CausalMind: True Causal Reasoning for Social Learning between LLM-Based Agents

by

Remi Lejeune

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Friday November 28, 2025 at 9:30.

Student number:	5040841	
Project duration:	November 28, 2024 – November 28, 2025	
Thesis committee:	Dr. C.A Raman,	TU Delft, Daily supervisor
	Dr.ir. M.J.T. Reinders ,	TU Delft, Advisor
	Dr. S. Tan,	TU Delft, External committee member
	Ir. O.K Shirekar,	TU Delft, Daily Co-Supervisor

TU Delft

CausalMind: True Causal Reasoning for Social Learning between LLM-Based Agents

Remi Lejeune

November 22, 2025

Abstract

Large language models enable rich communication and flexible planning in embodied agents, yet their updates to internal state remain correlation driven, making them prone to hallucinations that undermine reliability in interactive settings. To address this limitation, we investigate whether a true causal model can replace template based belief, desire, and intention (BDI) updates and provide more stable reasoning. We introduce CausalMind, an embodied agent architecture that integrates a causal model trained on text based BDI transitions drawn from a synthetic dataset. We evaluate both the causal model and the full agent in iTHOR across solo and cooperative tasks.

The results indicate that learning causal variables from text embeddings is difficult. The model focused primarily on action related features, failing to separate belief, desire, and intention and instead overfitting to biases in the dataset. Expanding and refining the synthetic dataset improved performance but did not resolve the core issues, highlighting the need for richer data and more suitable encoders. At the agent level, perception errors from the vision module frequently led to hallucinated task completion, limiting the usefulness of teacher guidance and making complex tasks difficult in the iTHOR environment.

Although the current implementation does not outperform template based baselines, the study clarifies the obstacles that arise when extending causal representation learning to natural language BDI states. These findings outline the key requirements for future work toward embodied agents capable of robust causal reasoning rather than correlation driven updates.

1 Introduction

While LLM-based agents [1] have transformed how we build intelligent systems, enabling richer communication, more effective cooperation, and advanced planning, a major challenge remains: hallucinations[2]. These occur when an LLM produces responses that are factually incorrect, nonsensical, or disconnected from the given input. In high-stakes real-world applications, whether acting autonomously or collaborating with humans or other agents, minimizing hallucinations is essential. For example, if a surgical assistant agent suggests the wrong instrument due to a hallucination, the consequences could be devastating.

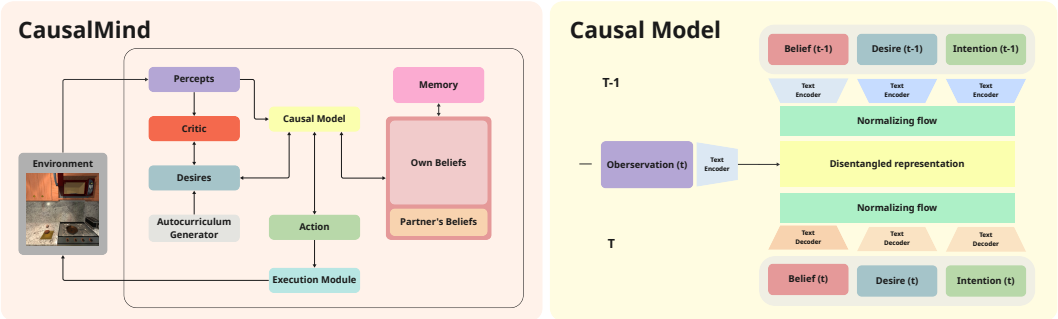


Figure 1: Core contributions of this work. The CausalMind agent (left) maintains a belief desire intention representation and adapts it through a causal model. The causal model (right) uses a normalizing flow to produce a disentangled latent space and integrates new observations to update internal beliefs.

However, these hallucinations cannot be fully eliminated due to the nature of LLMs[3]. They are sta-

tistical models that aim to approximate the underlying data distribution rather than recover the true causal[4] mechanisms that generate the data. To reduce this risk, recent work has explored incorporating causal reasoning into agent architectures[5]. MindForge[6] takes this approach by introducing a causal template that guides the LLM to follow structured causal reasoning, particularly for social reasoning and collaboration.

We argue that this causal template, while improving the agent's reasoning, is still not truly causal and only mimics causal structure rather than learning it directly[7]. As a result, hallucinations remain unavoidable. Although this template based approach enhances performance, it does not address the underlying issue. To further reduce hallucinations, a true causal model[8] is needed to replace the current template based reasoning by learning genuine causal variables.

In this paper, we propose a true causal model that replaces template based belief, desire, and intention (BDI) updates with a normalizing flow architecture designed to learn disentangled causal variables for each component of the BDI framework. We further conduct a detailed analysis of the model to assess whether it successfully captures the underlying causal structure rather than relying on correlated patterns. The main contributions of this work are:

- A causal model trained to learn and update beliefs, desires, and intentions (see Fig. 1)
- An agent architecture that integrates this causal model for internal and social reasoning (see Fig. 1)
- A synthetic dataset designed to train and evaluate BDI prediction

While some prior work on embodied agents has advanced the frontier of lifelong learning, emphasizing agents that continuously acquire, adapt, and refine skills across open-ended environments [9][6][10], other strands have primarily focused on collaborative task solving, where the emphasis lies on coordination, communication, and role specialization among multiple agents [11][12][13]. A third emerging direction explores causal reasoning, aiming to equip embodied agents with the ability to move beyond mere correlations and instead infer and leverage causal structures for more robust generalization and decision-making [14][15][16]. Yet, these areas have largely developed in parallel. Lifelong learning research often overlooks the demands of multi-agent collaboration, collaboration work tends to prioritize interaction dynamics without grounding them in causal understanding; and causal reasoning efforts rarely consider the challenges of continuous adaptation or multi-agent contexts. To the best of our knowledge, no prior work systematically integrates lifelong learning, collaboration, and causality within a single embodied framework. This gap is precisely where our contributions take shape, offering a unified perspective that bridges these domains and unlocks new capabilities for embodied intelligence.

The remainder of this paper proceeds as follows. Chapter 2 reviews related work, and Chapter 3 provides the necessary background. Chapter 4 introduces the problem statement, followed by Chapter 5, which outlines our proposed method. The results and discussion are presented in Chapter 6, while Chapter 7 offers future work. Finally, Chapter 8 concludes the paper.

2 Related Work

Causal Representation Learning Understanding causality is fundamental to building AI systems that can reason about the world beyond statistical correlations. Pearl's work on causality [4] established a formal framework for defining and identifying cause-and-effect relationships in data, it also introduced the Ladder of Causation[17], which states that causal reasoning operates on three levels: (1) *association*, which involves observing statistical correlations (e.g., $P(y \mid x)$); (2) *intervention*, which considers the effects of actions (e.g., $P(y \mid do(x))$); and (3) *counterfactuals*, which reason about alternate realities (e.g., what would have happened had x been different).

Pearl's structural causal models (SCMs)[8] enable AI systems to model interventions and counterfactual reasoning, which are critical for robust decision-making and generalization. While deep learning models excel at recognizing patterns, they often fail to distinguish correlation from causation, leading to unreliable or misleading inferences. Incorporating causal reasoning into AI systems helps overcome these limitations, allowing for more interpretable and reliable learning. Building on these ideas, the field of causal representation learning [18] emerged to study how high-level causal variables can be learned from raw data. Multiple type of causal representation learning model exists[19] for each level of Pearl's ladder: Observational[20, 21, 22], Interventional[23, 24, 25, 26], and Counterfactual[27, 28].

Focusing on interventional models, methods like CITRIS [23] and iCITRIS [24] leverage temporal sequences of images to extract causal structures from low-level pixel data. These approaches demonstrate that causal relationships can be inferred from high dimension data, but they rely on predefined intervention targets, which limits their applicability in real-world settings where interventions are often

unknown or unstructured. Building on this line of work, Hierarchical Causal Representation Learning[29] proposes the HERCULES framework which learns causal variables in a hierarchical manner, gradually refining coarse abstractions into more fine-grained causal factors as additional interventions are introduced. Similarly Temporally Disentangled Representation Learning[**?**], identifies latent time-dependent causal factors from sequential data, enabling causal discovery even under complex nonlinear temporal dynamics. More recent work unifies causal representation learning under the invariance principle, showing that causal variables can often be identified by exploiting invariant structure across environments rather than relying solely on intervention data[30]. To address the persistent challenge of requiring predefined intervention targets, BISCUIT [25, 16] was introduced as a framework for unsupervised causal discovery, allowing for more flexible and autonomous causal learning directly from observations.

LLM-based Multi-Agent Systems The rise of large language models (LLMs)[31] has led to the development of LLM-based multi-agent systems, where agents communicate through natural language to share knowledge and collaborate [32]. This shift allows researchers to study social interactions between AI agents, including cooperation, negotiation, and competition [33]. Experiments with multi-agent LLM systems have revealed emergent behaviors in simulated societies, such as the formation of norms and collective problem-solving strategies [11, 34].

These social interactions can be use in a multitude of ways, such as improving collaborative driving among multiple vehicles[35], learning new skills from other agents[6, 36], or even reasoning about a complex problem like mixed-autonomy traffic problems [37].

Embodied Causal Agents LLM agents[1] have diversified into multiple categories, one of which is embodied causal agents. These agents interact with environments and incorporate causal reasoning to guide decisions. In ADAM[14], causality is introduced through causal discovery to uncover new relationships and construct a causal graph of environment dynamics. Optimus-2[15] employs a Causal Perceiver to inform action selection. LLMCWM[16] integrates a causal model to support planning by performing interventions within its internal causal structure to identify optimal actions. MindForge[6] applies a causal template to enhance social and cultural learning.

3 Background

3.1 BISCUIT

BISCUIT (Binary Interactions for Causal Identifiability) is a framework for recovering latent causal variables and their transition dynamics from high dimensional temporal data [25]. The core idea is that external actions selectively intervene on specific latent dimensions, and these intervention patterns enable identifiability.

We consider observations $\{X^t\}_{t=0}^T$ generated from latent causal states $C^t \in \mathbb{R}^K$ through an injective observation function:

$$X^t = g(C^t). \tag{1}$$

The latent state evolves based on the previous state and the action:

$$p(C^t \mid C^{t-1}, R^t) = \prod_{i=1}^{K} p(C_i^t \mid C^{t-1}, I_i^t), \tag{2}$$

where $I_i^t \in \{0, 1\}$ indicates whether the action influences latent variable i. These binary interaction variables are produced by a small multi layer perceptron that takes the previous latent state and the action as input:

$$I^t = \mathrm{MLP}(R^t, C^{t-1}). \tag{3}$$

BISCUIT learns a representation $z^t = f(X^t)$ that recovers the causal variables up to invertible transformations. The latent variables are modeled using a normalizing flow, ensuring that the mapping between observations and latents remains bijective:

$$z^t = f_\theta(X^t). \tag{4}$$

The latent transition model follows the same factorized structure:

$$p(z^t \mid z^{t-1}, R^t) = \prod_{i=1}^{K} p(z_i^t \mid z^{t-1}, \hat{I}_i^t). \tag{5}$$

A key contribution of BISCUIT is the structured transition prior, which models how latent variables evolve over time. This prior is parameterized by an MLP that takes (z^{t-1}, R^t) as input and outputs the parameters governing the transition distribution:

$$p(z_i^t \mid z^{t-1}, R^t) = p_{\omega,i}(z_i^t \mid z^{t-1}, \text{MLP}_\omega^i(R^t, z^{t-1})). \tag{6}$$

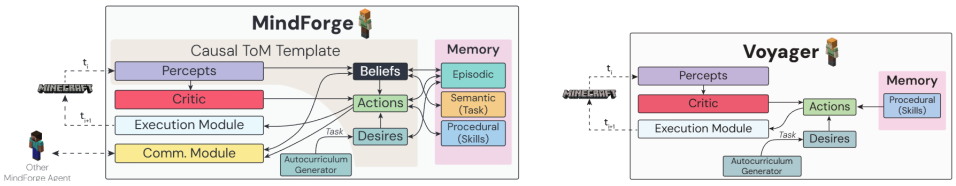
The MLP therefore specifies which latent dimensions are affected by an action and how those dimensions transition, allowing the model to distinguish natural temporal evolution from action driven changes. This structure enables BISCUIT to perform unsupervised causal discovery without predefined intervention targets and to operate on meaningful causal factors rather than correlations in raw observations.

For more details, we refer readers to the original BISCUIT paper [25].

