(G, G') = \frac{1}{|\mathcal{M}|} \sum_{r \in \mathcal{M}} \exp\left(-\frac{D^\mathcal{C}_{\mathcal{F}_g^\mathcal{C}}(x(G), x(G'))}{2r^2}\right)$$

where $0 < r < \infty$ for every r in the bandwidth multiplier set $\mathcal{M}$. For a finite index set I of categorical features, we use the weighted Hamming distance

$$D_I^\delta(x, x') = \frac{1}{|I|} \sum_{f \in I} \lambda_f \mathbf{1}[x_f \neq x'_f], \qquad \lambda_f > 0.$$

The node and graph categorical kernels are thus defined:

$$K_n^\delta(G, G') = \exp\left(-\frac{1}{|V|} \sum_{v \in V} D^\delta_{\mathcal{F}_n^\delta}(x(v), x'(v))\right), \qquad K_g^\delta(G, G') = \exp\left(-D^\delta_{\mathcal{F}_g^\delta}(x(G), x(G'))\right).$$

Where no features are present for a given type, we define $D_\mathcal{F}(G, G') = 0$. For the topology, we use the Jaccard kernel

$$K_J(G, G') = \frac{|E_G \cap E_{G'}|}{|E_G \cup E_{G'}|}, \quad \text{where } K_J(\emptyset, E) = 0 \text{ when } E \neq \emptyset \text{ and } K_J(\emptyset, \emptyset) = 1.$$

Finally, the joint kernel is defined as:

$$K_{\text{joint}}(G, G') = K_J(G, G') K_n^\mathcal{C}(G, G') K_n^\delta(G, G') K_g^\mathcal{C}(G, G') K_g^\delta(G, G'). \tag{5}$$

Lemma 1 *Assume that all graphs have the same finite, aligned node set. Assume also that the node and graph feature index sets are finite and that every categorical feature has a finite value space. Then K_{joint} is a characteristic kernel on this space of attributed graphs.* (Proof in Appendix A.1).

As the joint kernel is characteristic, the kernel can distinguish between different probability distributions over the space of attributed graphs. However, in the joint kernel formulation, any arbitrarily poor factor can drive the entire kernel toward 0, causing MMD K_{joint} to be dominated by that factor, compressing the range of the metric. As such, in order to have a useful metric for distinguishing both good and bad predictions, we are motivated to include additive terms which consider the factors independently. We define individual kernels for continuous and categorical features:

$$k_f^\mathcal{C}(x, x') = \frac{1}{|\mathcal{M}|} \sum_{r \in \mathcal{M}} \exp\left(-\frac{|x - x'|^2}{2r^2\sigma_f^2}\right), f \in \mathcal{F}_n^\mathcal{C} \cup \mathcal{F}_g^\mathcal{C}, \qquad k_f^\delta(x, x') = \exp\left(-\lambda_f \mathbf{1}[x \neq x']\right), f \in \mathcal{F}_n^\delta \cup \mathcal{F}_g^\delta.$$

We then define the additive node and graph kernels as

$$K^n_{\text{add}}(G, G') = \frac{1}{|V||\mathcal{F}_n|} \sum_{v \in V} \sum_{f \in \mathcal{F}_n} k_f(x_f(v), x'_f(v)), \;\; K^g_{\text{add}}(G, G') = \frac{1}{|\mathcal{F}_g|} \sum_{f \in \mathcal{F}_g} k_f(x_f(G), x'_f(G')).$$

For the additive topology term, we again use K_J. The full additive graph kernel is

$$K_{\text{add}}(G, G') = w_n K^n_{\text{add}}(G, G') + w_g K^g_{\text{add}}(G, G') + w_e K_J(G, G'),$$

where $w_n, w_g, w_e \geq 0$ control the relative importance of the three terms. Finally, the factorised graph kernel is

$$K(G, G') = K_{\text{add}}(G, G') + \epsilon K_{\text{joint}}(G, G'), \qquad \epsilon > 0. \tag{6}$$

Lemma 2 *Under the assumptions of Lemma 1, K is a characteristic kernel on the space of attributed graphs.* (Proof in Appendix A.2).

Using the factorised graph kernel, we define the GDD metric as the MMD between the predicted and ground truth graph distributions:

$$\text{GDD}(P, Q) = \text{MMD}_K(P, Q). \tag{7}$$

The corresponding squared GDD decomposes as

$$\text{GDD}^2 = w_n \,\text{MMD}^2_{K^n_{\text{add}}} + w_g \,\text{MMD}^2_{K^g_{\text{add}}} + w_e \,\text{MMD}^2_{K_J} + \epsilon \,\text{MMD}^2_{K_{\text{joint}}}.$$

Thus the additive terms retain sensitivity to errors in individual factors, while any $\epsilon > 0$ preserves distribution distinguishing. We set $w_n = w_g = w_e = 1/3$ and $\epsilon = 1$, providing equal weighting between the additive and joint terms, and between the three additive terms. In practice we use the biased MMD approximation in Equation (2) to calculate the GDD from sampled distributions. We demonstrate in Appendix F.1 that in-distribution samples produce a near-zero GDD estimate.

6 Experiments

Baselines. From Song & Cai (2026), we use the Error-Aware GWM (EA-GWM) as a baseline, which models deterministic transitions directly in the state space, and uses several regularising terms to mitigate multi-step rollout error. We also consider a Vanilla GWM baseline, which is EA-GWM without the regularisation terms. We also include G-RSSM (Karacelebi et al., 2026), which models stochastic transitions in latent space and uses all-pairs message passing to update the latent state rather than relying on a predicted adjacency matrix. For all baselines we adapt the encoders and decoders to the environments. Full adaptation details are provided in Appendix C.

Environments. We consider four evolving-topology environments with different levels of stochasticity and observability. We defer full environment descriptions to Appendix B.

Robust Graph Construction (RGC). Given an initial graph, the agent selects node pairs to construct new edges between, in order to minimise the probability of disconnection under node removals (Darvariu et al., 2021). The environment is deterministic and fully observable.

Opinion Dynamics (OD). The agent spreads a target opinion in an evolving network by selecting a set of nodes to try to activate at each step. At each step, a subset of nodes chosen by the environment may either be influenced by a neighbour or relocate an incident edge to a different neighbour, based on Holme & Newman (2006). The environment is stochastic and fully observable.

Search and Rescue (SAR). The agent must find and rescue as many survivors as possible by navigating a fragile network and transporting survivors to an exit node. Each node has a stability score, and movement through the node has a chance to reduce the stability, causing incident edges to collapse when below a threshold. The environment is stochastic and partially observable.

Cascading Failures (CF). Overloaded nodes in a network fail stochastically, causing incident edges to collapse and redistribute the load to neighbours (Motter & Lai, 2002). The agent accelerates this process by selecting a node to disconnect at each step. The safety margin of the nodes is determined by a random unobserved variable, making the environment stochastic and partially observable.

Evaluation Procedure. The primary evaluation metric is the mean per-episode GDD of the model's predictions under random rollouts, which we refer to as the GDD score. Full evaluation details are provided in Appendix D. We note here that the GDD score is dependent on the environment's features and is not inherently size-invariant, so it is not directly comparable across environments or graph sizes. However, it is a useful metric for evaluating the quality of a model's distributions, accounting for variance and capturing the dependence between node, graph, and topology features.

Training Details. For each model and environment, we train for 5000 epochs on a set of pre-collected rollouts of a random policy on a set of graphs with sizes in $\{15, 18, 20, 23, 25\}$. Hyperparameters were chosen from a grid search based on the mean GDD score achieved on a validation set consisting of 20 graphs of each size 20 and 30. Full training details are provided in Appendix E.2.

6.1 Size Generalisation for Next-State Distribution Predictions

We evaluate the size generalisation of each model by calculating the single-step GDD score on test sets of different sizes, with 20 nodes being within the training distribution, 30 nodes being only in the validation set, and all larger sizes being completely outside the training distribution. The results in Table 1 demonstrate that the GDM outperforms or matches all baselines on the stochastic environments. We do not evaluate RGC on large graphs due to episode lengths quadratic in $|V|$.

Table 1: GDD score (↓) for single-step predictions on test sets of different sizes (10 seeds).

