City St George's, University of London

A model of functional creativity for generalisation enhancement

June 2025

Author:

Cătărău-Cotuțiu Corina

Supervisors:

Dr. Esther Mondragón
Prof. Eduardo Alonso

This dissertation is submitted for the degree of
Doctor of Philosophy
City St George's, University of London
Department of Computer Science

Contents

List of Figures

List of Tables

Contributions

1. Cătărău-Cotuţiu, C., Mondragón, E., & Alonso, E. (2023, September). AIGenC: AI generalisation via creativity. In EPIA Conference on Artificial Intelligence (pp. 38-51). Cham: Springer Nature Switzerland.

2. Catarau-Cotutiu, C., Mondragon, E., & Alonso, E. (2025). A representational framework for learning and encoding structurally enriched trajectories in complex agent environments. arXiv preprint arXiv:2503.13194. (Submitted to Neural Networks - Under Revision)

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors for their invaluable guidance, support, and encouragement throughout this research journey. Their insights and feedback were critical in shaping the direction and depth of this work.

This research was supported by The School of Science and Technology's scholarships. I am grateful for this financial support, which made it possible to carry out the work presented here.

Finally, I would like to thank my partner, my dog and my family for their unwavering support, patience, and unwavering belief in me throughout the years. This work would not have been possible without them.

Declaration

I hereby declare that I grant powers of discretion to the University Librarian to allow the thesis to be copied in whole or in part without further reference to the author. I hereby declare that, except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Corina Catarau-Cotutiu June,2025

Abstract

This thesis investigates functional creativity as a mechanism for generalisation in reinforcement learning. Its main proposal is that agents can adapt to novel tasks by recomposing structured representations of past experience, rather than relying on reactive policies or overfitting pattern interpolation. Inspired by cognitive models of creativity, we introduce a framework (AIGenC) that conceptualises generalisation as the creative reuse of abstract knowledge. Under this view, functional creativity is formalised as the agent's capacity to extract, decompose, and reassemble task-relevant substructures, such as affordances, causal effects, and object relationships, into new behavioural strategies. This theoretical proposal is instantiated in a graph-based memory system that encodes episodic interactions as heterogeneous trajectories. Called Structurally Enriched Trajectory Learning and Encoding (SETLE), it captures high-level structure in an agent's experience, linking actions, objects, and outcomes across time and embedding these trajectories using a hierarchical contrastive learning objective. Stored in long-term memory, these structured representations are later retrieved and matched by similarity, allowing agents to reuse subgraphs from prior episodes selectively. SETLE is then integrated into a reinforcement learning loop, where memory-based enrichment informs both action selection and policy optimisation. Retrieved graphs are filtered via attention or clustering and injected into the agent's working memory, effectively conditioning decisions on structurally relevant past experiences. This enables zero-shot reuse of behavioural knowledge and promotes sample-efficient learning. The complete system is evaluated in two domains: CREATE, a continuous interaction environment with tools and physical affordances, and MiniGrid, a symbolic, discrete domain with sparse rewards and task diversity. In both settings, SETLE-enhanced agents outperformed baseline models in terms of learning speed, generalisation to new goals, and trajectory optimality, validating the practical impact of structured memory and creativity-inspired reasoning. The framework is discussed in the context of both the advances and limitations of current approaches, highlighting that generalisation remains bounded by the diversity of prior experience, and unsupervised perceptual pipelines (e.g., slot attention) often fail to transfer across domains. We conclude that integrating conceptual abstraction, episodic memory, and decision-making into a unified system is a step toward more adaptive, creative, and general artificial intelligence.

1 Introduction

In recent years, artificial intelligence (AI) has achieved remarkable progress, with reinforcement learning (RL) emerging as a powerful paradigm to enable autonomous agents to learn complex behaviours through interaction. Despite its successes in domains ranging from games to robotics, standard RL methods remain limited by their reliance on narrow, task-specific policies that often fail to generalise beyond the conditions encountered during training. When exposed to novel environments, unseen task configurations, or variations in object arrangements, these agents frequently exhibit a sharp performance degradation. This fragility not only limits their real-world applicability but also incurs significant computational cost: training agents from scratch for each new task demands extensive simulation time, large datasets, and considerable energy consumption. Recent studies have highlighted that even state-of-the-art RL models require millions of environment interactions to converge on a single task, often without transferable knowledge to similar ones (Cobbe et al., 2020). This inefficiency motivates the development of methods that promote generalisation and experience reuse, ideally enabling agents to learn new tasks faster, with less data and compute. Addressing both the problem of generalisation and the efficiency of learning thus constitutes a critical step toward more robust and adaptable AI systems. This thesis investigates the problem of generalisation in reinforcement learning (RL) by proposing a computational framework inspired by creativity informed by insights from cognitive science. It is grounded in the hypothesis that generalisation, as exhibited in human cognition, relies in part on creative problem solving, specifically the ability to reorganise and recombine prior knowledge to address novel challenges flexibly. Central to this process is the idea of bootstrapping from partial experiences, leveraging fragments of past situations to infer viable actions in new contexts. By emulating this principle, the proposed framework aims to go beyond memorisation, enabling agents to construct enriched internal representations that support compositional reasoning and transfer. To explore this idea, a conceptual architecture named AI Generalisation via Creativity (AIGenC) was developed and formalised in the accompanying paper (Catarau-Cotutiu et al., 2023). AIGenC is designed to support agents in structuring, storing, retrieving, and blending abstractions from prior experience, thereby enabling adaptive behaviour across tasks that vary in both surface form and underlying structure. By embedding this aspect of creativity into RL through concept spaces and enriched memory, this

work enables agents to tackle previously unseen tasks not through manually designed knowledge or hard-coded priors, but by autonomously retrieving, organising, and reapplying structures from their own past experiences.

At the centre of the AIGenC architecture lies a hierarchical concept space,a graph structure with various levels and types of representations procured by the different components. This memory enables agents to represent not just what occurred during an episode, but also how outcomes emerged. Three core computational modules mediate an agent's interaction with this concept space:

1. Concept Processing: responsible for transforming raw perceptual input into structured, graph-based representations

2. Reflective Reasoning: enabling the retrieval of past experiences based on relevance and structural similarity

3. Blending: allowing for the adaptation and combination of retrieved knowledge to generate novel solutions.

To instantiate these components in a concrete system, the second phase of this research introduces SETLE (Structurally Enriched Trajectory Learning and Encoding), as detailed in the paper "A Representational Framework for Learning and Encoding Structurally Enriched Trajectories in Complex Agent Environments" (Catarau-Cotutiu et al., 2025). SETLE represents the realisation of the Reflective Reasoning component of the AIGenC framework by proposing Structurally Enriched Trajectories (SETs), graph-structured representations that capture not only the temporal unfolding of agent experience through states and actions, but also the higher-order relational structure and affordance-based dependencies that span across episodes. By embedding these structural abstractions within memory, SETLE enables agents to retrieve, compare, and recombine meaningful sub-trajectories, thereby supporting creative problem-solving and generalisation in novel environments.

Within this thesis, SETs are integrated directly into the reinforcement learning training loop, enabling agents to generalise by using structured knowledge acquired from previous episodes, representing the final part of this research. This integration is realised through several key design elements:

1. A dual memory system comprising Working Memory (WM) and Long-Term Memory (LTM), in which SETs are created, stored, and selectively retrieved during learning

2. An attention-based enrichment mechanism that enhances the agent's current representation with subgraphs retrieved from LTM, based on structural similarity to the current context

3. A comprehensive evaluation protocol that compares multiple learning strategies on two different environments: (1) baseline reinforcement learning, (2) SET-based enrichment used only for action selection, (3) enrichment used for both selection and optimisation, and (4) approaches augmented with adaptive modules and similarity penalties.

The contributions of this thesis are as follows:

Novel Contributions.

1. **AIGenC: A Theoretical Framework for Creativity-Driven Generalisation** (Chapter 4): A novel architecture that formalises generalisation as creative knowledge reuse, comprising three integrated components: Concept Processing, Reflective Reasoning, and Blending, operating over a hierarchical concept space. This framework draws on principles from cognitive science and representational reasoning to provide a principled approach to structured generalisation in RL.

2. **Structurally Enriched Trajectories (SETs)** (Chapter 5): A novel graph-based representation that extends the standard RL trajectory ontology to include objects, interactions, affordances, and hierarchical dependencies. SETs capture not only temporal sequences but also the relational and functional structure of agent experience. And comes as an instantiation of several components of the AIGenC theoretical template.

3. **SETLE: A Hierarchical Graph Encoder for Trajectory Representations** (Section 5.3): A novel architecture for encoding SETs using heterogeneous graph neural networks with meta-path-aware attention. This encoder produces embeddings that support similarity-based retrieval and memory-guided reasoning.

4. **Memory-Augmented RL Integration** (Chapter 6): A novel method for integrating structured episodic memory into the RL training loop, including attention-weighted retrieval, graph-based state enrichment, and stabilisation mechanisms (adapter layers, matching penalties, soft

updates) that enable effective knowledge reuse during both action selection and policy optimisation.

5. **Dual Attention Mechanism** (Section 6.2.1): A novel combination of Mackintosh-based associative attention for memory consolidation and Transformer-style learned attention for retrieval, enabling complementary filtering of concepts for long-term storage and selective injection during inference.

Adopted Methods and Components. The framework builds upon several existing methods, which are evaluated and integrated rather than developed as part of this thesis:

- **Object Discovery**: We adopt the Segment Anything Model (Kirillov et al., 2023) for zero-shot object segmentation after evaluating alternatives including Slot Attention and DTI-Sprites (Section 5.2.2). SAM is used as a pre-trained module without modification.
- **Interaction Encoding**: We employ a ConvLSTM architecture with triplet loss for learning interaction representations (Section 5.2.2). While the training procedure is tailored to our environments, the underlying architecture follows established designs.
- **Base RL Algorithm**: The reinforcement learning substrate uses Double DQN (Van Hasselt et al., 2016) as the policy learning algorithm (Section 6.1), chosen for its stability and compatibility with our hybrid action space.
- **Graph Neural Network Foundations**: The SET encoder builds on Heterogeneous Co-contrastive Learning (Wang et al., 2021), adapting its contrastive objective for episodic trajectory data (Section 5.3.2).
- **Evaluation Environments**: Experiments are conducted in CREATE (Jain et al., 2020) and MiniGrid (Chevalier-Boisvert et al., 2023), both existing benchmarks selected for their support of structured variation and tool-based reasoning.

Empirical Contributions.

- Systematic evaluation across two domains (CREATE and MiniGrid) demonstrating that SETLE-enhanced agents exhibit improved generalisation, sample efficiency, and calibrated value estimation compared to baseline models.
- Ablation studies validating the contribution of individual architectural components (adapter, penalty, soft updates) to learning stability and performance.
- Cross-domain analysis showing that trajectory-level embeddings capture functional similarity across environments with different visual and symbolic characteristics.

In summary, this thesis outlines the AIGenC framework, its underlying architecture, and the outcomes it offers to advance the frontiers of AI and reinforcement learning.

2 Motivation

A persistent limitation in artificial intelligence lies in how knowledge is represented and reused. Many systems treat each task as an isolated problem, learning solutions from scratch without using structural regularities across experiences. This limitation is particularly pronounced in reinforcement learning (RL), where agents tend to specialise narrowly on the training distribution, lacking mechanisms for abstraction, transfer, or creative recombination. As a result, learning may be impaired in unfamiliar contexts, and agents often fail to apply prior knowledge meaningfully. Common symptoms include the development of overly optimistic policies due to recursive overestimation (Van Hasselt et al., 2016), as well as overfitting to task-specific cues that do not generalise beyond the original environment (Cobbe et al., 2020). These challenges highlight the need for structural representations and memory mechanisms that support flexible knowledge reuse across tasks

Interestingly, these shortcomings do not necessarily stem from catastrophic forgetting (Kirkpatrick et al., 2017) or memory decay, but often reflect a deeper limitation in the design of learning algorithms. Most machine learning systems, including many reinforcement learning agents, are built on the simplifying assumption that training and deployment data are drawn

from the same underlying distribution (i.i.d.). This assumption rarely holds in practice, where agents frequently encounter out-of-distribution (OOD) states due to variations in tasks, environmental shifts, or novel object configurations. Since these algorithms are not explicitly equipped to reason about domain shift, they often fail to adapt, resulting in brittle behaviour and sharp drops in performance when deployed beyond the training distribution (Zhou et al., 2022). Overcoming this limitation requires approaches beyond memorising patterns, toward mechanisms that support structural abstraction and experience reuse.

The situation intensifies when we consider AI's impressive pattern-matching capabilities, particularly in deep learning, where models have demonstrated superhuman proficiency in tasks within their training domain (Bengio et al., 2021; Lyre, 2020). However, a contradiction arises when these models struggle to apply the knowledge they have acquired to contexts that are significantly different from their prior experiences. This conundrum raises a fundamental question: Why do AI systems exhibit superhuman competence in one context while experiencing striking failures when confronted with a similar yet distinct challenge? To address this puzzle, it is essential to delve into the methods employed by deep neural networks. These networks occasionally resort to shortcut strategies to achieve specific goals (Geirhos et al., 2020). Shortcuts involve identifying spurious correlations within the data that enable networks to complete tasks without understanding the underlying structure of the input. They often encompass data interpolation, a process of making ad-hoc adjustments that, while reducing errors, lack a genuine understanding of the task. Interpolating agents, as a consequence, typically struggle to generalise beyond their training task unless supplemented with prior knowledge that extends their understanding beyond their training data. As a result, models may appear competent while relying on surface-level cues that fail to perform effectively under even minor distributional shifts.

RL exemplifies the challenge of poor generalisation in artificial agents. Despite their capacity to learn complex behaviours and achieve superhuman performance in certain domains, for instance in game-playing AI (Silver et al., 2017), robotics (Levine et al., 2018), and autonomous systems (Kiran et al., 2021), RL agents frequently struggle to transfer acquired skills across tasks, even when task structures are closely related. A common limitation arises from the typical training and evaluation paradigm, where agents are exposed to a single environment or a narrowly defined task. This restricted exposure often results in the memorisation of high-reward action sequences,

rather than the development of strategies that generalise beyond the training distribution. As a consequence, agents can overfit superficial task features, bypassing the acquisition of broadly applicable competencies (Malik et al., 2021). In the absence of structured internal representations, such as concepts that capture the functional or relational properties of objects and actions, agents tend to exploit transient regularities rather than acquire abstractions that support transfer.

The difficulty of generalisation becomes more apparent when agents are evaluated across tasks or environments that introduce structured variation. Although most RL benchmarks are designed to assess performance within a single task setting, only a limited number of testbeds, such as MiniGrid (Chevalier-Boisvert et al., 2023), CREATE, and Meta-World (Yu et al., 2020), offer systematic variations that facilitate the evaluation of generalisation. MiniGrid introduces symbolic and spatial variability through layout changes, whereas CREATE imposes tool-based variation by altering the set of actions available to the agent across episodes. This tool-centric dynamic forces agents to abandon fixed action policies and adapt to evolving task affordances, thus providing a more stringent test of generalisation.

On the topic of generalisation, an important part of the narrative is played by the recent advances in large language models (LLMs), which, in many ways, shifted the narrative around generalisation in AI. These systems show impressive performance on a wide range of benchmarks, often appearing to generalise across modalities and tasks (Achiam et al., 2023; Touvron et al., 2023; Bommasani et al., 2021). However, their success is largely attributable to extensive pre-training in broad and diverse datasets, rather than to an underlying capacity for structured, causal, or relational inference (Bender and Koller, 2020; Marcus, 2020). Although LLMs can exhibit zero-shot generalisation to novel inputs, they often fail to apply familiar concepts robustly in new contexts, a limitation observed even in foundational work on context-sensitive word representations (Dasgupta et al., 2020). This context dependence suggests that, in the absence of compositional abstractions, such systems rely heavily on the statistical breadth of their training data to approximate generalisation. As a result, achieving systematic reuse of knowledge across tasks often requires combinatorially large-scale data augmentation, incurring substantial computational costs.

This distinction is critical: although LLMs simulate generalisation through coverage and scale, they do not yet offer mechanisms for concept-level reasoning or grounded abstraction (Lake and Baroni, 2023; Mitchell and Krakauer,

2023). Without the ability to represent and manipulate meaningful relational concepts, AI systems remain limited in their ability to transfer information between structurally different contexts (Cătărău-Cotuţiu et al., 2023). Consequently, there is a continued need for architectures that support explicit compositional representations, particularly in domains characterised by high environmental variability, embodied interaction, or dynamic task structure, where exhaustively encoding all relevant conditions during training is infeasible (Garnelo and Shanahan, 2019; Cătărău-Cotuţiu et al., 2023).

Our hypothesis suggests that Creative Problem Solving (CPS) scenarios could enable AI systems to learn explicit, versatile concepts that can be adapted to novel situations, a step forward to solving the problem of generalisation in AI (Cătărău-Cotuţiu et al., 2023). Generalisation is a natural process by which individuals respond in the same way to different but similar information. Humans and other animals can map previous experiences to fresh situations, thus adjusting to their environments efficiently, which machines cannot (Momennejad, 2020; Murphy and Medin, 1985; Wu et al., 2018). Successful adaptation often requires more than simply reusing information; it calls for innovative solutions, such as crafting a tool from unconventional materials (Cătărău-Cotuţiu et al., 2023). This capacity is often referred to as transfer of learning, where prior knowledge can be efficiently applied to novel contexts. One proposed mechanism underlying transfer learning in humans is compositional generalisation (or compositional transfer), the ability to recompose learnt concepts into novel ones systematically.

In the context of AI, compositional generalisation has been studied in natural language (Johnson et al., 2017; Lake and Baroni, 2018; Bahdanau, 2020; Gordon et al., 2019; Keysers et al., 2019; Lin et al., 2023) but has only recently been approached by a few papers in RL (Kirk et al., 2023; Zhao et al., 2022). Hupkes et al. (2020) introduced a categorisation framework for compositional generalisation, developed for language but with parallels to RL. This framework identifies five distinct forms of compositional generalisation. Compositional generalisation involves aspects like *systematicity*, where AI systematically recombines known components and rules to extrapolate knowledge, *productivity*, which entails extending predictions beyond the length of training data to foster AI's capacity for innovation and adaptation, *substitutivity*, reflecting the ability to generalise by substituting components with synonyms or equivalents, *localism*, which examines whether model composition operations occur within specific data segments or apply globally across the dataset, and *overgeneralisation*, assessing AI models' adaptability

and resilience in the face of exceptions and outliers.

Achieving compositional generalisation ultimately hinges on how knowledge is internally structured. Systematic recombination, abstraction, and substitution of learned components require representations that encode not just features, but relations and dependencies. Although deep learning architectures, especially in reinforcement learning, have made substantial progress, they typically operate over dense, entangled representations that obscure conceptual boundaries and inhibit reuse. This structural opacity undermines the agent's ability to generalise beyond the training distribution, despite surface-level performance improvements.

Indeed, as deep networks increasingly match or surpass human benchmarks in perception and language tasks (e.g., Ramesh et al. (2021); Bayer et al. (2023); Yang et al. (2021)), their successes have fuelled the assumption that such models possess abstract, human-like conceptual understanding. Yet this impression is misleading. This illusion is nurtured by the assumption that human-like performance implies human-like strategies, that is, by the belief that behaving in a manner akin to humans presupposes similar underlying cognitive traits. This fallacy is rooted in the prevalent notion of computationalism and its disregard for the physical substrate of cognition (Catarau-Cotutiu et al., 2023; Searle, 1980). The challenge here is that the representations often need more structure to handle the nuanced forms of compositional generalisation mentioned earlier effectively. Deep RL architectures, while powerful, tend to simplify the environment's complexity by processing raw input. However, this reduction often results in representations that are not well-suited to systematically recombine known components (systematicity), extend predictions beyond training data (productivity), handle component substitution (substitutivity), discern the scope of model composition (localism), or exhibit adaptability to exceptions (overgeneralisation).

To address these issues, we draw inspiration from CPS theories and learning theories to investigate the requirements and corresponding modelling components that would enable a naïve agent, which learns by trial and error while interacting with an environment —i.e., an RL agent —to act efficiently across domains in multiple tasks. This approach differs from standard deep-RL architectures that merely reduce the complexity of the environment by condensing raw input. A strategy that does not capture the higher-order cognition needed to enable agents to acquire information that makes sense of the structure of the world. Stimulus generalisation in the natural domain is driven by the extant common elements present in the input. By virtue

of these commonalities, humans detect similarities between different sources of information and are capable of transferring sensory knowledge from one setting to another. Yet, stimulus generalisation relies exclusively on sensory attributes that are often embedded within numerous irrelevant cues, both similar and dissimilar, which may lead to inadequate or dysfunctional use of information. Furthermore, stimulus generalisation is only a fraction of human generalisation capabilities. For example, associations can also mediate generalisation to dissimilar cues, allowing the transfer of learning across different sensory dimensions (Eilifsen and Arntzen, 2021). In addition, the role of commonalities can be extended to different levels of information (Mondragón and Murphy, 2010). Thus, extracting suitable patterns and relationships capable of bearing forward resemblance across situations beyond simple sensory input is fundamentally adaptive as it enables humans to build up compressed information; that is, to progressively reduce otherwise overwhelming stimulation that would saturate their perception, and create hierarchies upon which knowledge can be transferred. Generalisation and abstraction compress our world to manageable proportions, make it familiar and permit transferring knowledge to analogous scenarios (James, 1890).

In addition, the agent's learning experience upon which concepts are to be acquired is usually limited to single scenarios with features predominantly extracted from a single sensory channel. Consequently, generalisation of performance can only be achieved based on interpolation within the sensory domain of the training set. Endowing artificial agents with core concepts, abstract representations that capture meaning beyond the training context (Chollet, 2019; Mitchell, 2021), that is, with a comparable capability of building and manipulating abstract information at different hierarchical levels, is essential to achieve human-like performance. In the absence of such concepts, current AI models produce non-optimal responses in unknown environments and rely on large datasets and large numbers of parameters to produce seemingly superhuman results (Reed et al., 2022), as is the case of transformers in the popular GPT-3, whose impressive performance is due to it using 175 billion parameters and 45TB of text data (Rae et al., 2021).

Following (Olteţeanu, 2020), we define different types of data collectively as concepts, a consolidated unit that enables us to look for similarities and match information parsimoniously. Accordingly, concepts are objects, events and properties, all of which constitute the building blocks of meaning and so-called common sense that allows humans to rapidly understand and interact with novel scenes (Shanahan et al., 2020; Shanahan and Mitchell, 2022).

This is in contrast to current approaches of AI, where concepts are learnt as undifferentiated feature vectors extracted from raw data, thus lacking the abstraction needed to structure our thinking at a higher cognitive level.

The ability to apply a learned concept in closely matching scenarios is referred to as near transfer. It involves allowing systems to reuse existing knowledge across highly similar contexts, but it offers limited adaptability when task structure or semantics deviate from prior experience. Achieving more human-like behaviour requires going beyond near transfer, toward the flexible reconstruction and modification of concepts to suit novel goals. This process, known as displacement or creative transfer (Haskell, 2000), involves recombining prior knowledge in new ways or forming entirely new abstractions to handle unfamiliar challenges. In such cases, simple generalisation or approximation based on past data is insufficient; instead, agents must engage in creative problem-solving, drawing from structural cues and conceptual dependencies to generate new, context-sensitive strategies. The more an agent can creatively reconfigure and repurpose previously acquired concepts, rather than simply reuse them intact, the closer its behaviour aligns with flexible, human-like generalisation. This motivates the need for architectural mechanisms that support not only transfer but conceptual transformation, particularly in reinforcement learning settings where environments may differ in goals, tools, or underlying relational structure. CPS shifts the focus of AI evaluation away from benchmark-specific performance and toward agents' capacity to generate novel solutions when faced with unfamiliar conditions. While standard benchmarks such as ImageNet (Deng et al., 2009), GLUE (Wang et al., 2018), and SuperGLUE (Wang et al., 2019) have driven progress in task-specific optimisation, they are increasingly limited in their ability to capture broader generalisation. As Bowman and Dahl (2021) observe, many popular NLP benchmarks are now saturated, state-of-the-art models have surpassed human-level performance on tasks like GLUE and SQuAD, yet continue to fail on simple test cases that probe the underlying reasoning skills those benchmarks were designed to assess. These models often exploit superficial patterns or annotation artefacts in the data, giving the illusion of competence while lacking a robust, transferable understanding.

Moreover, because many benchmarks are constructed under constrained or artificial conditions, they do not reflect the variability and ambiguity of real-world environments. As Raji et al. (2021) and others have argued, benchmarks may obscure progress by promoting metric-driven competition rather than structural improvement. This raises concerns not just about validity,

but also about the misalignment between benchmark success and meaningful generalisation. In contrast, CPS emphasises flexible knowledge reuse and concept recombination, capabilities that become increasingly important when the similarity between tasks is low or ambiguous. As Malik et al. (2021) notes, while reinforcement learning can exploit task similarity to aid generalisation, when structural overlap is limited, creative solutions may be essential. An agent displaying creative thinking should be able to generate novel solutions from existing knowledge (Frith et al., 2021).

A recurrent misconception is that the implicit relations learned by neural networks and captured in the weight matrices extracted from single sensory information are sufficient to learn concepts with meaningful knowledge (Doumas et al., 2008). Different studies challenge this assumption, showing that deep networks frequently rely on shallow heuristics. In vision, convolutional networks have been shown to prioritise local texture cues over global shape information, despite human visual perception being primarily shape-based (Geirhos et al., 2020). This reliance on low-level correlations suggests that such models often succeed by exploiting surface regularities rather than extracting high-level abstractions. Similarly, in natural language processing, large language models tend to encode token co-occurrence statistics without grounding their representations in semantic structure. Bender and Koller (2020) argue that models like BERT (Jawahar et al., 2019) and GPT-2 lack grounding in the external world and instead build internal representations based solely on distributional patterns in text corpora. As a result, they may produce outputs that are syntactically fluent yet semantically incoherent or insensitive to pragmatic context.

In contrast, recent work by Piantadosi and Hill (2022) proposes a more nuanced account grounded in conceptual role semantics. They argue that meaning may emerge not through reference to the external world but through the structured interrelations among internal representations. According to this perspective, what gives a concept meaning is not its external referent, but how it functions within a broader network of dependencies and inferences. Evidence supports this idea: large language models increasingly approximate human-like conceptual geometry and exhibit internal structure consistent with human judgments. However, these representations are still incomplete. As Piantadosi and Hill (2022) acknowledge, LLMs largely lack grounding in sensorimotor experience, causal reasoning, and goal-directed interaction.

This motivates the need for hybrid representational architectures that combine both implicit vector-based structure (capturing internal role seman-

tics) and explicit, externalised relational structure (capturing environmental dynamics). To achieve flexible cross-domain generalisation, agents must integrate richer forms of knowledge, including affordances, temporal dependencies, and causal relationships—features that describe how entities interact over time and how actions affect the world. These are not easily recoverable from static sensory correlations alone. Recent surveys underscore the limitations of vision-only systems and advocate for the integration of non-visual contextual cues in achieving robust generalisation (Zhai, 2023). By encoding both internal and external dimensions of conceptual structure, we aim to equip agents with something akin to common sense Shanahan et al. (2020): the ability to infer latent properties such as intentions, goals, and functional constraints that are not directly observable. In our approach, this form of grounded knowledge is not pre-specified, but acquired through structured interaction with the environment—simulated using reinforcement learning (Sutton et al., 1998).

In traditional RL scenarios, agents disregard by design most of the information provided by the environment, learning simple policies (sequences of actions that accumulate rewards) linked to global states. Although, in theory, states in RL can represent any type of information, in practice, RL implementations work primarily on states in which only sensory data is encoded (Badia et al., 2020; Vinyals et al., 2019). In so doing, states are monolithic entities that do not allow for the construction of concepts and their transfer. We argue that to achieve more robust concept representations, an agent cannot rely on raw sensory information only, but rather it must learn by trial and error the functional and contextual information that accompanies it. Thus, we need to exploit these enhanced representations as a first step towards achieving a creative agent that can adapt to new tasks in RL scenarios. Suppose we wish to provide an agent with the ability to adapt knowledge creatively. In that case, we need to expand the RL framework with a mechanism that enlarges the set of actions and makes creative problem-solving viable. There have been several approaches to enriching RL with CPS capabilities. Recently, Gizzi et al. (2020) developed a formal model that encapsulates states, actions, and policies under a single concept space on which they defined different creativity functions (e.g., combination, transformation). Similarly, Colin et al. (2016) drew a parallel between Wiggins' Creative System Framework (2006) and hierarchical reinforcement learning. The Creative System Framework (Wiggins, 2006) interprets creativity as a concept search within a problem space and a meta-search of problem spaces.