3.2 MindForge

MindForge[6] is a LLM-based agent built for cultural lifelong learning through explicit perspective taking. It is based of Voyager[9], a LLM-based agent designed to learn new skills through exploration and with the goal of achieving lifelong learning, it has an auto-curriculum modules that decides what will be the next task using the Minecraft wiki. However Voyager[9] performances fall short when the underlying LLM is not trained on the environment. To solve this issue, MindForge[6] introduces a new architecture with the goal to allow agents to communicate between each other and helping each other out to solve tasks and learn new skills. This architecture is composed of three components:

- Communication module: This module allows the agents to communicate between each other and ask for help when stuck on a task.
- Memory module: This module memorizes all important information about its past conversation, other agents, and skills learned.
- Causal Theory of Mind (ToM) template: This template's goal is to improve the reasoning of the agent, but also to allow for perspective taking and understand its peer agents better based on BigToM[38].



(a) MindForge Architecture (b) Voyager Architecture

Figure 2: Diagram of MindForge and Voyager architecture taken from MindForge paper[6]

Through these three components, [6] shows promising results for lifelong learning through perspective taking, its overall architecture can be observed in Figure 3 and for more information please refer to Appendix D

4 Problem Statement

Mindforge relies on a causal template filled by an LLM, which it uses to update its belief-desire-intention (BDI):

$$(B_t, D_t, I_t) = \text{Template}_{\text{LLM}}\left(O_t, B_{t-1}, D_{t-1}, I_{t-1}\right) \tag{7}$$

where B_t denotes the beliefs, D_t the desires, and I_t the intentions (or actions both will be use interchangeably) at time step $t \in T$. In its beliefs, the agent also keeps track of its partners' BDI, modeled through communication and observation using the same template.

$$(\hat{B}_t^j, \hat{D}_t^j, \hat{I}_t^j) = \text{Template}_{\text{LLM}}\left(O_t^j, \hat{B}_{t-1}^j, \hat{D}_{t-1}^j, \hat{I}_{t-1}^j\right), \quad \forall j \in [A] \tag{8}$$

where $[A]$ represents the set of other agents.

Instead, we would like to have a model $\mathcal{M}$ that uses causality to update the BDI, therefore directly acting on the causal variables of BDI:

$$C_{BDI,t} = \mathcal{M}\left(O_t, C_{BDI,t-1}\right), \quad \text{where} \quad C_{BDI,t-1} = \phi(B_{t-1}, D_{t-1}, I_{t-1}) \tag{9}$$

ϕ encodes the BDI into causal variables and θ decodes causal variables into BDI. The internal causal state is represented as:

$$C_{\text{BDI},t} = \{C^t_{B,1}, \ldots, C^t_{B,n},\ C^t_{D,1}, \ldots, C^t_{D,m},\ C^t_{A,1}, \ldots, C^t_{A,r}\} \tag{10}$$

Here:

- $C^t_{B,i}$ denotes the i-th causal variable associated with the agent's beliefs.
- $C^t_{D,j}$ denotes the j-th causal variable associated with the agent's desires.
- $C^t_{A,k}$ denotes the k-th causal variable associated with the agent's intentions.

This also applies to modeling its partners' BDI, however the causal variables modeled about the other agents are approximations since our agent does not know how they reason internally. These approximations come from observations and communication.

$$(\hat{B}^j_t, \hat{D}^j_t, \hat{I}^j_t) = \mathcal{M}\left(O_t, \hat{C}^j_{BDI,t-1}\right), \quad \text{where} \quad \hat{C}^j_{BDI,t-1} = g(\hat{B}^j_{t-1}, \hat{D}^j_{t-1}, \hat{I}^j_{t-1}), \quad \forall j \in [A] \tag{11}$$

To summarize, we aim to replace Mindforge's template system with a true causal model to model both our own BDI and our partners' BDI.

5 CausalMind

In this section we introduce CausalMind, our proposed embodied agent architecture that integrates causal reasoning into the BDI framework. Our design builds on Voyager[9], which showed how large language models can support lifelong learning through automatic curriculum generation and modular skill acquisition, and on MindForge[6], which extended this direction by incorporating a structured BDI representation, social learning, and adaptive memory. CausalMind departs from these approaches by introducing a causal model directly within the BDI structure to improve internal reasoning and social reasoning when interacting with other agents. This section first presents the causal model and its assumptions, then explains how it is integrated into the overall architecture, and finally describes the synthetic dataset designed to train this model.

5.1 Causal Model

We now describe the architecture of the proposed causal model $\mathcal{M}$, its underlying assumptions, and the design choices that integrate it into the overall CausalMind framework. The causal model serves as the backbone of the BDI representation, allowing structured reasoning over both the agent's internal state and approximations of other agents' states.

Our approach builds on BISCUIT [25], which learns to disentangle latent embeddings into causal variables. While BISCUIT was originally designed for image data, CausalMind stores the BDI state in natural language, denoted $BDI_{\text{text}} \in \mathcal{T}$, to facilitate seamless interaction between multiple LLM-based modules. Hence, instead of an image encoder, we require a text encoder $\phi : \mathcal{T} \to \mathbb{R}^d$ with the following properties: (i) fixed-length embeddings of dimension d, (ii) semantically meaningful latent space to improve interpolation and generalization, and (iii) a decoder $\theta : \mathbb{R}^d \to \mathcal{T}$ such that $\theta(\phi(x)) \approx x$, enabling round-trip reconstruction into natural language. After evaluation of different architectures (see Appendix A.2), we adopt Contra-T5 as encoder ϕ, yielding

$$BDI_{\text{text}} \xrightarrow{\phi} BDI_{\text{emb}} \in \mathbb{R}^d. \tag{12}$$

These embeddings are still too large to be computed efficiently and require further dimensionality reduction. Multiple choices were considered (see Appendix A.3) but PCA was chosen for its fast fitting and interpretability.

$$BDI_{\text{emb}} \xrightarrow{\text{PCA}} BDI_{\text{emb\_pca}} \in \mathbb{R}^m. \tag{13}$$

Next, PCA embeddings $BDI^{\text{emb\_pca}}$ are mapped to causal variables $C_{BDI} \in \mathbb{R}^k$ via the causal model $\mathcal{M}$, instantiated as a normalizing flow [39]:

$$BDI_{\text{emb\_pca}} \xrightarrow{\mathcal{M}} C_{BDI}. \tag{14}$$

Since $\mathcal{M}$ is composed of a sequence of invertible transformations, we have exact reversibility:

$$BDI_{\text{emb\_pca}} = \mathcal{M}^{-1}(C_{BDI}). \tag{15}$$

To incorporate observations, each new input $O^t_{text} \in \mathcal{T}$ is first encoded into an embedding. An MLP f_ω then modifies the causal state. Letting z^t denote the causal variables at time t, we define:

$$p_\omega(z^t \mid z^{t-1}, O^t) = \prod_{i=1}^{k} p_\omega\left(z_i^t \mid z^{t-1}, \hat{I}_i^t\right), \quad \text{with} \quad \hat{I}_i^t = \tanh\left(\frac{f_i(z_i^{t-1}, O^t)}{\tau}\right), \tag{16}$$

where $\tau > 0$ is a temperature parameter used to achieve continuous relaxation.

In summary, the causal model provides a bidirectional mapping:

$$BDI_{\text{text}} \leftrightarrow BDI_{\text{emb}} \leftrightarrow BDI_{\text{emb\_pca}} \leftrightarrow C_{BDI}, \tag{17}$$

and supports causal updates conditioned on observations. It is used both to maintain the agent's internal BDI state and to approximate the causal BDI states of other agents.

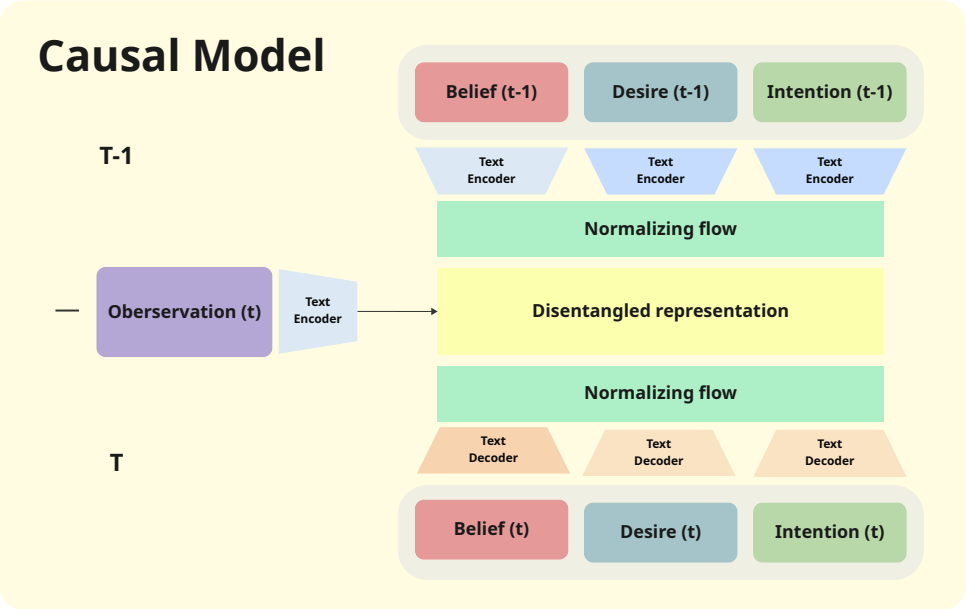


Figure 3: Architecture of the causal model. The previous BDI^{t-1}_{text} is encoded into embeddings BDI^{t-1}_{emb}, after which a normalizing flow produces a disentangled causal representation C^{t-1}_{BDI}. The observation O^t_{text} is then used to update this causal representation to obtain the new causal variables C^t_{BDI}, which are decoded back into text to form BDI^t_{text}

Figure 3 illustrates the full causal update pipeline. The previous BDI state is first encoded and mapped into a disentangled causal representation through a normalizing flow. The new observation is then encoded and used to update the causal variables, which are subsequently decoded back into text to yield the next BDI state.

Furthermore, for our causal model to work, it needs to respect 4 different assumptions, these assumptions are essential for causal variables to be correctly identified based on the theory. These 4 assumptions are the same as in BISCUIT since our model is based on it, however the inputs and interactions differs and therefore for each assumption a justification needs to be made to make sure it will identify causal variables correctly.

(A1) Interactions between the causal variables and observations can be described by binary interaction variables I^t. This means that interaction variables $\hat{I}^t$ are either in one state or another, for most environment this means either the causal variable is interacted with ($I^t = 1$) or it is not interacted with ($I^t = 0$). In CausalMind, interaction variables relate about the interactions between the given observation O^t_{text} and causal variables of BDI C^t_{BDI} . Based on BigToM[38], it is said that BDI_{text} is updated based on a given observation O^t_{text} (called percepts in their paper) and it is only updated if observation O^t_{text} impacts BDI, meaning that the interaction is binary. Therefore, since we model

BDI_{text} as causal variables C_{BDI} this interaction is also binary. It is to be noted that while theoretically it can only model binary variables, it also managed to model continuous variables as binary, in previous experiments[25].

(A2) Interaction variables I_i^t are independent of multiple causal variables $C_{BDI,i}^{t-1}$. Concretely, interaction variables I_i^t should not always interact with the same two (or more) causal variables $C_{BDI,i}^{t-1}$, otherwise it cannot disentangle them properly. Meaning there should be distinct interaction patterns for each causal variable $C_{BDI,i}^{t-1}$. This assumption can be respected by carefully designing the dataset used for training to ensure that included interactions are distinct and that the data is varied.

(A3) Causal relations can be resolved over time. A causal relation between $C_{BDI,i}^t$ and $C_{BDI,j}^t$ is not possible, it can only create causal relation with causal variable $C_{BDI,k} \in \{C_{BDI}^{t-1}, C_{BDI}^{t+1}\}$. In BigToM[38], causal variables of intentions $C_{intentions}^t$ are influenced by causal variables of both beliefs $C_{beliefs}^t$ and desires $C_{desires}^t$ from the same timestep t, nonetheless we argue that information about previous causal variables C_{BDI}^{t-1} and observation O_{text}^t should be enough to get current causal variables C_{BDI}^t. It is also important to note that environment run per timesteps and not in continue, allowing to compute the different causal variables at each timestep,

(A4) Sufficient variability of causal mechanisms (dynamics and time variability). This assumption ensures that causal relations are not static: each causal variable $C_{BDI,i}^t$ occasionally behaves differently under different conditions, allowing the model to disentangle cause and effect. Without such variability, the system could not tell whether a change in $C_{BDI,i}^t$ stems from its own dynamics or from an external intervention, this assumption is common among causal models[40, 41, 42]. This can respected by again having a enough varied dataset to have multiple different scenarios to allow for enough variability.