		$\lvert V\rvert = 20$	$\lvert V\rvert = 30$	$\lvert V\rvert = 50$	$\lvert V\rvert = 100$	$\lvert V\rvert = 200$	$\lvert V\rvert = 500$	$\lvert V\rvert = 1000$
RGC	GDM	**0.950±0.007**	**0.880±0.012**	**1.010±0.021**	**1.125±0.054**	1.175 ±0.065	–	–
RGC	Vanilla GWM	1.168 ±0.002	1.158 ±0.003	1.156 ±0.004	1.161 ±0.005	**1.167±0.006**	–	–
RGC	EA-GWM	1.167 ±0.002	1.161 ±0.005	1.160 ±0.008	1.170 ±0.007	1.177 ±0.006	–	–
RGC	G-RSSM	1.226 ±0.009	1.229 ±0.007	1.254 ±0.021	1.283 ±0.023	1.300 ±0.018	–	–
OD	GDM	**0.449±0.042**	**0.426±0.041**	**0.402±0.037**	**0.413±0.027**	**0.492±0.022**	**0.650±0.018**	**0.766±0.014**
OD	Vanilla GWM	0.916 ±0.005	0.913 ±0.004	0.930 ±0.005	0.972 ±0.007	0.995 ±0.007	1.016 ±0.007	1.025 ±0.007
OD	EA-GWM	0.920 ±0.011	0.920 ±0.009	0.935 ±0.010	0.973 ±0.010	0.994 ±0.010	1.013 ±0.009	1.023 ±0.009
OD	G-RSSM	1.034 ±0.002	1.024 ±0.002	1.027 ±0.002	1.044 ±0.001	1.053 ±0.001	1.062 ±0.001	1.068 ±0.002
SAR	GDM	**0.917±0.023**	**0.917±0.028**	**0.928±0.026**	**0.887±0.032**	**0.878±0.035**	**0.886±0.038**	**0.882±0.033**
SAR	Vanilla GWM	1.122 ±0.004	1.131 ±0.003	1.150 ±0.003	1.180 ±0.007	1.202 ±0.008	1.224 ±0.007	1.227 ±0.006
SAR	EA-GWM	1.126 ±0.006	1.133 ±0.005	1.152 ±0.004	1.183 ±0.007	1.205 ±0.008	1.226 ±0.007	1.227 ±0.006
SAR	G-RSSM	1.139 ±0.005	1.164 ±0.005	1.196 ±0.005	1.232 ±0.005	1.256 ±0.005	1.286 ±0.015	1.302 ±0.025
CF	GDM	**0.914±0.038**	**0.890±0.043**	**1.059±0.034**	**1.222±0.033**	**1.249±0.034**	**1.261±0.056**	**1.261±0.063**
CF	Vanilla GWM	1.179 ±0.002	1.198 ±0.003	1.212 ±0.004	1.227 ±0.006	**1.249±0.010**	1.271 ±0.013	1.277 ±0.014
CF	EA-GWM	1.171 ±0.002	1.193 ±0.003	1.211 ±0.005	1.233 ±0.009	1.259 ±0.013	1.284 ±0.017	1.292 ±0.018
CF	G-RSSM	1.240 ±0.010	1.277 ±0.010	1.363 ±0.042	1.436 ±0.053	1.470 ±0.060	1.485 ±0.066	1.493 ±0.069

We validate the GDD score by analysing the per-factor distribution metrics in Appendix F.3. The GDD score is consistent with the trend of these metrics but is also sensitive to the joint distribution of all factors, which is not captured by factor-wise metrics. GDM consistently demonstrates extremely low BCE on the adjacency matrix, owing to the explicit recurrent adjacency matrix. Accurate topology prediction is arguably the key component of a graph-based world model, as feature predictions are meaningless without an understanding of the underlying network.

6.2 Multi-Step Rollouts

We calculate the GDD score for rollouts of 2–5 steps on 50-node graphs in Figure 2, with full details in Appendix F.4. GDM consistently outperforms the baselines. Interestingly, there is little difference between Vanilla GWM and EA-GWM, suggesting that the regularisation terms in EA-GWM do not significantly improve multi-step rollout performance in these environments.

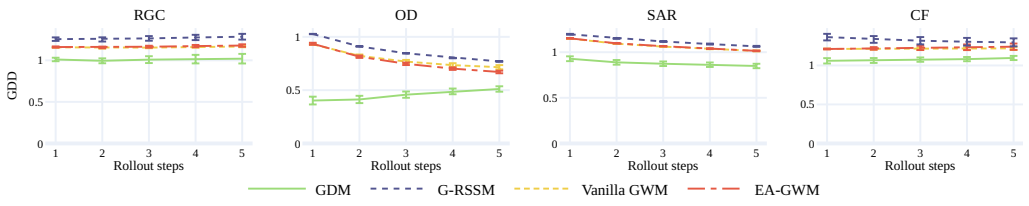


Figure 2: Multi-step rollout GDD on 50-node graphs (↓).

6.3 Ablations

We perform an ablation study to evaluate the contribution of each component of the GDM model. We test the contribution of the separate graph state by removing $\boldsymbol{h}(G)$ and $\boldsymbol{z}(G)$. We test the effect of setting β_h to 0, which removes the loss term drawing $\boldsymbol{h}^+$ and $\boldsymbol{h}^-$ together. We ablate the $\boldsymbol{h}^+$ posterior correction by using only the prior $\boldsymbol{h}^-$, aligning the model with the RSSM framework. We investigate the effect of the Hard Concrete sampling and SparseGAT message passing by using Bernoulli sampling for the adjacency update, and all-pairs message passing weighted by adjacency probabilities, respectively. We also test the effect of removing the recurrent adjacency, instead using all-pairs message passing and only calculating $\hat{\boldsymbol{A}}$ at the decoder output, as per G-RSSM.

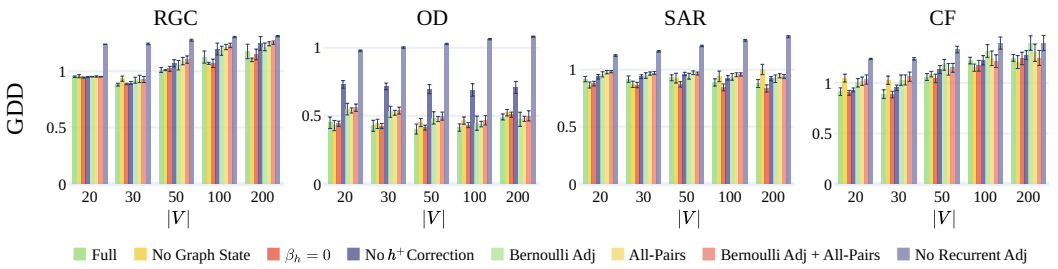


Figure 3: GDD scores for each ablation (↓).

Figure 3 shows the GDD score at different test sizes for each ablation on the environments, with further results in Appendix F.5. The removal of the graph component has variable effects across different environments. Removing the β_h loss term may slightly improve performance in particular environments. The posterior correction has a significant effect in the OD environment but less in others, likely due to the random node feature in the OD environment which is difficult to encode in a sampled categorical latent variable. The recurrent adjacency has a significant effect on the GDD across all environments. The use of SparseGAT message passing and Hard Concrete sampling is more important in some environments than others. Thus, we can conclude that the recurrent adjacency matrix is a key contributor to quality distribution predictions for evolving graphs.

7 Limitations and Future Work

While the aim of the GDM model is to predict the next-state distribution, there is some mismatch between the training objective, which considers reconstruction error in expectation, and the evaluation metric, which accounts for the full variance. Training the model to minimise a distributional score directly would require sampling multiple rollouts for each training transition. A limitation of GDM, which is shared with the other approaches considered in this work, is the independent sampling of edges. For some environments where edge events are inherently dependent, the model cannot accurately capture the dynamics, even if the probability of each edge is accurate. For the edge updates, GDM requires logit calculations for every pair of nodes, limiting the scalability. The model also does not include edge features, which restricts the environments to which GDM can be applied. Future work could explore the inclusion of edge features without $\mathcal{O}(|V|^2)$ computation.

A limitation of MMD is that it is kernel-dependent, reducing comparability across environments. MMD also scales quadratically with the number of samples, which can be limiting for multi-step rollouts. In addition, a characteristic kernel compresses the range of bad predictions, necessitating the addition of an extra term to the GDD metric to make it more informative, introducing an arbitrary weighting between the two terms. Lastly, the GDD score represents the ability of a model to predict next-state distributions, but does not necessarily reflect the model's ability to support downstream tasks. In future work we plan to explore the relationship between GDD and performance on downstream tasks, such as reinforcement learning, to better understand the utility of the metric.

8 Conclusion

In this work, we have introduced the GDM, a graph-based world model for stochastic and partially observable environments with evolving topologies, and the GDD metric for evaluating the quality of predicted next-state distributions. The GDM performs message passing on a sparse recurrent adjacency matrix and integrates a recurrent state-space model to capture stochastic transitions, enabling the modelling of stochastic dynamics in latent space with accurate topology predictions. The GDD metric uses MMD with a characteristic graph kernel to compare predicted and true next-state distributions, capturing dependencies between topology, node features, and graph features, and provably distinguishing between different distributions. We have demonstrated that the GDM outperforms the baselines in the GDD metric across a range of environments and graph sizes outside of the training distribution. Future work could explore the inclusion of edge features, and the relationship between GDD and performance on downstream tasks such as reinforcement learning.

AI USE STATEMENT

In this work, we used generative AI tools to provide critical ingredients for proving mathematical claims, assist in the writing of proofs, design or provide feedback on research methodology or experiments, and implement methods. We have not used generative AI tools to formulate mathematical claims, help develop theoretical models or conceptual frameworks, propose or refine hypotheses, interpret results, generate synthetic data sets, or clean and reformat datasets. Assisting with translation and supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create or edit software code, summarize or analyse existing literature, brainstorm, and identify relevant literature. We have reviewed all AI-assisted work. All LLM-generated code was verified and tested for correctness. All mathematical claims and proofs have been reviewed by a qualified researcher. Methodological feedback was verified by a manual literature review. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All code and data required to reproduce the results of this paper will be uploaded anonymously as supplementary materials and will be released publicly upon publication. GNN implementations use PyTorch Geometric (Fey & Lenssen, 2019). Models were trained on a cluster using GPUs with at least 16GB of memory.

REFERENCES

Pranav Agarwal, Sheldon Andrews, and Samira Ebrahimi Kahou. Learning to play Atari in a World of Tokens. In *ICML*, 2024.