Colin et al. (2016) made these two levels of search equivalent to searching the object space in the environment and searching the policies in the RL space. Despite the theoretical relevance of these approaches, they are high-level formal proposals which do not offer specifications on how to integrate CPS and RL. Our work aims at formulating a computational framework for machine learning implementations that incorporate CPS in a deep RL environment.

To address these shortcomings in generalisation, this thesis proposes a computational framework inspired by creativity, drawing from cognitive science theories of concept formation and transfer. The goal is to explore how agents can go beyond rote generalisation by creatively reorganising and reapplying prior knowledge in unfamiliar contexts. This perspective requires a data representation structure that supports abstraction, compositionality, and relational reasoning. Crucially, the thesis posits that knowledge must be structured in a discoverable and reusable way by the agent itself, without relying on human-curated symbolic scaffolding. This emphasis on self-organised, structured knowledge serves as the foundation for the next chapter, where we present the theoretical underpinnings of creativity, reinforcement learning, and graph-based representations that inform the proposed approach.

3 Background

In this section, we cover the theoretical background underpinning our proposed model. Our examination commences with an investigation into the types of creativity and creativity theories, specifically focusing on functional creativity. We examine the intricate landscape of concept spaces and their profound significance in cultivating creative cognitive processes. Concepts, recognized as fundamental units within the creative paradigm, are discerned not as isolated entities but as components intricately interlinked within a dynamic web of associations. Within the landscape of functional creativity, the critical notion of affordances emerges. Affordances serve as pivotal bridges between abstract conceptualizations and actionable, tangible outcomes, significantly shaping how individuals engage with their surroundings and manipulate their environment to manifest creative ideas.

Further in this theoretical exploration, we examine the principles of reinforcement learning (RL), a framework foundational to intelligent decision-making. RL describes how agents adapt to their environment through iterative interaction and feedback, learning to select actions that maximise

long-term outcomes. Crucially, it also raises questions about how knowledge is represented and reused, particularly in settings that demand adaptation to unfamiliar situations, a core concern for creative problem solving.

To address these challenges, we turn to representation learning, which seeks to construct internal models that capture the salient structure of the environment in a form amenable to generalisation. Within this context, memory becomes a critical component: agents must not only store past experiences but access and recombine them to inform future decisions. Structured memory architectures, particularly those that encode relational information, provide a foundation for this kind of reasoning.

We therefore introduce the notion of graphs and graph neural networks (GNNs), which offer a versatile and expressive framework for representing entities, their interactions, and temporal dynamics. In domains where creative adaptation is essential, graphs enable the encoding of compositional, relational, and affordance-based structures, qualities necessary for agents to move beyond pattern recognition and toward flexible conceptual reuse.

In summary, this section serves as the theoretical underpinning for our subsequent discussion of the proposed model. It elucidates the foundational theoretical elements that inform the model's design and functionality, laying an academic foundation upon which the creative capacities of our model will be elucidated and explored.

3.1 Creativity Theories

To achieve our research objectives, we conducted an extensive literature review on computational creativity and machine learning. Our exploration began with examining existing creativity taxonomies and theories, followed by an in-depth analysis of AI models that can assist us in achieving CPS.

Creativity is a complex concept that has been studied from various perspectives, including psychology, philosophy, cognitive science, and computer science. As a result, different definitions and requirements must be considered when developing a creative system. In computational creativity (CC), numerous definitions have been proposed over time, with some emphasizing the importance of value, novelty, and surprise.

3.1.1 The four P's Taxonomy

First, we will utilise Rhodes' four P's taxonomy of creativity to categorise our research properly. Even though our aim is not to replicate creativity in humans, we still need to scope out the creativity research pertinent to our goal of improving generalisation in RL agents by taking inspiration from CPS. Rhodes' taxonomy divides creativity research into four areas: person, process, press, and product. The "person" component focuses on the individual traits necessary for creativity, such as mental adaptability and the ability to redefine concepts. The "process" component involves the steps taken to produce a creative artifact. The "press" component refers to the influence of society and external factors on the creative process. Finally, the "product" component examines the characteristics necessary for a creation to be considered innovative.

To create a model that takes inspiration from the creative problem-solving abilities of a person, we need to evaluate how knowledge is structured in the minds of creative individuals. Understanding the process of acquiring concepts is crucial in developing a knowledge space which enables creativity. Although Rhodes' taxonomy separates "person" and "process", there is overlap in the research fields, as some traits of creative individuals are also necessary for the creative process.

The taxonomy also presents the social perspective of creativity - or the *press* element. Environmental factors at all stages of life form a *psychological press* (Rhodes, 1961) that may be either constructive or destructive to creativity. Humans usually belong to different environments, and the press component refers to how a person processes their environment and how the environment influences creativity, such that some products of creation are just responses to the social needs of time or climate. "The existing and overwhelming influence of causes for invention is proved by the frequency of duplicate invention, where the same idea is hatched by different minds independently at about the same time."

The final P refers to product, Rhodes (1961) specifies that an idea becomes a product the moment it is embodied in something tangible. Usually, product-based research focuses on the evaluation component of the artefact produced by a creative system; this can be anything from art pieces, mathematical theorems or even adaptability to the environment.

This taxonomy is particularly useful in framing our research goals, as it highlights that creativity-related constructs—such as process, product, and

person—are interdependent rather than mutually exclusive. Our primary focus lies in creative problem-solving, which emphasises the process of generating novel solutions by reorganising existing knowledge. In this thesis, we aim to emulate the dynamics of this process in artificial agents, specifically investigating how structural representations and memory mechanisms can support the emergence of creative behaviour. While our model operationalises creativity through process-based mechanisms (e.g. graph recombination and enrichment), we also recognise that these processes are shaped by aspects of mental representation (person) and result in measurable performance outcomes (product). The following section outlines key theoretical perspectives on the characteristics of creative output in problem-solving contexts, which will later serve as a comparative framework for analysing the behaviour of our system.

3.1.2 Product novelty taxonomies

Diverse interpretations of creativity exist, each accentuating aspects of the four P's, resulting in a plethora of assessment methodologies. In our endeavour, the central objective revolves around developing a model that takes inspiration from the process of generating creative output, necessitating an emphasis on process-oriented evaluation. This premise holds significance in light of the prevailing trend in creativity assessment, which frequently gravitates toward methods rooted in divergent thinking. Had our study primarily focused on the creative product, we might have readily adopted established theoretical frameworks such as the "Creative Product Semantic Scale" (O'Quin and Besemer, 1989) or the "Consensual Assessment Technique" (Baer, 2020). These frameworks typically involve external human evaluators who assess creative outputs. However, such conventional evaluations risk sidelining the creative potential of agents that may have yet to earn the favour of a specific audience.

Our process-oriented approach seeks to mitigate this inherent bias and instead focuses on the personal experience of creativity. This perspective, akin to the principles of compositional generalisation in machine learning, underscores the role of creative processes in reshaping and disseminating knowledge in novel and innovative ways. Just as compositional generalisation in machine learning emphasises the systematic recombination of known components and rules to extrapolate knowledge into novel ones, our creative process-centric evaluation seeks to unravel the dynamic processes through

which creative ideas and insights are generated, shaped, and communicated. This alignment between compositional generalisation and creativity underscores the synergy between these domains. It emphasises the potential for using insights from one field to enrich our understanding and modelling of the other.

As the research falls under the investigation of "personal creativity", even intra-personal creativity has different levels of magnitude. Therefore, we need to identify where on the creativity spectrum everyday creative problem-solving lies. Boden (1996) mentions two types of creativity: P-creativity (psychological), which refers to an idea being considered innovative with respect to the creator's mind, and H-creativity (historical), which includes being P-creative but also new to the whole of humanity. The second type is usually glorified and associated with brilliance and genius. However, our main aim in developing a model for creative behaviour is to enhance the ability of artificial agents to generalise, which hinges on their sense-making skills. So, we focus on P-creativity because the underlying creative process remains the same whether an idea is groundbreaking for everyone or just new to the agent creating it. Differences may arise from individual viewpoints or external influences (referred to as "person" or "press" factors), but these aspects are beyond our research scope. Our core goal is to understand and model the common processes involved in creative problem-solving, ultimately improving the generalisation capabilities of AI agents.

The realm of P-creativity has an extensive scope, encompassing a wide range of activities, from the simple act of using a pen to stir a cup to the more complex task of devising a novel method for adding numbers up to 100. To narrow our focus and effectively evaluate our agent's creative capacity, we must delineate the level of novelty we aim to attain. It's essential to recognise that creativity isn't a binary concept but rather a matter of degree.

Kaufman and Beghetto (2009)'s insightful perspective on creativity as a gradient phenomenon offers valuable guidance. He suggests that creators progress along a continuum, transitioning from one level of creativity to another. In this context, Kaufman's 4C model of creativity provides us with a framework to pinpoint the specific level of novelty and specificity at which we intend to assess our outputs.

Our interest primarily lies in the domain of creative problem-solving (CPS) and creativity's role in generating personally meaningful insights. CPS, as a form of everyday creativity, encompasses a wide range of cognitive activities, such as students articulating newly grasped concepts or individ-

uals reinterpreting ideas through novel metaphors. This level of creativity, often called mini-c creativity, is deeply embedded in the learning process. It does not require dramatic innovation but involves internal restructuring and reinterpretation of existing knowledge in a meaningful way to the learner.

In our context, this restructuring may not immediately appear "creative" in the colloquial sense. Yet the ability of an agent to adapt flexibly, reorganise prior experiences, and apply them to new situations lies at the core of creative cognition. The transition from mini-c to little-c creativity, marked by observable novelty and problem-solving efficacy, serves as a developmental trajectory in humans and artificial agents. This progression aligns with our broader objective of designing agents capable of robust generalisation and knowledge transfer.

Our approach embraces the idea that creativity is not solely about generating entirely new knowledge, but about reconfiguring existing structures to address novel challenges. In this thesis, we operationalise these processes through mechanisms that support structural abstraction, relational memory, and representational reuse, laying the groundwork for machine creativity grounded in adaptive problem-solving.

3.1.3 Process based creativity theories

Once we decided on our agent's desired level of creativity, we proceeded to review existing theories on the creative process. This included examining both computational creativity and psychology perspectives to understand the mechanism behind creativity better. By understanding this mechanism, we could select the appropriate data structures and components to incorporate into our model. Our goal was to create a more robust and transferable way for artificial agents to learn and solve problems by drawing inspiration from cognitive models of creativity.

3.1.4 Concepts and Conceptual space

Understanding theories of creativity requires first acknowledging the central structure within which creative processes are thought to occur: the conceptual space. This space serves as the cognitive substrate where concepts are represented, manipulated, and recombined, making it fundamental to models of creativity and generalisation. To determine which characteristics a computational model must incorporate, we must examine the theoretical

foundations of conceptual spaces, particularly the nature of the concepts that populate them. This involves surveying various cognitive and philosophical accounts that define what constitutes a concept and how these units contribute to structured thought and creative reasoning.

In the 1970s, Rosch's influential work on colour categorisation (Rosch, 1975) challenged the classical view of categories as being defined by necessary and sufficient conditions. Her findings demonstrated that category membership is often graded, with some instances perceived as more prototypical than others. This theory of graded category structure posits that concepts emerge not from formal logical criteria but from statistical regularities in perceptual and functional experience. In many ways, this view parallels how categories are learned in modern machine learning systems. For instance, unsupervised learning algorithms such as clustering identify groupings of data points based on feature similarity, often forming dense regions in representation space that reflect prototypical examples, analogous to human prototypes. These learned structures exhibit many of the properties described by Rosch, including graded membership and flexible boundaries, offering a computational realisation of cognitive theories of categorisation. A range of theoretical and computational approaches have since been proposed to explain the graded structure of categories, each offering distinct insights into how concepts are represented, organised, and flexibly applied—moving beyond the fixed, rule-based assumptions of classical categorisation theories (Gabora et al., 2008).

1. Prototypes (Rosch, 1975, 1973, 2024): A concept is defined by a collection of characteristic features, rather than specific defining ones. These features are weighted to create a prototype, which serves as a reference point for categorising new items. To be classified under a certain concept, an item must be similar enough to the prototype. The prototype consists of features, each with weight or applicability value. Unlike traditional category definitions, prototypes do not require uniform attributes or predetermined limits.

2. Exemplar Theories (e.g. (Medin et al., 1984; Nosofsky, 1988; Heit, 1996)): A concept is represented by a set of instances stored in memory. Each exemplar has uniquely weighted features, and a new item is categorised based on its similarity to the most salient exemplars.

3. Concepts as Theories: Concepts take the form of 'mini-theories' (e.g.

(Murphy and Medin, 1985)) or schemata (Rumelhart and Norman, 1983), in which the causal relationships amongst features or properties are identified. As such concepts do not stand independently but belong to larger systems and are viewed as interconnected within theories or networks, with context-dependent effects readily incorporated.

4. The Geometrical Approach: Gärdenfors (Gardenfors, 2004) introduced a geometrical approach that considered dimensions (e.g., color, pitch, temperature, weight) and their relationships. This approach distinguished between integral and non-integral dimensions and defined properties as convex regions in domains. Concepts were seen as sets of convex regions across multiple domains, with context-dependent combinations modelled by replacing regions from one concept with those from another.

We draw inspiration from the geometrical approach of Gardenfors (2004) and the concept of a conceptual space built from geometrical representations. This space is based on several quality dimensions, as mentioned earlier. Gardenfors's geometric representations serve as an intermediary between symbolic and sub-symbolic representations. In this conceptual space, quality dimensions are based on perception or sub-symbolic processing, while regions in the space are abstract symbols known as concepts. The distance between entities in this space represents their similarity, and the quality dimensions, as specified by Gardenfors (Gardenfors, 2004), represent various "qualities" of objects. Specifically, they are used to assigning properties to objects and specifying relations among them.

To expand a conceptual space, one can begin with a set of dimensions that are augmented through the learning process. These dimensions can include base concepts, perceptible through sensory inputs, as well as functional representations like object functions and social roles, which are defined by their actions. The development of complex concepts requires the construction of these higher-level ideas from the base concepts, requiring information to be encoded at multiple levels. Clark et al. (2021) delves into this topic by exploring formalising concepts. Bechberger and Kühnberger (2017) introduce several operations in a formalisation of the concept space that can be used to create new concepts from existing ones and describe relations between concepts. An important property is distance, which is a measure of similarity for concepts, as it allows for a natural representation of classes of entities.

One of the main hurdles with Gärdenfors' view when adapting it to an artificial setup is the origin of the features or dimensions. In artificial agents, a finite initial set of features can be provided manually by a knowledge engineer (Russell and Norvig, 2016). However, bottom-up concept building becomes unscalable as the problem complexity and feature count increase. Deep learning tools could enable an agent to extract the unsupervised features relevant to a given task and scenario, and build concepts based on those. This will remove the intermediator and allow the agent to filter an unlimited set of possible features. Yet, as the world becomes more complex, it is harder to define concepts in terms of features alone. Representing the concept space hierarchically, such that the combination of low-level concepts enables the definition of more complex ones, is thus essential.

In this section, we have discussed the fundamental features of concepts and conceptual space, as outlined in the theoretical literature. We aimed to highlight the importance of these traits, including adaptability within and across dimensions, as well as the ability to combine concepts to create a model that facilitates functional creativity. In the following sections, we will explore various theories related to computational creativity, with a focus on functional creativity. These theories will primarily operate within a conceptual space with properties similar to those outlined in this section.

3.1.5 Computational creativity theories

Boden (1996) conceptual space theory offers valuable insights into computational creativity, distinguishing between three primary types of creativity: combinatorial, exploratory, and transformational. Unlike earlier creativity theories that mainly emphasised novelty and the audience's perspective, Boden's theory delves into the underlying mechanisms that drive these creative processes.

Combinatorial creativity, the most accessible form, involves the novel juxtaposition of previously learned elements. It operates within a fixed conceptual space, drawing new connections between existing ideas, often through associative mechanisms. This process of forming associations is known as "associative learning" and enables us to link seemingly unrelated elements in our minds. This can be observed in artificial setups, where the agent recombines previously experienced object-action tuples retrieved from structurally similar episodes and reuses them in novel task configurations.

"Exploratory creativity," on the other hand, involves navigating through

a multidimensional conceptual space defined by various axes, such as x, y, and height. Within this conceptual space, individuals embark on a journey of exploration, uncovering new instances and ideas that have remained hidden within the existing boundaries. It is akin to charting uncharted territories within the existing creative landscape. This form of creativity can manifest in artificial agents as, rather than just copying past data, the agent selectively matches, filters, and recombines structurally relevant components (such as affordances or object relations) to construct plausible new solutions. This behaviour reflects an internal exploration of the space of achievable plans, enabled by structural generalisation rather than rote memorisation.

Notably, Boden's theory challenges the notion that "transformational creativity" is impossible. The feasibility of creativity, whether deemed possible or impossible, is closely tied to the current confines of the conceptual space in which an agent operates. When this concept space transforms, such as expanding to encompass new quality dimensions, the seemingly impossible can become possible, reshaping the creative landscape.

In exploring Boden's conceptual space theory, we gain insights into the essential functions that any creative agent must possess. This underscores the necessity for dynamic conceptual spaces capable of adaptation and evolution, accommodating the fusion of concepts, relations, and dimensions. Such adaptability fosters the generation of fresh and imaginative ideas. Furthermore, this discussion relates to our earlier conversations on the broader topics of generalisation in AI and compositional generalisation, emphasising the dynamic nature of conceptual spaces in creative AI systems.

Achieving all three types of creativity within a single system can be a formidable undertaking. To tackle this challenge methodically, we will commence our journey by striving for combinatorial creativity in the context of Creative Problem Solving (CPS). As per Boden's insights (Boden, 2009), combinatorial creativity stands out as the most accessible starting point. Our AI system will initially manifest combinatorial behaviour and subsequently progress towards embracing transformational creativity. The crux of our approach lies in leveraging the combinatorial operation, a cornerstone for artificial systems. This operation empowers systematic generalisation, facilitating the generation of fresh combinations, including those absent from the training data and even those bearing zero probability within the training distribution (Goyal and Bengio, 2022). It's crucial to note that these new combinations failed to surface in the training data and would remain absent even if we had access to an infinite volume of training data from our distribu-

tion. Boden's comprehensive theory equips us with a repertoire of operations that delineate diverse forms of creativity. Our primary aim is to adapt these creative facets to the realm of problem-solving, with a paramount objective of enhancing sense-making capabilities within our AI agents.

Boden's framework identifies combinatorial creativity as generating new ideas through novel combinations of familiar concepts, without altering the underlying conceptual space. Conceptual blending, as introduced by Fauconnier (Fauconnier and Turner, 1998), provides a cognitive mechanism for formally modelling this type of creativity. While Boden focuses on the kinds of creativity and the structure of conceptual spaces, Fauconnier provides a detailed account of how mental operations within and across these spaces can give rise to new meaning. This theory serves as a framework for problem-solving and sense-making, comprising four distinct spaces. Concept blending is a powerful domain-independent technique that allows us to avoid relying on specific dimensions. The Conceptual Integration Networks (CIN) space is abstract and further divided into three smaller mental spaces: input, generic, and blend. Mental spaces are small conceptual units constructed as we think and talk, with the purpose of local understanding and action. These spaces are partial assemblies that contain elements. Although conceptual and mental space definitions differ, they share the similarity of being represented geometrically, as Fauconnier portrays them. Mental spaces are represented by circles, with elements being points or icons within the circles, and lines represent connections between elements. This supports the idea of "the geometry of thought". Although Fauconnier's work (Fauconnier and Turner, 1998) did not directly investigate creativity, blending as a general cognitive operation enables dynamic and active concepts during thinking. Fauconnier identifies the construction and operation of blends as a creative process.

Our problem-solving approach involves interpreting the spaces of blending theory. The input space consists of concepts extracted and processed from the environment, such as the tools and context of a problem. The generic space contains elements shared among the input spaces and can be used to identify equivalence relations. In CPS, this can help us find similarities between previous and current problem requirements. The blend space is formed by combining elements from all the spaces (input and generic), but some elements from the input spaces can be merged before being added to the blend space.

A critical concept in this theory is selective projection. When using conceptual integration networks to solve a problem, only some aspects from the

input spaces will be chosen to be added to the blend space if they are helpful in solving the problem. This can lead to the creation of new structures in the blend space. Selective projection is a necessary and exciting process, and we need to explore different machine learning methods to achieve it. The mechanism behind selective projection is not described in Fauconnier and Turner (1998), but in our work on creative problem solving, we combine existing concepts to create new ones. This process required a selective projection mechanism, as not all features from each concept are necessary to solve the problem.

Both Boden's and Fauconnier and Turner's theories of creativity share a foundational emphasis on the recombination of ideas drawn from different conceptual sources. Whether through the traversal of conceptual spaces (Boden) or the dynamic integration of mental spaces (Fauconnier and Turner), creativity is consistently framed as a process of connecting previously unlinked structures in novel and meaningful ways. This theme resonates with Arthur Koestler's influential theory of bisociative thinking, articulated in The Act of Creation (Koestler and Burt, 1964), which further grounds the idea of creativity in the intersection of disparate cognitive domains.

Koestler introduces bisociation to distinguish between associative thinking within a single frame of reference and creative thinking that bridges multiple frames. If an idea can be associated with two frames of reference, F1 and F2, it becomes bisociated with both. The term bisociative implies that creative thinking should allow an idea to be perceived in multiple frames of reference.

What makes Koestler's account particularly relevant to our focus is its functional orientation. He defines a matrix as "any ability, habit, or skill, any pattern of ordered behaviour governed by a code or fixed rules," highlighting that creativity involves not only abstract reasoning but also the repurposing or recombination of functional behaviours.

This perspective aligns closely with our view that skills and actions are essential components of the concept space in Creative Problem Solving (CPS). The capacity to creatively reframe a task involves associating it with alternative functional structures or behavioural repertoires. This idea will be further developed in the following section on functional creativity and affordances, where we shift focus from conceptual abstraction to how agents perceive and act upon possibilities in the environment, and how these affordances enable novel solutions through the composition and repurposing of action schemas.

3.2 Functional creativity and Affordances

While the previous section surveyed general theories of creativity, this work focuses on a specific subset: functional creativity. We must distinguish classical problem-solving from Creative Problem Solving (CPS) to understand functional creativity. Traditional problem-solving tasks are typically defined by fixed initial and goal states and rely on well-established operators. In contrast, CPS tasks are characterised by incomplete or underspecified conceptual spaces, requiring the agent to expand or reorganise its internal representations to generate a solution. For example, CPS may involve designing a novel mechanism using a constrained set of parts or creatively reconfiguring an incomplete system to achieve a target function (Oltețeanu, 2020).

CPS aligns closely with Gestalt theories of insight, such as Karl Dunker's distinction between reproductive thinking, which involves applying known solutions, and productive thinking, which requires restructuring the problem representation to enable a novel insight. Functional creativity embodies this latter process, allowing agents to explore and extend their concept space, create new combinations, and reinterpret known structures in novel ways.

In our previous section, we discussed how an agent's tasks and environment can encourage or hinder their creativity. Humans gain different associations with stimuli by engaging with their environment in various everyday tasks, which they can utilise in the future. Functional creativity is a crucial mechanism that compels agents to explore and modify their concept space to develop alternative solutions to a problem. The ability to adapt and provide alternative solutions to a given environment is a vital aspect of generalisation in artificial agents.

According to Oltețeanu (2020), creative problem-solving should include varied desiderata such as visuospatial inference, creative use of affordance, concept generation and structure transfer, insight, and re-representation. Oltețeanu's research provides a theoretical framework for creative problem solving in artificial agents, allowing diverse creative problem-solving processes to occur. The knowledge encoding should be done at three different levels: feature space, concepts, and problem templates, stating that a conceptual space structure that is appropriate for CPS should contain hierarchical levels of representation going from sub-symbolic to symbolic, as supported by Oltețeanu (2020) in this framework.

Therefore, when an artificial agent is presented with a problem-solving task, it will process their environment and extract different sensory informa-

tion, which will help it build internal representations passively. We aimed to represent a concept in the concept space that can answer the questions of "what is?" and "what can be done with it?" according to a given context and goal. So far, we have concentrated on just one aspect of sensory information. However, as noted earlier, successful learning transfer requires more than sensory input. Different types of knowledge are crucial for creating useful concept representations (Cătărău-Cotuţiu et al., 2023). When faced with a problem-solving task related to objects, children typically interact with them first to learn about their characteristics and potential uses (Xie et al., 2021).

3.2.1 Affordances and temporality

The term affordance was introduced by Gibson in 1977 (Gibson, 1977) to describe the potential actions an environment offers to an agent, given the properties of objects and the context of interaction. Rather than being intrinsic properties of objects, affordances are understood as relational constructs, emerging from the dynamic interplay between agent and environment. This relational interpretation is reinforced by Chemero et al. (2003), who argues that affordances should be defined as static attributes but as agent-environment relationships that are grounded in perceptual and behavioural coupling.

The connection between affordances and functional creativity is especially salient when the initial representation or strategy for solving a problem proves insufficient. Functional creativity involves re-evaluating or reconfiguring object uses to achieve a goal in a novel context. Critically, visual similarity does not imply functional equivalence. For instance, while a beach ball and a bowling ball may appear similar in shape and size, their vastly different weights produce divergent affordances when, for example, attempting to balance a lever. As such, discovering affordances requires active exploration; agents must interact with objects through trial and error to infer their possible uses and behaviours (Cătărău-Cotuţiu et al., 2023).

Several formal frameworks have been proposed to articulate better and operationalise the concept of affordance, drawing from Gibson's foundational work and extending it through multiple disciplinary lenses. These frameworks typically define affordances through the interaction of three core elements: the agent, the environment, and an observer or interpreter. Key contributions include the perspectives of Chemero et al. (2003), Greeno (1994), Michaels (2003), Sanders (1997), Stoffregen (2018), Turvey (1992) and Wells

(2002). These interpretations vary in significant ways. For example, Turvey views affordances as invariant environmental properties, while Stoffregen also includes the observer's perceptual systems. Chemero's view is more dynamic, incorporating behaviour into the definition of affordance. On the other hand, Steedman conceptualises affordances as action possibilities, proposing their integration into planning systems.

Among these, we find the formalisation proposed by Şahin et al. (2007) particularly compelling. Like Chemero, they conceptualise affordances as relations between the agent and environment, but they further adapt this notion for practical use in autonomous robotic systems. According to their theory, an affordance is a relation between a specific effect and an (entity, behaviour) combination. This means that the effect is produced when an agent applies the behaviour to the entity. The refined formalisation is expressed as (effect, (entity, behaviour)). This formalisation makes it clear that affordances are relations consisting of an (entity, behaviour) pair and an effect, which has the potential to be generated when the agent applies the behaviour to the entity. This theoretical foundation is operationalised within the AIGenC framework in Section 4.1, where affordances are defined as acquired relations between (effect, reward) pairs and (concept-object, action) tuples, and formally instantiated as Affordance Transition Tuples in Section 5.1.2.

Therefore, we propose enriching the formation of a hierarchical concept space by including affordances related to objects' encoded physical features. Incorporating affordances enables an agent to form various complex representations in a given functional, spatial and temporal context, hence acquiring knowledge about object manipulation. We have also introduced a modification to Şahin's formalism such that we can capture the notion of an action outcome.

Time and context are foundational dimensions of creative problem solving (CPS), particularly in how agents form and apply concepts such as affordances. Many affordances derive their functional significance not from intrinsic object properties alone, but from the situational and temporal context in which they are encountered. This interdependence between structure, sequence, and consequence aligns with the neurocognitive construct of episodic memory, which serves as a mechanism for organising and retrieving temporally structured experiences.

Episodic memory organizes personal experiences based on particular times and places, effectively partitioning incoming sensory information into distinct

episodes or events. This segmentation of experiences, informed by associative learning principles, leads to a dynamic representation of stimuli over time. Comprehending the temporal relationships between events is pivotal for establishing causality in our actions. Furthermore, stimulus representations across different events may vary due to the temporal progression of the events themselves, as noted in Kokkola et al. (2019)'s research. Kokkola et al. (2019)'s study emphasises that stimulus representation undergoes temporal evolution, with certain elements exhibiting differential activity at various points during stimulus presentation. This temporal dynamics is key to understanding how our cognitive system processes and associates information. The notion of episodes, essentially groups of events, is a critical bridge between temporal reasoning and functional representations. According to Kokkola et al. (2019), when two temporal clusters, akin to what we call episodes, display similar activities, associations between elements within the predicting cluster and the outcome cluster strengthen more rapidly and achieve higher asymptotic strength. This temporal clustering and the ability to discern similarities among episodes are pivotal for facilitating temporal reasoning and supporting functional representations, ultimately enhancing the creative problem-solving capabilities of our cognitive agents (Goyal and Bengio, 2022).