5.2 General Architecture

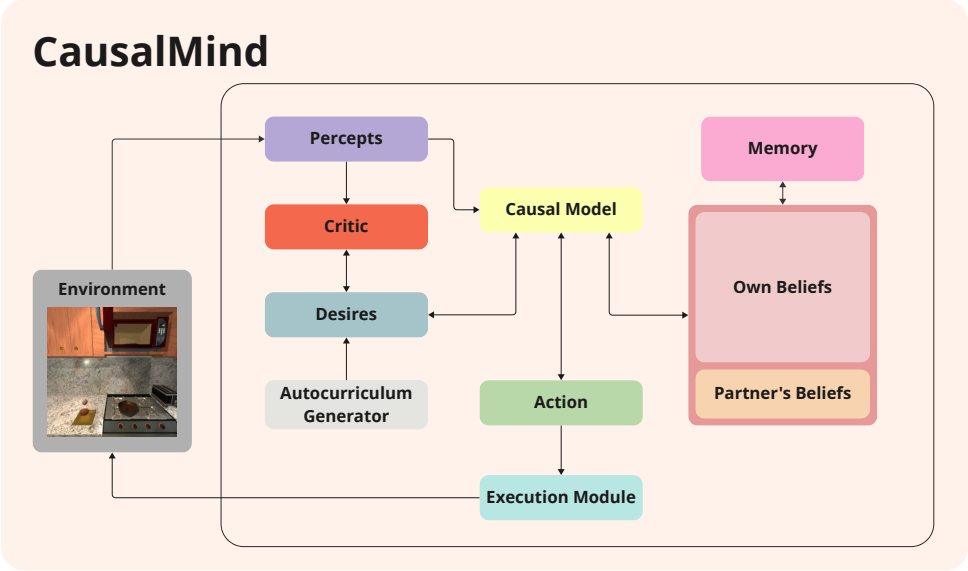


Figure 4: Overall architecture of CausalMind. The agent interacts with the environment through an execution module driven by the selected action at each step, producing new observations that update the BDI state through the causal model, which then propagates changes to the other modules

The overall agent architecture largely follows MindForge [6], as illustrated in Figure4, with the main modification being the inclusion of a causal model that updates both the agent's internal BDI state and its inferred BDI representations of other agents. Additional architectural changes were required to integrate this model and ensure proper interaction with the surrounding modules.

First, the input provided to the causal model is formatted to match the data used during training. This formatting is performed by an LLM that receives examples and produces text in the required structure. Only the most relevant information is passed to the causal model, since it was trained on single belief inputs due to the decoder's limited ability to reconstruct longer sequences. The task is obtained from the auto-curriculum and the action corresponds to the last action performed by the agent. The observation is also reformatted to match the training data.

The predicted BDI state is then processed as follows. The updated belief is stored in the memory module for future use. The updated task is checked for completion and preserved if still active. Finally, the

updated action is used to determine the next action to perform. Since the output of the BDI model is not directly executable, it cannot be passed to the execution module directly. Instead, it is passed through a prompt that converts the predicted action into a format understood by the environment, more information can be found in Appendix C.2.

Beyond the modifications required to integrate the causal model, we also introduce a change to the auto-curriculum module. In MindForge, the curriculum generator promoted open-ended exploration, aiming to cover the environment as broadly as possible, and could draw on external resources such as the Minecraft wiki to decompose tasks. In CausalMind, the auto-curriculum is goal-directed. It begins with a specific high-level objective (e.g. Cook an egg) and reasons about the intermediate steps needed to accomplish it. This change was introduced to better support controlled experimental evaluation.

5.3 Synthetic Dataset & Training

The causal model was trained on a synthetic dataset consisting of 200k samples over 150 epochs (see Appendix **??** for details). The dataset was generated using an algorithm designed to mimic how BDI^{t}_{text} should be updated from BDI^{t-1}_{text} and the current observation O^{t}_{text}, following the update rules introduced in BigToM[38].

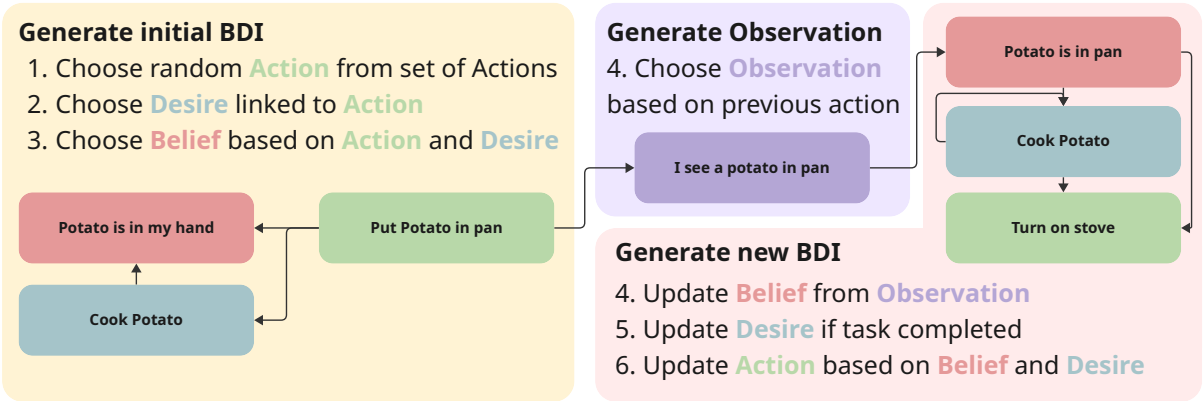


Figure 5: Overview of the procedure used to generate the synthetic dataset. The process begins by selecting an initial action, then assigning a desire related to that action and a belief relevant to the task and action. Next, an observation is generated based on the executed action, which is then used to update beliefs, revise the desire if the goal is achieved, and select the next action

As shown in Figure 5, the algorithm first selects an action at time step t. Actions include picking up an object (for example, a tomato), cutting an object, placing an object, or moving in the environment. Once an action is chosen, a corresponding desire is selected. This desire is typically related to the action. For example, picking up an egg implies a desire involving that egg, such as cooking it.

Next, a belief is selected. Due to the limitations of our current causal model, only a single belief is provided at each step. This belief is chosen to be the most relevant to the current task the agent is attempting to solve. For example, if the task involves picking up bread, the belief will focus on bread-related information. If no relevant belief exists, a different belief may be used as a fallback. Together, the selected action, desire, and belief form the initial BDI^{t-1}_{text} state.

We then construct an observation O^{t}_{text} conditioned on the selected action. Observations reflect the expected perceptual consequences of the action. For example, if the action is cutting bread, the observation would describe sliced bread. If the action is moving, the observation instead describes a random environmental perception such as "I see an apple on the table."

With the previous BDI^{t-1}_{text} state and the observation O^{t}_{text} defined, we generate the updated BDI^{t}_{text} state. Beliefs are updated based on new information obtained from the observation. For example, observing an apple on a table yields a new belief about the apple, which is added to memory so that the agent can reference it when solving future tasks. Desires are updated next. If the task is completed, the desire updates to "goal is achieved"; otherwise, it remains unchanged, following the update rules of BigToM[38]. Finally, a new action is generated based on the updated belief and desire.

Two versions of the synthetic dataset were created. The second version builds on the first to introduce more variation and correct unintended issues. One significant modification concerns how movement is represented. In the first version, exploratory movement actions were sampled uniformly from the available movement primitives in the environment: Move Ahead, Move Back, Move Right, Move Left,

Rotate Right, and Rotate Left. Because these actions were chosen at random, the model had no consistent signal indicating which movements were desirable and was penalized even when selecting a reasonable alternative. We hypothesized that this inconsistency could interfere with learning the underlying causal structure. To address this, all movement actions were replaced by a single generic action, Move Around, in the second dataset version.

Beliefs(t-1)	**Desires(t-1)**	**Intentions(t-1)**	**Observation (t)**	**Beliefs(t)**	**Desires(t)**	**Intentions(t)**
potato is on worktop	Cook potato	Pick potato	I see a potato in my hand	potato is in my hand	Cook potato	Rotate Right
I have a potato in my hand	Slice potato	Slice potato	I see potato sliced	sliced potato is in my hand	goal is achieved	No action
I have no beliefs	Slice bread	Rotate Left	I see a sliced tomato on dining table	sliced tomato is on dining table	Slice bread	Move Back
sliced tomato is in pan	Cook sliced tomato	Turn on stove	I see a cooked sliced tomato	sliced tomato is cooked	goal is achieved	No action

Table 1: Synthetic data generation process. For each step, an action is selected, a corresponding desire is assigned, and a belief is chosen based on task relevance (or randomly if none apply). The action produces an observation that updates beliefs, revises desires when goals are achieved, and informs the next action

Table 1 shows several example samples from our synthetic dataset V1. Each row illustrates how the agent's internal state evolves over time, including its previous BDI_{text}^{t-1}, the current observation O^t, and the updated BDI_{text}^{t} that results from processing this new information. These examples highlight the causal dependencies between beliefs, desires, and intentions, which are essential for training the causal model to reason about how new observations lead to state and action updates.

6 Experiments & Results

This section presents the experiments conducted using the Causal Model and CausalMind, along with their results. We begin by evaluating how well the Causal Model predicts the new BDI_{text}^{t} based on the previous BDI_{text}^{t-1} and the current observation O^t, and compare its performance to that of an LLM-based predictor. Next, we perform an in-depth analysis of the causal relationships learned by the model and their overall influence on system behavior. Finally, we evaluate CausalMind across multiple tasks of increasing difficulty, both in isolation and in collaboration with another agent, and compare its performance to Voyager, MindForge, as well as to an ablated version of CausalMind that uses a simple causal model instead of the full BDI causal model.

6.1 Causal Model

This experiment evaluates if the causal model managed to disentangle the different causal variables properly and is able to correctly predict BDI_{text}^{t}. Multiple version of our causal model are compared, they differ in the number of latent variables they have: 75, 96, or 300 (the latents are divided equally between BDI, see more in Appendix A.3). It will also be compared to an LLM (Gemini 2.0 Flash[43]) to see how it performs against the current method used to update BDI_{text}.

Figure 6 presents the cosine similarity between the predicted and ground truth embeddings of beliefs, desires, and intentions for the different causal models, as well as for the Gemini 2.0 Flash evaluated on our synthetic dataset V1. The results show that the causal models achieve moderate performance, with variability across the three components. Models with fewer latent variables (75L and 96L) generally outperform the larger 300L model, suggesting that excessive latent dimensionality may hinder effective disentanglement of causal factors. Among the three dimensions, the prediction of desires tends to yield the highest cosine similarity, followed by intentions and beliefs. Gemini 2.0 Flash model exhibits the highest overall performance across all components, demonstrating stronger alignment with the ground truth embeddings.

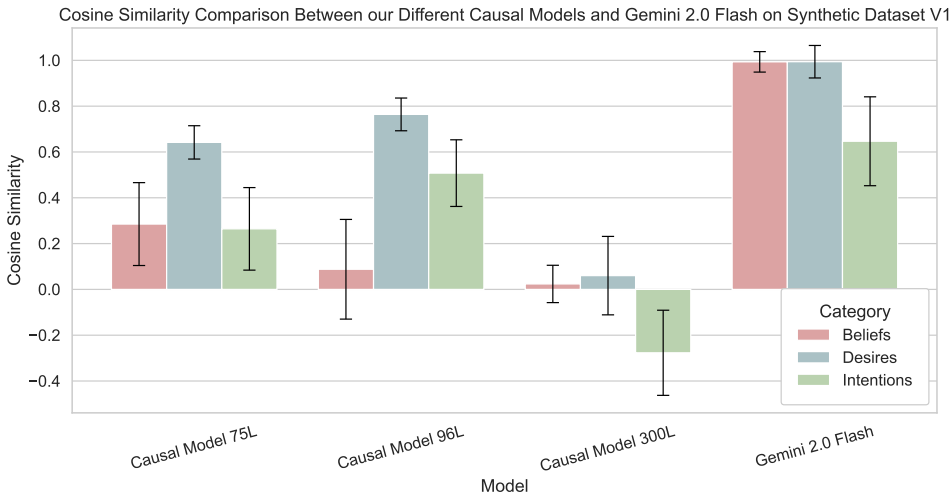


Figure 6: Cosine similarity between the predicted embeddings $\hat{BDI}^t_{emb}$ and the ground truth BDI^t_{emb} for different causal models and Gemini 2.0 Flash [43] evaluated on our synthetic dataset V1. Each causal model differs in the number of latent variables (75, 96, or 300, divided equally among beliefs, desires, and intentions). Higher values indicate better alignment with the ground truth, reflecting more accurate disentanglement and prediction of the causal variables.TODO add in figure the title saying it s dataset v1

With the model performing significantly worse than Gemini, we hypothesized that the issue stemmed from limitations in the synthetic dataset. We therefore improved the dataset and repeated the evaluation against Gemini 2.0 Flash. In this second run, only the best performing causal model (96L) was retrained and evaluated.