Lorenzo Mario Amorosa, Marco Skocaj, Roberto Verdone, and Deniz Gündüz. Multi-Agent Reinforcement Learning for Power Control in Wireless Networks via Adaptive Graphs. In *IEEE International Conference on Communications*, 2024.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation World Models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Aryan Amit Barsainyan, Jing Yu Lim, and Dianbo Liu. STORI: A Benchmark and Taxonomy for Stochastic Environments. *arXiv preprint arXiv:2509.01793*, 2025.

K. M. Borgwardt, C. S. Ong, S. Schonauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein Function Prediction via Graph Kernels. *Bioinformatics*, 21(Suppl 1):i47–56, 2005.

Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J. Smola. Integrating Structured Biological Data by Kernel Maximum Mean Discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

Karsten M. Borgwardt, Hans-Peter Kriegel, S. V. N. Vishwanathan, and Nicol N. Schraudolph. Graph Kernels for Disease Outcome Prediction from Protein-Protein Interaction Networks. In *Biocomputing*, 2007.

Mathieu Bouchard, Anne-Laure Jousselme, and Pierre-Emmanuel Doré. A Proof for the Positive Definiteness of the Jaccard Index Matrix. *International Journal of Approximate Reasoning*, 54 (5):615–626, 2013.

Xuewen Chen, Yulei Wu, and Dimitra Simeonidou. A Graph World Model for Action-Conditioned and Multi-Step Edge Orchestration. In *2026 IEEE/CIC International Conference on Communications in China (ICCC)*, 2026.

Stephen Chung, Ivan Anokhin, and David Krueger. Thinker: Learning to plan and act. In *NeurIPS*, 2023.

Victor-Alexandru Darvariu, Stephen Hailes, and Mirco Musolesi. Goal-directed graph construction using reinforcement learning. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2254):20210168, 2021.

Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian. Probabilistic Recurrent State-Space Models. In *ICML*, 2018.

Fan Feng and Sara Magliacane. Learning dynamic attribute-factored world models for efficient multi-object reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Tao Feng, Yexin Wu, Guanyu Lin, and Jiaxuan You. Graph World Model. In *ICML*, 2025.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. Vista: A Generalizable Driving World Model with High Fidelity and Versatile Controllability. In *NeurIPS*, 2024.

Colin Graber and Alexander G. Schwing. Dynamic Neural Relational Inference. In *CVPR*, 2020.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

David Ha and Jürgen Schmidhuber. World Models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. In *ICML*, 2019.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *ICLR*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Chenjie Hao, Weyl Lu, Yifan Xu, and Yubei Chen. Neural Motion Simulator Pushing the Limit of World Models in Reinforcement Learning. In *CVPR*, 2025.

Petter Holme and M. E. J. Newman. Nonequilibrium Phase Transition in the Coevolution of Networks and Opinions. *Physical Review E*, 74(5):056108, 2006.

Yanjia Huang, Yunuo Chen, Ying Jiang, Jinru Han, Zhengzhong Tu, Yin Yang, and Chenfanfu Jiang. Learn2Fold: Structured Origami Generation with World Model Planning. *arXiv preprint arXiv:2603.29585*, 2026.

Can Karacelebi, Yusuf Talha Sahin, Elif Surer, and Ertan Onur. Learning Ad Hoc Network Dynamics via Graph-Structured World Models. *arXiv preprint arXiv:2604.14811*, 2026.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data. In *ICLR*, 2017.

Jiawei Liu, Senqiao Yang, Mingjun Wang, Yu Wang, and Bei Yu. Graph World Models: Concepts, Taxonomy, and Future Directions. *arXiv preprint arXiv:2604.27895*, 2026a.

Renpeng Liu, Xiaozhe Wang, Yiyang Fu, Bo Hu, and Shanzhi Chen. Graph World Model for Energy-Efficient Hybrid Beamforming in LEO Satellite Networks. *IEEE Transactions on Green Communications and Networking*, 10:3772–3785, 2026b.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning Sparse Neural Networks through L_0 Regularization. In *ICLR*, 2018.

Yi Lu, Yaran Chen, Dongbin Zhao, and Dong Li. Mgrl: Graph Neural Network Based Inference in a Markov Network with Reinforcement Learning for Visual Navigation. *Neurocomputing*, 421: 140–150, 2021.

Paul Mattes, Rainer Schlosser, and Ralf Herbrich. Hieros: Hierarchical Imagination on Structured State Space Sequence World Models. In *ICML*, 2024.

Eli Meirom, Haggai Maron, Shie Mannor, and Gal Chechik. Controlling Graph Dynamics with Reinforcement Learning and Graph Neural Networks. In *ICML*, 2021.

Vincent Micheli, Eloi Alonso, and François Fleuret. Efficient World Models with Context-Aware Tokenization. In *ICML*, 2024.

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 *ICML*, 2025.

Adilson E. Motter and Ying-Cheng Lai. Cascade-Based Attacks on Complex Networks. *Physical Review E*, 66(6):065102, 2002.

Robin Schiewer, Anand Subramoney, and Laurenz Wiskott. Exploring the Limits of Hierarchical World Models in Reinforcement Learning. *Scientific Reports*, 14(1):26856, 2024.

Paul Schultz, Jobst Heitzig, and Jürgen Kurths. A random growth model for power grids and other spatially embedded infrastructure networks. *The European Physical Journal Special Topics*, 223 (12):2593–2610, 2014.

Andreas Sedlmeier, Michael Kölle, Robert Müller, Leo Baudrexel, and Claudia Linnhoff-Popien. Quantifying Multimodality in World Models. In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, 2022.

Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey. *IEEE access*, 9:79143–79168, 2021.

Xinyuan Song and Zekun Cai. Understanding Rollout Error in Graph World Models. *arXiv preprint arXiv:2606.27780*, 2026.

Bharath K. Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert R.G. Lanckriet. Hilbert Space Embeddings and Metrics on Probability Measures. *Journal of Machine Learning Research*, 11(50):1517–1561, 2010.

Ingo Steinwart. On the Influence of the Kernel on the Consistency of Support Vector Machines. *Journal of Machine Learning Research*, 2(Nov):67–93, 2001.

Wei Wang, Yaosen Chen, Han Yang, Yuegen Liu, Mingli Luo, Xinxin Jiao, Xuming Wen, and Ming Liu. A Structural Dynamics Graph World Model: Unified Modeling, Constrained Rollout, and Interpretable Calibration. *arXiv preprint arXiv:2608.08689*, 2026.

Yang Ye and Shihao Ji. Sparse Graph Attention Networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):905–916, 2021.

Cagatay Yildiz, Markus Heinonen, and Harri Lähdesmäki. Continuous-time Model-based Reinforcement Learning. In *ICML*, 2021.

A PROOFS

A.1 PROOF OF LEMMA 1

For the continuous features, after concatenating the coordinates over the aligned nodes and feature index sets, each term $\exp\left(-\frac{\|x-x'\|^2}{2r^2}\right)$ is an RBF kernel under a positive diagonal rescaling on a finite-dimensional Euclidean space and is therefore C_0-universal. Since $K_n^{\mathcal{C}}$ and $K_g^{\mathcal{C}}$ are finite positive mixtures of such kernels, they are also C_0-universal (Steinwart, 2001).

For the categorical features, each feature-index set and each corresponding value space is finite. If the categorical feature index set is empty, the distance is defined to be 0, and the corresponding kernel is identically 1 and can be omitted from the product. For a finite set, a kernel is universal if and only if its Gram matrix is strictly positive definite (Borgwardt et al., 2006). Let

$$d_{ij} = \frac{1}{|\mathcal{F}_g^\delta|} \sum_{f \in \mathcal{F}_g^\delta} \lambda_f \mathbf{1}[x_f(G_i) \neq x_f(G_j)] \,.$$

The corresponding Gram matrix is $\boldsymbol{K}_{ij} = \exp(-d_{ij})$. The distance d is a weighted Hamming distance. Weighted Hamming distance is conditionally negative definite when the weights are non-negative. Hence, by Schoenberg's theorem, $\boldsymbol{K}$ is positive definite. Moreover, because all weights satisfy $\lambda_f > 0$, this kernel is the product of strictly positive definite kernels on the individual finite categorical coordinates, and is therefore strictly positive definite. Thus K_g^δ is universal. The same argument applies to K_n^δ.

For the adjacency term, because the node set is finite, the set of possible edges is finite, and every graph topology is a subset of the finite set of possible edges. Bouchard et al. (2013) prove that the Jaccard kernel matrix on the non-empty subsets of any finite ground set is strictly positive definite. Under the stated convention for the empty set, K_J extends this matrix by an isolated diagonal entry $K_J(\emptyset, \emptyset) = 1$, with $K_J(\emptyset, E) = 0$ for $E \neq \emptyset$, and is therefore strictly positive definite on the full set of possible graph topologies. Since this domain is finite, strict positive definiteness makes the Gram matrix invertible, so the RKHS can represent every real-valued function on the domain. Hence K_J is universal on the finite space of graph topologies.

Finally, K_{joint} is the tensor-product kernel. The product of C_0-universal kernels is C_0-universal on the product domain. Therefore K_{joint} is C_0-universal and hence characteristic. □

A.2 PROOF OF LEMMA 2

Each continuous feature kernel $k_f^{\mathcal{C}}$ is a finite positive mixture of RBF kernels and is therefore positive definite. Each categorical feature kernel $k_f^\delta(x, y) = \exp\left(-\lambda_f \mathbf{1}[x \neq y]\right)$ is also positive definite: on the finite value space of feature f, its Gram matrix is $(1 - e^{-\lambda_f})I + e^{-\lambda_f}\mathbf{1}\mathbf{1}^\top$, which is positive semidefinite for $\lambda_f > 0$. Hence K_{add}^n and K_{add}^g, being nonnegative averages of positive-definite kernels, are positive definite. As in the proof of Lemma 1, K_J is a positive-definite kernel on finite sets. Therefore, since $w_n, w_g, w_e \geq 0$, K_{add} is positive definite.

