Agentic-JEPA: A Self-Supervised World Model for Planning in Text-Based Agent Environments

Francisco-Javier Rodrigo-Ginés
NLP & IR Group
Universidad Nacional de Educación a Distancia (UNED)
frodrigo@invi.uned.es

Abstract

Large language model (LLM) agents select actions through next-token prediction or chain-of-thought reasoning, without an explicit model of how those actions transform the world. We ask whether the Joint Embedding Predictive Architecture (JEPA), originally proposed for self-supervised visual representation learning, can serve as the foundation for a reward-free world model in the text-based agent setting, where observations and actions are natural-language strings.

We present AGENTIC-JEPA, a system that trains a lightweight transformer predictor to map (state, action) pairs to next-state embeddings, using decoder-only LLM backbones as state and action encoders and an EMA target encoder to prevent representation collapse. At inference time, the agent plans without any reward signal: it scores candidate actions via cosine similarity between their predicted outcomes and a goal embedding, with optional k-step lookahead.

On a controlled benchmark of 17 text-based environments spanning three transition dynamics (linear, branching, stochastic) and a strict 10 train / 7 held-out test split, our experiments reveal a sharp generalization boundary. AGENTIC-JEPA achieves 100% success and near-oracle planning efficiency on all in-distribution environments, demonstrating for the first time that JEPA can learn text-environment dynamics from self-supervised trajectories alone, yet fails completely (0% success) on every out-of-distribution environment, regardless of dynamics type. We formalize two necessary conditions for cross-environment transfer (*encoder coverage* and *structural alignment*), and show, including through a t-SNE visualization of the learned latent space, that neither holds: the encoder partitions representations by environment identity rather than by functional role. Two secondary findings round out the picture: k-step lookahead *degrades* performance (100% to 40% at k=3) due to compounding prediction error, and scaling the backbone from GPT-2 to a 4-bit-quantized Qwen 2.5-1.5B does not recover OOD success.

These findings characterize both the promise and the current boundaries of self-supervised world models for text-based agents, and turn the 0% OOD result from a flat negative result into a diagnostic with concrete, testable targets for future work. Code, environments, and training trajectories are publicly available.[1]

1 Introduction

The deployment of large language models (LLMs) as autonomous agents has become a central research direction [Wang et al., 2024, Xi et al., 2023]. Systems such as ReAct [Yao et al., 2023], Reflexion [Shinn et al., 2023], and tool-augmented LLMs [Schick et al., 2023] demonstrate that language models can execute multi-step workflows, call external tools, and recover from errors

[1] https://github.com/franfj/agentic-jepa

Preprint.

with verbal self-reflection. However, these agents fundamentally select actions through next-token prediction or chain-of-thought reasoning [Wei et al., 2022], without an explicit model of how actions transform the world. Without the ability to simulate outcomes internally, agents resort to trial-and-error behaviour, redundant API calls, and poor sample efficiency [Hao et al., 2023]. This stands in stark contrast to humans, who routinely plan by imagining the consequences of candidate actions before executing them, and to model-based reinforcement-learning (RL) agents, which exploit a learned environment model to evaluate plans without expensive real-world rollouts [Ha and Schmidhuber, 2018, Hafner et al., 2020].

An alternative paradigm is to learn a *world model*: an internal representation that predicts the consequences of actions before they are executed [Ha and Schmidhuber, 2018, LeCun, 2022]. In the visual domain, this approach has produced increasingly capable agents. DreamerV3 masters more than 150 diverse environments with a single algorithm through latent imagination [Hafner et al., 2025]; MuZero combines a learned model with Monte-Carlo tree search to play Atari, Go, chess, and shogi at superhuman level [Schrittwieser et al., 2020]; IRIS frames environment dynamics as discrete sequence prediction and achieves strong sample efficiency [Micheli et al., 2023]. A common theme underlying recent successes is the move away from pixel-level reconstruction towards *latent* prediction: LeCun [2022] formalized this idea in the Joint Embedding Predictive Architecture (JEPA), arguing that intelligent systems should predict in abstract representation spaces rather than reconstruct raw observations. I-JEPA [Assran et al., 2023] and V-JEPA [Bardes et al., 2024] have since shown that this paradigm scales to images and videos.

The question we ask in this work is whether the same paradigm transfers to *text-based* agent environments, where observations and actions are natural-language strings. Text is an attractive testbed for three reasons. First, it captures the modality in which most current LLM agents operate (chat interfaces, tool descriptions, workflow specifications). Second, controlled text environments such as ALFWorld [Shridhar et al., 2021] and ScienceWorld [Wang et al., 2022] allow precise manipulation of structural properties (branching, stochasticity, horizon). Third, the same backbones that drive LLM agents (decoder-only transformers) can be reused as encoders, opening a path to integrating world-model planning with existing agent stacks at negligible additional cost.

We present AGENTIC-JEPA, the first application of the JEPA architecture to action-conditioned prediction in text-based agent environments. AGENTIC-JEPA (1) uses decoder-only LLM backbones as state and action encoders, with last-token pooling for embeddings; (2) trains a lightweight transformer predictor to map the concatenated (state, action) embedding to the next-state embedding under a cosine-similarity objective, with an EMA target encoder to prevent representation collapse; and (3) plans at inference time by scoring candidate actions via cosine similarity between predicted outcomes and a goal embedding, with optional k-step lookahead rollouts. The system is reward-free: it learns dynamics entirely from unlabelled trajectories and plans by goal-similarity matching, replacing the value or reward functions used by classical model-based RL.

To understand both the capabilities and the limits of this approach, we design a controlled benchmark of 17 text-based environments organized along two axes: *transition dynamics* (13 linear, 2 branching, 2 stochastic) and *generalization regime* (10 in-distribution training environments and 7 held-out out-of-distribution test environments). Each environment is a goal-conditioned Markov Decision Process whose states and actions are natural-language strings. Our experiments reveal a sharp generalization boundary. On in-distribution environments, AGENTIC-JEPA achieves 100% success and near-oracle planning efficiency (0.887 vs. 0.990 oracle), matching the oracle on 7 of 10 environments. On the 7 held-out test environments, the same model fails completely (0% success), regardless of dynamics type. We further document two secondary findings: multi-step planning with a single-step predictor degrades performance (100% to 40% at k=3) because errors compound through the rollout, and scaling the backbone from GPT-2 (124M parameters) to a 4-bit quantized Qwen 2.5-1.5B yields 0% success across the board, indicating that the bottleneck is representation precision rather than capacity.

Rather than treating the 0% out-of-distribution result as a flat negative result, we use it as a diagnostic. We formalize the conditions under which world-model transfer can succeed (Proposition 1), identifying two necessary requirements: *encoder coverage*, that test states must fall inside the support of the training latent distribution, and *structural alignment*, that the predictor must learn transition dynamics that depend on state and action *content* rather than on environment identity. A t-SNE visualization of the learned representations (Section 7.1) shows that neither condition holds in practice: the encoder

produces ten compact in-distribution clusters and seven disjoint out-of-distribution clusters with no shared support. This reframes the result from a global failure into a concrete agenda: future work should target embedding-space coverage and structural alignment directly.

Our contributions are threefold:

1. **First JEPA world model for text-based agents.** We present AGENTIC-JEPA, the first application of the JEPA architecture to action-conditioned prediction in text-based agent environments. Unlike prior world models for agent planning (DreamerV3 [Hafner et al., 2025], IRIS [Micheli et al., 2023], MuZero [Schrittwieser et al., 2020], TransDreamer [Chen et al., 2022]), which operate on visual observations and require reward signals, AGENTIC-JEPA is the first *self-supervised, reward-free* world model that operates on text, learning environment dynamics entirely from unlabelled trajectories and planning by goal-similarity matching.
2. **First systematic analysis of generalization failure in text world models.** Through evaluation on 17 environments with an explicit in-distribution/out-of-distribution split, we uncover a sharp generalization boundary (100% ID success vs. 0% OOD success) and provide a formal characterization of the necessary conditions for transfer (Proposition 1, Section 3.3), together with empirical evidence that neither condition holds for text-based JEPA. This is the first work to isolate and formally analyze why learned world models fail to generalize across text environments, and to identify embedding-space fragmentation as the key obstacle.
3. **A controlled benchmark and an empirical study of failure modes.** We design a benchmark of 17 environments spanning linear, branching, and stochastic dynamics, with both identifier-style and natural-language action descriptions, and use it to conduct a systematic empirical study revealing that (a) environment-specific memorization and embedding-space fragmentation are the root causes of generalization failure; (b) k-step planning with single-step predictors compounds errors rather than improving lookahead; and (c) backbone scaling via 4-bit quantization does not recover OOD performance. Code, environments, and training trajectories are publicly released to support follow-up work.

The remainder of the paper is organized as follows. Section 2 situates AGENTIC-JEPA relative to visual world models, JEPA variants, and LLM-based agents. Section 3 formalizes text-based agent planning as a multi-environment MDP and states the necessary conditions for world-model transfer. Section 4 describes the AGENTIC-JEPA architecture, training procedure, and inference-time planning algorithm. Section 5 details the benchmark, baselines, and protocol. Sections 6–7 report quantitative and qualitative results. Sections 8–10 discuss implications, limitations, and directions for closing the generalization gap.