The findings from Kokkola et al. (2019) highlight that temporally organised episodes are not merely sequences of events but structured entities where relational and temporal coherence enables faster and stronger associative learning. For artificial agents to benefit from similar mechanisms, particularly in support of functional generalisation and creative reuse, these episodes must be encoded in a format that captures their internal structure, preserves their causal dynamics, and allows meaningful comparison across time and context. Graphs naturally meet these requirements. They provide a flexible and compositional framework for representing interactions between objects, actions, and affordances, while explicitly modelling temporal and causal relationships. This makes them especially well-suited for encoding episodes as structured knowledge traces that can be queried, matched, and recombined. In the following section, we introduce the graph-based foundations that underpin this representational strategy.

3.3 Graph Representations in Cognitive and Computational Models

Graphs are non-linear data structures composed of nodes (entities) and edges (relations) that can flexibly encode various interactions, dependencies, and abstractions. In the context of conceptual representation, graphs offer a powerful and expressive framework for modelling structured knowledge. Unlike fixed-vector representations or matrices, graphs are inherently compositional and modular: new relationships can be introduced or removed without restructuring the entire space. This makes them particularly well-suited for dynamic concept spaces. Additionally, graphs enable the representation of different levels of hierarchical abstraction, which means that a graph can be labelled and become a node in another higher-level graph.

Alternative relational structures have been proposed across various disciplines. Using graphs for structured memory representation has deep roots in cognitive science. The Quillian semantic network model (Quillian, 1967; Collins and Quillian, 1969) proposed that concepts are stored as nodes in a network, with pointers to their properties and categorical relationships. For instance, the node "canary" may point to bird as a superclass and to properties like yellow and can sing. More general properties (e.g., can fly) are not duplicated at every subordinate node but are inherited from higher levels (e.g., from the bird node). This architecture conserves memory and supports inference via traversal: to answer whether a canary can fly, the system must move up to "bird" and retrieve its associated features. Thus, Quillian's model provides early evidence that relational memory structures such as graphs can support compositional reasoning and hierarchical generalisation, principles that remain central in contemporary AI. However, the model also has significant limitations. It assumes a rigid taxonomic structure of knowledge, which does not reflect the graded, context-sensitive nature of human categorisation. Inference is symbolic and deterministic and lacks mechanisms for handling ambiguity, uncertainty, or noisy input.

Another influential model is frame theory (Minsky, 1974), which encodes stereotypical situations as structured memory units known as frames. Each frame comprises fixed top-level elements that define the general structure of a scenario, alongside lower-level slots that must be dynamically filled with context-specific data or linked sub-frames. Although frames provide a modular and interpretable way to represent knowledge, they are often handcrafted and domain-specific, limiting their scalability and flexibility.

In contrast, tensor-based models (Smolensky, 1990; Plate, 1995) and neural embedding approaches, such as relation vectors, seek to represent relational information in a continuous, distributed form. These data-driven and scalable methods make them appealing for large-scale applications. However, they often sacrifice interpretability and lack support for explicit symbolic reasoning or compositional generalisation, as relational structure is encoded implicitly and cannot be easily isolated or manipulated.

While each representation has its strengths, many struggle to jointly satisfy the requirements of interpretability, compositionality, and structural flexibility. Vector embeddings, for example, capture relational similarity through proximity in latent space but obscure the underlying structure, making it difficult to revise or reason about individual relationships. In contrast, graph-based representations explicitly encode both entities and their interrelations, supporting symbolic manipulation, relational abstraction, and goal-directed recombination, capabilities essential for planning and creative problem-solving.

Graphs offer flexibility in both vertical and horizontal dimensions. Vertical flexibility enables reasoning across multiple levels of abstraction within a hierarchy. In contrast, horizontal flexibility allows for structural expansion or modification at a given level, by adding, removing, or reconfiguring nodes and edges. To account for temporal dynamics, dynamic graphs extend this flexibility by enabling the graph structure itself to evolve over time. This makes them particularly well-suited for modelling creative cognition, temporally grounded reasoning, and adaptive knowledge structures, as they can reflect the agent's ongoing interaction with its environment. Such adaptability is essential for representing functional classes, episodic memory, and concept spaces that shift and reorganise in response to changing task demands.

However, using dynamic graphs in artificial systems requires an efficient and flexible memory structure. While static graphs are often stored as adjacency matrices, this format is ill-suited for dynamic, sparse, or partially connected graphs, where structure changes over time. A more suitable approach is adjacency lists, which represent a graph as an array of linked lists. Each node stores a list of its current connections, allowing for efficient updates and memory allocation as the graph evolves. This representation supports real-time modification and maintains scalability even in large, heterogeneous knowledge structures. This format supports partial connectivity and dynamic restructuring, making it well-suited for open-ended environments and incremental learning. Moreover, it aligns with theoretical requirements

for creativity-supportive representations—namely, that they be hierarchical, adaptive, and capable of reconfiguration in response to novel stimuli (Fauconnier and Turner, 1998; Gardenfors, 2004; Mednick, 1962; Wallas, 1970).

These advantages have driven a growing interest in graph-based approaches across artificial intelligence. Neural-symbolic systems (Garcez et al., 2008), knowledge graphs (Hogan et al., 2021), graph neural networks (GNNs) (Battaglia et al., 2018), and relational deep reinforcement learning frameworks (Zambaldi et al., 2018) all use graphs to model structured and flexible representations of knowledge. Their ability to support compositionality, abstraction, and generalisation has made them central to systems designed for multi-task reasoning, adaptive planning, and creative problem-solving.

Beyond their structural flexibility, graphs support various operations essential for reasoning and learning, including graph traversal (e.g., breadth-first or depth-first search), sub-graph matching and node and edge manipulation. These operations enable agents to perform dynamic inference, apply structural constraints, and recombine known patterns in novel ways, core capabilities for adaptive problem-solving and compositional generalisation.

Traditional approaches to graph comparison, such as isomorphism checks or sub-graph matching, often rely on rigid structural constraints and do not scale well in environments where graphs are large, noisy, or semantically heterogeneous. While theoretically grounded methods like Gromov-Wasserstein distances from optimal transport theory (Alvarez-Melis and Jaakkola, 2018) offer more flexible structural alignment, they can be computationally expensive and are limited in their ability to integrate semantic content or context-sensitive features. In our setting, for instance, comparing graph-structured concepts requires sensitivity to topological similarity and functional relevance: how affordances, objects, and actions interact over time to produce meaningful outcomes.

Recent advances have shifted toward learning-based methods that encode graph structure and semantics into vector representations to address these limitations. Graph neural networks (GNNs) provide a framework for learning such representations by propagating information along edges and aggregating context from local neighbourhoods. When combined with attention mechanisms, GNNs can learn to weight and prioritise relevant nodes and relationships dynamically, enabling fine-grained reasoning over graphs without requiring hand-crafted alignment criteria.

3.4 Representation Learning in Sequential and Hierarchical Tasks

As discussed in the previous section, graph-based representations offer a robust framework for encoding structured relationships between entities, actions, and outcomes. However, to use such structures effectively in learning systems, we must also understand how representations are acquired, how agents learn to organise, abstract, and generalise knowledge from experience. This is the central concern of representation learning, which enables agents to extract meaningful structure from high-dimensional data and apply it across diverse tasks and environments.

Representation learning transforms high-dimensional inputs, such as raw sensory observations or rich interaction histories, into compact and informative encodings that support efficient inference, prediction, and decision-making. These encodings can be learned through either supervised or unsupervised methods. While supervised approaches optimise for performance on specific labelled tasks, unsupervised and self-supervised methods aim to uncover meaningful structure in the data without external labels, an essential property for agents operating in interactive and open-ended environments.

Much of the existing work in representation learning has focused on compression-based techniques, where the goal is to project high-dimensional observations into lower-dimensional latent spaces that preserve task-relevant structure. These approaches aim to reduce redundancy, improve sample efficiency, and support policy generalisation in fields like reinforcement learning (RL) (Bengio et al., 2013; Lesort et al., 2018; Wang et al., 2021). Disentangled representation learning is a notable direction within this paradigm, which seeks to factor latent spaces into independently meaningful components, such as object identity, position, or controllable dynamics. By isolating such factors of variation, these representations promote modularity, interpretability, and transfer across tasks that share underlying structure (Higgins et al., 2018).

A widely adopted framework for learning latent representations is the variational autoencoder (VAE) (Kingma et al., 2013), which uses a probabilistic encoder-decoder architecture to reconstruct input data while regularising the latent space to follow a known prior distribution. VAEs are valued for their ability to produce smooth, continuous latent embeddings that support interpolation, sampling, and generative modelling. In reinforcement learning, VAEs have been employed to encode compact representations of states

and actions, enabling more efficient policy learning and decision-making.

However, standard VAEs are often limited in capturing structured dependencies, particularly in sequential, hierarchical, or temporally extended patterns. In such cases, compressing interactions into a single flat latent space can lead to losing important compositional or relational information. To address these limitations, various hierarchical extensions have been proposed, such as Hierarchical VAEs (HVAEs) (Zhao et al., 2017) and Hierarchical Variational Autoencoders for Control (HVACs) (Edwards and Storkey, 2016; Vahdat and Kautz, 2020). These models introduce multiple layers of stochastic latent variables, allowing the system to encode both low-level detail and high-level structure within a single generative framework. HVAC models, in particular, are designed for control settings, allowing agents to plan using temporally abstract latent goals while grounding them in low-level motor control. These hierarchical models are instrumental in domains with complex action semantics or task hierarchies.

Despite their advantages, both VAEs and their hierarchical variants often treat input as monolithic and overlook the fine-grained structure of the environment. This limitation becomes especially pronounced in sequential tasks, where the agent must reason not only about state identity but also about how actions, objects, and outcomes unfold over time. Generalisation in such settings depends on capturing temporal, causal, and functional dependencies, which cannot be easily inferred from static state representations.

Trajectory representations offer a richer alternative by preserving the sequence of decisions and their evolving consequences. They support long-horizon reasoning, temporal credit assignment, and planning under delayed feedback—core requirements in many RL domains. Various approaches have explored trajectory-level learning, including autoencoders for behaviour sequences, latent skill discovery, and segment-based abstraction. However, as we will discuss in the reinforcement learning section, many existing approaches continue to treat trajectories as unstructured or flat sequences, limiting their potential for reuse and transfer.

To overcome the limitations of treating trajectories as flat or purely sequential data, in our work, we shifted toward graph-based representations that model trajectories as structured, relational systems. Rather than viewing a trajectory as a list of temporally ordered states and actions, this perspective treats it as a dynamic network of interacting entities, linked through relations that reflect temporal, causal, or functional dependencies.

Such representations must be made tractable and comparable for down-

stream reasoning tasks. While theoretically grounded methods like Gromov-Wasserstein distances from optimal transport theory (Alvarez-Melis and Jaakkola, 2018) mentioned earlier offer flexible alignment between graphs with different topologies, they remain computationally demanding and often struggle to incorporate semantic content or contextual relevance. This motivates using learned graph encodings, which can preserve structure and semantics while enabling efficient comparison, retrieval, and policy conditioning. In the next section, we turn to Graph Neural Networks (GNNs) and attention mechanisms, which provide a framework for learning such representations from data.

3.5 Graph neural networks and attention

While previous chapters have provided the theoretical background behind graphs' suitability for our theoretical framework, this section will cover some of the information needed to use graphs in machine learning applications. Graphs provide a powerful and flexible structure for representing relational information. Formally, a graph G is defined as a pair (V,E), where V is a set of vertices (nodes) and E is a set of edges connecting pairs of nodes:

$$G = (V, E) \tag{1}$$

Edges may be represented explicitly as pairs (i,j) or encoded in an adjacency matrix or list A, where each entry $A_{i,j}$ denotes the presence or strength of a connection between node i and node j.

Unlike convolutional neural networks (CNNs), which assume a fixed-grid structure and exploit spatial locality, Graph Neural Networks (GNNs) generalise convolutional operations to arbitrary graph topologies. This flexibility allows GNNs to aggregate information from neighbouring nodes according to the graph's structure, learning over domains with non-Euclidean data such as knowledge graphs, trajectories, and causal systems.

A typical GNN takes as input a node feature matrix (NxD), where each of the N nodes has a D-dimensional feature vector, along with an adjacency matrix A of size (N×N). A standard fully connected layer propagates input as:

$$H^{(1)} = \sigma\left(W^{(0)\top} X + b^{(0)}\right) \tag{2}$$

where $X \in \mathbb{R}^{N \times D_0}$ is the input feature matrix, with N denoting the number of nodes and D_0 the input feature dimensionality. The matrix $W^{(0)} \in \mathbb{R}^{D_0 \times D_1}$ represents the learnable weight parameters of the first layer, and $b^{(0)} \in \mathbb{R}^{D_1}$ is the corresponding bias vector. The output $H^{(1)} \in \mathbb{R}^{N \times D_1}$ is the hidden node representation after applying the non-linear activation function $\sigma(\cdot)$, such as the Rectified Linear Unit (ReLU).

More generally, we denote by $H^{(k)} \in \mathbb{R}^{N \times D_k}$ the node feature matrix at layer k, where D_k is the dimensionality of the node embeddings at that layer. The updated node representations at the next layer are given by $H^{(k+1)} \in \mathbb{R}^{N \times D_{k+1}}$.

Graph Convolutional Layer In contrast to purely feature-based transformations, Graph Convolutional Networks (GCNs) explicitly incorporate graph topology by propagating information between neighbouring nodes (Yao et al., 2019):

$$H^{(k+1)} = \sigma\left(A^* H^{(k)} W^{(k)}\right) \tag{3}$$

where $W^{(k)} \in \mathbb{R}^{D_k \times D_{k+1}}$ is a learnable weight matrix and $\sigma(\cdot)$ is a non-linear activation function. The matrix $A^* \in \mathbb{R}^{N \times N}$ denotes the normalized adjacency matrix, typically defined as

$$A^* = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}, \tag{4}$$

where $\tilde{A} = A + I$ is the adjacency matrix augmented with self-loops and $\tilde{D}$ is the degree matrix of $\tilde{A}$. This normalization ensures numerically stable message aggregation and equal contribution from neighbouring nodes.

Message Passing Framework More generally, Graph Neural Networks (GNNs) can be expressed within a message passing framework. At each layer k, node representations are updated by aggregating messages from neighbouring nodes and combining them with the node's own state:

$$\begin{aligned} m_{si}^{(k)} &= \frac{1}{|\{(i,j) \in \mathcal{E}_s\}|} \sum_{j:(i,j) \in \mathcal{E}_s} f_{\text{msg}}\left(h_{sj}^{(k-1)}\right), \\ n_{si}^{(k)} &= f_{\text{node}}\left(h_{si}^{(k-1)}\right), \\ h_{si}^{(k)} &= f_{\text{update}}\left(m_{si}^{(k)}, n_{si}^{(k)}\right). \end{aligned} \tag{5}$$

Here, $\mathcal{E}_s$ denotes the edge set of graph s. The term $m_{si}^{(k)}$ represents the aggregated message received by node i from its neighbouring nodes, computed using a message function $f_{\text{msg}}(\cdot)$. The term $n_{si}^{(k)}$ corresponds to a transformation of the node's own previous representation via a node update function $f_{\text{node}}(\cdot)$. The updated node embedding $h_{si}^{(k)}$ is obtained by combining the neighbourhood message and the node-specific information using an update function $f_{\text{update}}(\cdot)$.

Graph Convolution Operator For notational convenience, the message passing process can be compactly expressed as a graph convolution operator:

$$h_{si}^{(k)} = \text{GConv}\Big(A_s, h_{si}^{(k-1)}\Big) \tag{6}$$

where $\text{GConv}(\cdot)$ denotes a generic graph convolution function parameterized by the adjacency matrix A_s and the node representations from the previous layer. This abstraction encompasses a wide range of GNN architectures, including GCNs, attention-based models, and relational message passing networks.

Crucially, GNN outputs are permutation-invariant, meaning that isomorphic graphs —those with identical structure but different node ordering —are treated equivalently. This property allows GNNs to generalise across structurally similar inputs, making them robust to superficial differences in representation and well-suited for reasoning over abstract relational structures.

While early GNN architectures were designed for homogeneous graphs, where all nodes and edges share the same type, many real-world domains are inherently heterogeneous. This is especially true in task-driven environments that involve diverse elements, such as objects, actions, affordances, and task outcomes. In such cases, different node and edge types carry different semantic roles, and modelling their interactions requires architectures capable of distinguishing and leveraging this heterogeneity.

Heterogeneous Graph Neural Networks (HGNNs) (Yang et al., 2023a) address this challenge by extending standard GNNs to support typed nodes and edges. These models utilise mechanisms such as meta-paths, edge schemas, and type-aware message passing to capture multi-relational structures across different semantic layers (Wang et al., 2021). A meta-path is a sequence of node and edge types (e.g., object–affords–action–achieves–goal) that defines a relational pattern or semantic pathway in the graph. By modelling how

information flows through such typed sequences, HGNNs can learn context-sensitive embeddings that reflect both local and global structural dependencies. As a result, HGNNs are particularly well-suited to domains that demand hierarchical reasoning, such as affordance-based object manipulation, compositional planning, or structured problem-solving.

Building on this foundation, recent research has explored the integration of causal reasoning into graph-based learning. For example, Jin et al. (2024) represent trajectories as causal graphs, applying counterfactual link prediction to model not just associations but potential causal dependencies between entities. While this approach allows for fine-grained causal inference, it relies on semi-supervised training, assuming access to ground-truth annotations or predefined graph structures. This assumption poses a limitation in reinforcement learning (RL) environments, where supervision is sparse, and agents must discover structure purely through interaction and exploration.

To overcome the dependency on labelled data, self-supervised HGNNs such as Heterogeneous Co-contrastive Learning (HeCo) have been introduced (Wang et al., 2021). HeCo employs two complementary views of the graph: one that captures global schema-level structure, and another that encodes meta-path-based local context. By contrasting these views, the model learns to embed nodes in a way that reflects both high-level semantics and neighbourhood-level structure, without requiring external labels. However, despite its strengths, HeCo is designed for static, node-centric graphs, making it less suitable for dynamic or episodic environments, where graphs evolve over time and agents must learn from sequences of interactions.

To address this, Graph Transformer Co-contrastive Learning (GTC) (Sun et al., 2025) introduces a dual-branch architecture that combines local graph neural network aggregation with Transformer-based global modelling. Its Hop2Token mechanism converts multi-hop neighbourhoods into structured token sequences, enabling the model to capture hierarchical and contextual dependencies better. This hybrid design improves representation quality and avoids over-smoothing, outperforming traditional HGNNs in static benchmarks. Nevertheless, like HeCo, GTC operates on fixed graph snapshots and lacks native mechanisms for temporal or episodic modelling, limiting its utility in RL settings where structure must be learned and adapted incrementally over time.

Taken together, these advances demonstrate the potential of HGNNs and attention-based models to capture rich relational and semantic structure. Yet they also highlight a gap: most current models are not designed to support

the dynamic, cumulative, and interactive nature of reinforcement learning. They assume static input graphs, overlook temporal evolution, and fail to track how task elements, affordances, and outcomes change throughout learning. These limitations point to the need for an expanded RL ontology that incorporates structured, hierarchical trajectory representations and enables reasoning over shared components across episodes.

These advances in graph neural networks and attention mechanisms provide the technical foundation for our approach. In Chapter 5, we employ heterogeneous GNNs with meta-path-aware attention to encode Structurally Enriched Trajectories (SETs), enabling agents to compare and retrieve structurally similar experiences from long-term memory.

While recent developments in graph-based learning offer promising tools for capturing relational structure, most reinforcement learning frameworks continue to rely on a classical ontology that models tasks using flat representations. To understand the representational gap, it is necessary first to review the foundational elements of reinforcement learning and how they shape current approaches to learning and generalisation.

3.6 Reinforcement Learning

Reinforcement Learning (RL) is a learning paradigm in which an agent interacts with an environment over time to learn a policy that maximises cumulative reward (Sutton et al., 1998). The core elements of an RL system include states, which represent the current configuration or situation of the environment; actions, which are the available decisions or operations the agent can take; and rewards, which provide scalar feedback on the desirability of outcomes resulting from those actions. RL agents learn through interaction, unlike other machine learning methods that rely on data. States represent the current situation of the agent in the world. The RL agent learns to optimise its policy to achieve a goal state by maximising the cumulative reward. In other words, it learns a strategy based on the distribution of rewards to achieve its objectives.

Markov Decision Processes The Markov Decision Process (MDP) is the standard formalism used in reinforcement learning (RL). An MDP is defined as a tuple

$$\mathcal{M} = (S, A, R, T, p),$$

where S is the state space, A is the action space, $R : S \times A \times S \to \mathbb{R}$ is the scalar reward function, $T(s' \mid s, a)$ is the (possibly stochastic) Markovian state transition function, and $p(s_0)$ is the initial state distribution.

At each discrete time step t, the agent observes a state $s_t \in S$, selects an action $a_t \in A$, receives a reward $r_t = R(s_t, a_t, s_{t+1})$, and transitions to the next state $s_{t+1} \sim T(\cdot \mid s_t, a_t)$.

The objective in an MDP is to learn a policy $\pi(a \mid s)$ that maximizes the expected cumulative discounted reward, also known as the return:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{7}$$

where $\gamma \in [0, 1)$ is the discount factor controlling the relative importance of future rewards.

The state-value function of a policy π is defined as

$$V^{\pi}(s) = \mathbb{E}_{\pi} \left[G_t \mid s_t = s \right], \tag{8}$$

and the action-value function (Q-function) is defined as

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi} \left[G_t \mid s_t = s, a_t = a \right]. \tag{9}$$

The optimal value functions are given by

$$V^*(s) = \max_{\pi} V^{\pi}(s), \quad Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a), \tag{10}$$

and the optimal policy π^* can be derived from the optimal Q-function as

$$\pi^*(s) = \arg\max_{a \in A} Q^*(s, a). \tag{11}$$

Deep reinforcement learning (deep RL) combines the sequential decision-making framework of reinforcement learning with the powerful function approximation capabilities of deep learning. In this setting, agents interact with their environment by feeding observations, often high-dimensional visual frames, into deep neural networks, which then output actions based on learned internal representations. Unlike classical RL approaches that operate on predefined, low-dimensional state spaces, deep RL agents can autonomously learn features from raw input, allowing them to operate in more complex and unstructured environments. Deep RL agents learn a variety of environment features, rather than simply learning policies tied to

global states. Some deep RL models train the neural networks responsible for learning environment representations and the RL task in a single loss function (e.g., Baker et al. (2019); Jaderberg et al. (2016); Raileanu et al. (2021)).

Deep Q-Learning In Q-learning, the optimal action-value function is learned iteratively using the Bellman optimality equation:

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a')\right]. \tag{12}$$

Deep Q-Networks (DQN) approximate the Q-function using a neural network $Q(s,a;\theta)$ with parameters θ. The network is trained by minimizing the temporal-difference (TD) loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta)\right)^2\right], \tag{13}$$

where $\mathcal{D}$ is a replay buffer and θ^- denotes the parameters of a target network, which are periodically updated to stabilize training.

Double Deep Q-Learning Double Deep Q-Networks (DDQN) reduce overestimation bias in DQN by decoupling action selection and evaluation. The target value in DDQN is defined as

$$y^{\text{DDQN}} = r + \gamma Q\left(s', \arg\max_{a'} Q(s',a';\theta), \theta^-\right). \tag{14}$$

This modification improves training stability and empirical performance, particularly in environments with noisy or stochastic rewards.

However, other works (Stooke et al., 2021; Yarats et al., 2021) and our work separate the representation learning task from goal-directed training. Representations are extracted using pretrained models, which are only updated inside the RL training loop instead of being trained from scratch. This approach allows for faster training and eliminates the need to optimise for two tasks simultaneously. Learning representations within the RL training loop is challenging because the only source of supervision is the reward, which is often noisy, sparse, and delayed (Zhang et al., 2022).

It is important to consider how these representations are formed, whether they are pretrained or not. A key limitation of conventional RL architectures is that they often encode experience as flat sequences of frames, disregarding

both structure and temporal abstraction. Simply encoding entire frames into latent vectors makes it difficult for the agent to differentiate between relevant and irrelevant features, including core features necessary for learning and those that are not, such as image quality, resolution, and colour. It also fails to represent temporal dependencies critical to tasks requiring delayed reward estimation, long-horizon planning, or compositional skill reuse (Hayman and Huebner, 2019; Zhang et al., 2022).

While outcomes, whether intended or incidental, offer a concise summary of performance, they do not capture the process by which the outcome was achieved. Understanding a task requires more than recognising the final state; it requires interpreting the trajectory of actions and interactions that led to it. As such, one of the central challenges in RL is to move beyond stepwise transitions and instead learn structured trajectory representations that capture the dynamics of behaviour over time.

Classically, trajectories are defined as sequences of state-action pairs that unfold from an initial state to termination (Sutton et al., 1998). While this introduces a basic temporal structure, it does not encode hierarchical or causal relationships beyond immediate transitions. One extension of this classical definition is the options framework (Precup, 2000), which introduces temporally extended actions that allow an agent to execute multi-step behaviours before concluding. An option consists of an internal policy, an initiation set, and a termination condition, enabling an agent to select and commit to more than a single-step action. Other approaches, such as RL with trajectory feedback, shift from evaluating individual steps to assessing the entire trajectory as a whole (Efroni et al., 2021). This is particularly useful in scenarios where immediate feedback is sparse or impractical, such as real-world robotics or autonomous systems, where long-term dependencies must be considered. Both options and trajectory feedback offer different ways to extend decision-making beyond stepwise transitions: one by grouping actions into temporally extended choices, and the other by evaluating entire trajectories. Yet, neither provides the level of structural detail necessary for effective generalisation and knowledge transfer. Options, as well as the trajectory feedback approach, treat elements of decision-making as indivisible units, lacking the granularity needed for structured generalisation. Options group actions into fixed temporal sequences, ignoring finer-grained execution details such as object properties, interaction dynamics, and environmental constraints. Similarly, trajectory feedback evaluates entire sequences without analysing individual contributions, making it difficult to identify structural

similarities across tasks or extract reusable behavioural patterns that support generalisation and adaptation. This limitation suggests the need for more structured trajectory representations that capture not just individual actions but also high-level task structures and reusable behavioural patterns.

An approach that deals with trajectories in a more structured way is the Self-Consistent Trajectory Autoencoder (SeCTAR) (Co-Reyes et al., 2018). SeCTAR frames hierarchical reinforcement learning (HRL) as a trajectory-level representation learning problem. Instead of learning discrete sub-policies, SeCTAR builds continuous latent spaces of trajectories, allowing agents to represent and generate structured behaviours. While SeCTAR's latent space encodes meaningful trajectory structures, it does not explicitly segment trajectories into transferable skills, which may limit its effectiveness for compositional learning and policy transfer. Methods such as the Constructing Skill Trees (CST) algorithm (Konidaris et al., 2011) address this limitation by automatically segmenting demonstration trajectories into reusable skills using change-point detection techniques. This approach enables agents to decompose long-horizon tasks into meaningful sub-policies, facilitating skill reuse and hierarchical planning. However, CST relies on human-provided demonstrations, which means that it does not autonomously explore or discover skills without predefined examples. A more generalisable approach would integrate structured trajectory representations with autonomous skill discovery, enabling agents to learn and refine reusable skills without requiring expert demonstrations.

Most existing trajectory representation methods in RL treat trajectories as flat sequences, focussing on linear dependencies rather than hierarchical structures. For example, the Outcome-Driven Actor-Critic (ODAC) framework (Rudner et al., 2021), trajectory-based prediction models (Haines et al., 2018), and memory-based RL frameworks such as Sequential Episodic Control (SEC) (Freire et al., 2024) all adopt flat representations of trajectories as a core design choice. ODAC models latent outcome representations by treating trajectories as flat sequences of state-action transitions, providing a structured yet non-hierarchical approach to inferring task success. Similarly, Haines et al. (2018) emphasises cumulative patterns in state sequences to predict outcomes, assuming linear trajectory structures to simplify trajectory modelling. Meanwhile, SEC improves adaptability and generalisation by retaining episodic state-action pairs, allowing agents to recall past interactions and improve learning efficiency in sparse-reward environments. However, SEC's flat memory structure prevents it from capturing hierarchi-

cal trajectory dependencies, which are essential for structured learning, skill reuse, and transferability.