By Lemma 1, K_{joint} is characteristic. For any probability measures P, Q, $\text{MMD}_K^2(P, Q) = \text{MMD}_{K_{\text{add}}}^2(P, Q) + \epsilon\,\text{MMD}_{K_{\text{joint}}}^2(P, Q)$, where both terms on the right-hand side are nonnegative. Since $\epsilon > 0$, $\text{MMD}_K^2(P, Q) = 0 \Rightarrow \text{MMD}_{K_{\text{joint}}}^2(P, Q) = 0$. Because K_{joint} is characteristic, this implies $P = Q$. Hence K is characteristic. □

B ENVIRONMENT DESCRIPTIONS

Each environment is implemented as a POMDP as described in Section 3. Within each environment, the observation space consists of the adjacency matrix for the current graph, the categorical and continuous node features, and the categorical and continuous graph features. These features are environment-specific and are described in detail below. For completeness, we include the number of node selections k in the graph features, and include the action mask in the node features.

B.1 Search and Rescue

The SAR environment is a stochastic, partially observable environment in which the agent navigates a graph to collect survivors and escort them to an exit node.

States consist of undirected graphs $G = (V, E, \{x(v)\}_{v \in V}, x(G))$. The node features are

$$x(v) = \{\text{survivor_present}(v), \text{stability}(v), \text{exit_node}(v), \text{current_location}(v)\},$$

where $\text{survivor_present}(v), \text{exit_node}(v), \text{current_location}(v) \in \{0, 1\}$, and $\text{stability}(v) \in [0, 1]$. The graph features are $\{\text{escorting}, k\}$. The $\text{escorting} \in \{0, 1\}$ feature indicates whether the agent is currently escorting a survivor.

The agent selects $k = 1$ adjacent node at each transition and moves to that node. If the agent is not escorting and the selected node contains a survivor, it collects the survivor. If the agent is escorting and moves to the exit node, it delivers the survivor. The environment begins with the agent at the exit node.

After the agent moves to node u, u loses 0.3 stability with probability 0.5, unless u is the exit node: $\text{stability}(u) \leftarrow \max\{0, \text{stability}(u) - 0.3\}$. The environment then removes every edge (u, v) incident to the visited node for which $\text{stability}(u) + \text{stability}(v) < 1$. Thus, movement may disconnect parts of the graph over time.

The reward is -1 per transition, with an additional reward of 1 for collecting a survivor, 10 for delivering a survivor to the exit, and 5 for rescuing all survivors. An episode terminates successfully when all survivors have been delivered. If the agent cannot reach the exit, the episode terminates with an additional penalty of $-30 - 5n_{\text{remaining}}$, where $n_{\text{remaining}}$ is the number of undelivered survivors. If the agent is not escorting a survivor and no remaining survivor is reachable from the exit, the episode terminates with an additional penalty of $-5n_{\text{remaining}}$. The episode is truncated after a number of transitions equal to $|V|$ times the initial number of survivors.

To create the partial observation, the true survivor-presence feature is revealed only for nodes within two hops of the agent's current location; survivor locations elsewhere are reported as absent. All other state features, including the graph topology, node stability, exit node, and current location, remain observable.

For current location u, the action mask is $\mathfrak{A}(s) = \{v \in V \mid (u, v) \in E\}$. Thus, the agent must select exactly one adjacent node at every transition.

We sample the initial graph from a Delaunay triangulation of a set of $|V|$ points located in the unit square. We select one exit node uniformly at random, select 30% of nodes, excluding the exit, to have survivors, and sample initial node stabilities uniformly from $[0.5, 1]$, setting the exit-node stability to 1. Parameters for the SAR environment were chosen via grid search to produce an environment with a reasonable margin between the performance of a greedy policy and a random policy without becoming trivial.

B.2 Opinion Dynamics

The OD environment is a stochastic, fully observable environment in which the agent selects a set of nodes to influence towards a target opinion.

States consist of undirected graphs $G = (V, E, \{\text{opinion}(v), \zeta(v)\}_{v \in V}, \{\text{k_frac}, \text{k_frac}_{\text{env}}\})$. For the node features, $\text{opinion}(v) \in \{0, \ldots, 4\}$ is node v's current opinion, and $\zeta(v) \in [-1, 1]$ is its fixed latent "vibe". In the graph feature, k_frac specifies the fraction of nodes selected by the agent and $\text{k_frac}_{\text{env}}$ specifies the fraction of nodes selected by the environment.

We use dynamics based on the model of Holme & Newman (2006). At each transition, the environment first selects $k_{\text{env}} = \lceil |V| \text{k_frac}_{\text{env}} \rceil$ nodes uniformly without replacement, with $\text{k_frac}_{\text{env}} = 0.1$. For each selected node, it either rewires one incident edge with probability $\phi = 0.458$ (Holme & Newman, 2006), or copies the opinion of one of its neighbours otherwise. Rewiring replaces an edge to a dissimilar-vibe neighbour with an edge to a same-opinion, similar-vibe non-neighbour. Let $d(u, v) = |\zeta(u) - \zeta(v)|$ denote the distance between two nodes' vibes, and let $T = 0.1$ be the

temperature. When selecting a vibe-similar node from a candidate set C, the environment uses

$$P(v \mid u, C) = \frac{\exp(-d(u,v)/T)}{\sum_{w \in C} \exp(-d(u,w)/T)}.$$

For a rewiring event, the existing neighbour to disconnect is instead selected with

$$P(v \mid u, C) = \frac{\exp(+d(u,v)/T)}{\sum_{w \in C} \exp(+d(u,w)/T)}.$$

It then connects the node to a currently unconnected node with the same opinion, sampled using the first, vibe-similar distribution. The agent then selects $k = \lceil |V| \mathrm{k_frac} \rceil$ nodes, with $\mathrm{k_frac} = 0.05$. Each selected node independently adopts the target opinion 0 with probability 0.3. The reward is the increase in the fraction of nodes holding the target opinion. Parameters for the OD environment were chosen via grid search to produce an environment with a reasonable margin between the performance of a greedy policy and a random policy, while still having some margin for improvement on the greedy policy.

With no intervention, the environment converges to a steady state within several hundred to several thousand transitions. Thus we set the horizon to $H = 50$ to evaluate the model's ability to predict the short-term dynamics of the environment.

We sample the initial graph from a connected Erdős–Rényi distribution with expected degree 4, assign each node one of five opinions uniformly at random, and sample each node's vibe uniformly from $[-1, 1]$. The action mask includes every node, so any distinct set of nodes is a valid action.

B.3 Cascading Failures

The CF environment is based on the model of Motter & Lai (2002), which describes a failure model of electrical power grids where each node has a load and a capacity, and overloaded nodes fail, redistributing their load to their neighbours. We introduce stochasticity and partial observability to the model.

States consist of undirected graphs $G = (V, E, \{\mathrm{load}(v), \mathrm{capacity}(v)\}_{v \in V}, \{k, \xi\})$. The $\mathrm{load}(v) \in [0, 1]$ describes the current load on node v, and $\mathrm{capacity}(v) \in [0, 1]$ describes the rated capacity of node v. The actual capacity of each node is determined by the unobserved safety factor $\xi \in [1.1, 1.4]$, which is a single graph feature fixed for the entire episode. Per Motter & Lai (2002), the load of each node is determined by the normalised betweenness centrality of the node,

$$c_t(v) = \frac{1}{P} \sum_{u,w \in V} \frac{\mathrm{paths}_t(u, w \mid v)}{\mathrm{paths}_t(u, w)},$$

where $\mathrm{paths}_t(u, w)$ is the total number of shortest paths between nodes u and w at step t, $\mathrm{paths}_t(u, w \mid v)$ is the number of those paths that pass through node v, and P is a normalisation factor given by the number of pairs of nodes in the graph. The rated capacity of each node is fixed to $c_0(v)$, the load of the nodes in the network before any failures.

The transition function is stochastic and proceeds as follows. At step t, the agent selects $k = 1$ node to disconnect from the graph, and all adjacent edges to that node are removed. Additionally, we define stochastic failures: any node v for which $\mathrm{load}_{t-1}(v) > \xi \mathrm{capacity}(v)$ fails according to

$$p(v \text{ failure}) = \min\left(1, \frac{\mathrm{load}_{t-1}(v) - \xi \mathrm{capacity}(v)}{\xi \mathrm{capacity}(v)}\right).$$

Failed nodes are disconnected from the graph by the removal of all incident edges. Subsequently, the loads of all nodes are recalculated based on the new graph topology.

Observations are given by $o_t = (V_t, E_t, \{\mathrm{load}_t(v), \mathrm{capacity}(v)\}_{v \in V_t}, \{k\})$. The action space is the set of nodes which have at least one incident edge: $\mathfrak{A}(s) = \{v \in V \mid \exists u \in V, (u, v) \in E\}$.