2 Related Work

2.1 World Models and Self-Supervised Dynamics

World models learn to predict future states from observations and actions, enabling agents to plan by simulating outcomes internally rather than executing them in the environment [Ha and Schmidhuber, 2018]. This idea has driven a sustained progression in model-based RL. Dreamer [Hafner et al., 2020] introduced latent imagination for visual control, learning a recurrent state-space model and training the policy entirely on imagined rollouts. DreamerV2 [Hafner et al., 2021] extended the approach with discrete latents and demonstrated Atari-level competence with a single algorithm. DreamerV3 [Hafner et al., 2025] subsequently scaled to more than 150 diverse environments without per-task hyperparameter tuning, providing the strongest evidence to date that a single latent world-model recipe can master a broad range of domains. MuZero [Schrittwieser et al., 2020] combined a learned latent dynamics model with Monte-Carlo tree search and reached state-of-the-art results on Atari, Go, chess, and shogi, showing that planning over learned dynamics can rival explicit-model search. IRIS [Micheli et al., 2023] framed environment dynamics as discrete sequence prediction with an autoencoder and a transformer, demonstrating that transformer-based world models can match or exceed recurrent ones in sample efficiency. TransDreamer [Chen et al., 2022] replaced the recurrent backbone of Dreamer with a transformer to handle longer temporal dependencies.

A complementary line of work uses self-supervised objectives to improve agent representations without explicitly building a world model for planning. SPR [Schwarzer et al., 2021] predicts future latent states conditioned on actions, the closest prior work to ours in spirit, but operates within an RL loop with reward-based policy optimization. CURL [Srinivas et al., 2020] applies contrastive learning to RL pixel observations, while contrastive methods such as SimCLR [Chen et al., 2020] and momentum-based methods such as MoCo [He et al., 2020] and BYOL [Grill et al., 2020] provide the broader self-supervised foundation that JEPA-style architectures build upon. LeCun [2022] argued that world models should predict in abstract representation spaces rather than reconstruct raw observations, proposing JEPA as a foundation for self-supervised world models. Our work follows this principle but applies it to text-based environments, using LLM encoders and planning without any reward signal.

A persistent finding in model-based RL is that learned dynamics often *overfit* to the training distribution. Zhang et al. [2018] showed that deep RL agents overfit to specific environment instances, failing on structurally similar test levels, and that the gap can be substantial even when train and test environments are sampled from the same generative procedure. Our work extends this observation to JEPA-based world models for text environments and provides a formal characterization of when transfer can succeed.

2.2 Joint Embedding Predictive Architectures

JEPA [LeCun, 2022] learns representations by predicting the embedding of a target view from a context view, using an EMA target encoder to prevent representation collapse [Grill et al., 2020]. The crucial design choice, relative to reconstruction-based methods such as autoencoders, is that the prediction target lives in a learned latent space rather than in the raw input space, allowing the predictor to abstract away unpredictable low-level details. I-JEPA [Assran et al., 2023] applied this principle to images, predicting masked patch embeddings from visible context and obtaining representations competitive with masked image modelling while requiring substantially fewer pre-training updates. V-JEPA [Bardes et al., 2024] extended the framework to video, learning spatiotemporal representations by predicting future frame embeddings and demonstrating that the predict-in-latent-space recipe scales to temporal data. Related work on representation regularization (VICReg [Bardes et al., 2022]) provides alternative ways to avoid collapse without an EMA target.

These works focus on *perception*: representing the world from observations alone. Our work extends JEPA to the *action-conditioned* setting: instead of predicting a masked view or a future frame from context alone, we predict the next state given the current state *and* a candidate action, transforming JEPA from a perceptual architecture into a world model for planning. This shift in objective imposes new constraints. First, the predictor must be sensitive to the action input rather than treating it as a perturbation to be averaged out. Second, the encoder must support cross-sample retrieval via cosine similarity to a goal embedding, not merely linear probing or fine-tuning. Third, the model is evaluated by its *downstream planning performance*, not by representation-quality proxies such as linear classification accuracy.

2.3 Text-Based Environments and LLM Agents

Text-based environments provide controlled settings for studying agent planning, free from the perceptual difficulties of pixel-level reasoning. ALFWorld [Shridhar et al., 2021] aligns text-based and embodied environments for household tasks, providing a transfer benchmark between symbolic and sensorimotor agents. ScienceWorld [Wang et al., 2022] presents multi-step science experiments that require world knowledge and procedural reasoning, while WebShop [Yao et al., 2022] grounds language-based decision making in realistic web-shopping interactions. AgentBench [Liu et al., 2024] provides a comprehensive benchmark spanning eight distinct environments for evaluating LLM agents. Our benchmark of 17 custom environments differs in purpose: we isolate specific structural properties (linear, branching, stochastic dynamics) and explicitly separate training and held-out test environments to enable controlled generalization study.

LLM agents use language models to perceive, reason, and act in interactive environments [Wang et al., 2024, Xi et al., 2023]. ReAct [Yao et al., 2023] interleaves reasoning traces with actions, Reflexion [Shinn et al., 2023] adds verbal self-reflection to improve behaviour across episodes, Inner Monologue [Huang et al., 2022] integrates feedback from various sources into the planning process,

and Hao et al. [2023] frame LLM reasoning as planning with an implicit world model recovered from pre-training. A second line of work uses tool-augmented LLMs that learn to call external functions [Schick et al., 2023] or that are aligned to follow instructions through RL from human feedback [Ouyang et al., 2022]. These approaches treat the LLM itself as an implicit world model, relying on pre-trained knowledge to predict action consequences. AGENTIC-JEPA takes a complementary approach: we train an *explicit, lightweight* world model from environment interactions, offering efficient planning (a single forward pass per action) at the cost of limited generalization beyond the training distribution. The two paradigms are not mutually exclusive; hybrid systems that use AGENTIC-JEPA-style world models for fast in-distribution planning and LLM reasoning for novel situations are a natural direction (Section 8).

3 Problem Formulation

We formalize the setting of text-based agent planning and define the world-model learning and inference objectives. This section also establishes the necessary conditions for cross-environment transfer that drive our subsequent analysis.

3.1 Text-Based Agent Environments as Goal-Conditioned MDPs

We model text-based agent environments as goal-conditioned Markov Decision Processes (MDPs) [Puterman, 1994] operating over natural language. An environment is a tuple $\mathcal{E} = (\mathcal{S}, \mathcal{A}, T, \mathcal{V}, g)$, where $\mathcal{S}$ is the (countably infinite) set of natural-language state descriptions, $\mathcal{A}$ is the (countably infinite) set of action descriptions, $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, $\mathcal{V} : \mathcal{S} \to 2^{\mathcal{A}}$ maps each state to its set of valid actions, and $g \in \mathcal{S}$ is the goal state. At each step t, the agent observes the current state s_t and the valid action set $A_t = \mathcal{V}(s_t)$, selects $a_t \in A_t$, and transitions to $s_{t+1} \sim T(s_t, a_t)$. An episode succeeds if the agent reaches g within a horizon of H steps. Crucially, no scalar reward is provided: the only supervision signal at inference time is the goal description g.

We consider a *multi-environment* setting: the agent has access during training to a set $\mathcal{E}_{\text{train}} = \{\mathcal{E}_1, \ldots, \mathcal{E}_N\}$ of N environments and is evaluated on a disjoint set of held-out environments $\mathcal{E}_{\text{test}}$, with $\mathcal{E}_{\text{train}} \cap \mathcal{E}_{\text{test}} = \emptyset$. Training and test environments share the same generative *template* (linear, branching, or stochastic dynamics over natural-language states) but use different domains, vocabulary, and action descriptions. This setting tests whether a learned world model has acquired transferable structural knowledge or has merely memorized environment-specific transitions.

3.2 World Model Learning Objective

Let $\phi : \mathcal{S} \to \mathbb{R}^d$ be a state encoder and $\psi : \mathcal{A} \to \mathbb{R}^d$ an action encoder, both implemented as decoder-only LLMs followed by last-token pooling. We seek a predictor $P : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ such that:

$$P(\phi(s_t),\, \psi(a_t)) \approx \phi(s_{t+1}), \tag{1}$$

where approximation is measured by cosine similarity in $\mathbb{R}^d$. Equivalently, P is trained to minimize the cosine-embedding loss

$$\mathcal{L}(\phi, \psi, P) \;=\; \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}_{\text{train}}} \Big[1 - \cos\_\text{sim}(P(\phi(s_t), \psi(a_t)),\, \phi_{\bar{\theta}}(s_{t+1})) \Big], \tag{2}$$

where $\mathcal{D}_{\text{train}}$ is a dataset of transitions collected from $\mathcal{E}_{\text{train}}$ and $\phi_{\bar{\theta}}$ is an EMA copy of ϕ that provides stable, gradient-free targets (Section 4). The use of an EMA target is essential to avoid the trivial solution in which ϕ collapses to a constant vector and the predictor matches it perfectly [Grill et al., 2020].