These limitations highlight the need to expand the ontology of reinforcement learning to include structured representations of task elements, hierarchical trajectory dependencies, and cross-episode relationships.

To summarise, this chapter has introduced the theoretical foundations and technical landscape underpinning our approach to structure-aware generalisation in reinforcement learning. We began by surveying key theories of creativity and functional abstraction, emphasising the cognitive importance of concept composition, affordance-based reasoning, and adaptive problem-solving. These insights motivated the need for systems that can not only learn from past experience, but flexibly reorganise and reuse that experience in novel contexts.

Building on this foundation, we introduced graph-based representations as a powerful formalism for modelling structured knowledge—particularly in domains where relationships, transitions, and causal dependencies are central to behaviour. We then explored how graph neural networks (GNNs), including heterogeneous and attention-based variants, provide mechanisms for encoding and reasoning over such structures. These models enable agents to learn relational patterns, attend to relevant substructures, and build abstractions that generalise beyond specific instances.

We also reviewed advances in representation learning, highlighting both the promise and limitations of compression-based methods for sequential and temporally extended tasks. In particular, we noted that many approaches struggle to capture the compositional and causal dynamics essential for generalisation. Finally, we examined the unique demands of reinforcement learning environments, where sparse rewards, evolving state spaces, and structural diversity make the case for richer, more expressive representations that integrate both trajectory history and conceptual abstraction.

Together, these discussions converge on the need for a unified framework that embeds structured experience into a form that supports retrieval, analogy, and reuse across tasks. At the conceptual level, this thesis builds on AIGenC (Cătărău-Cotuţiu et al., 2023), a cognitively inspired architecture for generalisation that identifies Reflective Reasoning, Conceptual Abstraction, and Creative Reuse as central components of adaptive intelligence. As part of this framework, we introduce SETLE (Structurally Enriched Trajectory Learning and Encoding) as a concrete realisation of AIGenC's Reflective Reasoning module. SETLE encodes agent experience as graph-structured

trajectories, capturing objects, affordances, and outcomes in a temporally and relationally grounded manner. It leverages graph neural networks and attention-based memory mechanisms to support structural matching, abstraction, and knowledge reuse. In the final stage of this research, SETLE is integrated into the reinforcement learning loop, allowing agents to enrich decision-making with structured representations derived from prior episodes. This integration not only improves data efficiency and policy performance but also marks a foundational step toward realising computational creativity through structure-aware learning and generalisation.

4 AIGenC: A Framework for Structured Generalisation and Creativity

The code for the different components of the system is available at the following repositories:

- `https://github.com/CatarauCorina/aigenc_main_rl_loop/` – implementation of the main RL training loop and agent integration.
- `https://github.com/CatarauCorina/setle_code` – implementation of the SETLE memory system for graph-based trajectory encoding and retrieval.

Before presenting the details of our framework, we begin by establishing its central objective: to enable artificial agents to learn, reuse, and generate structured representations that support flexible generalisation in reinforcement learning (RL). Rather than proposing a definitive cognitive theory of creativity, we draw inspiration from a range of cognitive models and theories, particularly those emphasising concept abstraction, affordance reasoning, and combinatorial learning. Our goal is not to mimic human functional creativity, but to harness its principles to improve representational capacity and knowledge transfer in artificial agents.

The AIGenC framework (Cătărău-Cotuţiu et al., 2023) provides a conceptual foundation for this endeavour. It proposes that generalisation in artificial systems should be approached not as a by-product of data scale or parameter tuning, but as a structured process grounded in the agent's ability to construct, adapt, and recombine concepts over time. In contrast to most machine learning models that operate on raw sensory data, AIGenC assumes

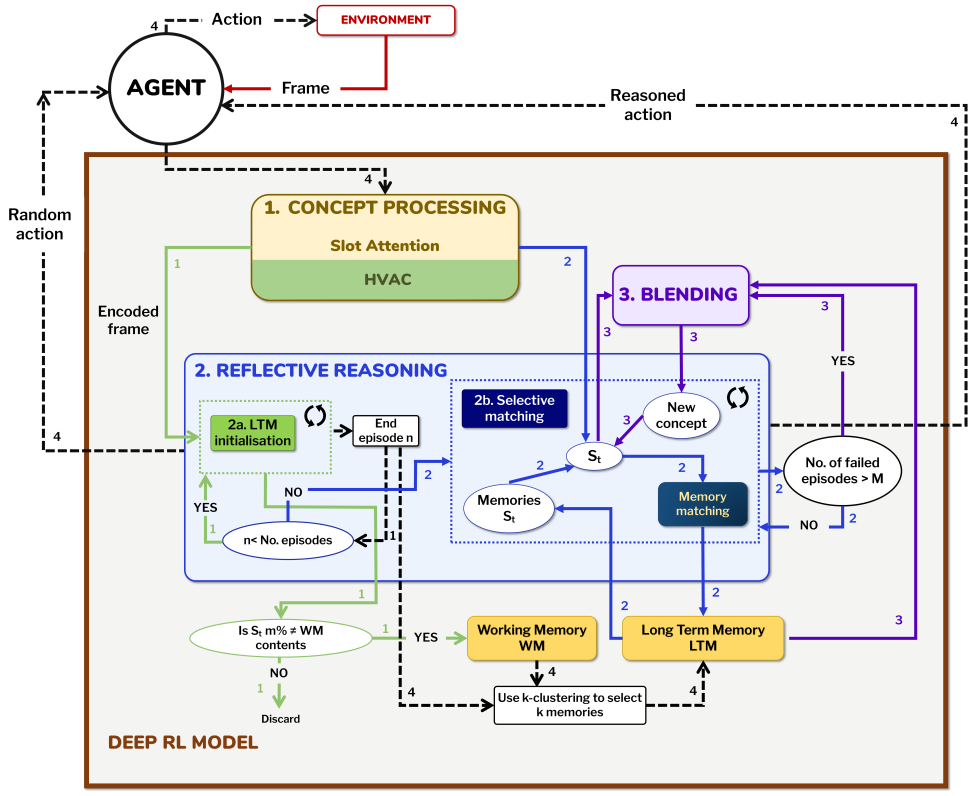


Figure 1: The three components of the model - Concept Processing (cream/olive), Reflective Reasoning (blue), Blending (purple) - and the algorithmic flow. The agent receives input in the form of frames from the environment. Frames are encoded in the Concept Processing component as vector representations by two unsupervised models. In the Reflective Reasoning component, LTM is initialised by random exploration (2a). The process of populating Working Memory and LTM is indicated by open tip arrows, a solid green (1) and a dashed black line (4), respectively. Following LTM initialisation, a process called Selective Matching (2b) is activated together with Deep Reinforcement Learning training. Selective Matching returns from LTM the concepts most similar to the current state. The retrieved concepts are then incorporated into the current state before they are inputted into the Deep Reinforcement Learning model. If the agent is unsuccessful at solving the task for several episodes, a Blending process is triggered, by which novel concepts are formed and then added to the current state (3).

that agents build internal concept spaces that encode both relational and functional knowledge, acquired through interaction with the environment.

AIGenC is presented in this chapter as a theoretical template, outlining the necessary functional components for creative generalisation—namely Concept Processing, Reflective Reasoning, and Blending. Importantly, this framework is not bound to a specific implementation; rather, it offers a model-theoretic architecture that outlines the minimal cognitive requirements for creative generalisation. While the functional roles of the components remain consistent, their realisation may differ across implementations depending on the affordances and observational richness of the environment.

Consequently, throughout this chapter we focus on the functional role of each component, deferring their specific computational instantiation to the subsequent chapters. Specifically, Chapter 5 (SETLE) provides the concrete algorithms for graph construction, encoding, and matching, while Chapter 6 details their integration into the reinforcement learning loop.

The framework comprises three core components, each corresponding to a distinct cognitive function essential for adaptive problem-solving:

1. **Concept Processing** is responsible for extracting and encoding objects, actions and affordances from sensory input, forming the foundational elements of the agent's internal concept space.

2. **Reflective Reasoning** enables the retrieval and comparison of stored conceptual structures from memory, aligning past experience with current task demands.

3. **Blending** facilitates the generation of novel concepts by creatively recombining previously learned elements when existing representations prove inadequate for solving new problems.

These components operate over a hierarchical concept space implemented as a graph-based structure. In this space, nodes represent learned concepts—such as objects, actions, or affordances—and edges encode relational, temporal, or causal dependencies among them. This structural representation supports flexible composition and abstraction, making it well-suited for transfer across diverse tasks and environments. Affordances and outcomes, which appear abstract at this template level, are formally defined with domain-specific examples in Sections 5.1.3 and 5.1.4 respectively.

Each high-level component includes finer-grained sub-processes. Concept Processing comprises two submodules: one for object discovery and another for action and affordance learning. Reflective Reasoning involves two key elements as well: the initialisation and structuring of Long-Term Memory (LTM), and a selective matching process that aligns current task representations with relevant prior knowledge. The interactions among these components and memory systems are illustrated in Figure 1.

The entire framework operates atop a goal-directed reinforcement learning substrate. The hierarchical concept space that underpins it leverages both implicit patterns—learned through distributed representations—and explicit relations derived from structural and functional interactions.

This chapter presents the architecture of AIGenC, describing each component and its interactions with the others. It also introduces the underlying dual-memory system, comprising a Working Memory (WM) and a Long-Term Memory (LTM). Working Memory holds the current context and encodes novelty, while Long-Term Memory accumulates concepts across tasks, enabling reuse and abstraction.

In the sections that follow, we describe each component in detail, beginning with Concept Processing, which encodes perceptual and functional information into a structured, internal concept space. This will form the basis for the subsequent chapters.

4.1 Concept processing component

A concept processing unit is needed to encode and add the input to the concept space. To separate sensory from dynamic information, we postulate two corresponding subunits, both unsupervised and pretrained.

Unsupervised models are used to ensure that the concept space is internal to the agent, in the sense that externally given classes do not predefine the discovered concepts, and to equip the agent with the ability to extract representations of objects or concepts regardless of whether it has experienced them before or not.

The first sub-unit aims to find and represent objects as latent vectors through unsupervised object discovery, while the second aims to encode action vectors. While the AIGenC template defines this as a generic perceptual role, our specific instantiation in Chapter 5 fulfils it using the Segment Anything Model; see Section 5.2.2).

The interaction of the two subunits and the environment is described

based on a modification of Şahin and collaborators' formalism (Şahin et al., 2007). Functionally, we define affordances here as an acquired relation between an *(effect, reward)* pair and a *(concept-object, action)* tuple. This relation is formally operationalised as the 'Affordance Transition Tuple' in Section 5.1.2, which details the mathematical structure of these connections. In this manner, when an agent applies an action to the object, an effect and associated reward pair is generated. The action representation and the effect will form the affordance (See Algorithm 1, lines 26-27).

The extracted representations are composed into graph-structured states, which form the input to the agent's working memory and ultimately drive policy learning and decision-making. As illustrated in Algorithm 1, this process establishes the foundational elements upon which Reflective Reasoning and Blending operate in later stages of the loop.

The functional role of the Concept Processing component is to ground the agent's reasoning in perceptual reality. Its purpose is to extract meaningful, generalisable representations from raw interaction data—transforming sensory inputs into the structured building blocks of the concept space. While we define its theoretical structure and cognitive motivation here, the specific neural architectures used to instantiate it, such as the object discovery and action encoding models, are detailed in Chapter 5.

Algorithm 1 AIGenC integrated within an RL setup

1: **Inputs:** environment $\mathcal{E}$, policy network π_θ, action set $\mathcal{A}$, number of outer iterations N
2: **Hyper-parameters:** memory init steps/phase, clustering method (e.g., KMeans), matching threshold τ (if applicable)
3: **Initialise LTM:** $ltm \leftarrow \langle keys[], \{\}\rangle$ ▷ *persistent long-term memory of abstracted graphs/centroids*
4: **Procedures (templates):**
5: ObjectDiscovery(*frame*) → *objects* ▷ *pre-trained; returns K object slots/encodings*
6: ActionEncoding(*interactions*) → *action_enc* ▷ *pre-trained; returns encodings of object-action interactions*
7: BuildStateGraph(*objects*, *action_enc*, s_t) → G_t ▷ *construct graph for state* s_t
8: Match(*mem*, *query*) → T/F ▷ *returns whether query already exists in mem*
9: EncodeState(s) → z_s ▷ *state encoder used for effect computation*
10: UpdateStateGraph($G_t, s_t, a_t, s_{t+1}, e_t$) → G_t ▷ *updates* G_t *with transition + effect*
11: Cluster(wm) → $\mathcal{C}$, Centroids($\mathcal{C}$) → $\{\mu_c\}$
12: **for** *iteration* $= 1, 2, \ldots, N$ **do**
13: **Initialise WM:** $wm \leftarrow \langle keys[], \{\}\rangle$ ▷ *working memory for this iteration*
14: Select $A_i \subseteq \mathcal{A}$ ▷ *e.g., random subset of actions/objects to explore*
15: **while** *episode not done* **do** ▷ *stop after fixed horizon or upon terminal reward*
16: Observe current state s_t
17: *objects* ← ObjectDiscovery(s_t)
18: *action_enc* ← ActionEncoding(A_i)
19: $G_t \leftarrow$ BuildStateGraph(*objects*, *action_enc*, s_t)
20: **if** *memory initialisation phase* **then**
21: $a_t \leftarrow$ Random($\mathcal{A}$)
22: **else**
23: $a_t \leftarrow \pi_\theta(G_t)$
24: Execute a_t in $\mathcal{E}$ and observe (s_{t+1}, r_t)
25: $z_t \leftarrow$ EncodeState(s_t), $z_{t+1} \leftarrow$ EncodeState(s_{t+1})
26: $e_t \leftarrow z_{t+1} - z_t$ ▷ *effect embedding captures action-induced change (representation-invariant transition cue)*
27: $G_t \leftarrow$ UpdateStateGraph($G_t, s_t, a_t, s_{t+1}, e_t$)
28: **if not** Match(wm, G_t) **then**
29: Add(wm, G_t) ▷ *store novel state graph in WM*
30: **else**
31: discard()
▷ *Post-episode / post-iteration consolidation: abstract WM and update LTM*
32: $\mathcal{C} \leftarrow$ Cluster(wm)
33: $\{\mu_c\} \leftarrow$ Centroids($\mathcal{C}$) ▷ *centroid representations (graphs or encodings)*
34: **if not** Match($ltm, \{\mu_c\}$) **then**
35: Add($ltm, \{\mu_c\}$) ▷ *store novel centroids/abstract graphs in LTM*
36: ▷ *Optimise policy network (details omitted here): update* θ *using transitions/graphs collected*

Algorithm 1 serves as the master orchestration of the AIGenC framework, integrating all three functional components into a unified learning cycle. It defines the complete operational flow: how concepts are extracted (Concept Processing, lines 5–6, with concrete implementations detailed in Section 5.2.2), how memory is managed (WM initialization line 13, LTM consolidation lines 32–35, instantiated in Section 5.2.3), how actions are selected via the policy network (lines 20–23, detailed in Section 6.1), and when specialized reasoning operations are invoked (Reflective Reasoning and Blending, whose detailed mechanisms are presented in the next subsections and fully integrated into the RL loop in Algorithm 5, Chapter 6).

Because Algorithm 1 represents the entire system, different sections of this thesis will revisit and explain specific portions of it from the perspective of each functional component. Section 4.1 focuses on Concept Processing's role (lines 16–27, 32–35); Section 4.2 details how Reflective Reasoning operates within the loop through selective matching; and Section 4.3 explains how Blending is conditionally activated during impasse situations. This layered explanation allows us to maintain the conceptual clarity of each component while showing how they integrate within the master algorithm.

In line 26 Algorithm 1, we calculate the 'effect' of an interaction as the vector difference between the resulting state and the current state ($z_{t+1} - z_t$). We use this difference formulation rather than the absolute final state because it isolates the transformation induced by the action from the specific environment configuration. By capturing the relative change, this representation is helpful in recognising the outcome across different contexts. For example, a 'push' action that moves an object forward produces a similar difference vector regardless of the object's initial coordinate. This invariance allows the agent to identify and transfer functional outcomes, such as displacing an obstacle or opening a door, even when they occur in novel locations or with different objects.

4.1.1 Concept space

The concept space functions as a dynamic, hierarchical structure that encodes the representations extracted during Concept Processing. It forms the substrate upon which Reflective Reasoning operates, supporting both similarity-based retrieval and compositional abstraction. Inspired by Olteteanu's cognitive formalism (Olteţeanu, 2020), the concept space is structured across three interconnected levels: a feature-level embedding space, a concept level, and a graph representation of an RL state-time configuration that defines the problem template (see Fig.2). The structure can be characterised as a graph that stores and relates information without a predefined data design and satisfies theoretical prerequisites such as being hierarchical and adaptive (Fauconnier and Turner, 1998; Gardenfors, 2004). Nodes encode entities such as objects, actions, affordances and outcomes, while edges capture their functional, temporal, or causal dependencies. Crucially, graphs enable dynamic adaptation—nodes and edges can be added, removed, or reconfigured without requiring a fixed schema, reflecting the evolving nature of agent experience. Higher-order concepts, such as states or outcomes, can be rep-

resented as subgraphs or macro-nodes themselves, enabling recursive reuse and transfer across tasks. Temporal structure is integrated through graph motifs that encode episodic transitions, allowing for fine-grained reasoning over sequences of events.

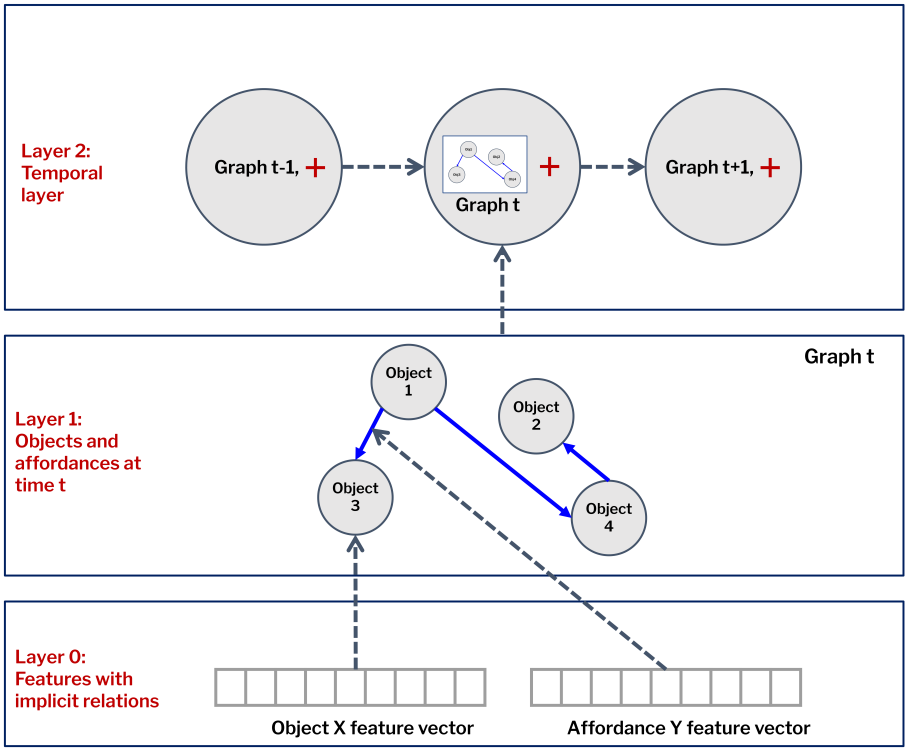


Figure 2: Hierarchical concept space: object concepts are represented by feature vectors at the bottom level (Layer 0). At the middle level (Layer 1), nodes and edges represent object concepts and affordances. At the top level (Layer 2), the graphs from the previous level become nodes along with a reward given to the agent in the RL setting (red +) in a graph whose edges mark temporal succesion (t).

Given the temporal and context-sensitive nature of creative problem solving, we represent the concept space as a dynamic graph stored in memory, using adjacency lists rather than fixed matrices to accommodate structural plasticity. This representation supports partial connectivity, episodic

binding, and continual expansion as the agent interacts with the environment. Unlike traditional memory architectures (e.g., Neural Turing Machines (Graves et al., 2014)), which operate on flat matrix-based memory banks, our graph-based memory enables rich topological encoding of experience, capturing not only what the agent has encountered but how elements relate and evolve over time. This hierarchical organisation of the concept space is concretely instantiated in Section 5.2.3, where we describe the three-level graph structure: Level 1 (objects and interactions), Level 2 (affordances and states), and Level 3 (SETs/trajectories).

4.1.2 Memory system

Our framework integrates two interacting memory components: a Working Memory (WM) and a Long-Term Memory (LTM). These memories operate at different temporal scales and interact continuously throughout the lifetime of the RL agent, enabling both short-term reasoning and long-term knowledge accumulation.

As introduced in the previous subsection, the content of both memories,the concept space, is represented using structured relational abstractions. Concretely, the concept space is assembled in the form of two interrelated lists (Fig. 3). First, an object list stores object representations extracted persistently by the Concept Processing component. Second, a hash map structure maps each object or concept to a set of related nodes, where each mapping encodes a relation (e.g., an affordance) between concepts.

This abstract representation corresponds directly to the graph-based memory structures used in Algorithm 1. In particular, the hash map structure depicted in Fig. 3 is instantiated as the data structure underlying both WM and LTM, with each key representing a node in the concept graph and each value representing a set of outgoing relational edges.

While such mappings could be maintained using conventional data structures, graph databases such as Neo4j (Lal, 2015) provide a more scalable and expressive implementation. Rather than relying on dense adjacency matrices, Neo4j represents graphs using adjacency lists and indexed node lookups, a model that naturally aligns with the object–relation structure shown in Fig. 3. Each concept node stores explicit references to its connected nodes and relations, enabling efficient traversal and modification.

This representation offers several computational advantages that are critical in an RL setting. It supports constant-time access to related concepts,

efficient insertion of new nodes and relations, and flexible subgraph queries. These operations are required as the agent incrementally constructs and updates state graphs during interaction with the environment. As demonstrated in later chapters, this structure allows the agent to compare and generalize experiences based on structural similarity rather than raw state identity.

The interaction between WM and LTM is explicitly realized in Algorithm 1. During each episode, the agent constructs a state graph G_t from object and action encodings (Algorithm 1, lines 18–20). If this graph is structurally novel with respect to the current contents of WM, it is added to WM (line 27). WM therefore acts as a temporary buffer of distinct state graphs encountered within an episode, supporting short-term reasoning and decision-making.

Once an episode terminates, either due to task completion or a predefined time limit, the contents of WM are abstracted through clustering. Specifically, representative centroid graphs are computed and used as compressed summaries of the episode's experiences (Algorithm 1, lines 34–38). These centroid graphs, together with their associated concepts and relations, are then stored in LTM if they are not already present. Unlike WM, LTM persists across episodes and serves as a repository of accumulated conceptual knowledge, enabling transfer of learning over the agent's lifetime (Loynd et al., 2020; Mezghan et al., 2022).

The operational interaction between these memory components corresponds directly to the information flow illustrated in Fig. 4. Reflective Reasoning is implemented through matching operations between the current state graph and the contents of LTM, allowing relevant past concepts to be retrieved and used to enrich the current representation. When repeated failures occur, the Blending component retrieves and combines concepts from LTM to construct novel state graphs, which are then reintegrated into WM and evaluated by the policy. Through this cycle, the agent incrementally refines its internal concept space while maintaining a clear separation between transient working representations and persistent long-term knowledge.

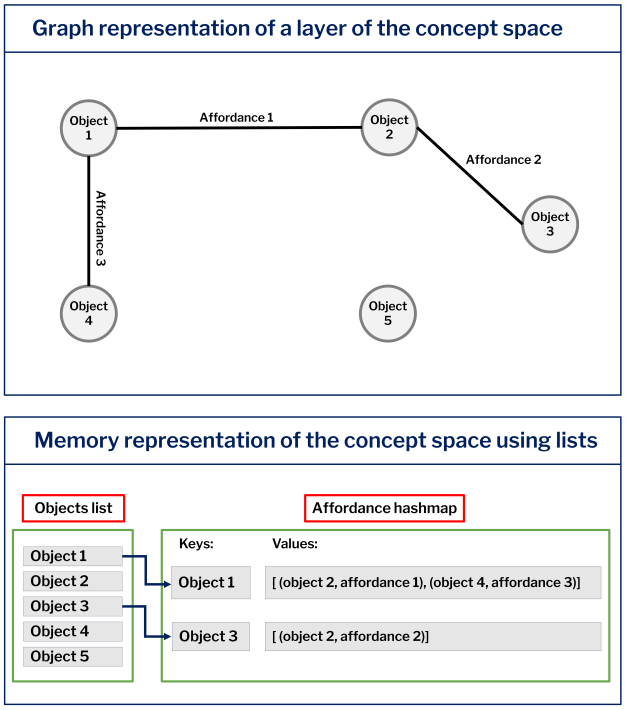


Figure 3: The top panel shows the concept space represented as a mathematical structure, a graph. The bottom panel displays how graphs are coded in memory, using two lists: an object list and a hash map of other nodes (i.e., other objects). The keys from the hashmap represent the objects (e.g., Object 1), and the values are tuples indicating the connected node (e.g., object 2) and the edge between two nodes, which in this case is an affordance relation (e.g., affordance 1).

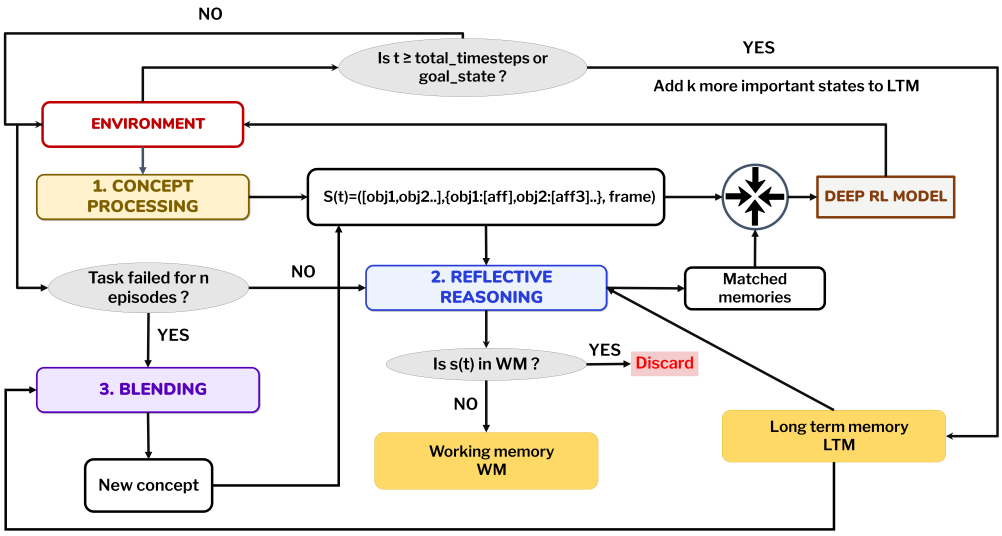


Figure 4: A snapshot of the flow of information through the system. The current state, encoded by the Concept Processing component, is first compared to the concepts stored in Long Term Memory at the Reflective Reasoning component. The concepts that match are used to expand the current state graph. The enhanced current state is then passed to the Deep RL model (detailed in Section 6) and applied to the environment. A condition set on the grey top ellipse determines whether the loop is repeated; if the condition is not met, the k most important states from Working Memory are added to Long Term Memory. At the end of a loop, the current state is added to Working Memory if it is different from its current content (bottom ellipse). If several sequential episodes end in failure, Blending is used to enhance the current state with new concepts created by retrieving and combining concepts from LTM (left ellipse).

4.2 Reflective reasoning component

The Reflective Reasoning component defines a procedure to choose which concepts from WM are to be permanently stored in LTM and incorporates a matching operation between the current state and LTM information (see Algorithm 2, line: 12). Matching allows selecting the concepts useful to fulfil the specific task at any given time point.

Processing in the Reflective Reasoning component is executed sequen-

tially (Fig.1). First, LTM is initialised; then, selective matching occurs (Algorithm 2, line: 12-14). The agent first explores the environment randomly (using a random action selection policy, Algorithm 1, line: 21), learning new concepts at each state and storing the most representative ones in LTM. The interplay of the two memory systems enables a process of matching novel states to past information (Doumas et al., 2022). Concepts that reach long-term memory (LTM) are selectively retrieved to enhance the agent's ability to choose an action that leads to a state closer to the goal. This selective matching operation is carried out in the Selective Matching unit (Fig. 2b). By storing structured representations of concepts in LTM, the agent can access past experiences and use them to inform current decisions, allowing adaptation to new tasks and environments.