The horizon is set to $H = |V|/2$, and the episode terminates after H steps or when the graph is completely disconnected. Each step incurs a reward of $-1/H$, and there is a terminal penalty of $|\mathrm{lcc}(G_t)|/|V|$ if the horizon is reached before the graph is completely disconnected, where $\mathrm{lcc}(G_t)$ is the largest connected component of the graph at step t.

We sample the initial graph from a Schultz-Heitzig-Kurths distribution (Schultz et al., 2014), with parameters $n_0 = 0.5|V|$, $p = 0.5$, $q = 0.5$, $r = 1/3$, and $s = 0.1$.

B.4 Robust Graph Construction

The RGC environment, introduced by Darvariu et al. (2021), is a deterministic, fully observable environment in which the agent selects $k = 2$ nodes to connect in order to increase the robustness of the graph under node removal. The states S consist of graphs $G = (V, E, \emptyset, \{\tau, k\})$, where the graph feature τ represents the remaining edge addition budget as a fraction of the total number of potential edges in the graph ($|V|(|V|-1)/2$). The transition adds an edge between the two selected nodes, and decreases the budget τ by $2/(|V|(|V|-1))$, unless an edge already exists between them. The reward function is the expected increase in robustness of the graph under targeted node removal, where the robustness is measured by the fraction of nodes required to be removed in order to disconnect the graph, with nodes being selected in descending order of initial degree. Termination occurs when the edge addition budget is exhausted, i.e., when $\tau = 0$, or after $H = |V|(|V| - 1)/2$ steps.

We sample the initial graph from an Erdős-Rényi distribution with $p = 0.2$, and set $\tau = 5\%$. For simplicity, we define $\mathfrak{A}(s)$ to be $\{v \in V \mid \exists u \in V, (u, v) \notin E\}$, which is the set of nodes with at least one available edge to add. This means that the agent may select a pair of nodes which are already connected, resulting in a noop.

C Baseline Implementations

We adapt the G-RSSM model to our environments by replacing its observation encoder and decoder with environment-specific networks that support the node- and graph-level scalar, categorical, and binary features defined by each environment. Categorical inputs are one-hot encoded, and separate output heads predict each state feature using an appropriate distribution. Per Karacelebi et al. (2026), graph-level features are broadcast to nodes during encoding and node-set actions are represented by a binary indicator at each node and supplied to the corresponding node-level recurrent transition.

The original adjacency decoder produces a fixed-size $|V| \times |V|$ output and therefore cannot be applied to graphs whose size differs from those used during training. We replace it with a size-independent node-wise decoder. For each node v, a shared MLP maps its deterministic and stochastic recurrent states to an embedding $\mathrm{emb}(v) = f_{\mathrm{dec}}(\boldsymbol{h}(v), \boldsymbol{z}(v))$, and the adjacency logit for a pair of nodes is $\ell_{uv} = (\boldsymbol{W}_{\mathrm{adj}}\,\mathrm{emb}(u))^{\top}\,(\boldsymbol{W}_{\mathrm{adj}}\,\mathrm{emb}(v))$. Because the same functions are applied to every node and node pair, the number of decoder parameters is independent of $|V|$, allowing the adapted G-RSSM to operate on graphs of different sizes. We also apply the feature scaling described in Appendix E.1 to the G-RSSM decoder outputs.

We similarly adapt EA-GWM and Vanilla GWM to multimodal, variable-size graph environments. The original models operate on continuous observable node states, whereas our environments may contain scalar, categorical, and binary node and graph features. We therefore introduce feature-specific encoders and prediction heads, represent categorical and binary predictions by their probabilities during differentiable rollouts, and retain node and graph features as distinct components of the predicted state. Continuous features are predicted as residual changes to their current values as per the original formulation (Song & Cai, 2026).

The adapted EA-GWM and Vanilla GWM share the same architecture and differ only in their training objectives. Vanilla GWM is trained using the supervised one-step losses for the next graph state, adjacency, reward, continuation, and action mask. The EA-GWM additionally uses rollout-consistency, spectral, and critical-node losses. We generalise the rollout and critical-node losses to use mean-squared error for scalar factors and classification losses for categorical and binary factors. Critical-node weighting is applied only to node-level state losses, with node weights proportional to degree and normalised to have mean one within each graph.

Adjacency Sampling. The GDM model uses Hard Concrete sampling during imagination to produce a sparse adjacency matrix update, applied to the previous recurrent adjacency matrix. The decoder for GDM is a pass-through network that outputs the same predicted adjacency matrix without resampling. For EA-GWM and Vanilla GWM, the adjacency matrix is predicted by a learned transformation of the node features: $\hat{\boldsymbol{A}}(i, j) = \sigma(\boldsymbol{h}(i)^{\top}\boldsymbol{Q}\boldsymbol{h}(j))$. Edges are sampled from the Bernoulli distribution defined by $\hat{\boldsymbol{A}}(i, j)$, and $\boldsymbol{Q}$ is regularised to have a norm of 1. Due to the Bernoulli sampling, the adjacency matrix distribution can be soft, even though the model is deterministic. This is disadvantageous in environments where transitions are stochastic but sharp, such as the SAR envi-

ronment. For the G-RSSM model, the adjacency matrix is predicted only by a decoder head and not used in the recurrent transition.

D Evaluation Details

The GDD score is calculated from predictions of the model under random rollouts. We use fixed test sets of 20 graphs of each size $|V| \in \{20, 30, 50, 100, 200, 500, 1000\}$, except for the RGC environment where the largest size is 200 due to episode length exceeding 6000 steps at 500 nodes. For each graph, we perform a random rollout of an episode to obtain a set of reference states. For a single transition, the environment is reset to the reference state and advanced one step using the reference action $\kappa = 30$ times to create the environment's transition distribution. Similarly, the model's transition distribution is obtained by performing a forward pass from the reference state and action κ times. For latent-state models, the model's initial latent state is based on the posterior-corrected latent state from the preceding trajectory, while for state-space models, the model's state is based on the environment's reference state.

The transition GDD is calculated using Equation (7) with all pairs of samples from the environment and model transition distributions, and the mean per-episode GDD is calculated by averaging the transition GDD across transitions in the episode. In some environments, the episode length varies based on the size of the graph, so for computational efficiency, we select 20 reference transitions per episode, always including the first and final transitions. For continuous and categorical features, bandwidths are fitted based on the training set using the median heuristic, with full details provided in Appendix D.1.

In the calculation of GDD, we include the reward, continuation, and action mask in the state features, so that the GDD score reflects the model's ability to predict these factors as well as the next graph state. The reward is considered a continuous graph feature, the continuation is a categorical graph feature, and the action mask is a categorical node feature. In a terminal transition, we consider only the reward and continuation, as the action mask and observation are not defined after termination.

D.1 Bandwidth Fitting

We fit MMD bandwidths for categorical and continuous node features separately for each environment, using transitions sampled from the training rollouts. For each environment we collect 8 transitions from each episode in the training set, resulting in a total of $8 \times 100 \times 5 = 4000$ transitions for each environment. All fitted continuous scales and categorical weights are precomputed and reused for every model evaluated in that environment.

For each continuous node or graph feature f, we compute the absolute one-step changes

$$\Delta_f = \{|x_{t+1,f} - x_{t,f}| : |x_{t+1,f} - x_{t,f}| > 0\}$$

over all collected transitions. For node features, changes are pooled over all nodes. The base bandwidth is the median nonzero one-step change,

$$b_f = \mathrm{median}(\Delta_f).$$

If a feature never changes in the sampled transitions, we instead use the median nonzero absolute value observed for that feature. If the feature is identically zero, we use the minimum bandwidth 10^{-8}. The multi-bandwidth RBF kernel uses bandwidth multipliers $\mathcal{M} = \{0.1, 0.5, 1, 2, 16\}$.

For categorical features, we fit a weight from the empirical one-step change rate. For every transition, we compute the fraction $\rho_{t,f}$ of valid entries whose value changes and set

$$\lambda_f = \frac{1}{\mathbb{E}_t[\rho_{t,f}]}.$$

Thus, changes in normally stable categorical features receive greater weight than changes in frequently varying features. If no changes are observed, we set $\lambda_f = 1$.

E Experiment Details

E.1 Feature Scaling

To prevent features with large magnitudes from dominating the decoder objective, we standardise each scalar prediction target using statistics calculated from the training data. For each scalar feature f, we calculate the mean μ_f and population standard deviation σ_f. The statistics are calculated over the scalar values in all stored pre- and post-transition observations. The reward mean and standard deviation are fitted separately using all transition rewards. If a target is constant throughout the training data, we set $\sigma_f = 1$.

Each scalar decoder head produces a value $\hat{z}_f$ in standardised coordinates. Before exposing the prediction to the model or evaluator, we transform it back to the original environment units: $\hat{x}_f = \sigma_f \hat{z}_f + \mu_f$. The reconstruction loss is calculated in standardised coordinates:

$$\mathcal{L}_f = \left(\frac{\hat{x}_f - \mu_f}{\sigma_f} - \frac{x_f - \mu_f}{\sigma_f} \right)^2 = \left(\frac{\hat{x}_f - x_f}{\sigma_f} \right)^2 .$$

Consequently, decoder predictions and sampled trajectories remain expressed in the original units of the environment, while each scalar feature contributes error relative to its variation in the training data. The same procedure is applied to the reward head.