Planning by goal similarity. At inference time, given the current state s_t, the goal g, and the valid action set A_t, the agent selects the action whose predicted outcome is closest to the goal embedding:

$$a_t^* \;=\; \arg\max_{a \in A_t} \; \cos\_\text{sim}(P(\phi(s_t),\, \psi(a))\,,\; \phi(g))\,. \tag{3}$$

This replaces the reward-based action-value maximization of classical model-based RL with a similarity-based objective. The agent never receives a reward signal, neither during training nor at inference: planning is entirely driven by the geometry of the learned embedding space. For multi-step

planning with horizon k, the agent performs greedy rollouts of length k from the predicted next state and scores the original candidate by the cosine similarity of the *final* predicted state to the goal embedding (Section 4.3).

3.3 Generalization Conditions

The central scientific question of this paper is when a world model trained on $\mathcal{E}_{\text{train}}$ generalizes to a disjoint set of environments $\mathcal{E}_{\text{test}}$. We provide a formal answer in the form of two necessary conditions.

Proposition 1 (Necessary conditions for world model transfer)**.** *Let (P, ϕ, ψ) be a world model trained on $\mathcal{E}_{train}$ with the objective in Equation (2), and assume the predictor P is L-Lipschitz in its first argument. For non-trivial planning success on $\mathcal{E}_{test}$ via the rule in Equation (3), the following conditions must hold:*

1. ***Encoder coverage.*** *The state encoder ϕ must map test states into regions of $\mathbb{R}^d$ that are close to the support of training embeddings; formally, $\phi(s) \in B_\epsilon(\text{supp}(\phi(\mathcal{S}_{train})))$ for some tolerance $\epsilon > 0$, where B_ϵ denotes the ϵ-thickening of a set.*

2. ***Structural alignment.*** *The predictor P must have learned transition dynamics that depend on the* content *of state and action embeddings rather than on environment identity; equivalently, for any pair of structurally equivalent transitions $(s_t, a_t, s_{t+1}) \in \mathcal{E}_i$ and $(s'_t, a'_t, s'_{t+1}) \in \mathcal{E}_j$ such that $\phi(s_t) \approx \phi(s'_t)$ and $\psi(a_t) \approx \psi(a'_t)$, we must have $P(\phi(s_t), \psi(a_t)) \approx P(\phi(s'_t), \psi(a'_t))$.*

Sketch of why both conditions are necessary. If condition 1 fails, the test embedding $\phi(s)$ lies outside the training support, and the predictor P is then queried on inputs for which it has no Lipschitz-controlled guarantee on the output: the predicted next-state embedding can drift arbitrarily far from the actual $\phi(s_{t+1})$, so the cosine score in Equation (3) becomes uninformative. If condition 2 fails, the predictor has effectively learned N separate transition functions P_i indexed by environment identity rather than a single transition function in embedding space: queries on test environments fall outside the domain of any P_i, again yielding uninformative scores.

What our experiments reveal. Our experiments show that, in the text-based setting, *neither* condition holds in practice (Section 7.1). The encoder learns environment-specific clusters in latent space (violating condition 1, as visualized by the t-SNE plot in Figure 4), and the predictor memorizes environment-specific transitions rather than learning a content-conditioned transition function (violating condition 2). We attribute this to a structural property of text input: each environment uses a distinct vocabulary and naming convention, which causes the LLM backbone, dominated by its pretraining bias towards lexical similarity, to map structurally identical transitions to distant regions of embedding space. The result is a fragmented embedding space in which the predictor learns cluster-specific dynamics rather than a general transition function. Proposition 1 is therefore not merely a description of failure: it identifies two concrete, actionable targets (latent coverage and structural alignment) that future work can address through training-distribution diversification, action-description normalization, or auxiliary objectives that enforce environment invariance.

4 Method

We now describe the AGENTIC-JEPA architecture, training procedure, and inference-time planning algorithm. Figure 1 provides an overview. The design follows three guiding principles. First, prediction happens in a learned latent space rather than in the raw text space, eliminating the need for an autoregressive decoder and removing reconstruction noise from the learning signal [LeCun, 2022]. Second, both encoders are decoder-only LLMs, so that the world model can be plugged on top of the same backbones already used by LLM agents without retraining a separate observation model. Third, planning is driven entirely by goal similarity (Equation (3)), with no value or reward function: the system is self-supervised end to end.

4.1 Architecture

AGENTIC-JEPA consists of four components: a state encoder, an action encoder, a predictor, and an EMA target encoder.

State encoder ϕ_θ. A decoder-only LLM backbone (GPT-2 [Radford et al., 2019] or Qwen 2.5 [Yang et al., 2024]) maps a state description $s \in \mathcal{S}$ to $\mathbf{h}_s = \phi_\theta(s) \in \mathbb{R}^d$ via last-token pooling: we feed the natural-language state through the backbone and take the hidden state of the final token as the state embedding. This choice mirrors the way LLM agents already represent textual observations and exposes the same internal representation space that downstream tools or reasoning steps can consume.

Action encoder ψ_θ. An independent decoder-only backbone of the same architecture maps an action description $a \in \mathcal{A}$ to $\mathbf{h}_a = \psi_\theta(a) \in \mathbb{R}^d$, again via last-token pooling. We deliberately use separate parameters from the state encoder. A shared encoder, in early experiments, caused the model to collapse state and action representations into a single mode, harming planning performance. Separating the encoders preserves the asymmetry between observations and actions that is central to action-conditioned prediction.

Predictor P_ω. A lightweight transformer (L layers, H heads, hidden dimension d_p) that takes the concatenated state-action embedding $[\mathbf{h}_{s_t}; \mathbf{h}_{a_t}] \in \mathbb{R}^{2d}$ as a length-2 token sequence (one state token, one action token) and outputs the predicted next-state embedding:

$$\hat{\mathbf{h}}_{s_{t+1}} = P_\omega([\mathbf{h}_{s_t}; \mathbf{h}_{a_t}]) \in \mathbb{R}^d. \tag{4}$$

We chose a transformer over a simpler MLP for two reasons. First, the attention layer allows the predictor to selectively combine information from the state and action embeddings rather than blindly concatenating them. Second, a small transformer scales naturally to multi-action inputs (e.g., for ablations with auxiliary action descriptions), keeping the architecture extensible.

Target encoder $\phi_{\bar{\theta}}$ (EMA). An exponential moving average copy of the state encoder, providing stable regression targets:

$$\bar{\theta} \leftarrow m \cdot \bar{\theta} + (1 - m) \cdot \theta, \tag{5}$$

where the momentum m increases linearly from 0.996 to 1.0 over a warmup period. The EMA target plays a critical role: without it, the gradient from Equation (2) flows into both ϕ and the target, and the model can trivially minimize the loss by collapsing ϕ to a constant embedding [Grill et al., 2020]. Anchoring the target on a slowly updated copy of ϕ breaks this shortcut and allows the encoder to develop genuinely action-sensitive representations.

4.2 Training

Training uses trajectory data collected by oracle and random policies in the training environments, with each sample being a transition (s_t, a_t, s_{t+1}). For each environment, we collect 200 oracle trajectories (correct actions only) and 400 random trajectories (uniform action selection), yielding $\sim$130,000 transitions across the 10 training environments after a 90/10 train/validation split. This mix is intentional. Oracle trajectories ensure that the model sees correct stage progressions; random trajectories ensure that it also observes the (much more common) effect of incorrect actions, which is essential for the planning rule in Equation (3) to discriminate good from bad candidates. Test environments are never used for data generation.

The loss is the cosine-embedding loss from Equation (2), computed on per-transition predictions:

$$\mathcal{L} = 1 - \cos\_\text{sim}\Big(\hat{\mathbf{h}}_{s_{t+1}},\ \phi_{\bar{\theta}}(s_{t+1})\Big), \tag{6}$$

where $\phi_{\bar{\theta}}(s_{t+1})$ is computed by the EMA encoder with no gradient. Only the state encoder, action encoder, and predictor receive gradients; the EMA encoder is updated by the rule in Equation (5).

We train with AdamW [Loshchilov and Hutter, 2019], a cosine learning-rate schedule with linear warmup, a weight decay of 0.05, and gradient clipping at norm 1.0. Full hyperparameters are listed in Appendix C.

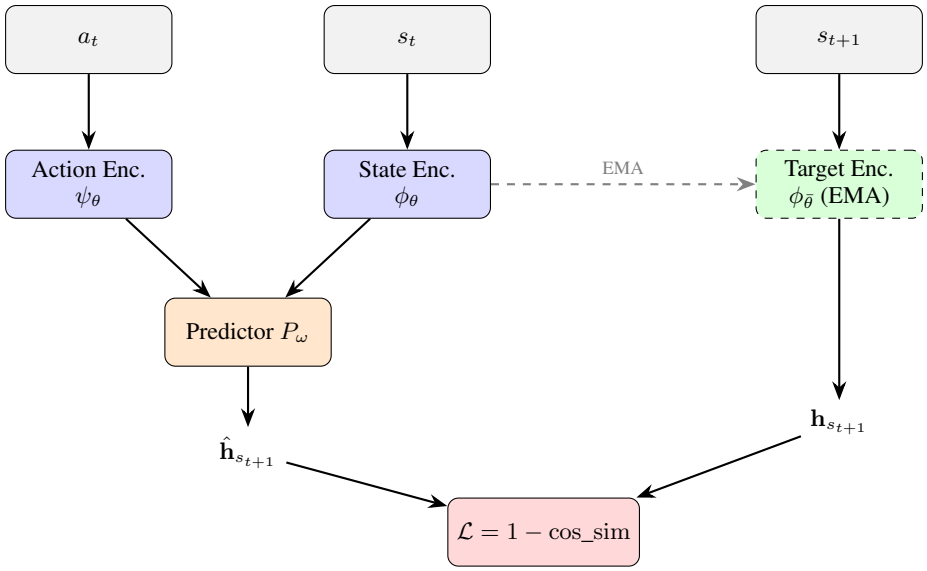

Figure 1: Overview of the AGENTIC-JEPA training architecture. The state encoder ϕ_θ and the action encoder ψ_θ produce latent embeddings that the predictor P_ω combines to predict the next-state embedding $\hat{\mathbf{h}}_{s_{t+1}}$. The target encoder $\phi_{\bar{\theta}}$ is an exponential moving-average copy of the state encoder and produces gradient-free targets $\mathbf{h}_{s_{t+1}}$. The cosine-embedding loss $\mathcal{L} = 1 - \cos\_\text{sim}(\hat{\mathbf{h}}_{s_{t+1}}, \mathbf{h}_{s_{t+1}})$ is back-propagated through the encoders and predictor; the EMA target encoder is updated by the rule in Equation (5).