Matching, however, does not convey sameness retrieval. Creative agents must adapt previous concepts to unfamiliar states by connecting relational concepts to new objects (Shanahan and Mitchell, 2022). The hierarchical structure of the concept space allows independent access to each level of abstraction, making this type of adaptation possible. Nonetheless, matching also requires a comparison process to assess similarity by measuring the distance between the elements involved, such as graph-based concepts. Rather than relying on shallow graph comparison techniques such as adjacency matrix overlap, which fail to account for semantic and functional variability, our approach encodes the graphs into vector representations that preserve relational and topological nuance. These learned embeddings are then compared using distance metrics to guide matching and enrichment, as will be elaborated in the subsequent chapters.

After the initialisation of LTM through agent exploration, selective matching on the initialised memory can be done using a trainable policy network for action selection (the policy network architecture and its integration with memory-based enrichment are detailed in Section 6.1 and Section 6.2 respectively). This shifts the agent's focus towards exploitation,where the goal is achieved by using objects in the environment or building new ones.

Algorithm 2 Instantiation of UpdateStateGraph via selective matching and supplementation

```
1: Input: current state graph G_t, long-term memory ltm, current state s_t
2:                          ▷ This algorithm expands the UpdateStateGraph operation in Algorithm 1
3:                                  ▷ It refines the state graph before action selection using LTM
4: selective–matching(G_t, ltm, s_t)                          ▷ Template operation from Algorithm 1
5:     returns a set of LTM subgraphs similar to G_t using graph–matching
6: supplement(G_1, G_2)                                       ▷ Template operation from Algorithm 1
7:     returns the graph union used to complete missing objects and affordances
8: for iteration = 1, 2, ..., N do
9:     Init wm ← ⟨keys[], {}⟩                                 ▷ As in Algorithm 1
10:    while episode not done do
11:        ...                                    ▷ State graph construction as in Algorithm 1
12:                                               ▷ Begin expansion of UpdateStateGraph
13:        matched–graphs ← selective–matching(G_t, ltm, s_t)
14:        enhanced–graph ← supplement(G_t, matched–graphs)
15:                                                 ▷ End expansion of UpdateStateGraph
16:        a_{i,t} ← π(enhanced–graph)                  ▷ Policy now acts on enriched graph
17:        ...                  ▷ Environment interaction and memory updates as in Algorithm 1
18:    Optimise policy network                                ▷ As in Algorithm 1
```

Applying a concept from memory to a given state entails two steps Fig.5: first, a match of the graph representing the state (e.g., G_t) to the graphs representing the long-term stored information (e.g., LTM); second, once the match is successful, G_t is supplemented with the objects and affordances present in the retrieved *LTM subgraph* but lacking in G_t in a process known as completion Shanahan and Mitchell (2022) (Algorithm 2, line: 13). Such completion will foster learning and bring to the agent's current state useful past experiences.

So far, we have demonstrated how an agent can solve a problem by utilising various existing concepts. The following component describes creating an entirely new concept by leveraging the existing concept space, considered (everyday) creativity.

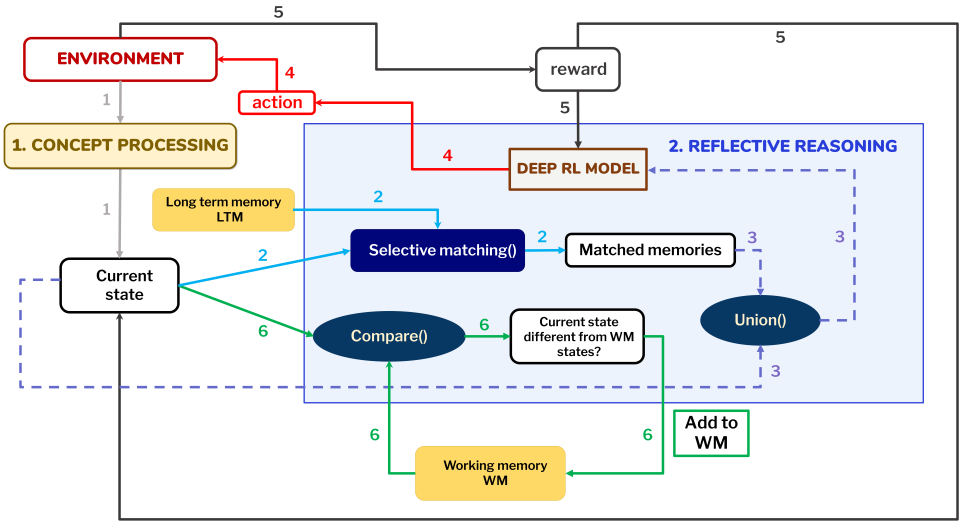


Figure 5: This figure provides a focused view of the Reflective Reasoning component, acting as a zoom-in of the memory interaction already introduced in Fig. 1. The processing is illustrated through numbered coloured arrows. First, the input is encoded and represented as a graph (1). The processed current state is passed to the memory matching function (2), which outputs the best matches between the current state and Long Term Memory concepts. The concepts retrieved represent the union of the best matches and supplement the current state (3), which is input into a deep RL model (detailed in Section 6). The deep RL model outputs an action which is applied to the environment (4). The deep RL model is updated with a reward and the current state becomes associated with it (5). The updated state is added to Working Memory if it is significantly different (6). The whole process runs in a loop until a final state is reached or a given number of time-steps have passed.

4.3 Blending Component

Creativity is triggered when existing concepts are insufficient to solve a task, leading an agent to a standstill. We are using the expression *impasse situation* (Laird et al., 2012) to refer to the inability of an agent to solve a task for several episodes. A CPS approach should help overcome such an impasse with a solution that satisfies the problem constraints. Hence, we are working

under the assumption that the impasse could be surmounted by developing a novel, useful concept. In the AIGenC framework, new concepts are generated by blending existing concepts in the latent space into new representations. Conceptual blending (Fauconnier and Turner, 1998) denotes the combination of meaningful features of two or more concepts into a new concept.

Two issues must be addressed when creating a new concept: selecting the relevant concepts that help the agent achieve a goal and combining them effectively. To filter the concept space for information that can solve a problem, the agent must have a high-level understanding of the problem, context, and task requirements; that is, acquire a general problem template to process the information, which requires summarising and organising information at a level of abstraction beyond the current hierarchy. Artificial agents lack this knowledge, so we propose a heuristic that widens the range of matched concepts by relaxing the similarity constraint in matching (i.e., we propose using a similarity criterion (say, X%) for selecting concepts to be blended, acknowledging that other approaches may also be appropriate). When Reflective Reasoning finds the concepts that lead to a satisfactory solution, the problem-solving task stops; otherwise, Blending is activated, making the two functionally complementary (see Fig.6). Both components match information from LTM to the current state. However, Blending retrieves a larger pool of information, allowing for a wider variety of environmental data to be operated on.

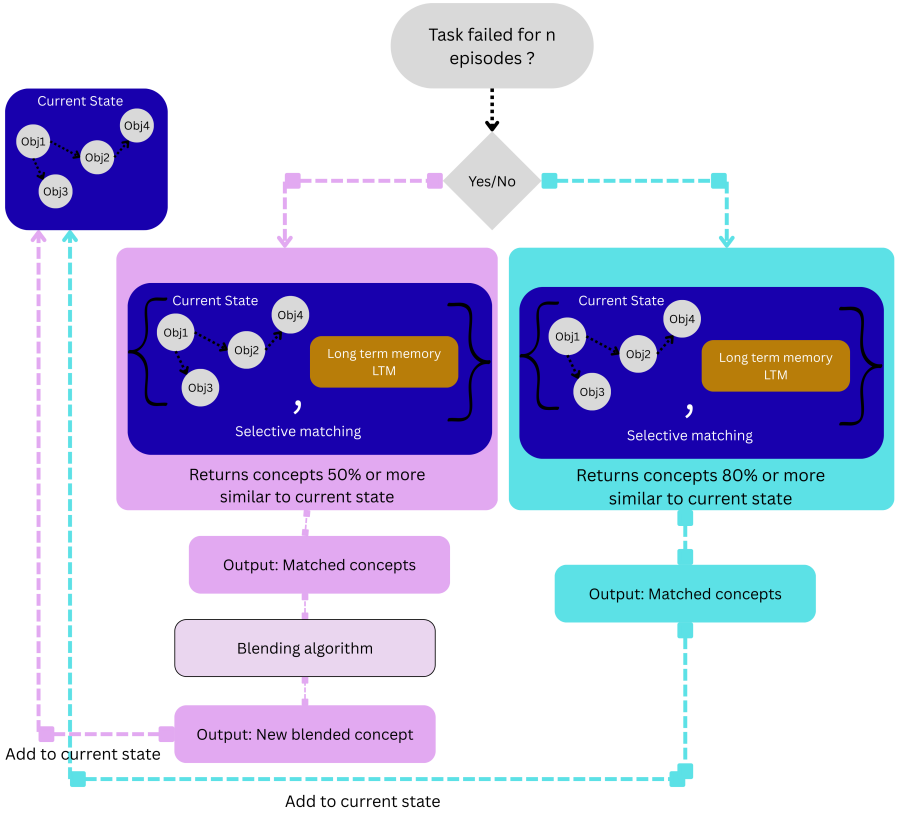


Figure 6: Reflective Reasoning and Blending complementary pipeline. Failure in solving a task on multiple episodes (grey top ellipse) activates Blending (left elements connected by pink lines); otherwise, the processing is carried out in Reflective Reasoning (right elements connected by blue lines). A process of Selective Matching between the current state and Long Term Memory follows. Reflective reasoning selects concepts with a high similarity ratio and applies a union function to supplement the current state graphs. Blending selects concepts with a lower similarity rate. A Blending algorithm is run. The outcome of this algorithm is applied to the current state.

Retrieved concepts are used to create novel concepts using a non-linear, trainable network, thereby expanding the concept space with new, diverse concepts that can be applied to the task. The network must respect the latent space's structure, meaning that the dimensions of the latent vectors should be maintained, and the network should be able to combine their feature

values by moving across their dimensions. Relevant features extracted by concept processing sub-modules can be identified post hoc by quantifying their contribution to the performance of the previous unsupervised models. SHAP (Shapley Additive Explanations, (Lundberg and Lee, 2017)) is an interpretability method that can be used to weigh the pertinence of individual features and select the latent space features that bear the most relevance to solving the task.

To keep the semantic features of data through the generation process, we propose decomposing the input into a vector and a latent code that targets the salient semantic features of the data distribution. This process is similar to Chen et al.'s approach Chen et al. (2016). Thus, the input vector would be replaced by the concepts to be merged with a set of trainable parameters for their combination function. In addition, the latent codes could be initialised with an average of the SHAP values. As the Blending function is intended to create novel concepts rather than reproduce input in the latent space, a clustering-specific loss function could be used to evaluate the similarity between the generated concept and existing concepts in the cluster, with a low value indicating a good match. By enforcing clustering similarity through the loss function, we could reduce the variability of the new concept, providing a mechanism for more meaningful concept building (Mukherjee et al., 2019).

It is essential to note that the proposed blending mechanism would only be possible in a system that incorporates functional components like those presented in our model. The underlying representational structure of the first component is critical to obtaining a comprehensive input representation. At the same time, Reflective Reasoning tools are essential for selecting, adding, and filtering concepts in the concept space, reducing computational costs and recycling useful concepts. Filtering, which involves understanding the high-level characteristics of the concept affordance, is not simply a search of the conceptual space. Instead, it requires a generalisable solution template, developed through repeated similar experiences. We anticipate that our agent's semi-random selection of relevant objects will establish a foundation for incremental abstract representations.

While the broader AIGenC framework initially relies on this semi-random retrieval, its instantiation through SETLE moves toward a more structured and informed mechanism. Central to this refinement is the concept of a problem template—an abstract pattern that emerges from generalising across previously encountered episodes and supports the selection and recombination of relevant knowledge components. Although agents do not begin with ac-

cess to such templates, the construction of Structurally Enriched Trajectories (SETs) and their outcome-based representations provides a concrete foundation for approximating them. By encoding temporal sequences, functional dependencies, and relational affordances, SETLE equips the agent to retrieve and compose partial solutions in a manner that mirrors template-based reasoning—facilitating more adaptive, transferable, and ultimately creative behaviour in novel tasks.

With Blending, the complete architecture has been presented. The architecture proposed serves as a blueprint for designing more robust systems equipped with transferable and adaptable knowledge capable of solving problems creatively.

While this section has outlined the conceptual role of Blending within the AIGenC framework, namely, the retrieval of loosely similar concepts and their recombination to generate novel representations, its concrete algorithmic realisation is deferred to Chapter 6. Specifically, Algorithm 5 (lines 18–24) implements the Blending mechanism through attention-weighted retrieval and graph supplementation, demonstrating how retrieved concepts are selected, scored, and injected into the agent's working memory during the reinforcement learning loop. However, it does not yet perform the full latent-space combination described above—where features from multiple concepts are interpolated or merged via a trainable network (e.g., using SHAP-weighted feature selection and InfoGAN-style generation).

This simplification reflects practical constraints: the CREATE and MiniGrid environments used in our experiments do not allow evaluation of—genuinely novel concept generation. Testing the full Blending mechanism would require environments specifically designed to reward creative solutions that cannot be achieved through retrieval alone—a direction we identify for future research.

5 AIGenC Concept Processing and Reflective Reasoning Instantiation: SETLE (Structurally Enriched Trajectory Learning and Encoding)

To instantiate the Reflective Reasoning component of the AIGenC framework, this chapter introduces a concrete computational model, namely, Structured Trajectory Learning and Encoding (SETLE). SETLE addresses a key representational bottleneck in reinforcement learning: the inability to encode and retrieve structured, relational knowledge from temporally extended interactions. This limitation hinders agents from abstracting past experience, thereby reducing their capacity to generalise or adapt creatively to novel tasks.

Rather than representing experience as flat sequences of state-action transitions, SETLE conceptualises episodes as hierarchically structured trajectories, called Structurally Enriched Trajectories (SETs), that encode objects, their interactions, and the affordances that mediate functional change over time. These hierarchical graphs preserve both the compositional nature of tasks and the dynamic dependencies between agent actions and environmental transformations. SETs are constructed incrementally during training and stored in a dual-memory system as described in the previous chapter, which enables retrieval, matching, and recombination across episodes. We detail how SETs are built from raw data, how their relational and temporal structure is encoded within heterogeneous graph memory, and how graph neural networks and attention-based mechanisms are used to compare current states with structurally similar past experiences. This includes SETLE's use of meta-path-aware semantics, outcome-based linking, and encoding strategies that allow the agent to identify functionally relevant substructures from its long-term memory.

Unlike traditional methods that aggregate information across entire graphs, we introduce a subgraph-centric approach tailored for episodic data. Each episode is treated as an independent subgraph, and its representation is solely based on its local neighbourhood. A key distinction of our framework is the ability to handle shared nodes that may appear in multiple trajectories or episodes. These shared nodes serve as bridges between episodes, allowing the emergence of cross-episode patterns. For example:

- An object, such as a ball, might be used across different episodes but in varying contexts (e.g., being pushed to a goal in one task and placed in a bucket in another).

- Similarly, an action like `push` could have different effects depending on the affordances and states in multiple episodes.

By encoding these shared nodes within their respective trajectory subgraphs, the relationships learnt from one trajectory or episode can inform the understanding of similar interactions in other episodes, enhancing generalisation across tasks.

By grounding abstraction in interaction, SETLE provides a principled mechanism for structured generalisation, transforming episodic traces into reusable, compositional knowledge. This not only enables behavioural reuse in new settings, but also supports the gradual emergence of task schemas or problem templates from accumulated experience.

We first define the ontology of our framework, then describe the process of constructing Structurally Enriched Trajectories (SETs) as heterogeneous graphs, ensuring they encode multi-level relationships between objects, interactions, states, and affordances. We then detail the methodology for learning SET embeddings, which can be used for downstream tasks, including integration into reinforcement learning pipelines.

Summarising, this chapter covers:

1. A conceptual framework for RL representations that extends RL's ontology to incorporate affordances and hierarchical trajectory structures, enabling agents to capture sequences of different states and their relational dependencies.

2. A formal conceptualisation of SET to expand the notion of trajectories to include not only state-action sequences but also objects, interactions, affordances, and task-relevant components.

3. A novel architecture SETLE to extract and encode SETs

   (a) An instantiation of the Hierarchical Memory Structure described in the previous chapter: A multi-level memory system that encodes task elements at varying levels of abstraction, capturing both fine-grained interactions and high-level trajectory patterns, supporting robust generalisation across tasks.

(b) Heterogeneous Sub-Graph Encoder: A specialised encoder designed to learn trajectory representations by leveraging subgraphs with shared components rather than treating full graphs as independent entities.

4. Experimental evidence suggesting that SETLE effectively captures both task-specific patterns and cross-task similarities, supporting structured trajectory learning in dynamic, episodic environments.

5.1 SETLE Ontology

This section defines key components of our framework, including interactions, Structurally Enriched Trajectories (SETs), affordances, and hierarchical memory, which together provide a structured representation of task execution. Although several of these concepts have been introduced in earlier sections, they are revisited here in the context of their integration within the full system instantiation, illustrating how they interoperate to support generalisation, abstraction, and decision-making.

5.1.1 Interactions and Objects

Objects and interactions are structured embeddings extracted from the environment that encode task-relevant features that influence decision-making. An interaction i_t represents the interaction between an agent-object or object-object relationship observed within the environment. However, the nature of these interactions varies across the environments used in our experiments. In CREATE, where agents and objects are embedded in a physics-based simulation, agent-object interactions can be explicitly observed and encoded, for example, when a ball pushes a block or navigates around obstacles. These interactions provide rich relational information that informs the agent's understanding of affordances and causal effects. In contrast, environments like MiniGrid do not explicitly expose physical interactions. Instead, transitions are primarily recorded as state-action changes, without observable agent-object contact, which limits the granularity of relational information. As a result, in MiniGrid, interaction nodes are omitted, and the graph representation focuses on object embeddings and action-induced state transitions instead.

5.1.2 Affordances

As described in the previous chapter, an affordance is conceptualised as the learnt relationships between an (effect, reward) pair and a (concept-object, interaction) tuple. When an agent applies an action, it produces an interaction, which in turn generates an effect (a state transition) and an associated reward, both of which are stored as an affordance. This structured representation is designed to improve generalisation across tasks by predicting how different interactions influence trajectories across varying contexts.

Formally, an affordance A encapsulates an agent's action, the current state, and the resulting trajectory (Catarau-Cotutiu et al., 2023):

$$A = \{s_t, a_t, s_{t+1}, r_t\} \tag{15}$$

where s_t is the current state, a_t is the agent's action, s_{t+1} is the resulting state, and r_t is the reward at time t.

5.1.3 Structurally Enriched Trajectories (SETs)

In reinforcement learning (RL), formally, we define the basic unit of interaction as a transition Tuple. This tuple captures the immediate causal link between a state, an action, and its outcome:

$$\mathcal{T} = s_t, a_t, s_{t+1}, r_t \tag{16}$$

where s_t is the current state, a_t is the action taken, s_{t+1} is the resulting state, and r_t is the immediate reward. A trajectory τ, then, is defined as a temporal sequence of these transition tuples, representing a complete episode of interaction:

$$\tau = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_T) \tag{17}$$

where $s_t \in S$ represents the agent's state at time step t, $a_t \in A$ is the action taken, and T denotes the length of the episode.

We define a **Structurally Enriched Trajectory (SET)** as an extension of this classical trajectory that incorporates additional structural components, capturing relational dependencies and affordance-based interactions. A SET is represented as a hierarchical graph:

$$\mathcal{G}_\tau = (V, E, \Phi) \tag{18}$$

where V is the set of nodes, E is the set of edges, and Φ assigns feature embeddings to nodes.

$V = \bigcup_{t=0}^{T} \{O_t, I_t, \hat{f}_t, s_t\}$ is the set of nodes across the episode. For each time step t:

- s_t is the state node at time step t.
- O_t is the set of object nodes present at time step t.
- I_t is the set of interaction nodes occurring at time step t.
- $\hat{f}_t$ is an affordance–outcome node associated with the transition from s_t to s_{t+1}. It represents the action-conditioned functional change observed in the environment and includes an outcome/effect term, e.g.

$$\Delta z_t = z_{t+1} - z_t, \tag{19}$$

  where $z_t = \mathrm{Enc}(s_t)$ is an encoded representation of the state (or observation).

The union over t ensures that V contains the complete set of states, objects, interactions, and affordance–outcome nodes encountered throughout the episode.

$E = E_{\text{temporal}} \cup E_{\text{structural}}$ is the set of edges, defined as the union of temporal and structural relations.

$E_{\text{temporal}} = \bigcup_{t=0}^{T-1} \{(s_t, \hat{f}_t), (\hat{f}_t, s_{t+1})\}$ encodes the temporal (causal) flow of the episode by linking each state to the affordance–outcome node and then to the subsequent state. This models the transition as a path $s_t \to \hat{f}_t \to s_{t+1}$, while keeping the outcome explicitly available as part of $\hat{f}_t$.

$E_{\text{structural}}$ encodes within-timestep functional dependencies:

$$E_{\text{structural}} = \bigcup_{t=0}^{T} \Big(\{(o, i) \mid o \in O_t,\, i \in I_t\} \cup \{(i, \hat{f}_t) \mid i \in I_t\} \Big). \tag{20}$$

That is, objects are linked to the interactions they participate in, and interactions are linked to the affordance–outcome they enable.

$\Phi : V \to \mathbb{R}^d$ assigns embeddings to nodes (states, objects, interactions, affordance–outcome nodes). In particular, $\Phi(\hat{f}_t)$ may concatenate or otherwise combine interaction features with the outcome embedding Δz_t, so that affordances capture both *what was done* and *what changed*.

Example. Consider an agent pushing a ball. At time t, the graph contains s_t, an object node o_{ball}, an interaction node i_{push}, and an affordance–outcome node $\hat{f}_t$ whose outcome component encodes the observed change (e.g., the ball's displacement), along with the next state s_{t+1}. The structural relation is encoded as $o_{\text{ball}} \to i_{\text{push}} \to \hat{f}_t$, while the temporal transition is encoded as $s_t \to \hat{f}_t \to s_{t+1}$.

This enriched representation allows us to model task execution beyond stepwise transitions, capturing how task elements evolve over time and interact within different contexts.

5.1.4 Hierarchical Memory Structure

We organise multiple Structurally Enriched Trajectories (SETs) within a hierarchical memory structure, which serves as a concrete instantiation of the concept space introduced in the previous chapter. The hierarchical memory structure, represented as a large heterogeneous graph, captures shared trajectory components across different episodes. The hierarchical memory is formally represented as:

$$\mathcal{M} = (\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N) \tag{21}$$

Where:

- $\mathcal{G}_i$ represents an individual SET subgraph, modelling a structured trajectory of an episode.

- Each subgraph $\mathcal{G}_i$ can be further decomposed into a sequence of lower-level graphs $\mathcal{G}'_i$, where each $\mathcal{G}'_i$ corresponds to a state in the episode, decomposed into its fundamental components (objects and interactions).

- SETs share low-level components, such as objects and interactions, creating relational dependencies across multiple trajectories.

The hierarchical structure of the memory captures different levels of abstraction. At the lowest level, each state is represented as a graph $\mathcal{G}'_i$, encoding the relationships between objects and their interactions. At a higher

level, SETs encapsulate structured trajectories, linking sequences of state-graphs through affordances into meaningful task representations. Unlike traditional memory structures that store isolated trajectories, our hierarchical memory structure connects trajectories through shared elements, preserving both task-specific dependencies and cross-task relationships. This structured representation allows agents to retrieve relevant experiences based on shared trajectory components, improving adaptability and transfer in reinforcement learning environments.

To formally represent the hierarchical graph for a SET, we model the task environment as a Heterogeneous Information Network (HIN), defined as a graph $G = (V, E, A, R, \Phi)$, where:

- V is the set of nodes (e.g., states, objects, interactions, affordances);
- $E \subseteq V \times V$ is the set of edges connecting nodes;
- $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ is the set of node types (e.g., `State`, `Object`, `Interaction`, `Affordance`);
- $\mathcal{R} = \{R_1, R_2, \ldots, R_m\}$ is the set of relation types, where each R_i defines a semantic relationship between node types (e.g., `has_object`, `participates_in`, `causes`, `temporal_next`);
- $\Phi : V \to \mathbb{R}^d$ assigns feature embeddings to nodes.

The network schema of G, denoted as $\Gamma = (A, R)$, is used to describe the structure of interactions between nodes of different types, capturing the local structure. The schema is used to define direct connections among various node types (such as states, interactions, affordances and trajectories), forming the foundation for encoding relationships in heterogeneous networks.

Additionally, we define *meta-paths* to capture the composite relationships within task episodes. A meta-path P is defined as a sequence $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_i} A_{i+1}$, where each A_i is a node type and each R_i is a relation. Meta-paths describe the indirect, higher-order connections between node types, such as sequences connecting states to the enriched trajectories, which are essential for capturing complex dependencies across task hierarchies.

5.1.5 Structured Data Source: CREATE Environment

To develop and evaluate our proposed framework, we required environments that support structured, temporally extended interactions and diverse task configurations. This necessitated platforms with multiple manipulable elements, variable goals, and the potential for multi-step planning—characteristics essential for studying generalisation, hierarchical reasoning, and creative adaptation. Our primary environment, CREATE (Construction, Reuse, and Extension of Actions through Transfer and Exploration) (Jain et al., 2020), meets these criteria. It provides a rich set of tasks involving dynamic object interactions—such as pushing a ball to a target, placing it in a container, or navigating around obstacles—allowing us to capture affordances and encode functionally diverse solutions. The variability in task configurations across episodes in CREATE forces agents to adapt strategies and recombine prior knowledge, making it particularly suited for testing creativity-inspired mechanisms like concept blending and structured reuse.

To complement this setup, we also incorporate the MiniGrid environment as a secondary testbed. MiniGrid offers lightweight, partially observable grid-based tasks that facilitate efficient data collection and evaluation of generalisation. While its state transitions and task diversity enable broad comparisons, MiniGrid lacks explicit physical interactions and continuous affordances, limiting its utility for developing the full creative potential of agents. Nevertheless, its inclusion allows us to assess the robustness of our framework across discrete and continuous settings and to evaluate the consistency of learned representations in environments with fundamentally different structural properties.

Together, CREATE and MiniGrid provide a dual testing ground: CREATE supports the emergence of structured, creative behaviour, while MiniGrid enables cross-environment evaluation of generalisation capabilities.

5.2 SETLE: Hierarchical Graph Construction

From template to instantiation. In Chapter 4, AIGenC was introduced at a template level, where the core operations object discovery, interaction encoding, affordance construction, memory matching, and graph supplementation were defined abstractly to foreground the architecture. In this section, we make that template concrete: SETLE instantiates these operations to construct *Structurally Enriched Trajectories (SETs)* that drive Reflective Rea-

soning.

More in particular, SETLE integrates the outputs of the Concept Processing component, transforming perceptual observations (objects and interactions) into graph nodes, and infers affordances as functional transformations that link successive states and bind them to the involved objects/interactions. Through this process, SETLE preserves both local relationships (e.g., agent-object interactions within a timestep) and longer-range patterns (e.g., action–effect sequences across time), grounding abstraction in observed experience. This structured representation forms the foundation for memory storage, retrieval, and downstream reasoning.

In the remainder of this section, we detail the graph construction operations, including object and interaction encoding, state transitions, and the integration of affordance-driven relations, all of which emerge from and build upon the framework defined by Concept Processing.

5.2.1 SET Definition Through Random Exploration

In RL, an episode consists of a complete sequence of interactions from an initial state to termination, encapsulating the trajectory an agent follows within a given task. To extract and formalise SETs (for a detailed definition, see Section 5.1.3) we employ an exploration strategy within the CREATE environment (Jain et al., 2020). In this setup, agents engage with tasks such as pushing a ball to a goal or placing it inside a bucket, generating diverse trajectories. To maximise variability in the trajectory space and better capture adaptive patterns and task-relevant structures, RL agents operate under a random policy rather than being optimised for specific rewards.