E.2 Training Details

For each model, we train for 5000 epochs on a set of pre-collected rollouts of a random policy on a set of graphs with sizes in $\{15, 18, 20, 23, 25\}$ (100 graphs per size). The random policy uniformly samples from the valid actions at each transition, so the training data never contains an invalid action. We use the Adam optimiser with a batch size of 8 and a sequence length of 20, training 10 seeds for each model. Hyperparameters for each model/environment pair were chosen from a grid search based on the mean GDD score achieved on a validation set consisting of 20 graphs of each node count 20 and 30. Hyperparameter details are provided in Appendix E.3.

E.3 Hyperparameter Search

The parameter search ranges for each model are provided in Table 2. Each combination was trained with 10 different seeds, and the minimum mean GDD score on the validation set over the seeds was used to select the best hyperparameters for each model/environment pair. The L parameter represents the number of rounds of message passing, being GNN layers for GDM and GWM and rounds of recurrent message passing for G-RSSM. For G-RSSM the Encoder L represents the number of GNN layers in the encoder.

Table 2: Hyperparameter search ranges for each model.

Parameter	GDM	EA-/Vanilla GWM	G-RSSM
LR	$10^{-3}, 5 \times 10^{-4}, 10^{-4}$	$10^{-3}, 5 \times 10^{-4}, 10^{-4}$	$5 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}, 10^{-4}$
L	$1, 2, 3, 4$	$1, 2, 3, 4$	$1, 2, 3$
Capacity	$\dim(\boldsymbol{h}) = 64$, $\dim(\boldsymbol{z}) = 32 \times 32$ $\dim(\boldsymbol{h}) = 128$, $\dim(\boldsymbol{z}) = 16 \times 16$	$\dim(\boldsymbol{h}) = 64, 128$	$\dim(\boldsymbol{h}) = 128$
β_{adj}	1	$0.1, 1, 10$	$0.1, 1, 10$
Encoder L	-	-	$1, 2, 3$
β_{rep}	$0.1, 0.05, 0.01$	-	0.1

E.3.1 Final Hyperparameters

The final hyperparameters for each model and environment are provided in Table 3. Across environments, we use Hard Concrete parameters of $\beta = 2/3$, $\gamma = -0.1$, and $\zeta = 1.1$. We use the pooling function $\text{Pool} = \text{mean}$ for all models and environments.

Table 3: Final hyperparameters for each model and environment. Unsearched parameters are indicated with an asterisk (*).

Parameter	RGC	OD	SAR	CF
		GDM		
LR	10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
L	1	1	1	1
$\dim(e)$	64	128	128	128
$\dim(\boldsymbol{h})$	64	128	128	128
$\dim(\boldsymbol{z}(V))$	32×32	16×16	16×16	16×16
$\dim(\boldsymbol{z}(G))$*	8×8	8×8	8×8	8×8
β_{adj}*	1	1	1	1
β_{dyn}*	1	1	1	1
β_{rep}	0.1	0.05	0.1	0.1
β_{pred}*	1	1	1	1
β_n*	1	1	1	1
β_g*	1	1	1	1
β_h*	1	1	1	1
λ_{FB}*	0	0.01	0.01	0.01
		EA-GWM		
LR	10^{-3}	10^{-4}	5×10^{-4}	10^{-4}
L	2	4	2	4
$\dim(\boldsymbol{h})$	64	64	128	128
β_{adj}	1	1	1	1
		Vanilla GWM		
LR	5×10^{-4}	10^{-4}	10^{-4}	10^{-4}
L	2	4	2	1
$\dim(\boldsymbol{h})$	64	128	128	128
β_{adj}	0.1	0.1	0.1	0.1
		G-RSSM		
LR	10^{-3}	10^{-3}	10^{-3}	10^{-3}
L	3	1	1	1
Encoder L	1	1	2	1
$\dim(e)$*	128	128	128	128
$\dim(\boldsymbol{h})$*	128	128	128	128
$\dim(\boldsymbol{z})$*	8×8	8×8	8×8	8×8
β_{adj}	10	1	10	0.1
β_{dyn}*	1	1	1	1
β_{rep}*	0.1	0.1	0.1	0.1
β_{pred}*	1	1	1	1
λ_{FB}*	0	0.01	0.01	0.01

F Additional Results

F.1 GDD Verification

For the calculation of GDD, we use the biased MMD estimator in Equation (2), which may produce small non-zero values between identical distributions due to finite sample effects. To measure the magnitude of this effect, we calculate the GDD between the reference environment distribution and itself under the same sampling procedure used for model evaluation. The results are provided in Table 4, which shows that the GDD bias is negligible for all environments and graph sizes. Results for RGC are exactly zero because the environment is deterministic, and therefore the reference distribution is a single point mass.

Table 4: Self-GDD between the reference environment distribution and a sampled environment distribution, with $\kappa = 30$ samples.

Env.	$\lvert V\rvert = 20$	$\lvert V\rvert = 30$	$\lvert V\rvert = 50$	$\lvert V\rvert = 100$	$\lvert V\rvert = 200$
RGC	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
OD	$0.031_{\pm 0.014}$	$0.033_{\pm 0.012}$	$0.036_{\pm 0.007}$	$0.035_{\pm 0.006}$	$0.035_{\pm 0.004}$
SAR	$0.012_{\pm 0.014}$	$0.012_{\pm 0.012}$	$0.010_{\pm 0.011}$	$0.007_{\pm 0.008}$	$0.006_{\pm 0.006}$
CF	$0.003_{\pm 0.011}$	$0.002_{\pm 0.010}$	$0.003_{\pm 0.010}$	$0.002_{\pm 0.009}$	$0.002_{\pm 0.007}$

F.2 Training Curves

Figure 4 shows the total loss curves, smoothed over 10 points, for each environment during training. Note that the loss equation and coefficients can differ between methods, so the values are not directly comparable across methods. In the OD environment, the loss for GDM increases after an initial stable period, which is due to the model learning to predict the reward function at the expense of higher KL loss in the node features.

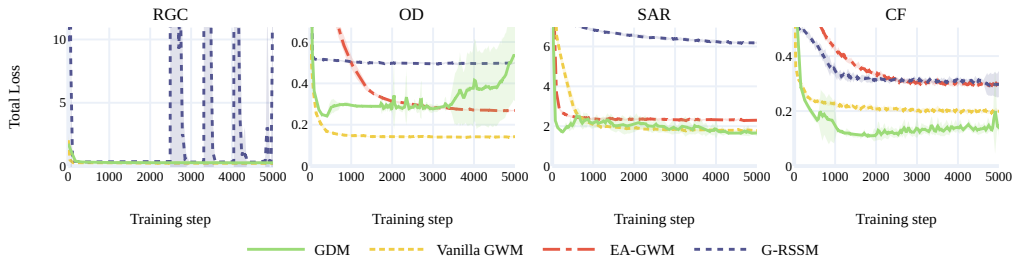


Figure 4: Training loss curves for each method/environment, averaged across 10 seeds.

F.3 Factor-Wise Distribution Metrics

For single-step predictions, we show the breakdown of the four components of GDD score in Figure 5. We also show the full factor-wise distribution metrics for each environment in Figure 6. GDM consistently predicts the adjacency matrix with extremely low BCE, demonstrating that the model is able to accurately capture the graph topology. The G-RSSM model does not explicitly model the adjacency matrix, and therefore has high BCE for the adjacency in all environments. The Vanilla GWM and EA-GWM models are able to make some predictions for the adjacency matrix, but struggle especially with static edges. Since the adjacency matrix is predicted from node features and there is no conditioning on previous adjacency, these models struggle to predict evolving topologies where many edges remain unchanged across transitions.

The Vanilla GWM and EA-GWM models are able to capture certain features more accurately than the G-RSSM and GDM models. This is due to a limitation of the RSSM framework, which requires the model to incorporate all posterior information through the sampled stochastic latent state. This

means that certain information, particularly information with high variance across episodes and little redundancy, can be difficult to represent accurately in the latent state. The GDM model partially mitigates this issue by using a deterministic state update on the posterior path, which improves its representational ability relative to G-RSSM but can still struggle with information loss.