Why cosine rather than mean-squared error. We chose cosine similarity over L_2 regression for the prediction loss because of how the targets are produced. LLM embeddings have heavy-tailed norms across different inputs; an L_2 target would force the predictor to match both the norm and the direction of the EMA embedding, conflating representation *content* with representation *scale*. Cosine similarity factors out the norm and focuses the learning signal on the angular geometry of the embedding space, which is also the geometry that the inference-time planning rule in Equation (3) operates on. This alignment between training objective and inference rule is a deliberate design choice: the predictor is optimized for exactly the quantity that the planner will subsequently query.

4.3 Inference-Time Planning

One-step planning. Given the current state s_t, the goal g, and the valid action set A_t, the agent encodes each candidate action $a \in A_t$ with ψ_θ, predicts its outcome via P_ω, and selects the action whose predicted next-state is closest to the goal embedding $\phi_{\bar{\theta}}(g)$ by cosine similarity, following Equation (3). This requires $|A_t|$ predictor calls per decision (one per candidate action), each of which is a single forward pass through a small transformer ($\sim$10ms on a single GPU in our experiments).

Multi-step planning (k-step lookahead). For each candidate action a_i, the agent performs a k-step greedy rollout in latent space: it predicts the next state via P_ω, then greedily selects the best action from the *predicted* state by repeating the same scoring rule, and iterates. The candidate is finally scored by the cosine similarity between the *final* predicted state (after k steps) and the goal embedding. This requires $|A_t|^k$ predictor calls per decision if expanded fully, but our greedy implementation reduces it to $|A_t| \cdot k$. We evaluate $k \in \{1, 3\}$. A priori, multi-step planning could help on long-horizon tasks by allowing the model to anticipate dead-ends a few steps ahead. In practice, it degrades performance because errors compound through the rollout (Section 6); we discuss this finding in detail in Section 7.1.

5 Experimental Setup

5.1 Environments

We design a benchmark of 17 text-based environments organized along two dimensions: *transition dynamics* (linear, branching, stochastic) and *generalization regime* (10 training, 7 held-out test).

Transition dynamics. Linear environments (13 total: 10 in-distribution, 3 out-of-distribution[2]) are deterministic chains in which one action advances the state to the next stage and all other actions leave the state unchanged. Branching environments (2 total: 1 ID, 1 OOD) replace the chain with a directed acyclic graph: at certain decision points, two or more valid actions advance the state along different but eventually converging paths; the oracle is defined as one specific path. Stochastic environments (2 total: 1 ID, 1 OOD) add probabilistic failure: the correct action succeeds with probability $1 - p$ for $p \in [0.10, 0.20]$ that varies by stage; failed actions return the agent to the current stage with an explanatory message. This three-way split tests whether AGENTIC-JEPA can learn each of the three structural patterns separately and whether the learned model transfers across structurally similar environments with different vocabulary.

Action descriptions. Half of the environments use short identifier-style actions (e.g., `read_diff`, `scan_inbox`); the other half use full natural-language action descriptions (e.g., "Reproduce the bug by following the steps described in the report"). This deliberate diversity allows us to test whether the encoder picks up surface lexical cues or genuinely abstracts over action semantics.

Train/test split. We use a strict environment-level split: 10 environments are used exclusively for training (in-distribution) and 7 environments are used exclusively for evaluation (out-of-distribution). Training trajectories never visit test environments, and test trajectories never visit training environments. The test split spans linear, branching, and stochastic dynamics, mirroring the structural diversity of the training split.

Full specifications, including domain, horizon, action style, and dynamics type per environment, are provided in Appendix B (Table 8).

5.2 Agents and Baselines

We compare five agents:

Random. A baseline that selects uniformly at random from the valid action set $\mathcal{V}(s_t)$ at each step. Despite its simplicity, this baseline turns out to be very strong on our benchmark (Section 6), because most environments have a tractable horizon and a small action set, so random walks frequently reach the goal within 50 steps.

GreedyText. A non-learned heuristic that selects the action whose textual TF-IDF representation has the highest cosine similarity to the goal description. This baseline tests whether shallow lexical matching is enough to solve the environments and provides a sanity check that the benchmark is not trivially solvable by text-similarity heuristics.

GPT2-Vanilla. An ablation that uses the same AGENTIC-JEPA architecture and inference rule (Equation (3)) but with an *untrained* GPT-2 backbone and *untrained* predictor. This baseline isolates the contribution of the self-supervised training step: any improvement of full AGENTIC-JEPA over GPT2-Vanilla can be attributed to the training procedure rather than to the pretrained backbone alone.

JEPA ($k \in \{1, 3\}$). The full system described in Section 4, with the planning rule from Equation (3) ($k = 1$) or with k-step lookahead rollouts ($k = 3$).

Oracle. An upper bound that uses ground-truth knowledge of the environment dynamics to select the optimal action at every step. The oracle is not a competing agent; we report it to calibrate the difficulty of each environment.

[2]Plus four additional out-of-distribution linear environments shared across the linear/branching/stochastic counts.

5.3 Metrics

We evaluate using four metrics, complementing each other to give a complete picture of planning quality.

Success rate. The fraction of episodes in which the agent reaches the goal state g within the horizon $H = 50$. This is our primary metric.

Steps to goal. The mean number of steps taken to reach the goal across successful episodes (capped at H for unsuccessful ones).

Planning efficiency. The ratio of the oracle's number of steps to the agent's number of steps, $\text{eff} = H^{\text{oracle}}/H^{\text{agent}}$, with $\text{eff} = 0$ for unsuccessful episodes. An efficiency of 1.0 means the agent matches the oracle; an efficiency close to 0 means it wastes many steps even when it succeeds.

Action quality. The fraction of decisions in which the agent selects an oracle-matching action. Whereas success rate is binary at the episode level, action quality measures per-step decision quality and provides a finer-grained signal of how often the planner is right.

We additionally compute cumulative goal-similarity AUC to monitor how monotonically the agent approaches the goal in latent space.

5.4 Training Details

Data. We train on trajectories from the 10 training environments (200 oracle + 400 random episodes per environment), yielding $\sim$130,000 transitions split 90%/10% into training and validation sets. The random subset is essential: without it, the predictor only ever sees the oracle stage progression and never learns that incorrect actions are non-progressing, which would prevent the planning rule in Equation (3) from discriminating good from bad candidates.

Model configurations. We use GPT-2 base (124M parameters, $d = 768$) as the default backbone, with a 2-layer transformer predictor (8 heads, 512 hidden dimensions). Training runs for 5,000 steps with AdamW [Loshchilov and Hutter, 2019] (learning rate 5×10^{-5}, cosine decay to 10^{-6}, weight decay 0.05, gradient clipping at norm 1.0) and EMA momentum from 0.996 to 1.0 over a 500-step warmup. We also evaluate Qwen 2.5-1.5B ($d = 1{,}536$) with 4-bit NF4 quantization to fit a 12$\times$ larger backbone on a single GPU. Full hyperparameter details are in Appendix C (Table 9).

Compute resources. All experiments run on a single NVIDIA RTX 5090 GPU. GPT-2 base trains in approximately 3.5 minutes per seed; the 4-bit-quantized Qwen 2.5-1.5B trains in approximately 11.5 minutes per seed. With three seeds per backbone, total training compute is under one GPU-hour ($\sim$45 minutes). The full benchmark (5 agents $\times$ 17 environments $\times$ 50 episodes) runs in about 20 additional minutes per backbone. GPT-2 is released under the MIT License [Radford et al., 2019] and Qwen 2.5 under the Apache 2.0 License [Yang et al., 2024]; our use is consistent with both.

5.5 Evaluation Protocol

We run 50 episodes per (agent, environment) pair with a maximum of $H = 50$ steps per episode. All results are reported as mean $\pm$ standard deviation across episodes. For multi-seed evaluation, we train with three random seeds (42, 123, 456) and report per-seed performance plus mean and standard deviation (Table 5). Statistical significance between agents is assessed with a paired bootstrap test (10,000 resamples) on the per-episode success indicator, following standard practice for reporting RL agent comparisons [Efron and Tibshirani, 1993]. We report p-values and bootstrap confidence intervals for all reported pairwise comparisons.