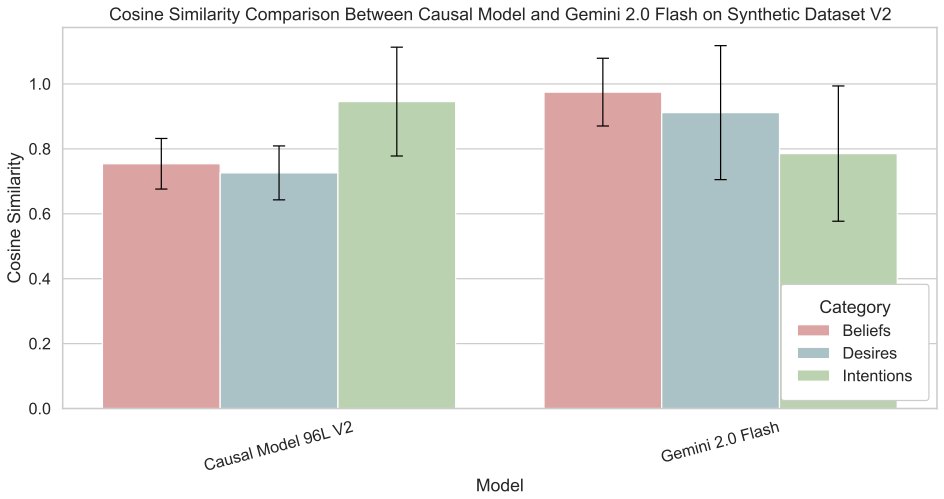
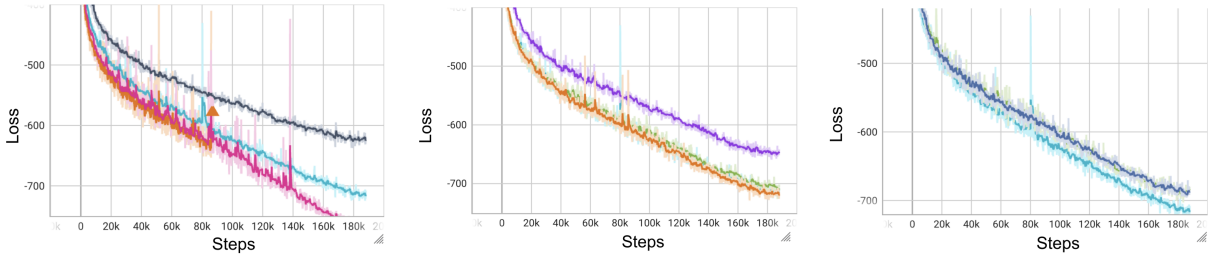


Figure 7: Cosine similarity between the predicted embeddings $\hat{BDI}^t_{emb}$ and the ground truth BDI^t_{emb} for our causal model (96L) and Gemini 2.0 Flash [43] evaluated on our synthetic dataset V2. Compare to our results on the V1 dataset, our causal model performs better, but still lacks behind.

Figure 7 compares the performance of our model and Gemini 2.0 Flash on predicting BDI^t_{emb}. Compared to results obtained using the first version of the synthetic dataset, the new dataset leads to improved predictions for both beliefs and intentions, while performance on desires remains similar. This suggests that the refined dataset contributes to better overall predictive accuracy. Gemini 2.0 Flash performs similarly on beliefs and desires, and shows improved performance on intentions.

After observing the model's poor performance, we conducted a series of experiments to identify potential causes. We trained multiple variants of the model using different learning rates, numbers of flow layers, and numbers of hidden layers to evaluate how these factors influenced performance.

Figure 8a shows the effect of learning rate on training dynamics. The tested values were 0.00003 (gray),



(a) Loss curve for the learning rates

(b) Loss curve for the number of flows

(c) Loss curve for the number of hidden layers

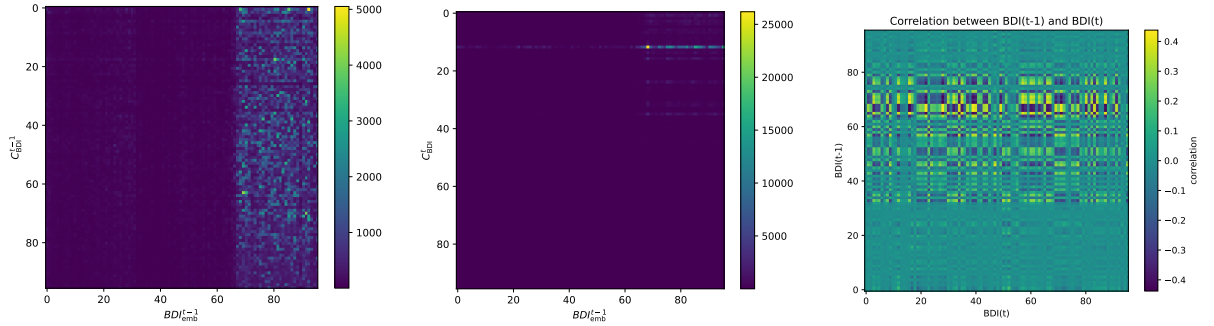
Figure 8: Comparison of loss curves for (a) learning rates, (b) number of flows, and (c) number of hidden layers. Varying these parameters has only a minimal effect on performance, indicating that they are not the primary source of the model's limitations.

0.0001 (pink), 0.0002 (blue), and 0.0003 (orange). A higher learning rate leads to numerical instability and produces NaN values, while a lower learning rate results in worse performance. The model trained with a learning rate of 0.0001 performs best and is therefore used in subsequent experiments.

Figure 8b compares different numbers of flow layers. The default model uses 4 flows (blue). Increasing the number to 5 (green) or 6 (orange) produces nearly identical performance, while reducing the number to 2 flows (purple) degrades performance. This suggests that increasing flow depth does not improve learning, while reducing depth harms expressiveness.

In Figure 8c, we varied the number of hidden layers used to condition the latent updates. The default model uses 196 layers (blue). Reducing the size to 128 (green) or 64 (dark blue) decreases performance. We did not test larger values due to instability during training, although based on the trends observed here we do not expect significant improvement.

These experiments indicate that factors such as learning rate, number of flow layers, and number of hidden layers do not explain the model's poor performance.



(a) Influence of the flow from BDI^{t-1}_{emb} to C^{t-1}_{BDI}

(b) Influence of the flow and prior from BDI^{t-1}_{emb} to C^{t-1}_{BDI}

(c) Correlation map between BDI^{t-1}_{emb} and BDI^{t}_{emb}

Figure 9: Comparison of (a) a heatmap of the normalizing flow, (b) a heatmap of the full model, and (c) a correlation map between BDI^{t-1}_{emb} and BDI^{t}_{emb}. The first heatmap shows how the normalizing flow transforms BDI^{t-1}_{emb} into C^{t-1}_{BDI}. The second heatmap shows how the normalizing flow transforms combined with the prior transforms BDI^{t-1}_{emb} into C^{t}_{BDI}. Both heatmaps indicate that information from the action dominates throughout the update process. The correlation map confirms these observations, showing that updated embeddings are primarily influenced by the action and to a lesser degree by the desire.

Figure 9 presents three heatmaps maps, (a) shows the influence of the flow, which corresponds to the Jacobian of the latent variables with respect to the embedding dimensions, (b) shows the influence of the full model, which corresponds to the Jacobian of the final output after both the flow and the MLP head.

We observe that the flow in subfigure 9a reacts strongly only to a narrow set of embedding dimensions corresponding to the action, while the rest of the embedding space exerts minimal influence on the latent representation. In subfigure 9b, this pattern becomes even more concentrated, with the MLP relying on only a few latent directions that originate from the action-related dimensions. This behavior is further

supported by the correlation map in subfigure 9c, which shows that BDI^{t}_{emb} is influenced primarily by the action-related dimensions of BDI^{t-1}_{emb}.

This result is unexpected. In principle, the flow should use information from all components of the BDI state, not only the action. Beliefs should contribute to updating desires and selecting new actions, while desires should affect both their own update and the next action. However, the heatmaps in Figure 9 shows that the update depends almost exclusively on the action embedding. Moreover, when generating the new BDI representation through the MLP, the update primarily affects a single latent dimension rather than reflecting a structured transformation across the full representation.

After observing this behavior, we conducted further analysis, this time focusing on the dataset itself. We identified a strong imbalance in the distribution of predicted actions, with approximately 90 percent of samples containing the action "Move Around." Since the dataset is intended to reflect an agent acting in an open environment, a high frequency of exploratory movement is expected. During dataset construction, we were aware that "Move Around" was overrepresented, but did not anticipate that it would cause issues, given that BISCUIT[25] exhibits a similar imbalance where most actions produce no changes.

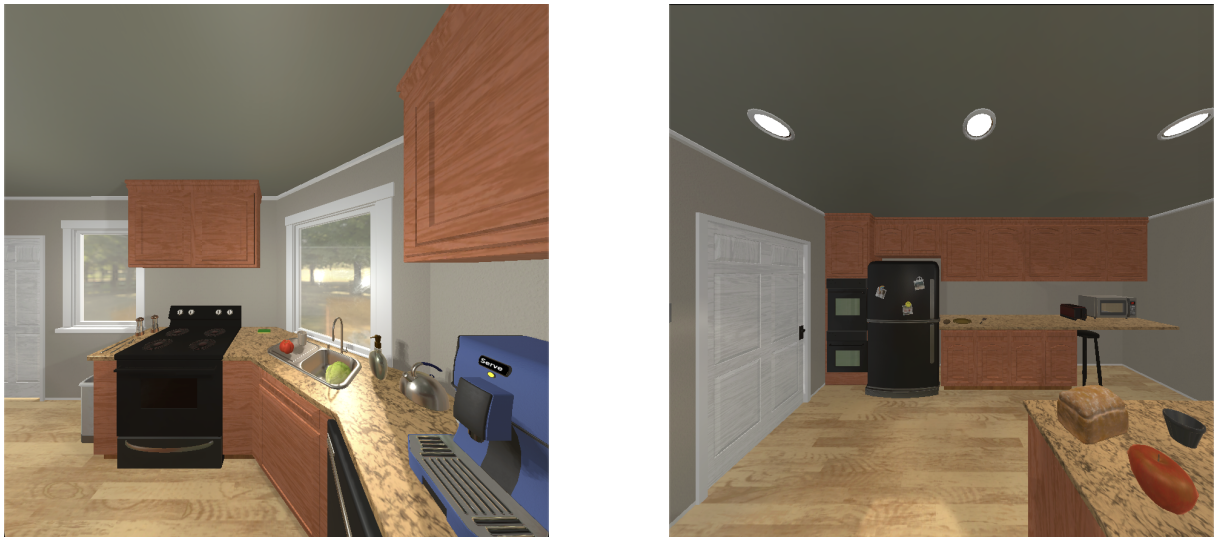
To test whether the model had overfit to this action, we evaluated performance only on samples where the correct action was not "Move Around". Under this filtered evaluation, accuracy on action prediction dropped from 0.95 to 0.36. Despite all targets being different from "Move Around", the model continued to output values close to this action, indicating that it had overfit to the majority class rather than learning distinct action representations.

We performed a second analysis to evaluate whether the model could correctly identify when a task is completed. Using a similar filtering procedure, we retained only samples where the correct desire was "goal is achieved." The model again failed to generalize, instead continuing to output the current desire rather than the completed-state label, causing performance to drop from 0.73 to 0.20.

6.2 CausalMind

6.2.1 Experimental Setup

To evaluate our agent against both Voyager and MindForge, we reimplemented both systems in Autogen, since their original implementations were designed specifically for Minecraft and were not compatible with our environment setup.



(a) (b)

Figure 10: Picture of iTHOR environment used for experiments

CausalMind will be evaluated in iTHOR[44], it is a 3D environment with four main characteristics:

- Object Manipulation: Allowing objects to be picked up, dropped, or placed on other items.
- State Changes: Letting items such as a micro wave to be turned on and off.
- Multi-agent: Multiple agents can interact together and the same environment at the same time.

- Realistic Setup: Graphics and physics of the environment are realistic allowing for more realistic scenarios to be tested.

Example pictures of iTHOR environment can be seen in Figure 10.