Each episode is represented as a SET graph (see Fig.13), encapsulating the task setup, actions taken, and the result of the task, which is stored for validation and further analysis in the hierarchical memory (see 5.1.4). This process directly instantiates the Concept Processing stage of the AIGenC framework (see Algorithm 1, Lines 4–6), where concepts are extracted from raw observations and structured into graphs. The resulting SETs are subsequently stored in Working and Long-Term Memory (Algorithm 1, Lines 7–8).

An episode E is defined as a sequence of state-action pairs:

$$E = \{(s_0, a_0), (s_1, a_1), \ldots, (s_t, a_t), \ldots (s_T, a_T)\} \tag{22}$$

where $s_t \in S$ is the state at time t, and $a_t \in A$ is the action taken at time t.

A SET is labelled as **success** if the agent reaches the goal; otherwise, it is labelled as **failure**. The SET of an episode $SET(E)$ is labeled as follows:

$$SET(E) = \begin{cases} \text{success}, & \text{if } s_T \in G \\ \text{failure}, & \text{otherwise} \end{cases} \tag{23}$$

where G is the set of goal states.

SETs are represented hierarchically (see Section 5.2.3) in the memory structure, including the episode's inventory and action sequence. For each episode, the corresponding SET graph G_{SET} is constructed, consisting of nodes representing states and edges representing state transitions (i.e., as seen in Fig.13).

5.2.2 Evaluating Methods for Concept Processing

The following section details the experimental methodology and evaluation criteria used to assess candidate models for the object discovery and interaction learning components of the Concept Processing module. This module plays a critical functional role by extracting transferable latent representations of objects and interactions, forming the perceptual foundation of the agent's internal concept space. These representations must be disentangled from raw sensory input and be capable of generalising across diverse environments. To meet these requirements, we focus on unsupervised or weakly supervised approaches that learn concepts from structural and statistical regularities in the data, rather than relying on predefined labels or dense supervision.

Object discovery: Methodological Considerations We adopt the premise that learned object concepts should be unsupervised, modular, and spatially disentangled. This motivates the use of representation learning methods such as autoencoders (AEs) (Rumelhart and Norman, 1983; Masci et al., 2011) and variational autoencoders (VAEs) (Kingma et al., 2013), which compress high-dimensional inputs into compact latent representations. However, standard VAEs treat the input image as a whole and typically learn entangled latent representations, which do not explicitly separate different objects or regions within the image. As a result, they lack the inductive bias needed for object compositionality, the ability to represent multiple, distinct

entities within a scene. To address this, we explored object-centric models specifically designed to decompose scenes into independent object-level components, thereby enabling more interpretable and transferable concept representations.

To filter the extracted object vectors and reduce computational overhead, we constrain the number of object slots to a fixed K, selecting only the most salient components from a total of N latent encodings. This reduces noise, limits memory bloat, and focuses attention on conceptually meaningful entities.

Object discovery: Model Selection Process To identify a suitable object discovery model for our framework, we trained and evaluated several existing approaches, focusing on their transferability, robustness across environments, and dependence on fine-tuning. All models were initially pretrained on standard object-centric datasets, and we systematically assessed their ability to generalise to novel RL environments without additional supervision.

Unsupervised object discovery requires identifying recurring patterns in visual input that can be interpreted as discrete entities, distinguishing objects from the background and one another. A range of popular methods tackle this challenge by learning to decompose visual scenes into interpretable components. These include approaches based on autoencoders, such as MoNet (Burgess et al., 2019), multi-object representation learning using VAEs, and the Slot Attention mechanism.

Our evaluation began with the Slot Attention module (Locatello et al., 2020), which uses an iterative attention process to assign parts of the latent representation to a fixed number of K object slots. Each slot is intended to capture a distinct object in the scene. This model has shown strong results in synthetic and structured environments and is often used as a benchmark in object-centric learning. This model has demonstrated strong performance in object-centric learning on structured datasets, such as CLEVR (Johnson et al., 2017). In our initial experiments, conducted on CLEVR, a custom 2D-shapes dataset, and two RL environments (CREATE and AnimalAI (Crosby et al., 2020)), Slot Attention demonstrated strong in-distribution performance. Still, it failed to generalise to more complex or noisy domains. Specifically, in CREATE and AnimalAI, the model struggled to produce stable segmentations and reconstructions, resulting in slot drift and low-quality

object representations even after fine-tuning (see Appendix 4.1).

Given these limitations, we next explored DTI-Sprites (Monnier et al., 2021), a model specifically designed to decompose images into object-like layers using learned appearance and motion cues. Unlike Slot Attention, which requires general-purpose pretraining, DTI-Sprites was trained directly on AnimalAI frames. This yielded more stable segmentations within the trained environment (Fig. 7, 8), with object layers that were better aligned with perceptual boundaries.

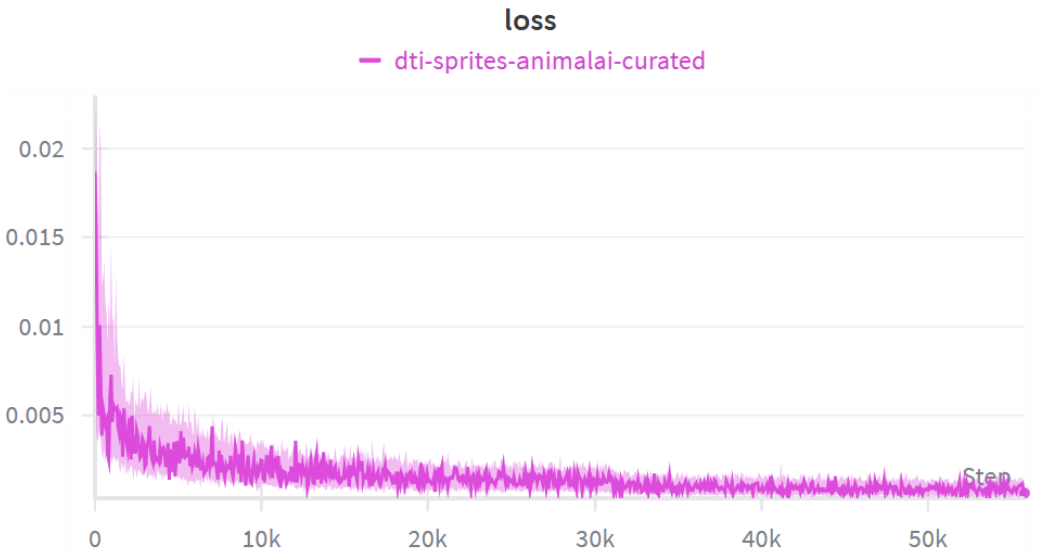


Figure 7: DTI-Sprites training on curated AnimalAI frames. The plot shows the reconstruction loss decreasing over time, indicating convergence.

However, despite its improved in-domain performance, DTI-Sprites exhibited poor transferability. The model failed to generalise to unseen environments, instead encoding dataset-specific priors that limited its utility as a general-purpose concept extractor. More critically, DTI-Sprites proved to be computationally inefficient, with long per-frame processing times that made it impractical for use in interactive or real-time learning settings. This posed a serious limitation for its integration into the AIGenC framework, where representations must be extracted online as part of a continual learning process. As such, DTI-Sprites did not meet the architectural or performance requirements of our system.

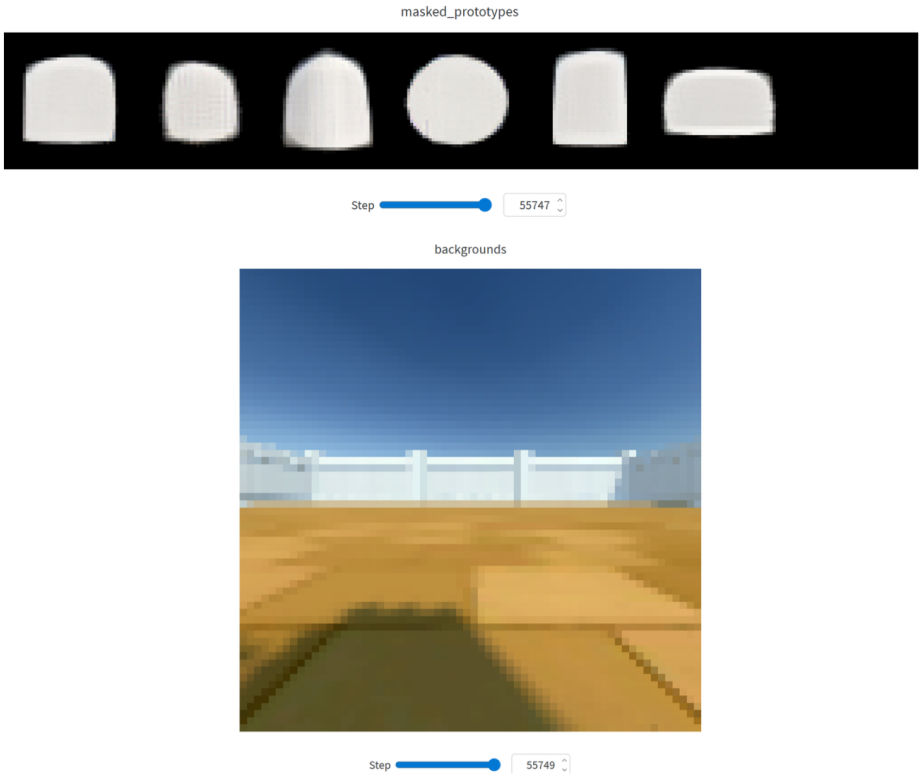


Figure 8: DTI-Sprites training on curated AnimalAI frames. The bottom panel displays a background sample from the environment. The top panel shows the masked prototypes generated by the model. Notably, these segmented shapes do not fully correspond to actual objects present in the scene, illustrating a key failure mode of the model—namely, the inability to discover meaningful and grounded object representations despite low reconstruction loss. This highlights the limitations of using DTI-Sprites for object-centric concept extraction in RL environments.

Finally, we evaluated the Segment Anything Model (SAM) (Kirillov et al., 2023), a large-scale foundation model trained on over 1 billion masks and 11 million images. Unlike previous models, SAM is designed for zero-shot segmentation and generalises well to previously unseen domains. In our experiments, we used SAM without any fine-tuning. We found that it consistently outperformed both Slot Attention and DTI-Sprites in segmenting objects across both CREATE and AnimalAI (Fig. 9). It produced clean, interpretable object masks that were robust to variation in background, shape, and viewpoint, making it the most viable solution for concept grounding in our pipeline.

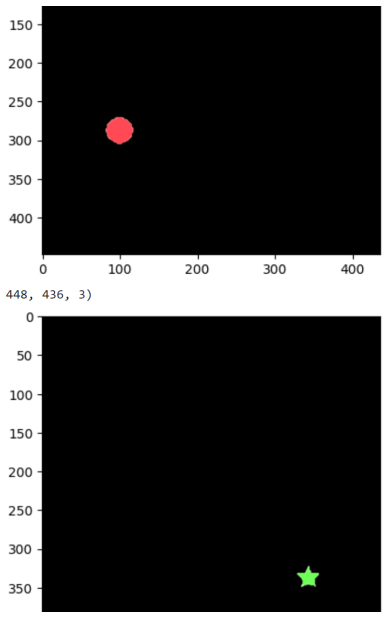


Figure 9: Segment Anything applied to CREATE environment frames. The model consistently produced object-consistent masks without fine-tuning, unlike previous object-centric models.

Object discovery: Final Design Choice Based on this evaluation, Segment Anything was selected as the default object discovery module in our system. This empirical success directly supports the theoretical requirements set forth by the AIGenC framework. In particular, SAM's ability to perform unsupervised, zero-shot segmentation aligns with the need for internally

grounded concept representations—those not dependent on pre-defined labels or task-specific tuning. Its robustness across domains ensures that extracted object concepts are stable and transferable, a core condition for building a reusable concept space. Moreover, SAM's inference efficiency makes it suitable for integration into online RL pipelines, fulfilling the practical constraint that concept processing must occur within the agent's active learning loop.

The discovered objects are encoded as latent vectors and added to the agent's working concept space. In the next sections, we describe how these encodings are combined with action representations, how we manage concept memory growth, and how structured trajectories are formed from these elements.

Interaction Learning: Methodological Considerations Modelling object–action interactions in reinforcement learning is a persistent challenge, particularly under unsupervised or weakly supervised constraints. While existing deep learning models have proven effective at encoding object-level features, capturing interactions, especially those with generalisable affordance-like properties, requires architectures capable of disentangling agent-object dynamics from contextual noise. Within the Concept Processing module, we experimentally evaluated two models: the Neural Statistician (NS) and a Convolutional Long Short-Term Memory (ConvLSTM) model with triplet loss. These models were chosen for their capacity to capture structured regularities across grouped interactions and to support generalisation across variations in object configurations.

Importantly, the latent vectors produced are not referred to as affordances in the classical sense, since they do not fully capture functional invariances across tasks. Instead, we refer to them as interaction representations, which encode interaction dynamics observed during episodes. To align with our formalism (see Algorithm 1, lines 26–27), these vectors are further enriched with outcome-level features, including the effect and reward, thereby capturing both behavioural and functional attributes necessary for downstream learning and reasoning.

Interaction Learning: Model Selection Process To evaluate effective encoding strategies, we explored two distinct models: the Neural Statistician and a ConvLSTM with triplet loss.

The Neural Statistician (NS) (Edwards and Storkey, 2016) is explicitly

designed to separate context-level (shared) and instance-level (specific) variables in grouped data. This architecture allows us to treat sequences of interactions involving the same object as a single group, where the object's identity remains fixed and only the configuration varies. This aligns with our affordance formalism, where interaction effects (e.g., bounce, slide) are learned across similar contexts. We trained the NS on grouped action sequences from the CREATE environment, where each group captured multiple configurations of one object undergoing interactions. However, CREATE frames exhibit high visual similarity across configurations, and many object interactions differ only subtly in spatial layout or motion trajectory. NS learns latent representations for unordered datasets by generating a posterior $q(c|D)$ over the context c of each dataset D, as such, their application to environments with high task similarity, such as CREATE, reveals notable limitations. Specifically, the shared context c across episodes often results in poor separability between distinct tasks, hindering the generalisation of representations. As shown in Fig. 10, although reconstructions capture coarse contextual structure, the training was plagued by unstable variational lower bounds (VLBs) and inconsistent gradients.

The PCA projection of interaction embeddings from the Neural Statistician model (see Fig.11) reveals that the model learned to cluster certain object types, particularly those with high semantic consistency such as Fixed_Ball or Bouncy_Hexagon—into distinct regions. This indicates that the context-level variables were partially successful in capturing object identity across multiple configurations. However, the significant overlap between other categories (e.g., Fixed_Box vs Bouncy_Box, Triangle variants) suggests that the separation between context and instance-level variables was not fully achieved. These issues, combined with the NS model's sensitivity to group partitioning and lack of scalability, ultimately hindered its use as an online component in the reinforcement learning loop.

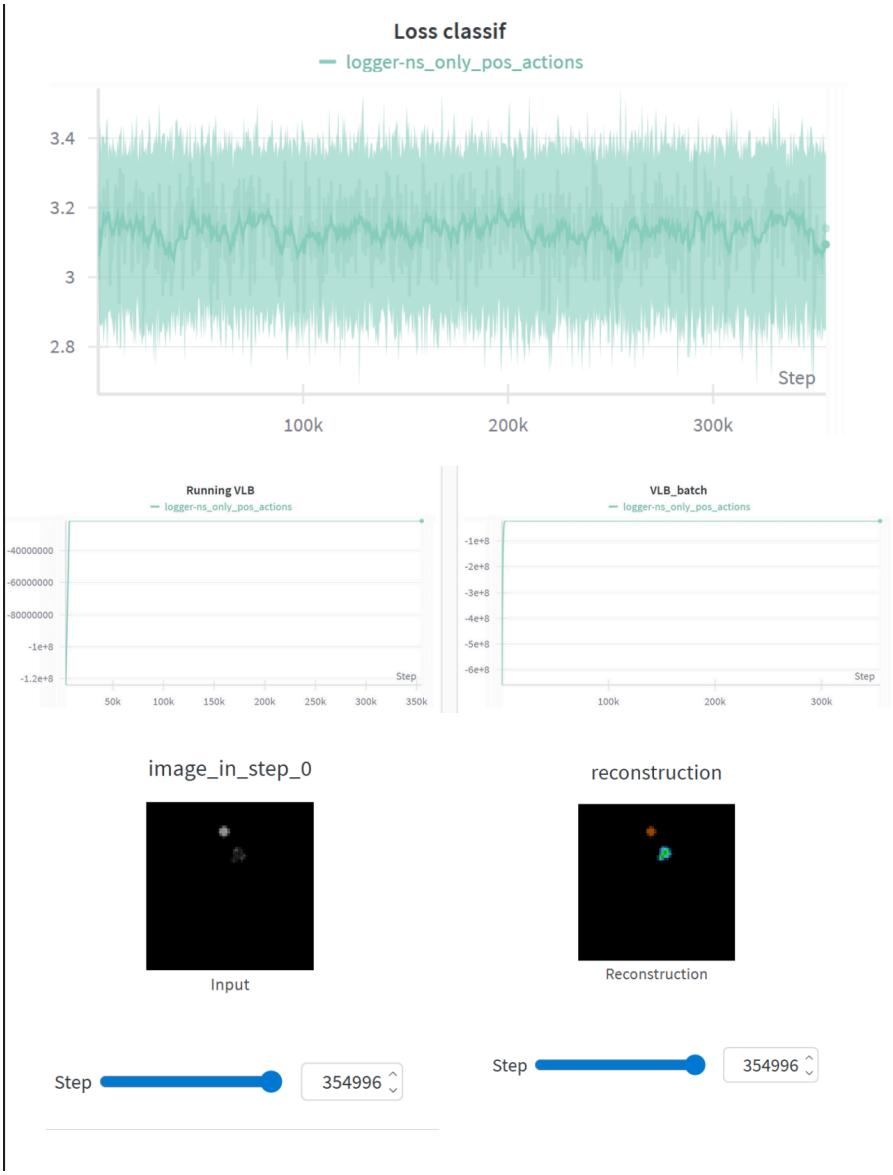


Figure 10: Training metrics for the Neural Statistician model on grouped interaction sequences. The classification loss (top panel) and variational lower bound (VLB) (middle panel, left and right) plateau early, suggesting limited optimisation progress or an insufficient gradient signal. While reconstruction samples show coarse recovery of object-level patterns, the instability may stem from both the model's sensitivity to grouping structure and the high visual similarity of interactions in the dataset.

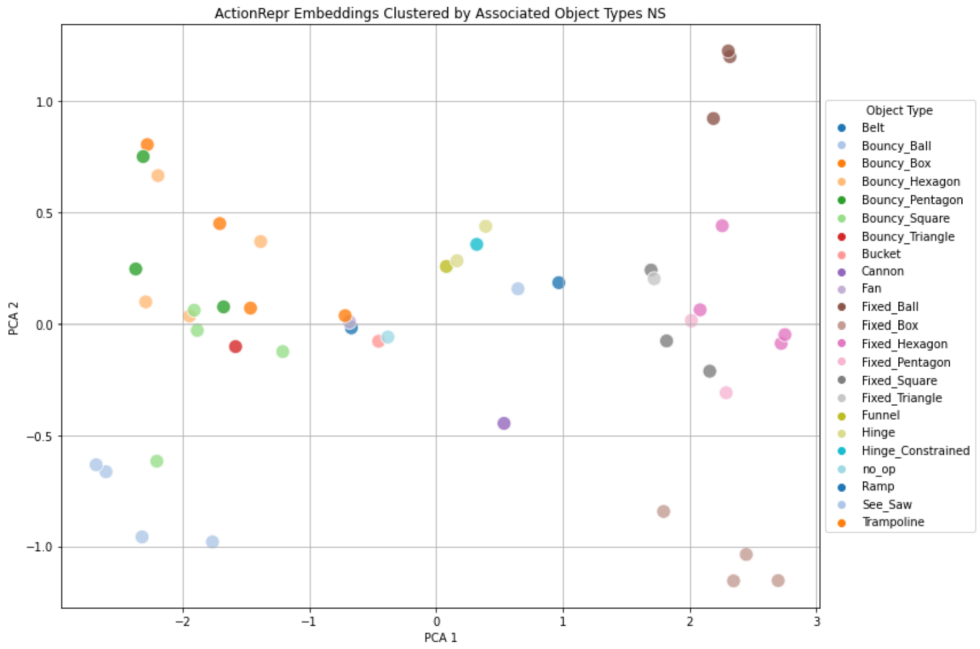


Figure 11: PCA projection of action representation embeddings produced by the Neural Statistician. Each point represents an embedding associated with an object interaction. While some clusters (e.g., Bouncy_Box, Fixed_Hexagon) are distinguishable, there is notable overlap across certain categories, reflecting the difficulty of learning disentangled interaction dynamics in a high-similarity environment like CREATE.

We utilise a ConvLSTM-based encoder to address the challenges of distinguishing between similar objects and their interactions across various tasks. This encoder is designed to capture the temporal dependencies inherent in sequences of object interactions. By encoding temporal dynamics, the ConvLSTM model provides a richer and more structured representation of the task environment. Furthermore, to ensure that the embeddings effectively distinguish between similar and distinct interactions, we couple the ConvLSTM with a triplet loss function. The triplet loss enforces separation in the embedding space by bringing embeddings of similar interactions closer together while pushing embeddings of distinct ones further apart. As such we were able to produce robust and discriminative interaction representations, forming a foundational component for subsequent hierarchical encoding.

Let $\mathcal{F}_a = \{f_1, f_2, \ldots, f_n \mid \text{all frames associated with interaction } a\}$ repre-

sent the set of frames associated with a specific interaction a. For a given frame $x_a \in \mathcal{F}_a$, we denote $x_p \in \mathcal{F}_a$ as a positive sample (a frame associated with the same interaction a) and $x_n \notin \mathcal{F}_a$ as a negative sample (a frame associated with a different interaction).

The triplet loss L_{triplet} is defined as:

$$L_{\text{triplet}} = \max(0, d(f(x_a), f(x_p)) - d(f(x_a), f(x_n)) + \alpha) \tag{24}$$

Where $f(x)$ is the embedding function that maps a frame into the embedding space, $d(\cdot)$ is a distance function (e.g., Euclidean distance), and α is the margin. In our experiments, we set $\alpha = 0.6$ after a hyper-parameter search of values ranging from 0.1-1.0. The final value provided optimal separation between embeddings of similar and distinct interactions.

This margin value helps the network distinguish subtle differences between interactions while maintaining adequate separation between positive and negative pairs.

Using this approach, we observed a clear separability in the embedding space. We applied K-means clustering to the learned embeddings and revealed the distinct clusters corresponding to different types of interactions. In the environment used (CREATE), all interactions involve a ball object interacting with various other objects. Thus, our analysis focuses on the second object, as it determines the dynamics of the interaction.

While clustering separability is not perfect, the learned representations exhibit an emergent understanding of interaction similarity. For instance, interactions involving swings or trampolines may occasionally overlap in certain clusters, yet the model consistently groups bouncy objects together, distinguishing them from interactions involving fixed objects (Fig. 12). This behaviour highlights the network's ability to capture underlying object properties that are essential for solving tasks in the CREATE environment.

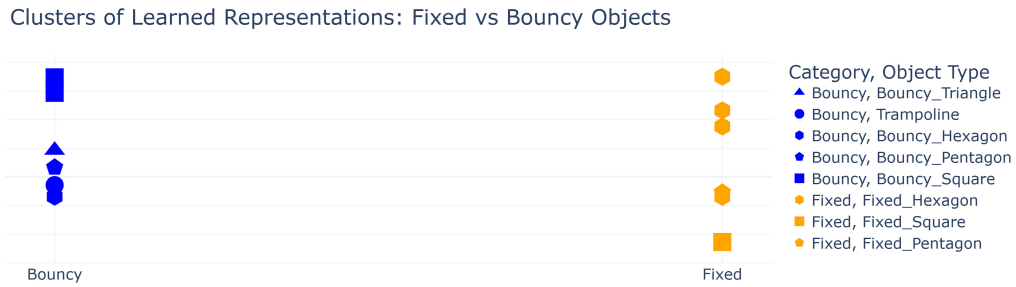


Figure 12: The scatter plot visualises the second object in interactions categorized as Fixed (orange, right side of the plot) or Bouncy (blue, left side of the plot), with each symbol representing a unique object (e.g. triangle, hexagon etc). The type of symbol along with their color indicates specific objects (e.g., Bouncy_Triangle, Fixed_Hexagon). The y-axis represents unique object IDs, while the x-axis categorizes objects into Bouncy or Fixed.

5.2.3 Hierarchical Memory Structure

To effectively model SETs, we structured episodic data into hierarchical heterogeneous graphs, where the feature extraction functions $\Phi : V \to \mathbb{R}^d$ (see Section 5.1.3) assigned high-dimensional embeddings to nodes representing objects, interactions, and states. This structure instantiates the conceptual memory architecture introduced in Section 4.1.2, where the concept space was defined as a hierarchical graph supporting both Working Memory and Long-Term Memory operations (see Fig. 3 and Fig. 4).

The nodes and their different relations are organised into hierarchical heterogeneous graphs with three distinct levels of abstraction: (1) a base layer for objects and interactions, where objects are extracted using the SAM model (Kirillov et al., 2023) and interaction dynamics are encoded via a ConvLSTM; (2) a mid-layer for states and affordances, capturing the transition dynamics and relationships between interactions and their effects (affordances here realise the (effect, reward)–(object, action) relations defined in Section 4.1); and (3) a trajectory layer for task-level abstractions, representing the cumulative impact of interactions and sequences (see Fig. 14).

This hierarchical organisation is stored in a graph database, as motivated by the computational advantages discussed in Section 4.1.2. It is not simply a stratification of layers but a structured representation where lower levels influence higher-level abstractions. Objects and actions/interactions influence affordances, which in turn shape state transitions and contribute to the

formation of SETs. This dependency across levels enables agents to reason over task structure, capturing both local dynamics and high-level task generalisation in complex environments. Formally, the hierarchical memory graph **G** representing a SET consists of:

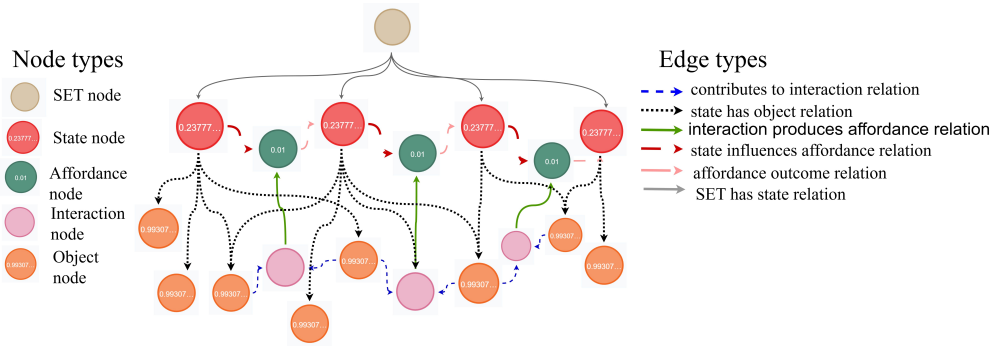


Figure 13: This illustration represents the hierarchical structure of a SET, capturing relationships between different node types in task representation. At the lowest level, interactions (light-filled circular nodes) depend on objects (darker circular nodes). Moving upward, affordances (smaller darker nodes) emerge from interactions, which then influence states (highlighted in red with distinct borders) which in turn influence the next affordance. At the highest level, the SET node connects to the structure through temporal state dependencies. The edges indicate relationships such as object dependencies, contributions, and effects. Varying line styles help differentiate these relationships. This hierarchical structure showcases how task execution is represented by linking objects, interactions, affordances, and states. Note that this figure instantiates the conceptual hierarchy presented in our AIGenC template and shown in Figure 2 (Chapter 4): the progression from objects to interactions to affordances to states to SETs reflects increasing levels of abstraction in the latent embedding space. Unlike the conceptual template where affordances were depicted as edges, here they are implemented as explicit nodes to support independent embeddings and graph-database storage.