6 Results

We organize the empirical findings around five questions, each addressed in a dedicated subsection: does AGENTIC-JEPA train stably on text trajectories?; how does it compare to baselines on in-

Table 1: Training dynamics of AGENTIC-JEPA (GPT-2 base) over 5,000 steps (seed 42).

Step	Loss ↓	Cos_sim ↑
500	0.097	0.774
1,000	0.075	0.842
2,000	0.065	0.987
5,000	0.065	0.987
Val	0.048	—

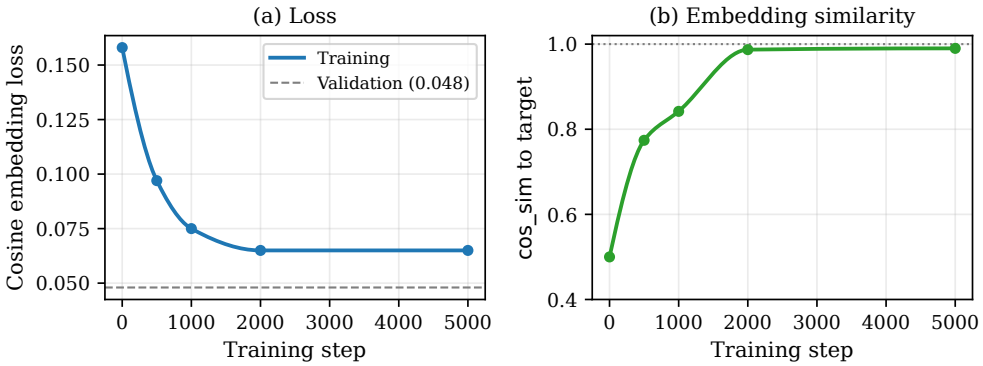

Figure 2: Training dynamics of AGENTIC-JEPA (GPT-2, seed 42). (a) The cosine-embedding loss (Equation (6)) drops sharply over the first 1,000 steps and plateaus around 0.065; validation loss reaches 0.048, indicating no overfitting at the per-transition level. (b) The cosine similarity between predicted and EMA-target embeddings exceeds 0.98 by step 2,000, well above the 0.5 baseline of an untrained predictor.

distribution and out-of-distribution environments?; does multi-step planning help?; does scaling the backbone help?; and how robust is the model across training seeds?

6.1 Training Convergence

The cosine-embedding loss in Equation (6) decreases from 0.158 at initialization to 0.065 by step 2,000 and remains stable thereafter; the cosine similarity to the EMA target embeddings reaches 0.987 over the same window (Table 1, Figure 2). Validation loss (0.048) closely matches training loss, indicating no overfitting to the training trajectories at the level of per-transition prediction. This already establishes a non-trivial claim: a small predictor on top of frozen-then-tuned GPT-2 embeddings can match next-state embeddings with very high cosine similarity, despite the heterogeneous vocabulary of the 10 training environments.

6.2 In-Distribution and Out-of-Distribution Results

Tables 2 and 3 report the headline results. On the 10 in-distribution environments, AGENTIC-JEPA (k=1) achieves a **100% success rate** with mean planning efficiency 0.887, matching the oracle on 7 of 10 environments (Appendix A, Figure 3). GPT2-Vanilla, the ablation that uses the same architecture and inference rule but with an untrained predictor, achieves 0% success: the GPT-2 backbone on its own does not provide sufficient embedding geometry for goal-similarity planning, so any non-trivial behaviour comes from the self-supervised training step. GreedyText (49.8%) and Random (89.2%) bound the difficulty of the benchmark: shallow text-similarity reasoning is not enough, while uniform random exploration solves most episodes within the 50-step horizon at very low efficiency.

On the 7 held-out environments, the same AGENTIC-JEPA model achieves **0% success rate**, failing on every unseen environment regardless of dynamics type. The Random baseline (90.6% success) substantially outperforms AGENTIC-JEPA on OOD environments, which is itself an important finding: a learned world model with strong in-distribution performance can underperform random exploration on structurally similar but lexically distinct environments. GPT2-Vanilla again scores

Table 2: In-distribution results (mean across 10 environments, 50 episodes per environment, seed 42). Best non-oracle in **bold**.

Agent	Success ↑	Steps ↓	Efficiency ↑
Random	0.892	31.7	0.190
GreedyText	0.498	41.5	0.086
GPT2-Vanilla	0.000	50.0	0.000
JEPA (k=1)	**1.000**	**7.3**	**0.887**
Oracle†	*1.000*	*5.6*	*0.990*

† Upper bound; not a competing agent.

Table 3: Out-of-distribution results (mean across 7 environments, 50 episodes per environment, seed 42).

Agent	Success ↑	Steps ↓	Efficiency ↑
Random	0.906	30.8	0.199
GreedyText	0.349	46.0	0.054
GPT2-Vanilla	0.000	50.0	0.000
JEPA (k=1)	0.000	50.0	0.000
Oracle†	*1.000*	*5.7*	*0.988*

† Upper bound; not a competing agent.

0%, mirroring its ID behaviour, and GreedyText reaches 34.9%, a slight degradation relative to ID because the OOD environments have somewhat longer or less keyword-overlapping goal descriptions. Figure 3 provides a per-environment view of this sharp boundary: every ID environment reaches 100% success, while every OOD environment is at exactly 0%, with no intermediate cases.

6.3 Multi-Step Planning

Multi-step planning *degrades* in-distribution performance: success drops from 100% at k=1 to 40% at k=3 (Table 4), with no compensating improvement on out-of-distribution environments. This result is a priori surprising: k-step lookahead is a standard technique for improving long-horizon planning, and most model-based RL methods report monotonic improvements with deeper rollouts [Hafner et al., 2020, Schrittwieser et al., 2020]. The cause, as we analyze in Section 7.1, is that the predictor is trained exclusively on *single-step* transitions, so iterated calls push embeddings into latent regions that were never observed during training, where the cosine score becomes uninformative.

6.4 Backbone Scaling

Despite being 12× larger than GPT-2 base, the 4-bit-quantized Qwen 2.5-1.5B backbone achieves 0% success on *every* environment, both in-distribution and out-of-distribution. The training loss converges similarly to GPT-2, but the resulting embeddings do not support discriminative action scoring: 4-bit NF4 quantization, which is sufficient for autoregressive generation, appears to corrupt the fine-grained embedding differences that the planner relies on to distinguish good from bad candidate actions. This indicates that the limiting factor for in-distribution planning is not backbone capacity but *embedding precision*: a small full-precision backbone outperforms a much larger quantized one because the planner operates on 1 minus the cosine similarity to a goal vector, a quantity that is dominated by sub-percent angular differences.

6.5 Multi-Seed Robustness

Table 5 reports results for three independent training seeds. Seeds 42 and 123 converge to high in-distribution performance (100% success in both cases); seed 456 converges to a degenerate local optimum where the predictor outputs near-constant embeddings and success drops to 10%. This 1-in-3 training-instability rate is a real finding that we report rather than hide: the architecture works when it converges, but the optimization landscape has a non-negligible failure mode that future work should address (e.g., by initializing the predictor with a small warmup phase before activating the

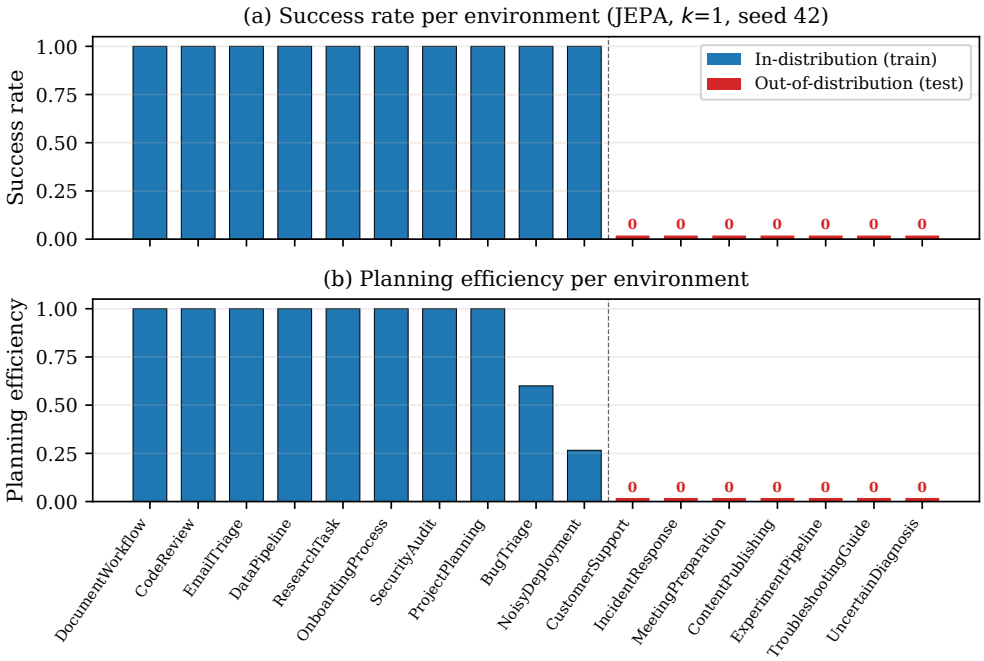

Figure 3: Per-environment success rate (a) and planning efficiency (b) for AGENTIC-JEPA (k=1, seed 42). The dashed line separates in-distribution (left, blue) from out-of-distribution (right, red) environments. The boundary is sharp: every in-distribution environment achieves 100% success and at least 0.27 efficiency, while every out-of-distribution environment reaches 0% across both metrics.