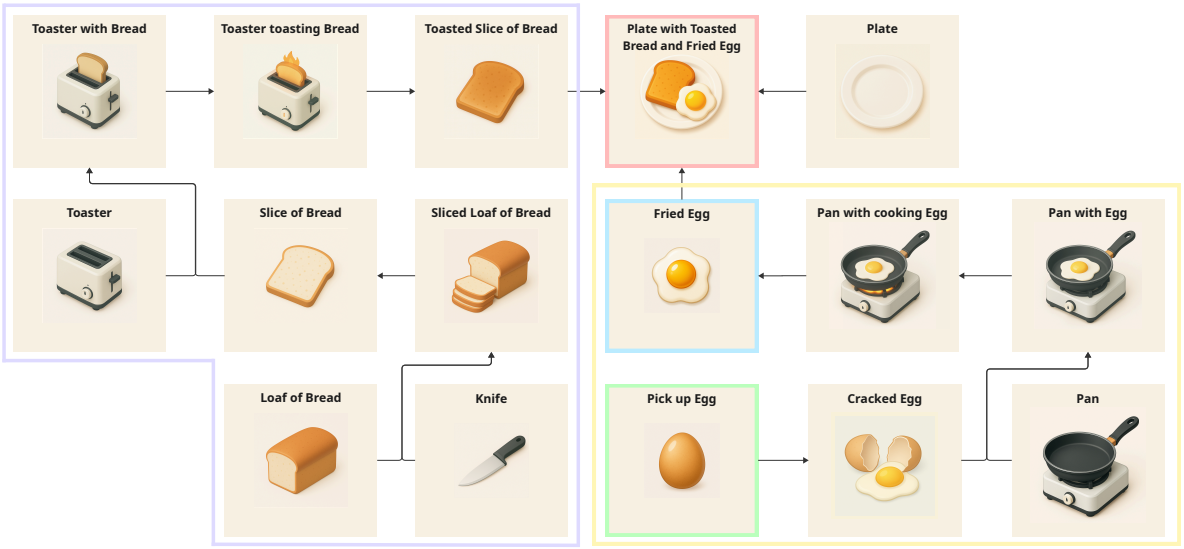


Figure 11: Causal skill tree outlining the actions required for the three evaluation tasks. Each task is represented by its final highlighted outcome: green for Task 1 (easy), where the goal is to pick up the egg; blue for Task 2 (medium), where the goal is to obtain a fried egg; and red for Task 3 (hard), where the goal is to produce a plate containing fried egg and toasted sliced bread

There are three tasks used to evaluate the different difficulty levels, as shown in Figure 11:

1. **Pick up Egg (Easy)** (green): The agent must locate the egg and pick it up. This task is considered easy because it requires only one interaction after finding the target object.
2. **Fried Egg (Medium)** (blue): This task extends the easy task by requiring the agent to crack the egg, place it in a pan, put the pan on the stove, turn the stove on, and retrieve the cooked egg. It is considered medium difficulty because it involves multiple sequential steps that must be executed in the correct order while reasoning about a single object.
3. **Fried Egg and Toasted Sliced Bread (Hard)** (red): This task builds on the medium task. After cooking the egg, the agent must find a knife, cut the bread, pick up the sliced bread, place it in the toaster, turn the toaster on, retrieve the toasted bread, and serve both items together on a plate. This task is considered hard because the agent must complete two multi-step procedures without losing track of the overall objective. Working with bread and a knife is also more complex due to the greater number of required interactions.

Each task must be completed within a maximum of 100 steps. Experiments are divided into two settings: a solo setting and a multi-agent (duo) setting.

In the solo setting, four agents are evaluated: Voyager, MindForge, CausalMind, and CausalMind (Combined). Voyager and MindForge are included as baselines, since CausalMind builds upon these architectures. In particular, MindForge serves as a direct comparison to assess the difference between updating BDI with an LLM versus a causal model. The CausalMind (Combined) variant uses a single embedding of size 96 rather than separate embeddings for beliefs, desires, and intentions (32 each). This variant evaluates whether separating BDI components into distinct embeddings provides benefits over a unified representation. It is important to note that CausalMind (and CausalMind Combined) uses the critic module to detect task completion. As shown in the earlier analysis, the causal model alone cannot reliably recognize completed tasks, which would cause the agent to remain stuck on the initial task.

In the multi-agent setting, experiments involve two agents. Pairs may consist of two MindForge agents, two CausalMind agents, or one of each. This setup allows us to compare both architectures in social settings and to test whether they can cooperate when paired together. In these experiments, one agent acts as an expert and provides guidance to the other, allowing us to evaluate social learning.

Two metrics are used to evaluate the agents. The first metric is the success rate, which measures how often an agent completes a task. This metric reflects the robustness of the agent: a higher success rate

indicates that the agent can reliably solve the task.

The second metric is the average number of steps, which counts how many steps the agent takes on average to complete a task. A step is defined as either executing an action in the environment (e.g. Pick up egg) or communicating with another agent. This metric evaluates the efficiency of the agent in acquiring skills and solving tasks.

6.2.2 Causal Tasks Results

Benchmark Metric	Voyager	MindForge (solo)	CausalMind (solo, ours)	CausalMind (solo, C)
Task 1 success rate	50%	90%	80%	0.6%
Task 1 avg. steps	8.4	8	9.4	12
Task 2 success rate	0%	20%	0%	0%
Task 2 avg. steps	—	17.5	—	—
Task 3 success rate	0%	0%	0%	0%
Task 3 avg. steps	—	—	—	—

Table 2: Benchmark results comparing Voyager, MindForge (solo), CausalMind (solo, ours), and CausalMind (Combined). On the 3 tasks introduced in the previous section

In Table 2, we report solo benchmark results for Voyager, MindForge, CausalMind, and CausalMind (Combined). Voyager struggles even on the easiest task and performs significantly worse than the more advanced agents. CausalMind (Combined) performs worse than the standard CausalMind model, indicating that separating beliefs, desires, and intentions into distinct embeddings leads to better performance. MindForge is the only agent capable of completing the medium task, showing that our causal model remains inferior to using an LLM for BDI updates in this setting. No agent successfully completed the hard task.

Benchmark Metric	MindForge (duo)	CausalMind (duo, ours)
Task 1 success rate	100%	100%
Task 1 avg. steps	5.77	9
Task 2 success rate	20%	0%
Task 2 avg. steps	8.5	—
Task 3 success rate	0%	0%
Task 3 avg. steps	—	—

Table 3: Benchmark results comparing MindForge (duo) and CausalMind (duo, ours). On the 3 tasks introduced in the previous section

Table 3 reports results for MindForge and CausalMind in a teacher–student setup. Both agents perform similarly to their solo evaluations, with MindForge again outperforming CausalMind.

Comparing Table 2 and Table 3 shows that performance remains largely unchanged across settings, suggesting that the teacher–student setup does not provide measurable benefits. The primary reason is that the critique frequently hallucinates task completion. In many runs, the critique incorrectly declares a task as completed, causing the agent to move on despite failing to achieve the required outcome. This occurs most often in the medium task, where the agent places the egg in the pan and incorrectly assumes it is cooked without activating the stove. This failure is partly due to limited visual understanding, leading the model to misinterpret observations. As a result, the agent does not recognize that help is needed and therefore does not request assistance from the teacher. It may also hallucinate new tasks and prematurely assume they are completed.

None of the agents completed the hard task. Beyond hallucinated completion, two additional issues contribute to failure. First, the auto-curriculum sometimes generates plans that do not align with the environment's causal skill tree, because the agent lacks full knowledge of environment constraints. This can result in goals that are impossible (for example, flipping the egg) or unnecessary (for example, requiring a spatula). Combined with hallucinated progress, this causes agents to pursue irrelevant subtasks. Second, the environment is highly unforgiving. If the agent drops an item on the floor, the item often becomes unrecoverable. In many trials, essential objects were lost, making task completion impossible. These problems collectively make even the medium task difficult to complete reliably.

Finally, action analysis confirms behavior observed in our earlier causal model experiments: the model predicts nearly the same action throughout execution, most often "Move Around". This matches our previous findings that the action dimensions dominate the embedding and that the model overfits to this majority class. Interestingly, the agent occasionally executes a different action only due to the language model responsible for translating predicted actions into environment-specific commands. Because the causal model frequently produces outputs close to "Move Around", the LLM interprets these text strings and sometimes maps them to alternative valid actions. In these cases, progress is achieved through linguistic interpretation rather than through a causally grounded action update.

7 Future Work

To strengthen the causal model, several directions are possible. The first step would be to address the imbalance present in the synthetic dataset so that all causal patterns are represented more uniformly, helping the model avoid overfitting to a subset of causal relations.

A key long term goal is to develop a causal model that operates directly on natural language. Since the original architecture was designed for images, this work represents its first application to text input. We assumed that this model would translate directly onto text, since it acts on embedding. But, due to our inability to make it work, we would recommend starting from a smaller version of the model, since as shown in Figure 9 most dimensions are not used and then verify that it can reliably recover the intended causal variables. Once this baseline is stable, the architecture can be expanded incrementally, with repeated evaluation at each stage. Inspecting the Jacobian matrix after each training cycle remains a useful diagnostic to verify that the discovered relationships align with the target causal structure.

When the appropriate number of latent variables has been identified, the PCA based encoder can be replaced with a more expressive representation method tailored to the task. PCA remains useful for early experimentation since it trains quickly, but more powerful encoders would be beneficial once the rest of the pipeline stabilizes.

Another limitation lies in the current methods used for text embeddings. Many modern embedding models lack decoders and produce high dimensional representations that cannot be passed directly through the normalizing flow, which is why PCA is required as an intermediate compression step. An alternative direction would be to replace the normalizing flow with another generative backbone, for example a diffusion model, though this would require further investigation to ensure that causal variables can still be extracted and controlled reliably.

The largest obstacle remains data. There is no existing dataset for this task, and the synthetic dataset used here cannot capture the full richness of real interactions. Two paths are worth exploring. The first is to continue expanding the synthetic dataset by introducing more diverse cases, potentially through a state machine that systematically enumerates plausible scenarios. The second is to generate larger datasets using generative models and either manually curate the outputs or leverage crowdsourcing to validate them. Both strategies could help bridge the gap between synthetic training conditions and the complexity required for robust causal reasoning.

As for CausalMind, the main limitation lied in the causal model, some other improvements can only be made by swapping Gemini 2.0 Flash to a better model, such as its vision capability. One improvement could be to let the agents always communicate to ensure the teacher can correct the student directly, reducing the risk of hallucinating the success of tasks and of doing non-useful tasks. Additionally, further information could be saved about where the agent dropped items to make sure it can find them again.

8 Conclusion

This work explored whether a causal representation of beliefs, desires, and intentions could improve the reasoning abilities of LLM driven embodied agents. We introduced CausalMind, an architecture that replaces template based updates with a causal model built on top of text embeddings and normalizing flows. The idea was to move beyond correlation based updates and toward structured causal reasoning that supports both solo and collaborative tasks.

Our analysis of the causal model exposed several key factors that shaped its performance. The number of latent causal variables had a strong impact on stability, and the Jacobian analysis showed that the model failed to distribute influence across the BDI space. Instead, it leaned almost entirely on the action dimensions and ignored most belief and desire signals, a clear sign that the causal variables were not learned as intended. This narrow sensitivity came from overfitting to an imbalanced synthetic dataset and from the difficulty of training a flow based causal model when data are limited. The improved

synthetic dataset did raise performance, which confirms that data quality is a major bottleneck, but the model still struggled to recover the underlying causal structure.

At the agent level, similar limitations emerged. CausalMind and its baselines performed inconsistently in iTHOR because the vision model often misinterpreted observations, which caused the agent to believe that a task was complete when it was not. This created a feedback loop that weakened both planning and social learning, since a student agent that believes it has already succeeded will not seek help from the teacher.

Overall, the results show that integrating a causal model into an embodied LLM agent remains challenging. Both the causal model and the agent depend heavily on the quality of the data, the embedding structure, and the perceptual pipeline. Still, the experiments make clear which components require redesign and provide a path toward agents that can eventually ground their reasoning in causal structure rather than surface level correlations.

References

[1] Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiaoming Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Mengxue Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, Yiqiao Jin, Fan Zhang, Xianhong Wu, Hanqing Zhao, Dacheng Tao, Philip S. Yu, and Ming Zhang. Large language model agent: A survey on methodology, applications and challenges. *ArXiv*, abs/2503.21460, 2025. URL `https://api.semanticscholar.org/CorpusID:277350072`.

[2] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

[3] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. Hallucination is inevitable: An innate limitation of large language models, 2025. URL `https://arxiv.org/abs/2401.11817`.

[4] Judea Pearl. *Causality*. Cambridge university press, 2009.

[5] Adib Bazgir, Amir Habibdoust, Yuwen Zhang, and Xing Song. Causal mas: A survey of large language model architectures for discovery and effect estimation. *arXiv preprint arXiv:2509.00987*, 2025.

[6] Mircea Lică, Ojas Shirekar, Baptiste Colle, and Chirag Raman. Mindforge: Empowering embodied agents with theory of mind for lifelong collaborative learning, 2024. URL `https://arxiv.org/abs/2411.12977`.