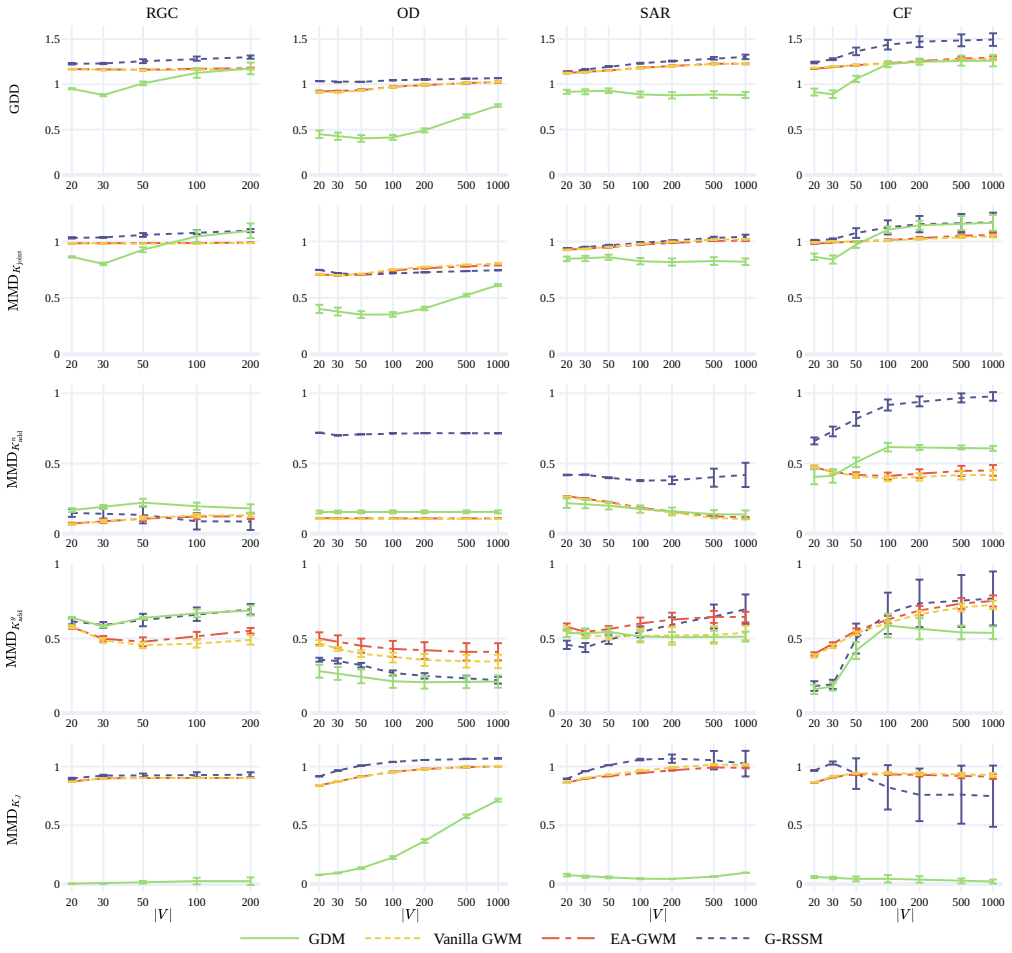


Figure 5: GDD score and individual components of the GDD for each environment and model.

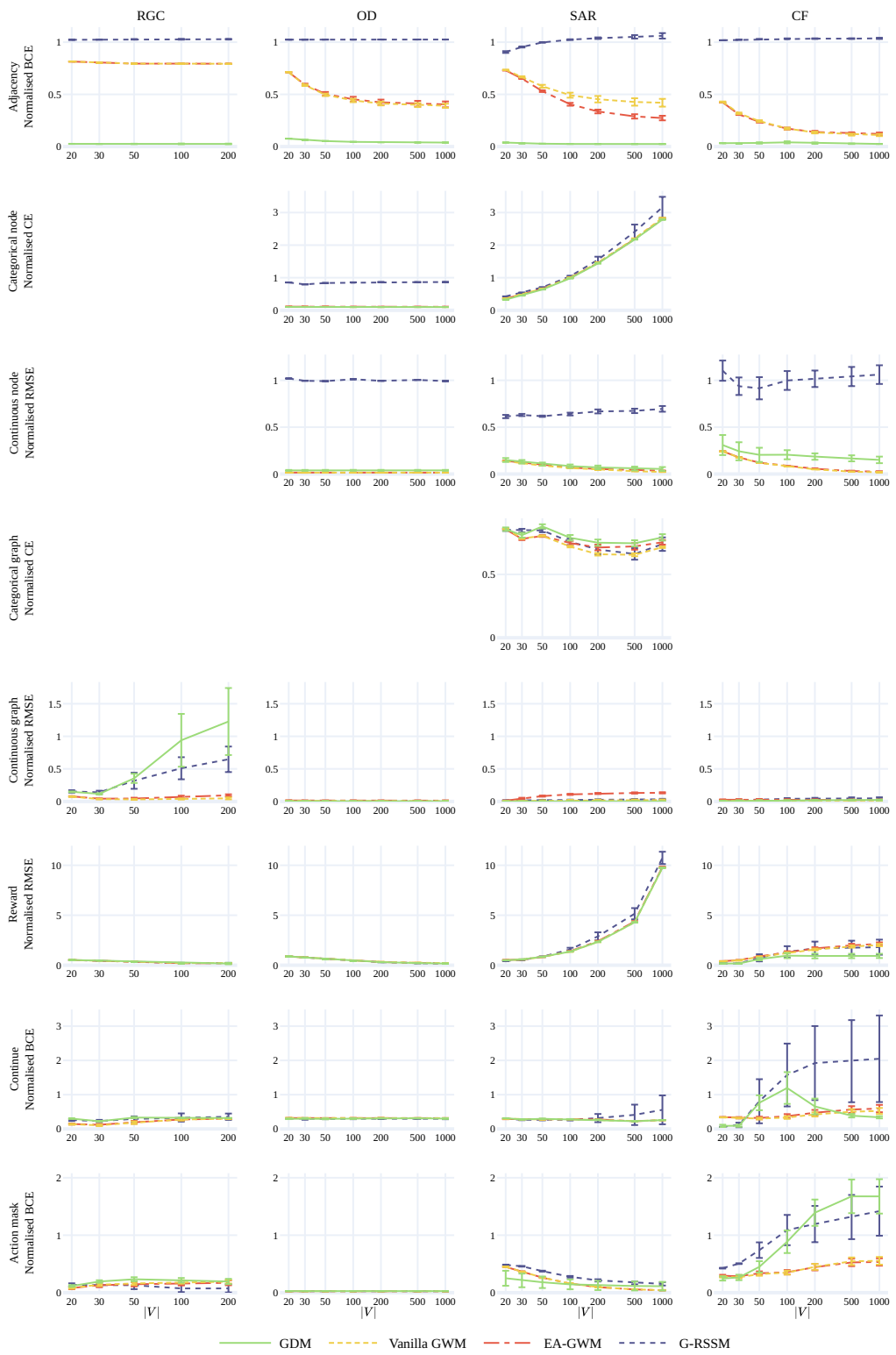


Figure 6: Factor-wise distribution metrics for each environment. Empty plots indicate absence of features for that environment.

F.4 Multi-Step Rollout Results

When calculating the GDD score for multi-step rollouts, we select initial states as per single-step rollouts, then perform a rollouts of T steps according to the pre-determined actions. In some cases, this can lead to invalid actions being taken, as the stochastic transition may result in a state where the action mask does not permit the action that was valid in the reference trajectory. For the purpose of the rollouts, we allow invalid actions in the environment by performing a noop, and in the environment by running the forward pass with the invalid action. Due to the diffusion of the distribution with more steps, we sample 100 points for the multi-step GDD calculation. Figure 7 shows the breakdown of the GDD score for each environment and model across rollout lengths from 1 to 5 steps, as well as the rate of invalid actions taken during the rollout, with the environment shown in grey.

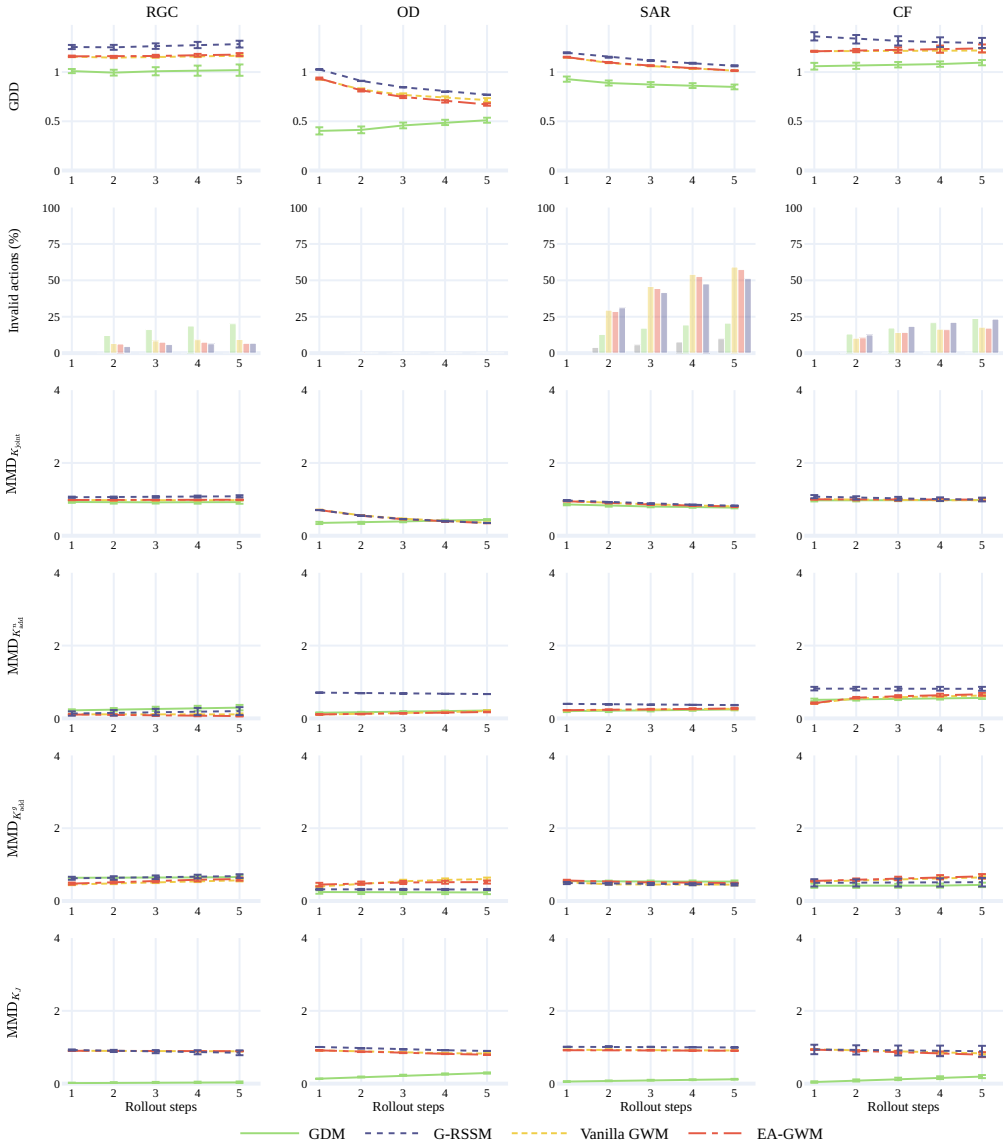


Figure 7: Multi-step rollout GDD, invalid action rate, and MMD components on 50-node graphs.