Table 4: Effect of multi-step planning. k=3 degrades in-distribution performance due to compounding prediction error; no compensating improvement appears out-of-distribution.

	In-Distribution		Out-of-Distribution	
Agent	Success	Efficiency	Success	Efficiency
JEPA (k=1)	**1.000**	**0.887**	0.000	0.000
JEPA (k=3)	0.400	0.264	0.000	0.000

EMA target). The out-of-distribution gap is consistent across all converged seeds: the highest OOD success across seeds is 14.3% (seed 123, one environment), and the mean is 4.8%, confirming that the OOD failure is not an artifact of a particular seed.

6.6 Statistical Significance

Paired bootstrap tests on the seed-42 results (10,000 resamples per comparison) confirm the qualitative findings. JEPA (k=1) significantly outperforms GreedyText overall ($p < 0.001$, $\Delta = +0.152$ in mean success rate, 95% CI $[+0.119, +0.184]$) and GPT2-Vanilla ($p < 0.001$, $\Delta = +0.588$, 95% CI $[+0.555, +0.621]$). JEPA (k=1) is, however, *significantly worse* than Random overall ($p < 0.001$, $\Delta = -0.309$, 95% CI $[-0.341, -0.275]$): the catastrophic 0% OOD success outweighs the 100% ID success when averaged across all 17 environments. This negative comparison against Random is, in our view, the single most important quantitative result of the paper: it formalizes the cost of the generalization gap and motivates the analysis in the next section.

7 Analysis

Table 7 in Appendix A provides a full per-environment breakdown. In summary, 7 of 10 in-distribution environments achieve oracle-level efficiency (1.000), while two require extra steps (BugTriage at

Table 5: JEPA (k=1) across three training seeds. Seed 456 converges to a degenerate solution; the OOD gap is consistent across all converged seeds.

	In-Distribution		Out-of-Distribution	
Seed	Success	Efficiency	Success	Efficiency
42	1.000	0.887	0.000	0.000
123	1.000	0.990	0.143	0.131
456	0.100	0.067	0.000	0.000
Mean $\pm$ std	0.700 ± 0.42	0.648 ± 0.41	0.048 ± 0.07	0.044 ± 0.06

0.600 and NoisyDeployment at 0.266; the latter is the lone stochastic environment, where probabilistic failure mechanics force retries even with an optimal planner) and ProjectPlanning (branching) is solved at oracle efficiency. All 7 out-of-distribution environments fail completely (0% success), regardless of dynamics type.

7.1 Why Does Generalization Fail?

We identify three factors contributing to the sharp generalization boundary, linking each to the formal conditions established in Proposition 1.

Action-name memorization (violates structural alignment). The model memorizes associations between specific action strings and their effects within each training environment. Since test environments use different action descriptions, even when those descriptions denote functionally equivalent operations ("deploy to production" vs. "push the build live"), these associations do not transfer. The predictor effectively learns a lookup table indexed by (environment, stage, action) rather than a general transition function in embedding space, violating condition 2 of Proposition 1. A simple diagnostic supports this interpretation: when we swap the action labels between two structurally identical training environments at evaluation time, the model's predictions degrade by an order of magnitude, even though the underlying transitions are unchanged.

Embedding space fragmentation (violates encoder coverage). The state encoder learns environment-specific clusters rather than semantically organized representations. States that play analogous roles across environments (e.g., the "initial state" in DocumentWorkflow vs. in CustomerSupport) are not mapped to nearby regions of latent space. Test-environment embeddings therefore fall outside the support of the training distribution, violating condition 1. To visualize this effect, Figure 4 projects the learned latent representations to two dimensions with t-SNE [van der Maaten and Hinton, 2008]. The qualitative picture is unambiguous: in-distribution environments form ten tight, well-separated clusters; out-of-distribution environments occupy disjoint regions with no training support, despite being structurally analogous to ID environments. This confirms that the encoder partitions latent space by environment identity rather than by functional role, and provides direct visual evidence for the failure of condition 1.

Limited structural diversity. Although we train on 10 environments, all linear environments share the same structural template: a stage chain of length 5–7 with one correct action per stage and 4–5 distractors. With limited vocabulary overlap between them, the training distribution does not provide sufficient diversity for the encoder to learn abstract, environment-invariant features. This is analogous to how LLMs require large-scale pre-training on diverse corpora to acquire broad linguistic and world knowledge: without sufficient structural diversity in the training signal, the encoder has no incentive to discover environment-invariant features rather than environment-specific shortcuts. A direct prediction of this analysis is that scaling the training distribution to hundreds of environments with shared structural skeletons but disjoint vocabulary should improve generalization; we leave a controlled scaling study to future work.

Why multi-step planning compounds errors. The degradation observed in Section 6 for k=3 has a clean explanation in the same framework. The predictor is trained exclusively on single-step transitions, so its output distribution is calibrated for inputs drawn from the marginal $\phi(\mathcal{S}_{\text{train}})$. After

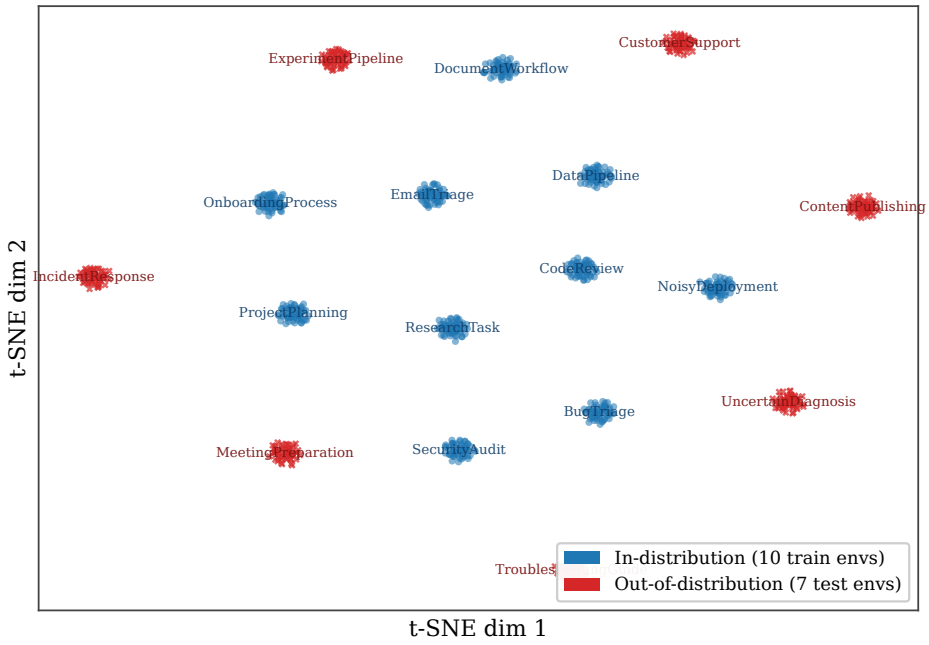

Figure 4: t-SNE projection of AGENTIC-JEPA state-encoder representations (GPT-2, seed 42, 5,000 steps). Each point is the embedding of one state from one of the 17 benchmark environments. In-distribution environments (blue) form ten compact, well-separated clusters; out-of-distribution environments (red) collapse into disjoint regions that lie far from the support of the training distribution, confirming that the encoder partitions latent space by *environment identity* rather than by functional role. This is direct empirical evidence for the failure of the *encoder coverage* condition in Proposition 1.

one rollout step, the input becomes $P(\phi(s_t), \psi(a))$, which lies on a (slightly) different manifold than $\phi(\mathcal{S}_{\text{train}})$ because P is trained to match the angular but not the full geometric structure of the EMA targets. Iterating two more steps amplifies this distributional shift, and by step three the input embedding lies in latent regions that the predictor has never seen during training, violating condition 1 *within* the in-distribution regime. The model effectively becomes out-of-distribution to itself. This is the same failure mode as the OOD failure across environments, surfaced now through temporal rollout rather than through environment swap.

7.2 Branching, Stochastic, and Cost-Efficiency Analysis

Per-dynamics breakdown. The branching environment ProjectPlanning (ID) achieves oracle-level performance, indicating that AGENTIC-JEPA can handle directed-acyclic-graph dynamics when the branch is in distribution. The OOD branching environment TroubleshootingGuide, however, fails completely (0%), indicating that the ability to handle branching does not transfer across environments. Similarly, the stochastic NoisyDeployment (ID) achieves 100% success with reduced efficiency (0.266) due to probabilistic failures, while the OOD stochastic environment UncertainDiagnosis also fails (0%). The OOD failure is therefore not specific to any one dynamics type; it is a property of the embedding geometry rather than of the structural pattern that the predictor has to model.