[7] Matej Zečević, Moritz Willig, Devendra Singh Dhami, and Kristian Kersting. Causal parrots: Large language models may talk causality but are not causal. *arXiv preprint arXiv:2308.13067*, 2023.

[8] Judea Pearl. Causal inference. In Isabelle Guyon, Dominik Janzing, and Bernhard Schölkopf, editors, *Proceedings of Workshop on Causality: Objectives and Assessment at NIPS 2008*, volume 6 of *Proceedings of Machine Learning Research*, pages 39–58, Whistler, Canada, 12 Dec 2010. PMLR. URL `https://proceedings.mlr.press/v6/pearl10a.html`.

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

[10] Jin-Fang Gao, Xiao Ding, Yiming Cui, Jianbai Zhao, Hepeng Wang, Ting Liu, and Bing Qin. Self-evolving gpt: A lifelong autonomous experiential learner. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL `https://api.semanticscholar.org/CorpusID:271162185`.

[11] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

[12] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models, 2024. URL `https://arxiv.org/abs/2307.02485`.

[13] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008, 2023.

[14] Shu Yu and Chaochao Lu. Adam: An embodied causal agent in open-world environments, 2024. URL `https://arxiv.org/abs/2410.22194`.

[15] Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-2: Multimodal minecraft agent with goal-observation-action conditioned policy, 2025. URL `https://arxiv.org/abs/2502.19902`.

[16] John Gkountouras, Matthias Lindemann, Phillip Lippe, Efstratios Gavves, and Ivan Titov. Language agents meet causality – bridging llms and causal world models, 2024. URL `https://arxiv.org/abs/2410.19923`.

[17] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018.

[18] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[19] Aneesh Komanduri, Xintao Wu, Yongkai Wu, and Feng Chen. From identifiable causal representations to controllable counterfactual generation: A survey on causal generative modeling, 2024. URL `https://arxiv.org/abs/2310.11011`.

[20] Weiran Yao, Yuewen Sun, Alex Ho, Changyin Sun, and Kun Zhang. Learning temporally causal latent processes from general temporal data. *arXiv preprint arXiv:2110.05428*, 2021.

[21] Weiran Yao, Guangyi Chen, and Kun Zhang. Temporally disentangled representation learning. *arXiv preprint arXiv:2210.13647*, 2022.

[22] Xiangchen Song, Weiran Yao, Yewen Fan, Xinshuai Dong, Guangyi Chen, Juan Carlos Niebles, Eric Xing, and Kun Zhang. Temporally disentangled representation learning under unknown nonstationarity. *Advances in Neural Information Processing Systems*, 36:8092–8113, 2023.

[23] Phillip Lippe, Sara Magliacane, Sindy Löwe, Yuki M Asano, Taco Cohen, and Stratis Gavves. Citris: Causal identifiability from temporal intervened sequences. In *International Conference on Machine Learning*, pages 13557–13603. PMLR, 2022.

[24] Phillip Lippe, Sara Magliacane, Sindy Löwe, Yuki M Asano, Taco Cohen, and Efstratios Gavves. icitris: Causal representation learning for instantaneous temporal effects. In *UAI 2022 Workshop on Causal Representation Learning*, 2022.

[25] Phillip Lippe, Sara Magliacane, Sindy Löwe, Yuki M Asano, Taco Cohen, and Efstratios Gavves. Biscuit: Causal representation learning from binary interactions. In *Uncertainty in Artificial Intelligence*, pages 1263–1273. PMLR, 2023.

[26] Sébastien Lachapelle and Simon Lacoste-Julien. Partial disentanglement via mechanism sparsity. *arXiv preprint arXiv:2207.07732*, 2022.

[27] Julius Von Kügelgen, Yash Sharma, Luigi Gresele, Wieland Brendel, Bernhard Schölkopf, Michel Besserve, and Francesco Locatello. Self-supervised learning with data augmentations provably isolates content from style. *Advances in neural information processing systems*, 34:16451–16467, 2021.

[28] Johann Brehmer, Pim De Haan, Phillip Lippe, and Taco S Cohen. Weakly supervised causal representation learning. *Advances in Neural Information Processing Systems*, 35:38319–38331, 2022.

[29] Angelos Nalmpantis, Phillip Lippe, and Sara Magliacane. Hierarchical causal representation learning. In *Causal Representation Learning Workshop at NeurIPS 2023*.

[30] Sébastien Lachapelle, Pau Rodríguez López, Yash Sharma, Katie Everett, Rémi Le Priol, Alexandre Lacoste, and Simon Lacoste-Julien. Nonparametric partial disentanglement via mechanism sparsity: Sparse actions, interventions and sparse temporal dependencies, 2024. URL `https://arxiv.org/abs/2401.04890`.

[31] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

[32] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[33] Chen Gao, Xiaochong Lan, Nian Li, Yuan Yuan, Jingtao Ding, Zhilun Zhou, Fengli Xu, and Yong Li. Large language models empowered agent-based modeling and simulation: A survey and perspectives. *Humanities and Social Sciences Communications*, 11(1):1–24, 2024.

[34] Jinghua Piao, Yuwei Yan, Jun Zhang, Nian Li, Junbo Yan, Xiaochong Lan, Zhihong Lu, Zhiheng Zheng, Jing Yi Wang, Di Zhou, et al. Agentsociety: Large-scale simulation of llm-driven generative agents advances understanding of human behaviors and society. *arXiv preprint arXiv:2502.08691*, 2025.

[35] Senkang Hu, Zhengru Fang, Zihan Fang, Yiqin Deng, Xianhao Chen, and Yuguang Fang. Agentscodriver: Large language model empowered collaborative driving with lifelong learning. *ArXiv*, abs/2404.06345, 2024. URL `https://api.semanticscholar.org/CorpusId:269009845`.

[36] Xinran Li, Chenjia Bai, Zijian Li, Jiakun Zheng, Ting Xiao, and Jun Zhang. Learn as individuals, evolve as a team: Multi-agent llms adaptation in embodied environments. *ArXiv*, abs/2506.07232, 2025. URL https://api.semanticscholar.org/CorpusId:279250751.

[37] Zhiwei Liu, Linsey Pang, Justin Turnau, Vishnu Nandam, Hua Wei, Longchao Da, and Huaiyuan Yao. Comal: Collaborative multi-agent large language models for mixed-autonomy traffic. *ArXiv*, abs/2410.14368, 2024. URL https://api.semanticscholar.org/CorpusId:273482255.

[38] Kanishk Gandhi, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D. Goodman. Understanding social reasoning in language models with language models, 2023. URL https://arxiv.org/abs/2306.15448.

[39] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2016. URL https://arxiv.org/abs/1505.05770.

[40] Aapo Hyvarinen, Hiroaki Sasaki, and Richard Turner. Nonlinear ica using auxiliary variables and generalized contrastive learning. In *The 22nd international conference on artificial intelligence and statistics*, pages 859–868. PMLR, 2019.

[41] Weiran Yao, Guangyi Chen, and Kun Zhang. Temporally disentangled representation learning. *arXiv preprint arXiv:2210.13647*, 2022.

[42] Weiran Yao, Yuewen Sun, Alex Ho, Changyin Sun, and Kun Zhang. Learning temporally causal latent processes from general temporal data. *arXiv preprint arXiv:2110.05428*, 2021.

[43] Google DeepMind. Introducing gemini 2.0: Our new ai model for the agentic era. https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/, 2024. Accessed: YYYY-MM-DD.

[44] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai, 2022. URL https://arxiv.org/abs/1712.05474.

[45] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[46] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7871–7880, 2020.

[47] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, 2025.

[48] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

[49] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[50] Jiasen Lu, Christopher Clark, Sangho Lee, Zichen Zhang, Savya Khosla, Ryan Marten, Derek Hoiem, and Aniruddha Kembhavi. Unified-io 2: Scaling autoregressive multimodal models with vision language audio and action. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26439–26455, 2024.

[51] Paul-Ambroise Duquenne, Holger Schwenk, and Benoît Sagot. Sonar: sentence-level multimodal and language-agnostic representations. *arXiv preprint arXiv:2308.11466*, 2023.

[52] John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M Rush. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, 2023.

[53] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[54] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

A Causal Model

A.1 Choice of Causal Model

The choice of BISCUIT[25] was motivated by several factors. First, it takes embeddings as input which allows it to work with any high dimensional data that can be compressed into embeddings (i.e. images or text). Second, it provides disentangled causal variables which is useful for interpretability and ensures the model captures causal structure. Third, it allows interception on these causal variables, enabling modifications to the input, in our case updating the BDI. However, any other framework that supports interception over text or embeddings could also be used with our agent.

A.2 Text Encoder-Decoder

To use BISCUIT[25] as a causal model we needed to convert our BDI to latents, which required an encoder decoder architecture. We first looked at text encoders such as BERT[45], BART[46], ModernBERT[47], Matryoshka[48], and T5[49], but none of them allowed decoding the embeddings back into text. Through our research we found only four models that include a decoder: Unified IO 2[50], SONAR[51], Vec2Text[52], and Contra-T5[1].

Unified IO 2[50] was discarded because the embedding size depends on the input format which makes it harder to use as a fixed latent space for causal modeling. Vec2Text[52] was also discarded because it relies on OpenAI text embeddings (text-embedding-ada-002) which are paid. This left us with two viable options: SONAR[51] and Contra T5.

Original
`I see an egg in the fridge`
Reconstructed
`It is in the Time in it was not in the City in it to be in the City. - it does not have it be in the City. - it does not start as an in the up-to-the-time in in in the in the up-and in the up-and in in the in the up-and in in in the up-and in in in in the up-and in in in in in the up-and in in in in the up-and in in in in the up-`

Table 4: Example of reconstruction failure when applying PCA on SONAR[51]

Both options performed similarly when using their full embedding size. However, after applying PCA, SONAR[51] collapsed as shown in Figure 4, which displays the original sentence and the reconstructed sentence after passing it through SONAR and PCA. This left us choosing Contra T5. Contra-T5 has multiple model sizes:

- Small: 60M params, 512 embedding dimensions
- Base: 220M params, 768 embedding dimensions
- Large: 770M params, 1024 embedding dimensions
- XL: 3B params, 2048 embedding dimensions

Through manually testing the different models, we found out that models below large are not able to reconstruct embeddings properly for our use case. Therefore we chose to use the large model.

A.3 Reducing Embeddings Size

Using the default size of our text embeddings would lead us to have an input size of 3072 (1024 for Beliefs + 1024 for Desires + 1024 for Intentions) for our causal model, however due to scaling issues of the normalizing flow, this would lead our model to have more than 11B parameters. The solution being reducing the embedding size, two options were considered, PCA[53] and a simple multilayer perceptron (MLP). The simple MLP did not work at all for us, text was when run through and converted back to text it would output incomprehensible gibberish, we do think using an MLP is a viable option but we did not have time to make it work. We also tried using PCA[53], and this method worked perfectly for multiple reasons. First, fitting PCA[53] to our dataset is very fast, allowing us to experiment with different embedding sizes easily. Secondly, it outputs the explained variance ratio, helping us understand how much performance we lose when reducing the number of dimension.

[1] https://huggingface.co/thesephist/contra-bottleneck-t5-xl-wikipedia

BDI Embedding Size	Explained Variance Ratio	Cosine Similarity	Causal Model Size
75	0.909	0.975	47.3M
96	0.943	0.987	57.5M
300	0.997	1.000	224M
1024	1.00	N/A	2.3B

Table 5: Effect of PCA-reduced embedding dimensionality

As mentioned in the Result Section, we test 3 different embedding sizes. Table 5 displays how reducing the dimensionality impacts both the variance preserved by PCA and the similarity between the original and compressed embeddings, as well as the resulting model size. We observe that even with a small embedding size of 96, we retain more than 94 percent of the variance with almost identical cosine similarity, while reducing the model size by more than an order of magnitude compared to a 1024 dimensional representation. The 1024 dimensional case is included in the table to highlight how quickly the model size grows when using the full embedding.

A.4 Training

Causal models were trained on NVIDIA L40 chips (48 GB VRAM) through the DAIC (Delft AI Cluster). Each model was trained for 150 epochs, a value chosen empirically. Dataset V1 contained 200k datapoints and V2 contained 340k datapoints. In both cases the data was split into 80 percent training, 10 percent validation, and 10 percent testing. A high learning rate above 0.0002 often resulted in NaN values and early termination of the training run. Training the main model required approximately 12 hours.

The main model used for experimentation had the following parameters:

- Learning rate: 0.0002
- Number of flows: 4
- Number of hidden layers: 192
- Batch size: 256

Other values were kept to their defaults.