F.5 Ablation Results

The GDD for all ablation studies is found in Table 5. We additionally show an ablation combining the removal of graph state, $\boldsymbol{h}^+$ correction, and recurrent adjacency.

Table 5: GDD score (single-step) for each ablation and environment. Mean and 95% confidence interval over 10 seeds.

Ablations	$\|V\| = 20$	$\|V\| = 30$	$\|V\| = 50$	$\|V\| = 100$	$\|V\| = 200$
			RGC		
Full	$0.950_{\pm 0.007}$	$0.880_{\pm 0.012}$	$1.010_{\pm 0.021}$	$1.125_{\pm 0.054}$	$1.175_{\pm 0.065}$
No Graph State	$0.955_{\pm 0.013}$	$0.933_{\pm 0.023}$	$1.009_{\pm 0.004}$	$1.069_{\pm 0.009}$	$1.100_{\pm 0.016}$
$\beta_h = 0$	$0.944_{\pm 0.005}$	$0.886_{\pm 0.006}$	$1.021_{\pm 0.022}$	$1.070_{\pm 0.037}$	$1.148_{\pm 0.048}$
No $\boldsymbol{h}^+$ Correction	$0.947_{\pm 0.004}$	$0.895_{\pm 0.013}$	$1.070_{\pm 0.032}$	$1.197_{\pm 0.053}$	$1.246_{\pm 0.061}$
Bernoulli Adj	$0.951_{\pm 0.003}$	$0.920_{\pm 0.026}$	$1.053_{\pm 0.042}$	$1.181_{\pm 0.040}$	$1.215_{\pm 0.034}$
All-Pairs	$0.952_{\pm 0.006}$	$0.931_{\pm 0.033}$	$1.089_{\pm 0.033}$	$1.214_{\pm 0.023}$	$1.242_{\pm 0.018}$
Bernoulli Adj + All-Pairs	$0.950_{\pm 0.003}$	$0.926_{\pm 0.027}$	$1.103_{\pm 0.032}$	$1.228_{\pm 0.020}$	$1.253_{\pm 0.015}$
No Recurrent Adj	$1.238_{\pm 0.001}$	$1.242_{\pm 0.007}$	$1.275_{\pm 0.008}$	$1.303_{\pm 0.006}$	$1.310_{\pm 0.005}$
No h^+ Correction, No Graph State, No Recurrent Adj	$1.238_{\pm 0.001}$	$1.245_{\pm 0.002}$	$1.363_{\pm 0.023}$	$1.496_{\pm 0.046}$	$1.542_{\pm 0.048}$
			OD		
Full	$0.449_{\pm 0.042}$	$0.426_{\pm 0.041}$	$0.402_{\pm 0.037}$	$0.413_{\pm 0.027}$	$0.492_{\pm 0.022}$
No Graph State	$0.428_{\pm 0.037}$	$0.441_{\pm 0.034}$	$0.450_{\pm 0.029}$	$0.465_{\pm 0.027}$	$0.523_{\pm 0.024}$
$\beta_h = 0$	$0.441_{\pm 0.019}$	$0.425_{\pm 0.018}$	$0.413_{\pm 0.019}$	$0.432_{\pm 0.019}$	$0.508_{\pm 0.019}$
No $\boldsymbol{h}^+$ Correction	$0.730_{\pm 0.028}$	$0.716_{\pm 0.024}$	$0.696_{\pm 0.033}$	$0.690_{\pm 0.046}$	$0.708_{\pm 0.044}$
Bernoulli Adj	$0.549_{\pm 0.043}$	$0.529_{\pm 0.040}$	$0.485_{\pm 0.045}$	$0.446_{\pm 0.052}$	$0.474_{\pm 0.052}$
All-Pairs	$0.539_{\pm 0.019}$	$0.522_{\pm 0.018}$	$0.476_{\pm 0.018}$	$0.439_{\pm 0.020}$	$0.477_{\pm 0.019}$
Bernoulli Adj + All-Pairs	$0.559_{\pm 0.027}$	$0.538_{\pm 0.025}$	$0.498_{\pm 0.030}$	$0.467_{\pm 0.034}$	$0.499_{\pm 0.037}$
No Recurrent Adj	$0.977_{\pm 0.006}$	$1.001_{\pm 0.005}$	$1.028_{\pm 0.005}$	$1.062_{\pm 0.005}$	$1.081_{\pm 0.004}$
No h^+ Correction, No Graph State, No Recurrent Adj	$0.980_{\pm 0.007}$	$1.004_{\pm 0.011}$	$1.031_{\pm 0.017}$	$1.070_{\pm 0.024}$	$1.089_{\pm 0.026}$
			SAR		
Full	$0.917_{\pm 0.023}$	$0.917_{\pm 0.028}$	$0.928_{\pm 0.026}$	$0.887_{\pm 0.032}$	$0.878_{\pm 0.035}$
No Graph State	$0.865_{\pm 0.026}$	$0.872_{\pm 0.027}$	$0.925_{\pm 0.043}$	$0.941_{\pm 0.046}$	$1.001_{\pm 0.045}$
$\beta_h = 0$	$0.877_{\pm 0.020}$	$0.865_{\pm 0.024}$	$0.871_{\pm 0.024}$	$0.844_{\pm 0.030}$	$0.835_{\pm 0.032}$
No $\boldsymbol{h}^+$ Correction	$0.937_{\pm 0.020}$	$0.938_{\pm 0.016}$	$0.962_{\pm 0.015}$	$0.927_{\pm 0.019}$	$0.920_{\pm 0.018}$
Bernoulli Adj	$0.958_{\pm 0.023}$	$0.948_{\pm 0.025}$	$0.942_{\pm 0.026}$	$0.936_{\pm 0.029}$	$0.920_{\pm 0.033}$
All-Pairs	$0.977_{\pm 0.015}$	$0.966_{\pm 0.014}$	$0.972_{\pm 0.016}$	$0.955_{\pm 0.017}$	$0.946_{\pm 0.018}$
Bernoulli Adj + All-Pairs	$0.982_{\pm 0.010}$	$0.971_{\pm 0.012}$	$0.966_{\pm 0.013}$	$0.957_{\pm 0.014}$	$0.940_{\pm 0.017}$
No Recurrent Adj	$1.123_{\pm 0.007}$	$1.159_{\pm 0.007}$	$1.207_{\pm 0.007}$	$1.256_{\pm 0.008}$	$1.288_{\pm 0.009}$
No h^+ Correction, No Graph State, No Recurrent Adj	$1.065_{\pm 0.009}$	$1.124_{\pm 0.005}$	$1.201_{\pm 0.005}$	$1.264_{\pm 0.010}$	$1.307_{\pm 0.019}$
			CF		
Full	$0.914_{\pm 0.038}$	$0.890_{\pm 0.043}$	$1.059_{\pm 0.034}$	$1.222_{\pm 0.033}$	$1.249_{\pm 0.034}$
No Graph State	$1.049_{\pm 0.039}$	$1.028_{\pm 0.041}$	$1.089_{\pm 0.027}$	$1.152_{\pm 0.036}$	$1.208_{\pm 0.064}$
$\beta_h = 0$	$0.903_{\pm 0.024}$	$0.885_{\pm 0.033}$	$1.045_{\pm 0.042}$	$1.171_{\pm 0.053}$	$1.244_{\pm 0.061}$
No $\boldsymbol{h}^+$ Correction	$0.932_{\pm 0.019}$	$0.952_{\pm 0.028}$	$1.133_{\pm 0.039}$	$1.225_{\pm 0.046}$	$1.276_{\pm 0.044}$
Bernoulli Adj	$1.002_{\pm 0.042}$	$1.029_{\pm 0.051}$	$1.180_{\pm 0.053}$	$1.317_{\pm 0.063}$	$1.396_{\pm 0.072}$
All-Pairs	$1.018_{\pm 0.041}$	$1.030_{\pm 0.049}$	$1.136_{\pm 0.057}$	$1.244_{\pm 0.072}$	$1.301_{\pm 0.086}$
Bernoulli Adj + All-Pairs	$1.035_{\pm 0.046}$	$1.062_{\pm 0.046}$	$1.150_{\pm 0.044}$	$1.216_{\pm 0.063}$	$1.248_{\pm 0.074}$
No Recurrent Adj	$1.239_{\pm 0.006}$	$1.239_{\pm 0.011}$	$1.332_{\pm 0.033}$	$1.396_{\pm 0.060}$	$1.396_{\pm 0.075}$
No h^+ Correction, No Graph State, No Recurrent Adj	$1.200_{\pm 0.015}$	$1.224_{\pm 0.008}$	$1.286_{\pm 0.011}$	$1.351_{\pm 0.025}$	$1.386_{\pm 0.042}$