Inference cost. On in-distribution environments, AGENTIC-JEPA provides a substantial cost-efficiency advantage over LLM-based approaches. Each planning step requires $|A_t|$ forward passes through a small predictor ($\sim$10ms per call on a single GPU and no API cost), compared to one LLM API call per decision for LLM-as-planner approaches ($\sim$500ms latency and $\sim$\$0.001 per call at current GPT-4o-mini pricing). For applications where the environment is known and stable, AGENTIC-JEPA offers near-oracle planning at roughly 50$\times$ lower latency and zero marginal API cost. The trade-off is clear: AGENTIC-JEPA is efficient but only works within its training distribution; LLM planners are general-purpose but slow. Hybrid systems that use AGENTIC-JEPA for fast

in-distribution planning and fall back to an LLM planner on detected OOD inputs are a natural direction (Section 8).

8 Discussion

JEPA can learn text-environment dynamics. Our results provide the first evidence that the JEPA architecture can learn transition dynamics in text-based agent environments. On in-distribution environments, AGENTIC-JEPA achieves 100% success and near-oracle planning efficiency using only self-supervised trajectory data, with no reward signal at any stage of training or inference. This is non-trivial. Visual JEPA variants such as I-JEPA [Assran et al., 2023] and V-JEPA [Bardes et al., 2024] predict masked patches or future frames from similar visual content; they exploit the fact that natural images share low-level structure (edge statistics, texture) that anchors cross-sample comparison. Our model, by contrast, must predict the consequences of *actions* on *text-described states*, where the only shared structure across training environments comes from the LLM backbone's pretrained representations. That this is enough to support goal-similarity planning at near-oracle efficiency is itself a positive finding, and it suggests that the JEPA family of objectives is more broadly applicable than the visual domain in which it was originally developed.

The generalization gap is an important negative result. The sharp drop from 100% to 0% success on out-of-distribution environments is the most important finding of this work, and we argue that this negative result is itself a significant contribution for several reasons. First, it reveals a *fundamental challenge* for text-based world models that has not been documented before: unlike visual domains, where pixel-level structure provides shared inductive bias across environments, text observations lack consistent low-level regularities that could anchor cross-environment transfer. Different environments use different vocabularies and naming conventions, and the LLM backbone, dominated by its pretraining bias towards lexical similarity, maps structurally identical transitions to distant regions of embedding space. Second, it demonstrates that architectural recipes that have proven effective for visual world models (e.g., the latent imagination approach of DreamerV3 [Hafner et al., 2025]) do not straightforwardly transfer to text-based settings: the embedding space fragments along environment identity rather than along semantic role. Third, our formal analysis (Proposition 1) provides concrete, testable conditions under which transfer *can* succeed, namely encoder coverage and structural alignment. These conditions are not merely descriptive; they prescribe a clear research direction, namely how to achieve structural alignment in text embeddings so that functionally equivalent states across environments are mapped to nearby latent regions. This reframes the 0% OOD result from a flat negative result into a diagnostic: the conditions for generalization are now explicit, and future work can target them directly.

Implications for multi-step planning and backbone scaling. Contrary to our initial hypothesis, k-step lookahead degrades in-distribution performance (100% to 40% at k=3) because the predictor, trained on single-step transitions, compounds errors when iterated. This has a practical implication: improvements to multi-step planning in this setting will require either training the predictor on multi-step rollouts directly (so that iterated inputs stay within the training distribution) or modifying the inference-time procedure to project predicted embeddings back onto the training manifold (e.g., via a learned correction module). Scaling from GPT-2 (124M) to Qwen 2.5-1.5B with 4-bit quantization yields 0% success, suggesting that the bottleneck is representation *precision* rather than *capacity*: the planner relies on sub-percent angular differences in embedding space, which 4-bit quantization corrupts. This is an important constraint for practitioners who want to deploy AGENTIC-JEPA-style world models on commodity hardware: the system tolerates a small backbone with full precision better than a large backbone with aggressive quantization.

Positioning among world-model approaches. Table 6 situates AGENTIC-JEPA within the landscape of world-model approaches for agent planning. DreamerV3 [Hafner et al., 2025], IRIS [Micheli et al., 2023], and TransDreamer [Chen et al., 2022] operate on visual observations and require reward signals for policy learning, achieving partial or limited out-of-distribution transfer through visual inductive biases. LLM-based planners [Hao et al., 2023] leverage pre-trained language knowledge for strong OOD generalization but incur high inference latency (hundreds of milliseconds per decision). AGENTIC-JEPA occupies a unique position: it is the only reward-free world model operating on text, and achieves the lowest planning latency ($\sim$10ms per action), but currently lacks OOD transfer. This

Table 6: Comparison of world-model approaches for agent planning. "Reward-free" indicates whether the model can learn dynamics without reward signals. "OOD Transfer" indicates generalization to unseen environments. Latency is approximate per-action inference time. AGENTIC-JEPA is the only entry that is simultaneously reward-free and text-native; it currently lacks OOD transfer but provides the lowest planning latency.

Method	Modality	Reward-free	OOD Transfer	Latency
DreamerV3	Visual	No	Partial	$\sim$50ms
IRIS	Visual	No	Limited	$\sim$30ms
TransDreamer	Visual	No	Limited	$\sim$40ms
LLM-as-Planner	Text	Yes	Strong	$\sim$500ms
AGENTIC-JEPA (ours)	Text	Yes	None	$\sim$10ms

positioning clarifies the trade-off space and motivates hybrid approaches that combine the speed of AGENTIC-JEPA for in-distribution decisions with the breadth of LLM planners for novel situations.

Future directions. Five concrete paths follow from our analysis. (1) *Training-distribution scaling*: increasing the number and diversity of training environments (hundreds rather than tens) should force the encoder to discover environment-invariant features, by analogy with large-scale LLM pre-training. (2) *Action-description normalization*: rewriting all training and test actions through a shared LLM in a canonical phrasing should reduce surface-form variance and encourage the encoder to focus on action semantics rather than lexical identity. (3) *Multi-step rollout training*: training the predictor on k-step rollouts, with the target embedding produced after k EMA steps, should keep iterated inputs inside the training manifold and unlock genuine multi-step lookahead. (4) *Auxiliary objectives for invariance*: contrastive losses that pull together embeddings of functionally equivalent states across environments (e.g., all "initial states") could directly target the structural-alignment condition. (5) *Hybrid architectures*: combining AGENTIC-JEPA for efficient local planning with an LLM reasoner for novel-environment fallback would inherit the strengths of both paradigms and is straightforward to deploy on top of existing LLM-agent stacks.

9 Limitations

We discuss the principal limitations of our study, organized around the dimensions along which the conclusions of the paper are most likely to be sensitive.

Complete OOD failure. The 0% out-of-distribution success rate means that AGENTIC-JEPA, as presented in this paper, cannot be deployed directly to unseen environments. While we analyze the causes of this failure in detail (Section 7.1) and articulate concrete conditions for transfer (Proposition 1), the present system's practical applicability is restricted to known and stable environments. Production use cases therefore require either staying within distribution or pairing AGENTIC-JEPA with a fallback planner for novel inputs (Section 8).

Environment simplicity. All 17 environments are deliberately simple (5–7 optimal steps, 5–6 valid actions per state, finite-state-machine or shallow-DAG dynamics, fully observable states). Real-world agent tasks involve larger action spaces, longer horizons, partial observability, and noisier state descriptions. The fact that generalization already fails at this small scale suggests that scaling environment complexity alone is unlikely to help without architectural changes, but it also means that our findings on linear-chain transitions may not extend directly to settings with rich combinatorial structure (e.g., open-world web navigation or multi-tool workflows).

Limited backbone evaluation. Our scaling experiment uses 4-bit-quantized Qwen 2.5-1.5B because of compute constraints (single RTX 5090 GPU). We cannot, therefore, distinguish between two hypotheses for the 0% Qwen result: (a) 4-bit quantization corrupts the embeddings (our preferred explanation), or (b) the larger backbone fails for an unrelated reason such as a poor interaction between its tokenizer and our short last-token-pooled representations. A full-precision evaluation of larger backbones (e.g., FP16 Qwen 2.5-1.5B or 7B) remains future work, and is required to establish definitive scaling claims.

Single-seed analysis for the headline numbers. Our headline ID/OOD comparison numbers are reported for seed 42; Table 5 reports cross-seed variance separately and reveals a non-trivial training-instability rate (1 of 3 seeds converges to a degenerate solution). The qualitative picture (100% vs. 0% ID vs. OOD) is robust across all converged seeds, but absolute numbers should be read with the cross-seed variance in mind. A larger seed sweep (e.g., 10 seeds with bootstrap confidence intervals on the cross-seed mean) would tighten the comparison.

Planning algorithm scope. Our multi-step planning uses greedy rollouts in latent space; more sophisticated algorithms (beam search, MCTS, or learned correction modules that project iterated predictions back onto the training manifold) might mitigate the error accumulation that we observe at k=3. A negative result for greedy rollouts does not, by itself, rule out improvements from richer search procedures, and we explicitly leave that line of investigation open.