B Environment

B.1 Choice of Environment

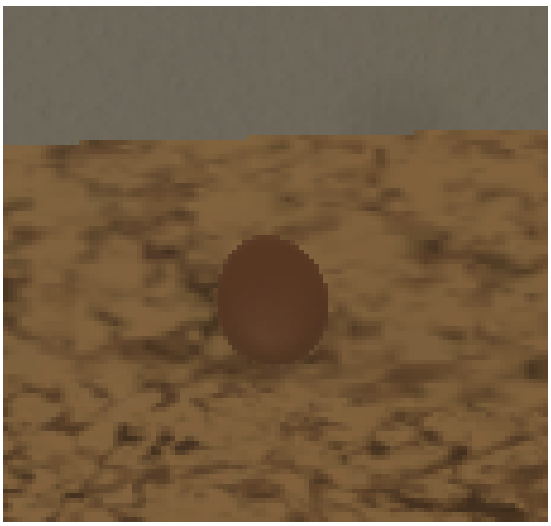
The first environment considered was Minecraft since Mindforge is built on it. However, Minecraft was not designed to be used as a research environment, which led to many issues such as crashes and slow performance. This becomes particularly problematic when running a large number of experiments. We also wanted to use a more realistic environment to mimic a real use case as much as possible. This led us to two choices: iTHOR from Ai2 and Habitat 3.0 from Meta, which are both realistic indoor simulation environments focused on embodied agents, interactive objects, and goal directed tasks in everyday scenes. Our choice leaned towards iTHOR for two reasons. First, it is the same environment used in BISCUIT, which gives us some confidence that causal modeling is feasible. Second, it supports a wide range of state changes such as slicing or cooking food, enabling us to build a larger and more structured causal skill tree.

B.2 Quality of Life Modifications

A few quality of life modifications were made to help the agent interact with the environment and to restrict which interactions it can perform. First, the agent is only allowed to pick up objects that are both visible and close enough to reach. By default the agent can grab any item even if it is far away or outside its field of view. Then, the field of view of the agent was increased from the default value of 60 to 100 to allow it to see a larger portion of the environment and improve its understanding of the scene. Next, items can only be sliced when the agent is holding a knife. By default the agent can execute the slice command without holding a knife and still perform the action.

Another issue was that the potato looked too similar to the egg, as shown in Figure 12. This led to the agent repeatedly picking up potatoes while searching for eggs. Since this issue occurred often we replaced

(a) Potato



(b) Egg

Figure 12: The egg and potato items look very similar which leads to the agent confusing the potato for an egg

all potatoes in the environment with eggs to avoid blocking our results because of a few ambiguous potatoes.

Finally, the experiments were conducted in FloorPlan27. iTHOR[44] provides a wide range of floor plans with different kitchen layouts that support similar interactions. However, most of them do not have a plate large enough to hold both a slice of bread and an egg, which is required for the final task. FloorPlan27 was one of the few that supported this, so it was selected for our experiments.

C CausalMind

C.1 Choice of Framework

When researching which framework to use to build multi agent applications, two options stood out: LangChain[2] and AutoGen[54]. Since we had never built something similar in the past, it was difficult to determine which one to choose. In the end our decision came down to two factors. LangChain appeared to be oriented more toward enterprise use, with some features locked behind a paid wall, while AutoGen[54] was designed with research use cases in mind. We also received a recommendation from our supervisors to use AutoGen[54], which reinforced our choice.

C.2 Prompts

As mentionned in Subsection 5.1, multiple prompts are used to help format the agent's belief, observation, and action. These prompts can be found below:

Return the description of the given image, here's a few example how the output should look:
Example 1: observation: "I see sliced potato" Example 2: observation: "sliced apple is cooked" Example 3: observation: "knife is in my hand" Example 4: observation: "I see a sliced tomato in cabinet" Example 5: observation: "I see nothing in fridge" Example 6: observation: "I see a pot on the stove"
Focus on object from the task: task. If there is nothing relevant to the task, just do a normal description of the image.
IMPORTANT your response should be in the following JSON format: "observation": "observation text"

Figure 13: Format prompt for observation

[2]https://www.langchain.com

ou have the following belief: belief
Format it like the following example: Example 1: belief: “the egg is in the fridge” Example 2: belief: “the knife is on the kitchen counter” Example 3: belief: “the pan is on the stove” Example 4: belief: “I have no beliefs” (if you have no beliefs)
IMPORTANT your response should be in the following JSON format, return only the most relevant belief: “belief”: “belief text”

Figure 14: Format prompt for belief

Given the following action: action, polish it to fit the format below:
Choose between the following movement actions: [“MoveAhead”, “MoveBack”, “MoveLeft”, “MoveRight”, “RotateRight”, “RotateLeft”, “NoOp”] and object actions: [“PickupObject”, “PutObject”, “ToggleObject”, “SliceObject”, “OpenObject”, “DropHandObject”] Object actions MUST be followed by an object name. Example: “PickupObject Knife” or “PutObject Pan”. The environment is simple and interactions are binary (ON/OFF, Open/Closed, Sliced/Not Sliced). Sometimes the pan is called pot. You can hold an item in your hand and perform actions on other items at the same time.
Only modify the action if it does not fit the format above. Other information that might be needed: Last action, The action performed last round: last_action Critique, A proposal of what you could do differently to achieve the task: critique Error, An error message returned by the environment. None if no error was returned: error
You also have some observation observation, belief: belief and a task: task to help you fix the action if needed.
You are controlling a robot in an environment with other agents. Your goal is to exactly follow the task, don’t try to pick other stuff up, if you can’t find what you are searching fore just explore more by using Rotation and Move. You can hold an item in your hand and perform actions on other items at the same time. IMPORTANT To perform an action on an object you in a hand reaching range, first MoveAhead till you are as close as possible. IMPORTANT do not wait for stuff to happen, do the action yourself. Also, only pickup/use items you need based on the given tasks. ONLY Respond STRICTLY in valid JSON format with ‘task’, ‘thoughts’, ‘action’, and ‘communication’ fields.
IMPORTANT Example full response: “task”: “I need to find the Egg”, “thoughts”: “I see a kitchen counter”, “action”: “RotateRight”, “communication”: “I am moving to the kitchen”

Figure 15: Format prompt for action

For modules that are in common with MindForge, the same prompts are used.

D Voyager & Mindforge

Voyager & Mindforge

Both Voyager and Mindforge have been reimplemented using Autogen, a framework made to build LLM-based multi-agents systems. Compare to their original implementation, our framework is modular and work in any environment, not just Minecraft or iTHOR. One major change was made, happening on the percepts module, where data observed was originally was given through the meta-data of the game, in our implementation we leverage the power of VLMs to observe the environment.

Percept

The percept module uses the modal capacity of the VLM powering our agent to translate the visual observation into text. Done using a simple prompt such as: “describe the given image”. This method allows our percept module to understand any environment that can be observed in an 2D image, while this allow for more versatility, it can lead to hallucinated observation.

Critic

The critic module goal is to verify if the previous taken action was successful or not. It can either reason about the given percept or get information directly from the environment to know if the action was successful. In case of failure, it will propose ideas to the agent on how the issue can be fixed, as an

example let's say we want to open the fridge. However, when trying to perform "open fridge" the action fails, the critic will then propose solutions such as trying to perform "open refrigerator" instead.

Memory

The memory is used to give extra information to the beliefs, it is divided in 3 parts: episodic memory, semantic memory, and procedural memory. The episodic memory is used to store past action that lead to failure and information about the other agents useful for perspective taking in the goal of learning from its mistakes and tracking progress of the other agent.

The semantic memory stores general information about the world and completing a task. It is used as information that could be important for future task solving.

The procedural memory stores the different skills learned by the agent. It is used as the skill library of the agent that can be used when wanting to use a previous learned skilled. For example, if our agent wants to open the fridge, it will check if this action is available in its procedural memory and if available perform the retrieved action in the environment.

Auto-curriculum

The auto-curriculum module is there to define the desires of our agent, it is given a final goal and creates multiple subgoals needed to reach it. Once one goal is reached, it will move to the next and go through each subgoal literately till solving the final goal.

D.1 Prompts

For each module prompts are used, this section lists all prompts used in the agent.

General prompts

You are controlling a robot in an environment with other agents. Every round you must select actions for it to perform in the environment. Your ultimate goal is first to pick up an egg, then cook it in a pan then serve it on a plate. ONLY Respond STRICTLY in valid JSON format with 'task', 'thoughts', 'action' and 'communication' fields. environment.environment_prompt. Example full response: "tasks": "I need to find the Egg", "thoughts": "I see a kitchen counter", "action": "RotateRight", "communication": "I am moving to the kitchen"
Each round I will give you: Task: The task you should try to achieve Last action: The action performed last round Critique: A proposal of what you could do differently to achieve the task Error: An error message returned by the environment. None if no error was returned Relevant skills: Descriptions of passed skills that succeeded and the exact command used to do them Past episodes: A summary of relevant past episodes Task belief: Beliefs about the current task Perception belief: Beliefs about the current environment around you Interaction beliefs: Beliefs based about what the other agents have told you Partner beliefs: Beliefs about your partner agents Communications from other agents: ... An image of what you can currently see infront of you.

Figure 16: System prompt used by the Agent

"task": "Current task you need to do", "thoughts": "Description of what you see and what you want to do", "action": "action to execute in the environment", "communication": "Send message to other agents"
Task: task Last action: last_action Critique: critique Error: error Relevant skills: skill_memory Past episodes: episode_summary Item held: item Task belief: task_beliefs Perception belief: perception_beliefs Interaction beliefs: interaction_beliefs Partner beliefs: partner_beliefs

Figure 17: Instruction prompt is an additional prompt to system prompt to give relevant information to the agent

Choose between the following movement actions: [“MoveAhead”, “MoveBack”, “MoveLeft”, “MoveRight”, “RotateRight”, “RotateLeft”, “NoOp”] and object actions: [“PickupObject”, “PutObject”, “ToggleObject”, “SliceObject”, “OpenObject”, “DropHandObject”] Object actions MUST be followed by an object name. Example: “PickupObject Knife” or “PutObject Pan”. The environment is simple and interactions are binary (ON/OFF, Open/Closed, Sliced/Not Sliced). Sometimes the pan is called pot. You can hold an item in your hand and perform actions on other items at the same time. IMPORTANT you are in the iTHOR environment from AI2-THOR.

Figure 18: Environment prompt used to give information about the environment

Curriculum

You are a helpful assistant that asks questions to help me decide the next immediate task to do. My ultimate goal is first to pick up an egg, then cook it in a pan, and putting it in a plate. After finishing cooking the egg, I need cut some bread and toast one slice of it and putting it in a plate. The environment is simple and interactions are binary (ON/OFF, Open/Closed, Sliced/Not Sliced). First think the simplest way to complete a task and if it fails then reflect on it and try something more complex.
Completed tasks so far: completed_tasks Failed tasks that are too hard: failed_tasks
Last communications received from other agents: communications
I will also provide you with an image of what the robot is currently looking at.
You must follow the following criteria: 1) You should ask at least 5 questions (but no more than 10 questions) to help me decide the next immediate task to do. Each question should be followed by the concept that the question is about. 2) Your question should be specific to a concept relevant to the current environment. Bad example (the question is too general): Question: What is the best action to do in the environment? Concept: unknown Bad example (cooking is still general, you should specify the type of cooking such as cooking egg): Could the egg be cooked in the pot? Concept: cooking Good example: Question: How would you slice bread? Concept: sliced bread 3) Your questions should be self-contained and not require any context. Bad example (the question requires the context of my current room): Question: What are the items that I can find in my current room? Concept: unknown Bad example (the question requires the context of the environment): Question: Do you have knife next to you? Concept: knife Bad example (the question requires the context of my nearby items): Question: Is there a stove nearby? Concept: stove Good example: Question: What would be the items that I could find in the fridge? Concept: fridge contents 4) Do not ask questions about other rooms or leaving the room (such as opening or seeing what is behind the door) since they are too hard for me to do and I must stay in the current room.
Let’s say you see a fridge in the image. You can ask questions like: Question: How could I interact with the fridge? Concept: fridge Question: What could I find inside the fridge? Concept: fridge contents
Let’s say other agents have communicated to you that they are trying to cook an egg. You can ask a question like: Question: What could I do to help the other agent cook an egg? Concept: help cook egg
Let’s say your last completed task is “pick up a knife”. You can ask a question like: Question: What are the suggested tasks that I can do after having picked up a knife? Concept: knife
Here are some more question and concept examples: Question 1: What can I do with a slice of bread?, Concept 1: slice of bread, Question 2: How could I turn on the stove?, Concept 2: stove, Question 3: How could I use the toaster to create new items and learn new skills?, Concept 3: toaster, Question 4: How to obtain an egg?, Concept 4: egg, Question 5: What are the benefits of using a pot on top of the stove?, Concept 5: stove pot, Question 6: What new items require a potato to be made?, Concept 6: potato, IMPORTANT only have questions that relate to the ultimate goal, for example if there is a potato in the image but the goal is to cook a tomato then ignore it.
IMPORTANT, You should only respond in the format as described below: RESPONSE FORMAT: “‘ reasoning: ..., questions: [Question 1: ..., Concept 1: ..., Question 2: ..., Concept 2: ..., Question 3: ..., Concept 3: ..., ..., Question n: ..., Concept n: ...,] “‘