Level 1: low-level abstractions (objects and interactions) The lowest level of the hierarchy captures individual **objects** (Fig. 15) and **interactions representations** (Fig. 16), which form the fundamental units in the environment (Fig. 17). In the graph visualisation (Fig. 13), objects appear

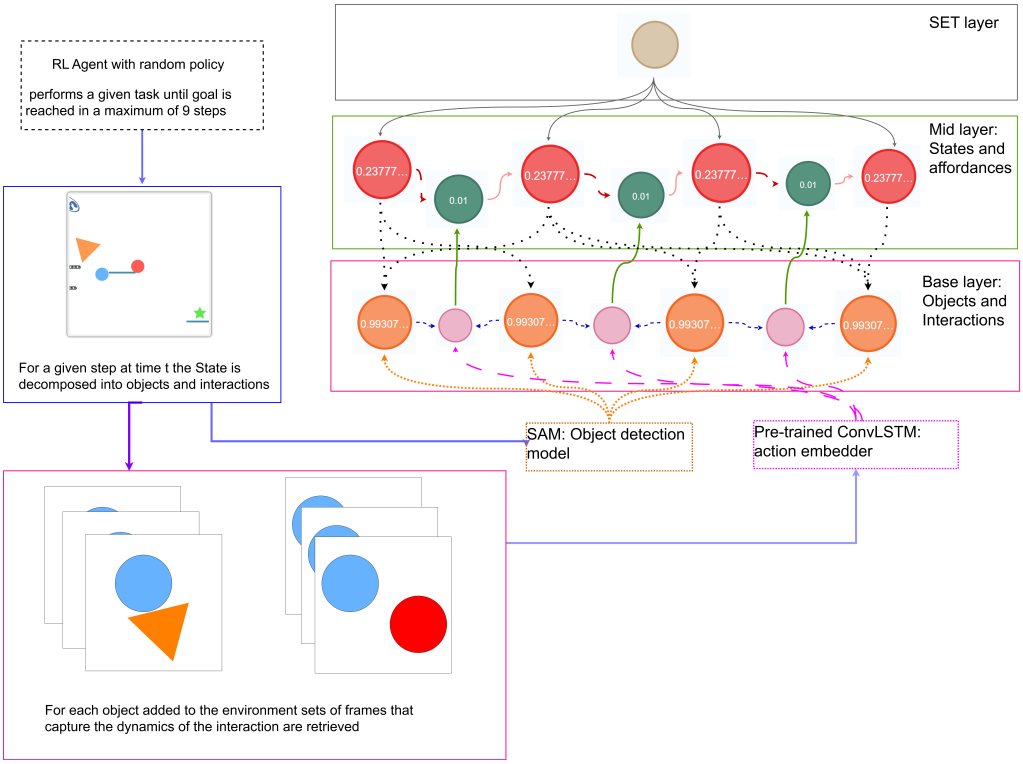


Figure 14: Hierarchical Memory building: The image shows the models used to build the hierarchical memory. The base layer extracts objects and interactions using the SAM object detection model and a pre-trained ConvLSTM, capturing fundamental task dynamics. The mid-layer represents states and affordances, modelling how interactions influence future states. At the highest level, the SET layer encodes trajectory-level abstractions, capturing long-term dependencies across sequential states. This hierarchical organisation enables efficient reasoning over task structures and adaptive decision-making.

as orange nodes, each connected by `has_object` edges to higher-level states, while interactions appear as pink nodes encoding the dynamics between objects.

The relationship between objects and interactions is fundamental to model structured learning. Fig. 17 illustrates how objects interact with each other, emphasising that an interaction is defined not only by its motion dynamics but also by the entities involved in its execution.

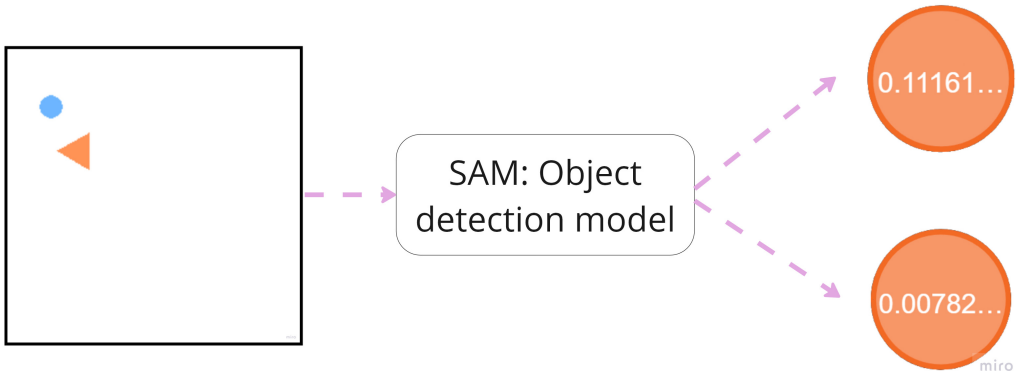


Figure 15: Base-level representations: Object-level representations extracted at the base layer using the SAM object detection model. These representations capture object features in the environment and are stored in a graph database.

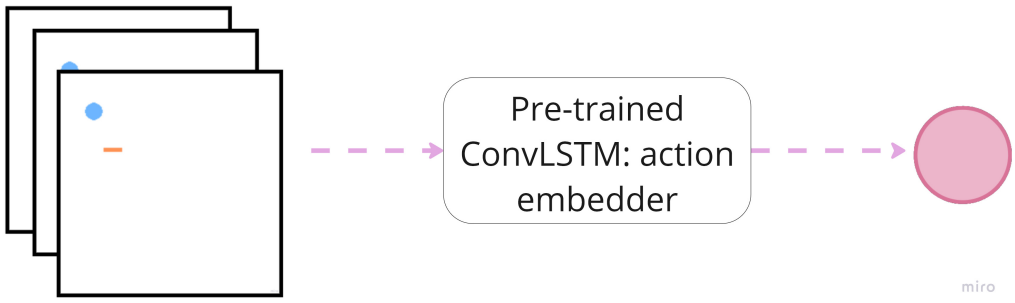


Figure 16: Base-level representations: Interaction-level representations extracted at the base layer using a pre-trained ConvLSTM action embedder. These embeddings capture the dynamics of object interactions and are stored in the graph database.

To construct structured representations of objects and interactions, we employed a combination of object detection and interaction representation learning techniques. This section presents the final design choices resulting from our evaluation process. In the previous section, we have detailed the methodological considerations and experimental results that informed these decisions.

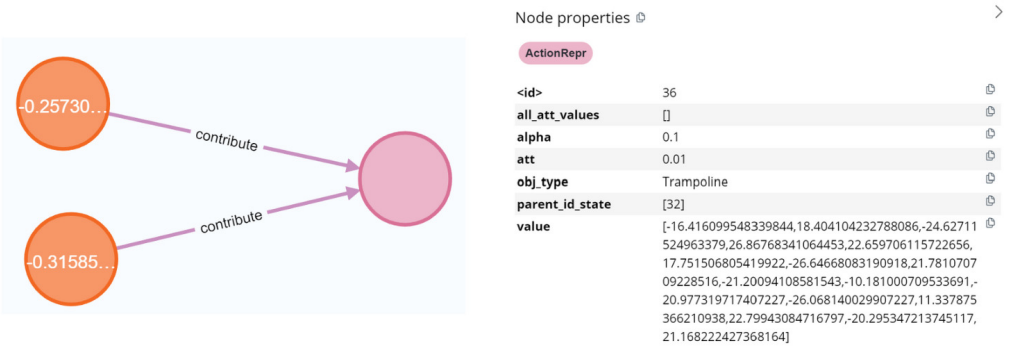


Figure 17: Representation of two objects interacting. Interactions are characterized not only by their execution but also by the relationships they share with the objects involved. On the right, the different parameters of an Interaction are shown such as the type, and the vector representation. Representation obtained from the Neo4j (Lal, 2015) graph database interface.

1. Object Detection for extraction of object nodes

   Objects within each episode are extracted using the SAM (Segment Anything Model) object detection model Kirillov et al. (2023), which identifies and classifies entities in the environment. A process of checking whether the object exists already is carried out using a cosine similarity function.

$$\text{Similarity}(o_{\text{new}}, o_i) = \frac{\mathbf{v}_{\text{new}}^\top \mathbf{v}_i}{\|\mathbf{v}_{\text{new}}\| \|\mathbf{v}_i\|}, \quad \forall i \in \text{Memory} \tag{25}$$

$$\text{If } \max_i \text{Similarity}(o_{\text{new}}, o_i) < \tau, \text{ add } o_{\text{new}} \text{ as a new node.} \tag{26}$$

2. Learning Interactions Representations using ConvLSTM for the extraction of interaction nodes

   A ConvLSTM encoder with triplet loss is used to model object interactions effectively, capturing their temporal and spatial dynamics within the environment.

Level 2: Affordances and State Transitions At the intermediate level, we represent states as heterogeneous graphs, denoted as G'_{it} (See Section

5.1.4) at time t. Each state comprises multiple interacting objects, connected through interaction representations that shape the decision-making process (Fig. 13).

Affordances, whose conceptual role was introduced in Section 4.1 as acquired relations between (effect, reward) pairs and (object, action) tuples, are here instantiated as directed edges (green nodes in Fig. 13) linking consecutive states (Fig. 18). They capture both the immediate effect of an interaction and its associated reward, enabling the model to prioritise interactions that contribute to successful task completion.

By explicitly modelling affordance relationships, we improve the system's ability to reason about sequential dependencies and facilitate adaptive decision-making in complex environments, capturing richer contextual information beyond traditional action-reward pairs.

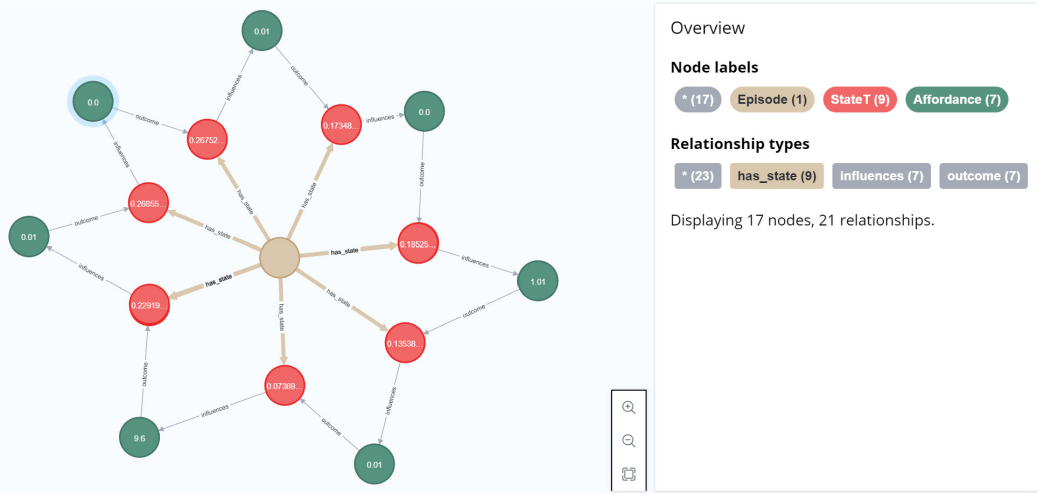


Figure 18: Illustration of the hierarchical structure in SETLE as extracted from the Neo4j graph database (Lal, 2015). The second level of the hierarchy consists of states (red, from the middle node the first set of connections) connected by affordances (green, the outermost nodes), illustrating the transitions between states via affordances.

By explicitly modelling affordance relationships, we improve the system's ability to reason about sequential dependencies and facilitate adaptive decision-making in complex environments that move beyond traditional action-reward, capturing richer contextual information.

Level 3: High-Level Abstraction of Structurally Enriched Trajectories (SETs) The topmost layer in the hierarchical structure corresponds to the overall SET representation. In its raw form, a SET is defined by the underlying levels, comprising states, affordances, interactions, and objects (see Section 5.1.3). The hierarchical graph structure illustrated in Figure 13 captures these relationships, with the SET node connecting to states through temporal dependencies.

However, utilising full graph structures for downstream tasks is computationally impractical. To address this, we learn a **latent representations** of SETs (see Section 5.3), effectively compressing the rich hierarchical information into a compact embedding. This latent representation forms the core of our proposed AIGenC algorithmic template, enabling efficient and scalable utilisation in downstream tasks without losing critical task-relevant information.

5.3 SETLE: SET encoder

The SETLE encoder transforms Structurally Enriched Trajectories (SETs) into compact, task-relevant embeddings, allowing the agent to generalise across episodes. Built upon the structured representations defined in Section 5.2, each SET encodes object configurations, agent interactions, affordances, and temporally linked states as a heterogeneous graph. The encoder projects these graphs into a latent space where structurally and functionally similar trajectories are positioned nearby, supporting relational abstraction and cross-task retrieval.

Within the AIGenC framework, the encoder is a critical component of Reflective Reasoning. It provides the mechanism by which the agent compares its current trajectory with stored experiences—not based on surface-level similarity but by recognising deeper structural regularities across tasks. These trajectory embeddings serve as functional approximations of problem templates, which are abstract patterns that capture how actions and affordances combine over time to produce successful or unsuccessful outcomes.

Significantly, this process extends beyond simple retrieval. The encoder supports the enhancement of the agent's current working memory by enabling partial matches to structurally relevant past episodes. When no high-similarity match is found, this exact mechanism underpins blending, facilitating the recombination of multiple weakly related experiences to generate novel solutions.

To learn this embedding space, the SETLE encoder is trained using a triplet loss objective, which enforces that trajectories with similar causal and functional structures—and especially similar outcomes—are embedded close together while dissimilar or unsuccessful episodes are pushed further apart. This encourages the emergence of a structured trajectory space that supports flexible retrieval and generalisation during learning and decision-making.

The remainder of this chapter details the core components of the SETLE training pipeline. We begin with the data collection process, outlining how episodic graphs are constructed from CREATE and MiniGrid interactions. We then describe the SETLE graph encoder architecture, which builds on a Heterogeneous Co-contrastive Learning (HeCo) architecture adapted for structured trajectory data. Finally, we detail the experimental protocol, where the encoder is optimised using triplet loss to cluster structurally similar outcomes and support trajectory-level generalisation. These components together instantiate the Reflective Reasoning process in AIGenC.

5.3.1 Collecting data for training

We collect SETs and label them to create a dataset for training, by following these steps:

1. Random exploration: We begin with an exploration phase where the agent performs multiple tasks within the CREATE or Minigird environment. During this phase, a random policy agent operates in the environment until it reaches the goal state. For tractability, we have limited the number of steps the agent can take in an environment to 9, if it has not reached a goal state by then, the SET is labelled as unsuccessful. We repeat the process until sufficient successful trajectories have been collected for training.

2. SET Labelling: After each episode, we assess whether the agent successfully completed the task. If the episode ends with the agent achieving the goal state, the episode SET is labelled as: success; otherwise, it is labelled as: failure. Alongside the label, we store relevant metadata about the setup, actions taken, and objects encountered, allowing for later validation and analysis. Successful SETs are stored in the agent's memory (see Section 5.1.4), completed with the inventory of objects (in the case of CREATE), interactions, and state transitions that led to the successful completion.

Once a sufficient number of trajectories have been collected, we train the SETLE encoder using a triplet loss function.

5.3.2 Graph Encoder Architecture Based on Heterogeneous Co-contrastive Learning (HeCo)

A graph encoder is a neural architecture that transforms graph-structured data into meaningful embeddings, preserving the relationships and dependencies between entities. Unlike standard feature extractors, graph encoders operate over relational data, using the graph topology to learn structured representations.

Graph encoders rely on message passing, a core operation in Graph Neural Networks (GNNs) (Scarselli et al., 2008), where each node iteratively aggregates information from its neighbours to refine its representation (Gilmer et al., 2017). In its general form, message passing is defined as:

$$H^{[l+1]} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{[l]}W^{[l]}\right) \tag{27}$$

Where:

- $H^{[l]} \in \mathbb{R}^{N\times d^{[l]}}$: Node representations at the l-th layer, extending the notation from Equation 3.
- $d^{[l]}$: Hidden dimension of the l-th layer.
- $W^{[l]} \in \mathbb{R}^{d^{[l]}\times d^{[l+1]}}$: Learnable weight matrix for the l-th layer.
- N: Number of nodes in the graph.
- $\tilde{A} = A + I$: Adjacency matrix A with added self-loops (identity matrix I), a common extension of the normalised adjacency A^* from Equation 3.
- $\tilde{D}$: Degree matrix of $\tilde{A}$, used for symmetric normalisation.
- σ: Nonlinear activation function.

This process allows nodes to incorporate structural context from their neighbours, making graph encoders particularly effective for tasks that require relational reasoning.

Most existing approaches either encode entire heterogeneous graphs or focus on subgraph encoding for homogeneous data. Full-graph encoding enables global feature propagation, but it does not preserve localised dependencies. We model the hierarchical memory in **SETLE** as a collection of SET subgraphs, unlike standard GNNs that operate on a global graph, SETLE learns representations for independent trajectory (SET) subgraphs, where each SET node aggregates information solely from its connected states, interactions, objects, and affordances. By structuring trajectories (SETs) as subgraphs within the larger task graph (see Section 5.1.4), SETLE captures common elements, such as objects and interactions, at lower levels, allowing **localised learning while maintaining global coherence across tasks**.

Building on the principles of HeCo (introduced in the Background section), we adapt its contrastive learning approach for episodic graphs. We treat each SET as a subgraph, with the *SET node* acting as the central node. The central node refers to the primary node within a graph that aggregates information from surrounding nodes through both meta-path and schema-based message passing, ensuring that its embedding captures local structural dependencies as well as high-order semantic relationships within the heterogeneous graph.

The encoding operation can be formally represented as:

$$\text{SETLE-encoder}(G_{SET}) \rightarrow z_{SET} \tag{28}$$

where z_{SET} is the latent representation of a Structurally Enriched Trajectory G_{SET}.

To train the graph encoder, we use a triplet loss and a hybrid loss function to organise SET embeddings. Successful SETs from the same task are drawn together, while unsuccessful SETs or SETs from different tasks are pushed apart. This allows us to differentiate between successful and unsuccessful strategies, enhancing generalisation. To ensure that episodes with similar trajectories are encoded closely in the latent space, we employ a triplet loss as such:

$$L_{triplet}^{SET} = \max(0, d(g(y_a), g(y_p)) - d(g(y_a), g(y_n)) + \alpha) \tag{29}$$

Here:

- $g(y)$ is the SET embedding function.
- y_a, y_p, and y_n are the anchor, positive, and negative SET samples.

- d is a distance metric.
- α is a margin parameter.

By minimising this loss, the model learnt to align trajectory-specific embeddings, ensuring that episodes with similar trajectories (i.e., trajectories belonging to the same tasks) remained close while distinguishing those with different results.

To further improve the quality of SET representations, a modified **hybrid loss function** that combines the triplet loss with a cross-view contrastive loss is also used. This approach not only pushes similar SET embeddings together but also aligns two graph views: meta-path and schema, ensuring consistency across different perspectives of the episodic graph.

The two key views from HeCo are adapted to SETs as follows:

1. Network Schema View

   In the network schema view, we encode the structural dependencies of each episode by considering relationships between node types (e.g., state, interaction, affordance) around the central SET node. This view allows us to capture local patterns directly connected to the episode.

2. Metapath View

   The meta-path view captures high-order semantic relationships by encoding dependencies across sequences of connected nodes, allowing the model to learn relationships that span multiple levels of the episode's hierarchy. The following steps outline the process of computing node embeddings under the meta-path view:

   (a) For a given node "i" and a set of M meta-paths $\{P_1, P_2, \ldots, P_M\}$, which represent high-order semantic relationships (e.g., SET-St-Obj-Inter-Aff or SET-St-Aff-St):

      i. For each meta-path P_n, we apply a meta-path specific Graph Convolutional Network (GCN) to encode its characteristics:
         - Calculate the projected features h_i and h_j for nodes "i" and "j" in the meta-path.
         - Use degree information d_i and d_j for nodes "i" and "j" to compute the updated embedding $h_i^{P_n}$ based on the meta-path's semantic similarity.

(b) For all M meta-paths, we obtain a set of embeddings $\{h_i^{P_1}, \ldots, h_i^{P_M}\}$ for node "i."

(c) Use **semantic-level attention** to combine these embeddings into the final embedding z_i^{mp} under the meta-path view:
   - Calculate the weight of each meta-path P_n for node "i."
   - Determine the importance of each meta-path in the embedding fusion.
   - Perform a weighted sum of meta-path embeddings to produce the final node embedding.

The selected metapaths, `SET-St-Obj-Inter-Aff` and `SET-St-Aff-St`, are designed to capture high-order semantic relationships within each episode in the RL environment.

- `SET-St-Obj-Inter-Aff` **(SET - State - Object - Interaction - Affordance):**
  - This meta-path reflects the hierarchical progression from the episode as a whole, through individual states, down to specific objects and actions, ultimately linking to affordances.
  - By following this sequence, SETLE can capture the causal chain of events where each *state* within a *SET* is defined by its constituent *objects* and the *interactions* involving those objects. The inclusion of *affordances* in the meta-path allows the model to understand how specific object interactions influence the likelihood of transitioning to desirable or goal-oriented states.
  - This path is crucial for learning how particular combinations of objects and their interactions contribute to successful results, enabling the model to capture the semantic importance of specific object relationships in each episode. It supports the agent in recognising patterns of interaction that are effective across similar tasks, improving object selection and result prediction in new scenarios.
- `SET-St-Aff-St` **(SET - State - Affordance - State):**
  - This meta-path emphasises the transitions between states within a SET, specifically focussing on how each *affordance* (a state-action-state link with associated rewards) facilitates state changes.

– By encoding this meta-path, SETLE captures the dynamics of state transitions within the trajectory. The affordance links provide a representation of the agent's actions and the effects, highlighting which transitions are likely to lead to success or failure.
– This metapath is valuable for modelling temporal dependencies in SETs, as it reveals the sequences of states the agent encounters. Understanding these transitions is critical for building a representation that reflects the path to the goal, rather than just the individual states, thus helping the agent generalise across different task setups that share similar transition dynamics.

Together, these meta-paths allow us to capture both **static relationships** (e.g., object interactions within a state) and **dynamic relationships** (e.g., state-to-state transitions driven by affordances) in the SET graph. By incorporating both aspects, we can construct a more comprehensive and semantically meaningful representation of each trajectory, allowing it to effectively generalise across tasks and improve decision-making in complex reinforcement learning environments.

Considering these two views we define the hybrid loss as:

$$L_{\text{hybrid}} = \lambda L_{\text{triplet}}^{\text{SET}} + (1 - \lambda) L_{\text{contrastive}}^{\text{view}} \tag{30}$$

Here:

- $L_{\text{triplet}}^{\text{SET}}$ organises SET embeddings in the latent space, as previously defined:

$$L_{\text{triplet}}^{\text{SET}} = \max(0, d(g(y_a), g(y_p)) - d(g(y_a), g(y_n)) + \alpha) \tag{31}$$

where $g(y)$ is the SET embedding function, y_a, y_p, and y_n are the anchor, positive, and negative SET samples, d is a distance metric, and α is a margin parameter.

- $L_{\text{contrastive}}^{\text{view}}$ aligns the meta-path and schema view embeddings to ensure consistency across views:

$$L_{\text{contrastive}}^{\text{view}} = -\log \frac{\exp(\text{sim}(z_{\text{mp}}, z_{\text{sc}})/\tau)}{\sum_{z' \in \mathcal{Z}} \exp(\text{sim}(z_{\text{mp}}, z')/\tau)} \tag{32}$$

Where:

- z_{mp} and z_{sc} are the meta-path and schema view embeddings for the same SET.
- $\text{sim}(\cdot, \cdot)$ is a similarity function, such as cosine similarity.
- τ is the temperature scaling parameter.
- $\mathcal{Z}$ represents the set of embeddings, including positive and negative samples.

- λ is a hyperparameter that controls the balance between the triplet loss and the contrastive loss, with $0 \leq \lambda \leq 1$.

By combining these objectives, the hybrid loss ensures that SET embeddings are both *task-specific* (via the triplet loss) and consistent across graph views (via the contrastive loss).

5.4 SET encoder experimental procedure

This section evaluates the effectiveness of SETLE's hierarchical graph encoder in encoding task-specific information. The evaluation is structured into two key sets of experiments: margin sensitivity analysis and ablation studies. In the first set of experiments (S1), we assessed how different margin values in the triplet loss function affect the quality of learnt embeddings. By varying the margin parameter, we analysed its influence on the clustering of success and failure SETs in the embedding space. This experiment provided insights into the optimal margin settings that enhanced separability while maintaining robust intra-cluster cohesion.

The second set of experiments (S2) consisted of ablation studies to assess the impact of hierarchical structures and relational dependencies in SETLE's SET design. We systematically modified or removed key components, such as hierarchical organisation, sequentiality, and affordance-based state transitions, to understand their contributions to SET representation. By comparing the clustering performance of modified versions of SET's structure against its complete hierarchical design, we demonstrated the necessity of structured graph-based encoding for effective trajectory learning.

Finally, we present results in the MiniGrid environment—a simplified, discrete setting in which the concept space is restricted to objects and state transitions, as interaction-level dynamics cannot be extracted meaningfully. This allows us to test SETLE's capacity for abstraction under reduced representational complexity.

5.4.1 S1: SET Embedding Analysis: Margin and Loss Function Impact

The first set of experiments aimed to evaluate the ability of SETLE's hierarchical graph encoder to represent enhanced trajectory information effectively. The primary focus was on assessing whether the embeddings generated by SETLE can distinguish between successful and unsuccessful episodes within and across tasks. Using episodic data collected from the CREATE environment, each episode was encoded as a heterogeneous graph (SET) that comprises objects, actions, states, and affordances.

We investigated how well SETLE's SET embeddings captured the underlying task dynamics by analysing clustering quality using various configurations of hyper-parameters and different loss functions.

The evaluation focused on specific parameters including:

- **Margin Values:** We experimented with margin values for the triplet loss, specifically testing values of [0.1, 0.2, 0.5, 1.2, 1.5]. This parameter regulates the separability of embeddings in the latent space by controlling how far apart unsuccessful SETs are pushed from successful ones.
- **Loss Functions:** Two types of loss functions were employed:
    1. **Classical Triplet Loss:** Ensures SET embeddings for similar episodes are close, while those for dissimilar SETs are further apart.
    2. **Hybrid Triplet Loss:** Combines triplet loss with a cross-view contrastive loss. The contrastive component aligns embeddings from the schema and meta-path views, while the triplet component enhances task-based separability.

The following metrics were used to evaluate the separability of success and failure clusters:

- **Silhouette Score**: Measures the cohesion and separation of clusters, the values can range between -1 and 1, with higher values indicating more distinct clusters. Values closer to 1 suggest that episodes are well-clustered within their respective success or failure groups.
- **Davies-Bouldin Index (DBI)**: Quantifies the average similarity ratio of each cluster with its most similar cluster, with lower values indicating better clustering performance and less overlap between clusters.

- **Dunn Index**: Assesses the ratio between the minimum inter-cluster distance and the maximum intra-cluster distance, with higher values representing greater separability and well-defined clusters.

Margin Impact and Results Effect of Margin Values

We evaluated the impact of varying margin values on the clustering quality of SET embeddings. The results provided insights into the cohesion and separation of clusters for both successful and unsuccessful episodes. High cohesion within success or failure clusters indicated that the embeddings effectively captured the underlying patterns of similar SETs, while high separation between these clusters reflected the model's ability to distinguish the dynamics of successful and unsuccessful episodes. These insights were critical for evaluating the suitability of different margin values in shaping the latent space for SET-specific encoding.

The clustering results were analysed using three standard metrics: **Silhouette Score**, **Davies-Bouldin Index (DBI)**, and **Dunn Index**.

Table 1 summarises the clustering performance across margin values. Higher margins (1.2 and 1.5) generally yielded better separability, reflecting clearer distinctions between successful and unsuccessful episodes. We extended our analysis to $\alpha = 2.0$ to verify behaviour at larger values. Results showed comparable performance to $\alpha = 1.5$, indicating a plateau rather than degradation. This suggests $\alpha \approx 0.6$–1.5 represents the effective operating range, with diminishing returns beyond this interval.