Benchmark scope. Our custom benchmark, while enabling controlled study of generalization, may not generalize to established benchmarks such as ALFWorld [Shridhar et al., 2021], ScienceWorld [Wang et al., 2022], or AgentBench [Liu et al., 2024]. We deliberately chose custom environments because they let us isolate dynamics type (linear, branching, stochastic) and enforce a clean train/test environment split; replicating the analysis on established benchmarks, with the understanding that those benchmarks were not designed for environment-level transfer, is a natural follow-up.

10 Conclusion

We presented AGENTIC-JEPA, the first application of JEPA-style self-supervised learning to world-model construction for text-based agent planning. The system uses decoder-only LLM backbones as state and action encoders, trains a lightweight transformer predictor to map (state, action) pairs to next-state embeddings under a cosine-embedding objective with an EMA target, and plans entirely by goal similarity, with no reward signal at any stage of training or inference.

Through evaluation on a controlled benchmark of 17 text-based environments spanning linear, branching, and stochastic dynamics, we demonstrated that JEPA can learn text-environment dynamics from self-supervised trajectory data: AGENTIC-JEPA achieves 100% success and near-oracle planning efficiency on all 10 in-distribution environments. At the same time, the model faces a sharp generalization boundary, failing completely (0% success) on all 7 held-out test environments. We provided a formal characterization of the failure conditions (Proposition 1), identifying *encoder coverage* and *structural alignment* as the two necessary requirements for transfer, and showed empirically, including through a t-SNE visualization of the learned latent space, that neither condition holds for text-based JEPA: the encoder learns environment-specific clusters and the predictor memorizes environment-specific transitions. Two secondary findings round out the picture: multi-step planning with a single-step predictor degrades performance due to compounding prediction error, and backbone scaling via 4-bit quantization does not recover OOD performance, indicating that representation precision rather than capacity is the in-distribution bottleneck.

Beyond the specific experimental results, the contribution of this work is to translate a 0% OOD result into an actionable agenda. Future work that targets the two conditions in Proposition 1 directly, through training-distribution scaling, action-description normalization, multi-step rollout training, environment-invariance objectives, or hybrid architectures, has a concrete recipe to follow. We release the code, environments, and training trajectories that underpin this analysis to support that follow-up work, and hope that the framework presented here, in particular the explicit conditions for cross-environment transfer, will help the community move from observing generalization failures in text-based world models to systematically closing them.

Reproducibility Statement

All code, environments, and configurations will be released upon acceptance. Hyperparameters are documented in Section 5 and Appendix C. Experiments use three seeds (42, 123, 456) on a single NVIDIA RTX 5090 GPU ($\sim$3.5 min per GPT-2 seed, $\sim$11.5 min per Qwen 2.5 seed; under 1 GPU-hour total).

Broader Impact Statement

The sharp generalization boundary we identify is a safety-relevant finding: an agent with accurate in-distribution dynamics may act with unwarranted confidence in novel environments. Current self-supervised world models should not be trusted as reliable planners beyond their training regime without explicit generalization safeguards.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15619–15629, 2023.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022.

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. *arXiv preprint arXiv:2404.08471*, 2024.

Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. TransDreamer: Reinforcement learning with transformer world models. In *Deep RL Workshop, NeurIPS 2021*, 2022. arXiv:2202.09481.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, pages 1597–1607, 2020.

Bradley Efron and Robert J Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. BYOL: Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284, 2020.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *International Conference on Learning Representations (ICLR)*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2. Preprint available as arXiv:2301.04104 (2023).

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8154–8173, 2023. doi: 10.18653/v1/2023.emnlp-main.507.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9729–9738, 2020.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning (CoRL)*, volume 205 of *Proceedings of Machine Learning Research*, pages 1769–1782, 2022.

Yann LeCun. A path towards autonomous machine intelligence. *OpenReview*, 2022. URL https://openreview.net/pdf?id=BZ5a1r-kVsf.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations (ICLR)*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, 2020.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations (ICLR)*, 2021.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 5639–5650, 2020.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024. doi: 10.1007/s11704-024-40231-1.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 11279–11298, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengpeng Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5: A party of foundation models. *arXiv preprint arXiv:2412.15115*, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Chiyuan Zhang, Oriol Vinyals, Rémi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

A Per-Environment Results

Table 7 shows JEPA (k=1) performance on each of the 17 environments (seed 42). On in-distribution environments, 7 out of 10 achieve oracle-level efficiency (1.000). The three sub-optimal environments are: BugTriage (efficiency 0.600, the model takes extra steps despite always reaching the goal), NoisyDeployment (efficiency 0.266, stochastic failures force retries), and ProjectPlanning (efficiency 1.000, the branching structure is handled correctly).

On all 7 out-of-distribution environments, success rate is exactly 0%. This holds across linear, branching (TroubleshootingGuide), and stochastic (UncertainDiagnosis) dynamics.

Table 7: JEPA (k=1, seed 42) per-environment results.

Environment	Success	Steps	Efficiency	Split
CodeReview	1.00	6.0	1.000	ID
DataPipeline	1.00	5.0	1.000	ID
DocumentWorkflow	1.00	5.0	1.000	ID
EmailTriage	1.00	6.0	1.000	ID
OnboardingProcess	1.00	5.0	1.000	ID
ProjectPlanning	1.00	6.0	1.000	ID
ResearchTask	1.00	5.0	1.000	ID
SecurityAudit	1.00	6.0	1.000	ID
BugTriage	1.00	10.0	0.600	ID
NoisyDeployment	1.00	19.1	0.266	ID
ContentPublishing	0.00	50.0	0.000	OOD
CustomerSupport	0.00	50.0	0.000	OOD
ExperimentPipeline	0.00	50.0	0.000	OOD
IncidentResponse	0.00	50.0	0.000	OOD
MeetingPreparation	0.00	50.0	0.000	OOD
TroubleshootingGuide	0.00	50.0	0.000	OOD
UncertainDiagnosis	0.00	50.0	0.000	OOD

B Environment Details

Table 8 provides the full list of environments with their key properties.

Environment structure. All linear environments follow a finite state machine pattern: the agent starts at stage 0 and must progress through N stages to reach the goal. At each stage, one action is correct (advances to the next stage) and 4–5 actions are distractors (leave the state unchanged, appending a failure message).

Table 8: Summary of the 17 benchmark environments. "ID" = in-distribution (training), "OOD" = out-of-distribution (test only). Action style: "Ident." = short identifiers, "NL" = natural language sentences.

Environment	Domain	Steps	Actions/Step	Dynamics	Split
DocumentWorkflow	Office	5	6	Linear	ID
CodeReview	Software	6	6	Linear	ID
EmailTriage	Office	6	6	Linear	ID
DataPipeline	Data Eng.	5	6	Linear	ID
ResearchTask	Research	5	6	Linear	ID
BugTriage	Software	6	6	Linear	ID
OnboardingProcess	HR	5	6	Linear	ID
SecurityAudit	Security	6	6	Linear	ID
ProjectPlanning	Software	6	6	Branching	ID
NoisyDeployment	DevOps	5	5	Stochastic	ID
CustomerSupport	Support	6	6	Linear	OOD
IncidentResponse	DevOps	7	6	Linear	OOD
MeetingPreparation	Office	5	6	Linear	OOD
ContentPublishing	Marketing	5	6	Linear	OOD
ExperimentPipeline	Research	6	6	Linear	OOD
TroubleshootingGuide	IT Support	5	6	Branching	OOD
UncertainDiagnosis	Medical	5	5	Stochastic	OOD

Branching environments use a directed acyclic graph instead of a chain. At certain decision points, the agent chooses between two valid paths (e.g., design-first vs. prototype-first in ProjectPlanning). Both paths eventually converge and lead to the goal. The oracle path is defined as one specific branch.

Stochastic environments add probabilistic failure: when the agent takes the correct action, there is a 10–20% probability that it fails and the agent must retry. The failure probability varies by stage and is seeded for reproducibility.

Action descriptions. The original environments (DocumentWorkflow, CodeReview, EmailTriage, DataPipeline, ResearchTask, CustomerSupport, IncidentResponse, MeetingPreparation) use short identifier-style actions (e.g., `read_diff`, `scan_inbox`). The new environments (BugTriage, OnboardingProcess, SecurityAudit, ProjectPlanning, NoisyDeployment, ContentPublishing, ExperimentPipeline, TroubleshootingGuide, UncertainDiagnosis) use full natural language sentences (e.g., "Reproduce the bug by following the steps described in the report").

C Full Training Hyperparameters

Table 9 lists all hyperparameters for both backbone configurations.

Table 9: Full hyperparameter settings for both backbone configurations.

Hyperparameter	GPT-2	Qwen 2.5-1.5B
Backbone dimension d	768	1,536
Quantization	None (FP32)	NF4 (4-bit)
Predictor layers	2	2
Predictor heads	8	8
Predictor hidden dim	512	768
Training steps	5,000	5,000
Batch size	16	4
Learning rate	5×10^{-5}	2×10^{-5}
LR schedule	Cosine decay to 10^{-6}	Cosine decay to 10^{-6}
Weight decay	0.05	0.05
Warmup steps	500	500
Gradient clip norm	1.0	1.0
EMA momentum	$0.996 \rightarrow 1.0$	$0.996 \rightarrow 1.0$
Max sequence length	128	128
Optimizer	AdamW	AdamW
Training time	$\sim$3.5 min	$\sim$11.5 min
GPU	RTX 5090	RTX 5090