Figure 19: Curriculum question prompt that is used to generate the different questions that will be used to generate tasks

You are a helpful assistant that answer my question about the environment.
You will answer a question and based on the context (only if available and helpful), your own knowledge of the environment. Always give the simplest answers when multiple answers are possible. 1) IMPORTANT always answer in the following JSON format: "answer": answer 2) Answer "answer: Unknown" if you don't know the answer.
Question: question Relevant Past Contexts: relevant$_{p}ast_{c}ontext$

Figure 20: Curriculum answer prompt used to answer the curriculum question prompt

You are a helpful assistant that tells me the next immediate task to do. My ultimate goal is first to pick up an egg, then cook it in a pan, and putting it in a plate. After finishing cooking the egg, I need cut some bread and toast one slice of it and putting it in a plate. The environment is simple and interactions are binary (ON/OFF, Open/Closed, Sliced/Not Sliced). First think the simplest way to complete a task and if it fails then reflect on it and try something more complex.
I will give you the following information: Question 1: ... Answer: ... Question 2: ... Answer: ... Question 3: ... Answer: ... ... Completed tasks so far: ... Failed tasks that are too hard: ... My last action: ... Previous Thoughts: ... Last action was a success: True/False Last action critique: ...
I will also provide you with an image of what the robot is currently looking at.
You must follow the following criteria: 1) You should act as a mentor and guide me to the next task based on my current learning progress. 2) Please be very specific about what items I need to collect, what I interaction I need to do with them. 3) The task should involve doing some action(s) inside the environment. Do not ask me to "describe" or "think". 4) The next task should follow a concise format, such as "Pick up [item]", "Put [item] in [item]", "Slice [item]", "Bring [item] to [item]", "Open [item]", "Move to [area]" etc. It should be a single phrase. Do not propose multiple tasks at the same time. Do not mention anything else. 5) The next task should not be too hard since I may not have the necessary resources or have learned enough skills to complete it yet. 6) The next task should be novel and interesting. I should look for new items, find unique interactions with items, and discover new things. I should not be doing the same thing over and over again. 7) I may sometimes need to repeat some tasks if I need to collect more resources to complete more difficult tasks. Only repeat tasks if necessary.
You should only respond in the format as described below: IMPORTANT always answer in the following JSON format: "reasoning": Based on the information I listed above, do reasoning about what the next task should be, "task": The next task
Here's an example response: "reasoning": "I see a piece of sliced bread and a toaster", "task": "Put bread in toaster"

Figure 21: Curriculum prompt that is used to select what is the next task to do

question$_{a}nswers$
Completed tasks so far: completed$_{t}asksFailedtasksthataretoohard$: $failed_{t}asksLastaction$: $last_{a}ctionPrevious thoughts$: $last_{t}houghtsLastactionwasasuccess$: $successCritique$: $critique$
Last communications received from other agents: communications

Figure 22: Curriculum info prompt used to add information the curriculum prompt and that takes curriculum question prompt as input

Critic

You are an assistant that assesses my progress of discovering this environment and provides useful guidance.
You are required to evaluate if I have completed the task. Exceeding the task requirements is also considered a success while failing to meet them requires you to provide critique to help me improve.
IMPORTANT: only return success as true if the task is fully completed. A task is considered fully completed only when all its requirements are met. A task is fully completed when the last action to perform it was successful and no error was returned. A task is completed when you can visually confirm in the image that the task has been accomplished.
I will give you the following information:
Task: The objective I need to accomplish. Last action: The last action I performed in the environment. Context: The context of the task. Communication: Messages sent by other agents in the environment. Error: Optionally, you may receive an error message resulting from the previously attempted action. If error is 'None', then no error happened. An image frame displaying what I can currently see.
If you receive an error you MUST set "success" to false. Critique why you think the error happened and what I should do to overcome it. If you do not receive an error but nothing happens when trying to pick up an object, it means the object is not reachable. You should try to get closer to the object before picking it up again.
IMPORTANT You should only respond in JSON format as described below: "reasoning": reasoning, "success": boolean, "critique": critique, Ensure the response can be parsed by Python 'json.loads', e.g.: no trailing commas, no single quotes, etc.
Here are some examples: INPUT:
Task: Open the fridge and look inside
Context: ...
Error: None
RESPONSE: "reasoning": "The fridge door is open therefore the agent successfully opened the fridge", "success": true, "critique": "" INPUT:
Task: Open the fridge and look inside
Context: ...
Error: ValueError('No object found with name: Refrigerator')
RESPONSE: "reasoning": "The environment did not find any objects called 'Pan'. However, there is a pan in the image so it might be called something different.", "success": false, "critique": "Try using 'Pot' instead of 'Pan'. "
"reasoning": "The environment did not find any objects called 'Refrigerator'. However, there is a fridge in the image so it might be called something different.", "success": false, "critique": "Try using 'Fridge' instead of 'Refrigerator'. "

Figure 23: Critic prompt used to judge if a task has been completed and to give relevant information in case of failure

Task: task Last action: last_action Context: context Communication : communication Error: error

Figure 24: Critic info prompt used to give pass information to the critique

Skill description

You are a helpful assistant that writes a name and description for a given action command made by another agent. You will be given the action the agent made as well as their thoughts.
1) Do not mention the actual action command 2) Try to summarize the function in no more than 3 sentences. 3) Your response should be a single line of text.
IMPORTANT You should only respond in JSON format as described below: "name": "action name", "description": "description of the action", For example, if the action command is: SliceObject Bread
Then you would write: "name": "slicing bread", "description": "The agent will slice the bread into bread slices",

Figure 25: Skill description prompt used to generate new skills that are then saved in the memory

Please generate a name and a description for the following action:
Agent thoughts: agent_thoughts Agent action: action

Figure 26: Skill description info prompt that is combined with the skill description prompt to save a new skill

Episodic Memory Summary

You are a helpful assistant tasked with summarizing past episodes and pointing out the causes of failure. Create a concise summary. Episodes:

Figure 27: Episodic memory summary prompt used to summarize past episodic memory

Belief

You are a robot exploring an environment with other agents. You just had a conversation with another agent based on a task you are trying to solve. Based on the contents of the conversation and the previous beliefs, you have to create a set of beliefs that that can help you complete the task. The new interaction beliefs should encapsulate useful information from the conversation that can help you complete the task given the conversation and your previous beliefs. Aim to create a maximum of 5 beliefs. Beliefs should be concise and relevant to the task.
Conversations: conversations Previous beliefs: previous_interaction_beliefs Current task: task
IMPORTANT you response should be in the following JSON format: "beliefs": response

Figure 28: Interaction belief prompt used to generate new interaction beliefs

You are a robot exploring an environment with other agents. You just had a conversation with another agent based on a task you are trying to solve. Based on the contents of the conversation and the previous beliefs, you have to create a set of beliefs that represent your perception of the other agent.
The new partner beliefs should contain your perception of the other agent based on the conversation and your previous beliefs. Beliefs should be informative of the other's agent state.
Previous Partner Beliefs: previous_partner_belief Conversation: convo
IMPORTANT you response should be in the following JSON format: "beliefs": response

Figure 29: Partner belief prompt used to generate updated partner beliefs

You are a robot exploring an environment with other agents. Provide a set of beliefs that encapsulate the robot's perception of the environment. You will be given an image frame of the environment, recent communications with other agents and possible execution errors. Include information about notable items you can see nearby, interactable objects, execution errors and important chat logs. Communications: communications Execution errors: error
IMPORTANT you response should be in the following JSON format: "beliefs": response

Figure 30: Perception belief prompt used to generate updated perception beliefs

Create a new context based on the task and the interaction beliefs. If necessary, update the context to include the new information. If the previous context and interaction beliefs contradict, assume the interaction beliefs are true. Keep the new context concise and informative.
Previous context: previous_context Current task: task Interaction beliefs: interaction_beliefs
IMPORTANT you response should be in the following JSON format: "beliefs": response

Figure 31: Update context belief prompt used to generate updated beliefs about context/task

E Teacher-Student Setup

If the student agent struggles with a task, meaning it does not manage to complete it in 5 steps, it will send a message to the teacher agent for help by sending relevant information (i.e. beliefs, task information, last action, last frame).

You are a teacher agent guiding a learner agent that is struggling with a task.
Your purpose: - Offer short, specific advice and ask diagnostic questions that help the learner uncover reasoning gaps. - Base your feedback on your own beliefs (about the task, environment, and interaction) and your beliefs about the learner (the partner). - Maintain a collaborative, supportive tone.
Each round, you will receive: Perception beliefs: Your beliefs about the current environment around you. Task beliefs: Your beliefs about the current task you and the learner are trying to achieve. Partner beliefs: Your beliefs about the learner agent based on your interactions. Interaction beliefs: Your beliefs about what the learner has communicated to you.
You will also receive a communication from the learner agent with: Their successful and failed tasks. Their current task. Their current beliefs about the task and environment. Based on this information, provide feedback to help the learner improve their performance.
You also have your own knowledge about the task and environment that you can use to inform your feedback.

Figure 32: System prompt used for the teacher

When receiving the message, the teacher will update its beliefs and then evaluate the situation of the student and answer it with relevant information to complete its goal throught the use of a System prompt32 and a User prompt33. From this answer, the student will also update its beliefs.

Perception beliefs: perception_beliefs
Task beliefs: task_beliefs
Partner beliefs: partner_beliefs
Interaction beliefs: interaction_beliefs
Learner update: learner_update
Knowledge: knowledge
Based on the information provided, give concise, specific feedback to help the learner improve their performance on the task. Your feedback should include advice and diagnostic questions that encourage the learner to reflect on their reasoning and actions. Maintain a collaborative and supportive tone throughout your response. Tell it what task it needs to first focus on completing. Also focus on spreading your knowledge about how to complete the task. Also if it's attempting to do something that is not required for the task, tell it to stop doing that and redirect its efforts towards the main task.
Important: Your response should be in JSON format as follows: content: "your feedback content here"

Figure 33: System prompt used for the teacher

To cook an egg in a pan, follow these steps in this exact order: 1. Find the egg 2. Pick up the egg 3. Find the pan/pot: Look for the pan/pot on the stove. 4. Put the egg in the pan/pot 5. Cook the egg: turn on the stove 6. Take the cooked egg out of the pan/pot 7. Find a plate 8. Put the cooked egg on the plate
To cook a toasted sliced bread: 1. Find the knife: Look for the knife next to the sink. 2. Pick up the knife 3. Find the bread: Locate the bread on table. 4. Slice the bread using the knife 5. Drop the knife 6. Pick up the sliced bread 7. Find the toaster: Look for the toaster located next to the fridge. 8. Put the sliced bread in the toaster 9. Toast the sliced bread: turn on the toaster 10. Take the toasted sliced bread out of the toaster 11. Find a plate 12. Put the toasted sliced bread on the plate
other information: Egg could be in the sink, next to stove, or in the frigde. Pan is already on the stove no need to move it Bread needs to be sliced before toasting Do the steps in the exact order as described above Pan is sometimes referred to as pot Plate is located on the table To turn on the stove you need to use the StoveKnob Stove knob is called StoveKnob

Figure 34: Knowledge file

To be able to answer the student, the teacher has a knowledge file that contains all the relevant information to answer the student as shown in Figure 34.

F Causal Model (Combined)

Original model	**Combined model**
Input: `[‘‘Potato is in my hand", ‘‘Slice the potato", ‘‘Move around"]`	**Input:** `Potato is in my hand. Slice the potato. Move around`
Encoding: $E_{\text{Beliefs}}, E_{\text{Desires}}, E_{\text{Intentions}}$ computed separately	**Encoding:** Single embedding E_{BDI} from full concatenated text
Final latent: $E_{\text{BDI}} = [E_{\text{Beliefs}}; E_{\text{Desires}}; E_{\text{Intentions}}]$	**Final latent:** E_{BDI}

Table 6: Comparison between the original BDI encoding pipeline and the combined model

This causal model was trained similarly to the main one with one exception. Instead of taking as input three separate embeddings for beliefs, desires, and intentions that are later concatenated, it uses a single embedding computed from the full concatenated text. This structure can be see in Table 6, where the

BDI information is encoded jointly rather than encoded independently.