Table 1: Clustering Results for Success and Failure for SET encodings at Different Margin Values

Margin	**Silhouette Score (Success)**	**Silhouette Score (Failure)**	**DBI (Success)**	**DBI (Failure)**	**Dunn Index (Success)**	**Dunn Index (Failure)**
0.1	0.7632	0.8531	0.4514	0.2319	2.2497	3.4662
0.2	0.7959	0.8659	0.3205	0.2233	2.9952	3.3552
0.5	0.7943	0.8798	0.3378	0.1771	2.5417	5.3031
1.2	0.7588	0.8535	0.3603	0.2305	2.3577	3.5070
1.5	0.8137	0.8990	0.2974	0.1548	2.7778	5.5415

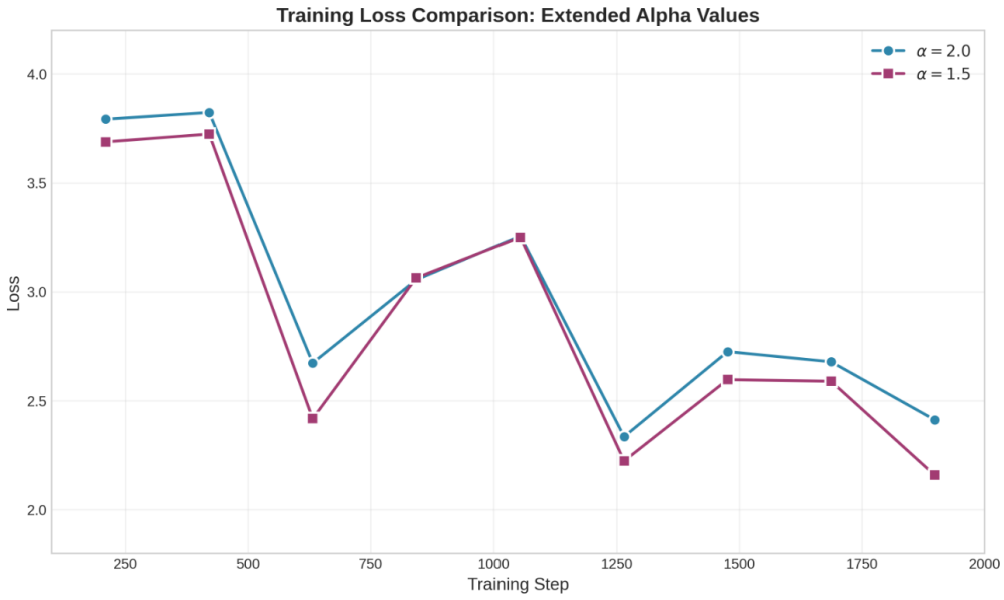


Figure 19: Training loss comparison between $\alpha = 1.5$ and $\alpha = 2.0$ over training steps. Both configurations exhibit similar loss trajectories, with $\alpha = 1.5$ achieving marginally lower values (e.g., 2.16 vs 2.41 at step 1898). The absence of performance degradation at $\alpha = 2.0$ indicates that the loss plateaus beyond the optimal range rather than increasing, confirming that $\alpha \in [0.6, 1.5]$ represents the effective operating range with diminishing returns at higher values.

From the results in Table 1, it was shown that higher margin values generally improved clustering quality for both success and failure SETs. Notably:

- The **Silhouette Score** showed a consistent improvement with increasing margin, peaking at a margin of 1.5, where the success and failure clusters were well-separated.
- The **Davies-Bouldin Index (DBI)** decreased as the margin increased, reaching its lowest value at a margin of 1.5. This suggested that higher margins reduced the overlap between clusters, leading to more distinct success and failure groupings.
- The **Dunn Index** improved significantly with higher margins, especially for failure SETs, which achieved their highest separability at a margin of 1.5. This indicated that increasing the margin value results in more well-defined clusters, with greater distances between the success and failure groups.

Higher margin values proved necessary for effective clustering in this context because the CREATE environment is relatively simple, with low-resolution frames and limited visual complexity. In such environments, embeddings based on episode encodings can appear similar due to the simplicity of the scenes and fewer unique features across frames. Using higher margins (1.2 and 1.5), we ensured that embeddings for episodes with different outcomes or tasks are separated more distinctly, overcoming the limitations posed by the low frame quality and simplifying environment.

In terms of epoch loss, we did not observe significant changes with varying margin values Fig.20. The overall epoch loss progression remained relatively stable, suggesting that while higher margins did not heavily impact convergence or loss reduction during training, they did have a positive effect on the quality of clustering in the latent space. In the cluster analysis below, we examined how these higher margin values enhanced episode separability, supporting generalisation and task-based decision-making. Figure 20 illustrates the training loss for different margin values.

Visualization of Outcome Embeddings To complement the quantitative analysis of clustering metrics such as Dunn Index, DBI, and Silhouette Score, we employed K-means clustering and visualisation techniques to provide a qualitative assessment of the learnt SET embeddings. By reducing the

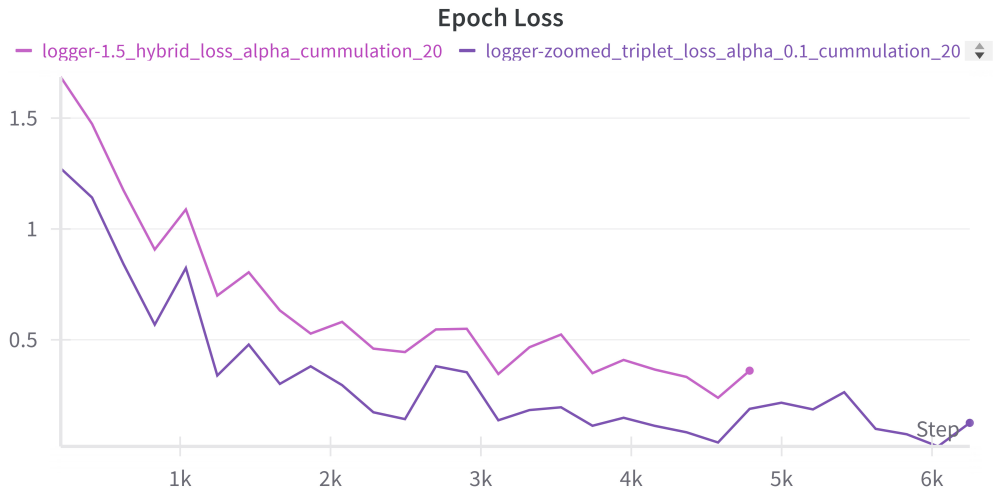
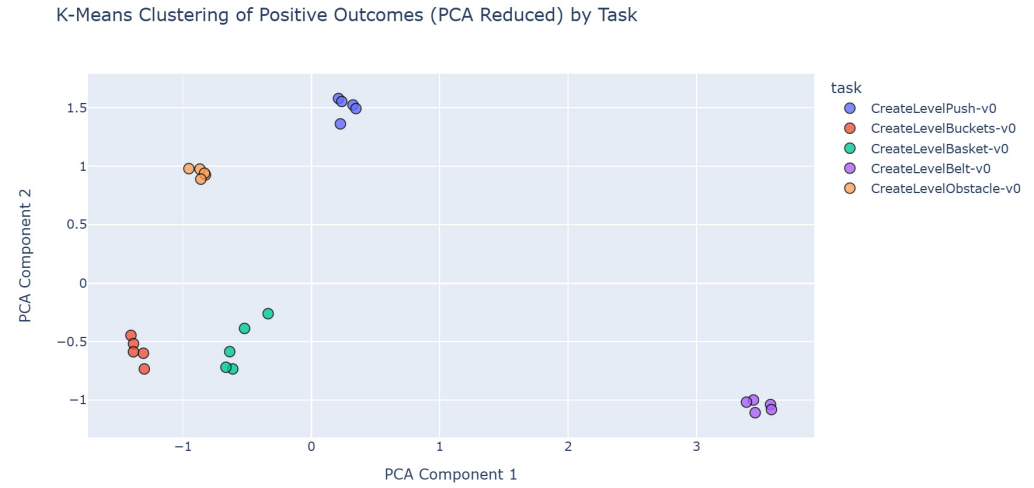


Figure 20: Training loss for various margin values. Higher margins improve clustering quality without significantly affecting convergence.

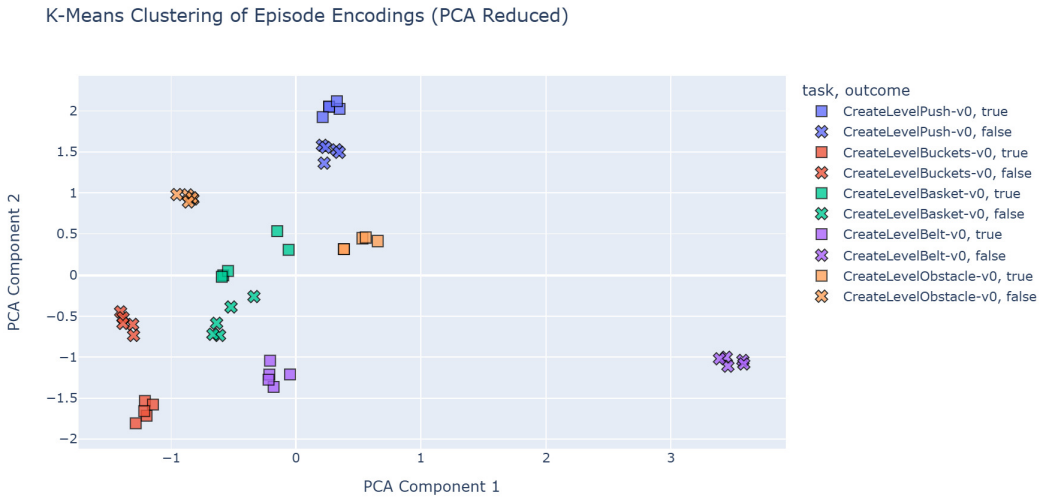
dimensionality of the embeddings using PCA, we examined the spatial distribution of clusters in two dimensions, revealing the distinctiveness of task and result representations. These visualisations offered additional insights into how well SETLE's graph encoder captured the structural and outcome-specific features of the tasks, highlighting its ability to group similar episodes while maintaining separation between dissimilar ones. The tight grouping of points within each task cluster, along with minimal overlap between clusters of different tasks, indicated that the graph encoder has learnt meaningful representations that distinguish both the task type and the outcome.

Figure 21a shows the K-means clustering of SET embeddings, color-coded by task. The results highlight SETLE's ability to capture task-level distinctions and result-specific information.

Additionally, the separation of successful and unsuccessful SETs within the same task, indicated by distinct markers in the plot, further highlighted SETLE's effectiveness in encoding task-specific information. For example, episodes from CreateLevelPush that successfully reached the goal state were grouped closely, while failed attempts were separated, providing evidence that SETLE captures not only the task identity but also the quality of task execution Figure 21b.



(a) Task-level clusters: K-means Clustering of SET Encodings (PCA Reduced). Each point represents a SET in the CREATE environment, color-coded by task.



(b) Outcome and task-specific clusters: Each point represents a SET in the CREATE environment, colour-coded by task, where circles mark successful SETs and crosses mark failed ones.

Figure 21: K-means clustering of SET embeddings. (a) Clustering by task only. (b) Clustering by task and outcome (success vs. failure). SETLE effectively captures both task-level distinctions and outcome-specific information.

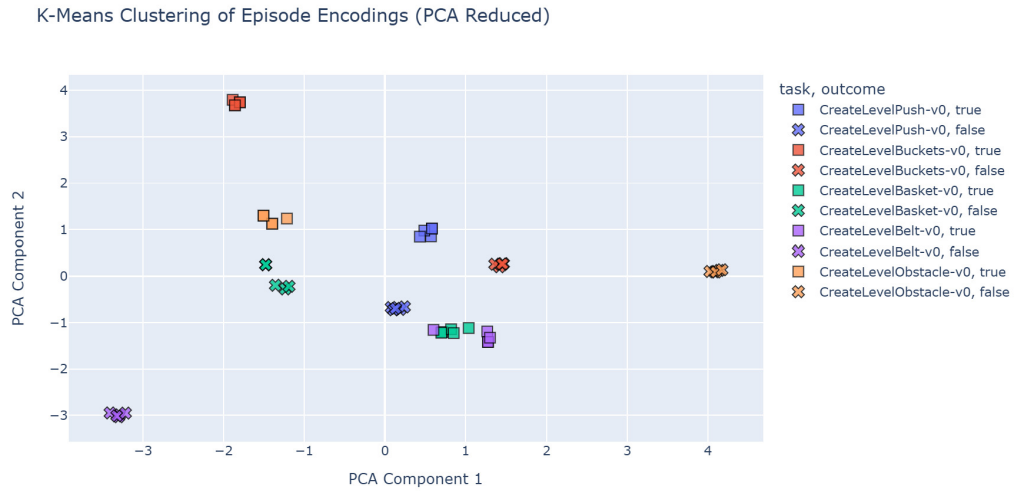
Loss Functions Impact and Results To further analyse the impact of different optimisation strategies on SET representation, we compared the performance of two loss functions: the simple triplet loss and the hybrid loss. While the triplet loss focusses on enforcing distance constraints between similar and dissimilar SET embeddings, the hybrid loss integrates contrastive learning by aligning embeddings across schema and meta-path views. This combination aims to enhance both structural consistency and task separability.

To assess the effectiveness of the hybrid loss in improving representation compared to the standard triplet loss, we visualised the SET embeddings generated by SETLE under both loss functions. Each point represents a SET, colour-coded by task, with successful episodes marked as $\square$ and unsuccessful episodes marked as $\times$. The visualisations are provided in Figures 22a and 22b, where subfigure (a) shows the results with triplet loss and subfigure (b) shows the results with hybrid loss.

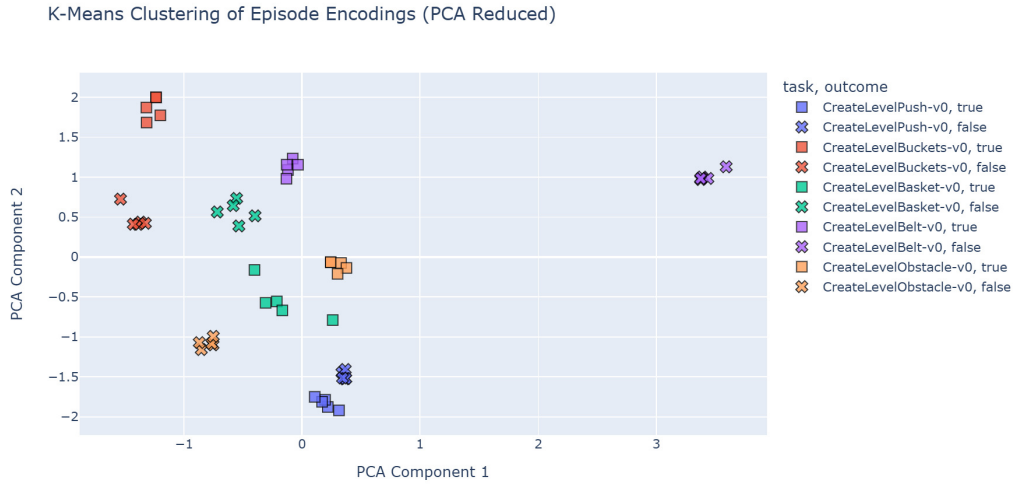
Observations on Inter-task Improvements: The hybrid loss demonstrated inter-task improvements compared to the triplet loss:

- **Task Separation:** The hybrid loss resulted in better separation between tasks compared to the triplet loss. In Figure 22b, tasks with shared structural dynamics (e.g., tasks using similar objects like buckets or baskets) were positioned closer in the embedding space, while maintaining distinct clusters for unrelated tasks. The hybrid loss leveraged cross-view contrastive signals (e.g., schema and meta-path views) to capture shared dynamics between tasks. This led to embeddings that better encode structural similarities and inter-task relationships, facilitating generalisation.

- **Outcome-specific Encoding:** While both loss functions distinguished successful and unsuccessful SETs within the same task, the hybrid loss brought positive and negative outcomes from the same task closer together compared to the triplet loss. This behaviour arose from the cross-view contrastive learning mechanism in the hybrid loss, which emphasised shared structural and task-specific features across both positive and negative outcomes. By leveraging complementary views (e.g., schema and meta-path), the hybrid loss encouraged the embeddings of episodes from the same task, regardless of their outcome, to cluster more closely while maintaining sufficient separation between success and failure states. This reflected a more nuanced encoding of



(a) Triplet Loss trained encoder clustering results.



(b) Hybrid Loss trained encoder clustering results.

Figure 22: Comparison of K-means clustering of SET embeddings under different loss functions. (a) Triplet loss: Provides reasonable task separation but limited inter-task alignment. (b) Hybrid loss: Enhances task separation and aligns result-specific embeddings more effectively by leveraging cross-view contrastive learning. Successful SETs (○) are closer within the same task, while failed ones (×) remain distinct, reflecting improved generalization and result differentiation.

the shared task dynamics and structural similarities inherent in both successful and unsuccessful trajectories.

The hybrid loss significantly enhanced the representation of inter-task relationships while maintaining strong intra-task and outcome-level differentiation, by using cross-view contrastive learning in addition to triplet-based alignment.

Comparison with State-of-the-Art Heterogeneous Graph Encoders While state-of-the-art heterogeneous graph encoders such as HeCo and GTC excel in static graph settings and node-level tasks like classification and clustering, they are not directly comparable to SETLE due to differences in problem setting, data structure, and application focus. SETLE addresses the unique challenge of encoding episodic, dynamic graphs, where each episode forms a structured subgraph (SET) representing trajectories. Unlike the static, node-centric graphs typically used by HeCo and GTC, the data in SETLE comes from episodic environments like CREATE, where the graphs evolve dynamically over time and must encapsulate hierarchical abstractions across objects, interactions, states, affordances, and trajectories.

This fundamental difference in problem formulation and the nature of the data makes direct comparisons with existing heterogeneous graph encoders unsuitable and underscores the distinct contribution of SETLE to learning hierarchical SET representations.

Comparison on CREATE environment In Jain et al. (2020), action representations were evaluated by encoding agent actions across episodes and analysing their separability within the embedding space. Their results highlighted effective clustering of actions, particularly for distinguishing tasks requiring distinct agent behaviours.

In contrast, SETLE shifted the focus to trajectory representations, encoding not only the actions but also the intermediate states, transitions, and affordances that cumulatively define the task's success or failure. The comparison reveals that action representations, as emphasised by Jain et al. (2020), excel in precision for short-term predictions, such as determining the next best action in a sequence. SETLE, however, takes a broader approach by also learning interaction representations using a ConvLSTM architecture and then integrating them with state, affordance, and object information to enable comprehensive SET representation learning. By combining detailed

interaction-level encoding with higher-level abstractions of task trajectories, SETLE provides a unified framework that supports both immediate decision-making and generalisation across tasks. This integration highlights the complementary nature of interaction and trajectory representations, with SETLE excelling in task-level analysis while maintaining robust interaction representation capabilities.

5.4.2 S2: Ablation Studies Emphasising the Model's Design

To further investigate the importance of SETLE's hierarchical structure, we conducted three additional experiments: random state removal, sequentiality disruption, and flattened representations. These experiments highlighted the robustness of the model and its reliance on structured representations for accurate outcome embeddings.

1. Random State Removal

Objective: Demonstrate the importance of sequential states in accurately representing SETs.

Design: Randomly removed a fixed proportion (e.g., 10%, 20%, 40%) of the states in the graph. Encode the modified SET using SETLE and compare clustering quality. Below we presented the results of removing 40% of the states, having observed that for lower values the results were insignificant.

Results:

Table 2: Comparison of Clustering Results for Random State Removal vs. Complete Graphs

Metric	State Removal	Complete	Remarks
Silhouette Success	0.756	0.797	Reduced cohesion and separation of success clusters.
Silhouette Failure	0.827	0.820	Less cohesive clustering.
DBI Success	0.472	0.345	Higher overlap and less distinct success clusters.
DBI Failure	0.298	0.314	Marginally better cluster definition.
Dunn Index Success	1.838	2.561	Reduced separation and compactness of success clusters.
Dunn Index Failure	2.904	2.618	Less compact and separated failure clusters.

Remarks: State removal disrupted the sequential flow and relational dependencies, leading to poorer cluster cohesion and separability (see Table 2).

2. Sequentiality Disruption

Objective: Test how breaking the sequential order of states impacts SET representation.

Design: Randomly shuffled the sequence of states in each episode while keeping all nodes intact.

Remarks: Sequential disruption reduced the ability to encode cumulative effects, leading to embeddings that are less meaningful and cohesive.

Results:

- **Silhouette Scores:** Shuffled sequences exhibit lower clustering quality (success: 0.785 compared to 0.748).
- **Dunn Index:** Noticeable degradation in success cluster compactness (2.285 compared to 1.613).
- **DBI:** Increased overlap between shuffled task clusters (success: 0.378 compared to 0.494).

3. Flattened Representations

Objective: Tested the importance of hierarchical elements in trajectories by replacing the hierarchical graph with a flat graph structure.

Remarks: Flattening the graph significantly degraded the quality of the learnt task representations, highlighting the importance of relational data for meaningful task dynamics (see Table 3).

Results:

Table 3: Impact of Flattening SET Hierarchical Structure on Clustering Results

Metric	Flattened	Complete	Impact of Flattening
Silhouette Success	0.705	0.770	Decreased cohesion and separation of success clusters.
Silhouette Failure	0.782	0.823	Reduced distinct boundaries for failure clusters.
DBI Success	0.595	0.473	Increased overlap and less distinct success clusters.
DBI Failure	0.425	0.305	Higher overlap between failure clusters.
Dunn Index Success	1.326	1.652	Reduced separation and compactness of success clusters.
Dunn Index Failure	1.864	2.888	Poorer separation and compactness of failure clusters.

5.4.3 Performance Evaluation in Reduced-Complexity Environments: MiniGrid

While the CREATE environment (Jain et al., 2020) showcased SETLE's ability to learn from rich, dynamic, and continuous visual environments, a central hypothesis of this thesis is that the same underlying mechanism can generalise across varying representational regimes—including low-resolution, symbolic environments. To test this, we extended SETLE to the MiniGrid benchmark suite, a grid-based environment that offers highly abstract, discrete observations and minimal visual complexity.

Unlike CREATE, MiniGrid lacks continuous interactions or a persistent inventory system and features a limited, discrete action space. Accordingly, the graph construction process was modified: "Action Representation" nodes, previously used to encode rich agent-tool-object dynamics, were omitted. The resulting MiniGrid-specific SETs contain only Object, State, and Affordance nodes, linked via relational edges that trace the agent's symbolic trajectory.

This adaptation serves to isolate and evaluate the contribution of SETLE's graph-based inductive bias in a minimal setting.

To mitigate the limitations of low-resolution symbolic input in MiniGrid, further refinements were applied to the object extraction pipeline. Rendered frames were first pre-processed using contrast equalisation and spatial upscaling to enhance visual saliency. Each segmented crop was then passed through CLIP (Ramesh et al., 2022) to obtain a robust vision-language embedding. CLIP was chosen explicitly over conventional encoders such as ResNet (He et al., 2016) or VGG (Simonyan and Zisserman, 2014) because these standard architectures, trained on natural image statistics, catastrophically fail when applied to synthetic grid-based environments with flat colours and uniform textures. CLIP, by contrast, demonstrated the ability to align symbolic patterns with semantic concepts even in low-visual-complexity domains. This design choice was critical for preserving the object-centric nature of the SETLE framework and maintaining consistency between the original CREATE experiments and the MiniGrid adaptation.

Once the static objects were extracted and stored, SETLE followed its standard graph memory construction: State nodes were created at each timestep, Affordance nodes were linked to represent the agent's chosen action at each transition, and the object nodes were consistently connected to all subsequent states in the episode.

We trained the SETLE encoder on MiniGrid episodes using the same procedure as described for CREATE. Figure 23 illustrates the impact of varying the triplet loss margin α in the hybrid loss function during training. As in CREATE, all models converge to similarly low loss values, with lower margins (e.g., 0.2 and 0.5) exhibiting slightly smoother convergence. However, convergence stability alone does not capture the quality of the learned embeddings. To better assess the impact of margin settings, we evaluated clustering performance using the same standard metrics: Silhouette Score, DBI and Dunn Index, computed separately for success and failure clusters. For example, with $\alpha = 0.2$, we observed a strong Silhouette Score of 0.69

for successful episodes and a low DBI of 0.45, indicating compact and well-separated clusters. By contrast, $\alpha = 1.2$ achieved a slightly lower silhouette (0.57) and a higher DBI (0.78), suggesting less distinct cluster boundaries. In contrast to CREATE, where higher margin values (e.g., 1.2 and 1.5) were needed to enforce separation between similar-looking episodes due to visual simplicity and frame-level redundancy, MiniGrid presented a reverse pattern. Despite its symbolic and low-resolution observations, MiniGrid's discrete and abstract representation space contains more structured and semantically distinct transitions. As a result, lower margin values (e.g., 0.2 and 0.5) were sufficient to stabilise training and encourage meaningful embedding separation.

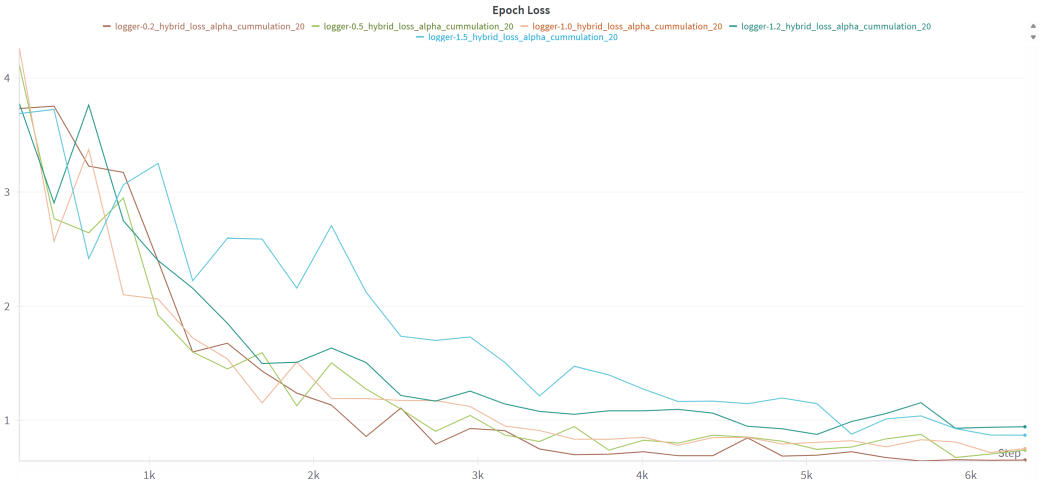


Figure 23: Epoch-wise training loss curves for different values of the hybrid loss weight α during training with alpha cumulation.

We now turn to visual clustering analyses to further examine this effect. Using PCA-reduced embeddings and K-Means clustering, we visualise how different tasks and outcomes are organised in the latent space. These findings provide insights into the agent's ability to separate task semantics, represent outcome distinctions, and recognise structural task similarity.

The results reveal several important trends. First, as shown in Fig. 24, tasks with structurally distinct layouts—such as "MultiRoom-N4-S5-v0", "UnlockPickup-v0" and "SimpleCrossing-S9N1", form clearly separated clusters, suggesting that the encoder successfully captures high-level differences in spatial configuration and affordance structure. When both successful and

unsuccessful episodes are considered (Fig. 25), we observe a consistent separation between outcomes, even within the same task. This indicates that SETLE encodes not only the task setting but also the behavioural trajectory and its result, capturing outcome-sensitive variations in agent interaction.

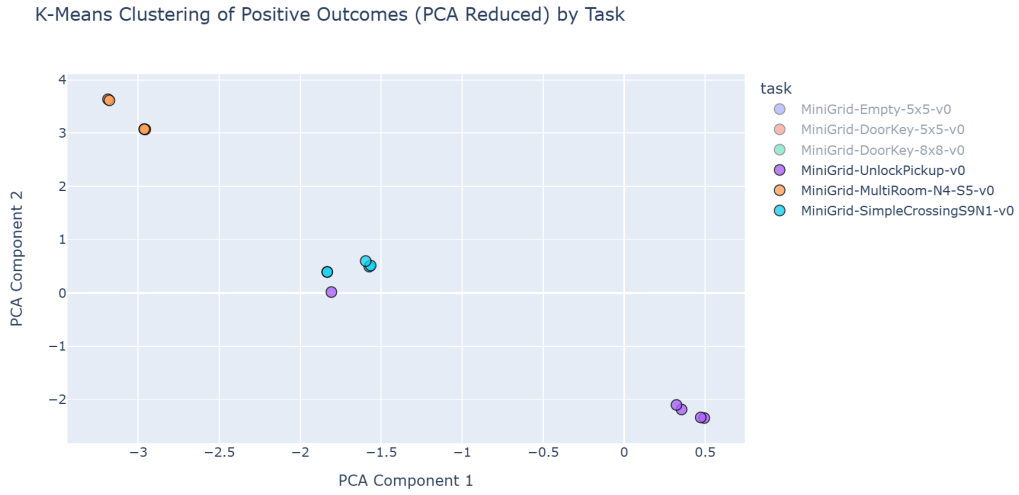


Figure 24: PCA-reduced embeddings of successful episodes from multiple MiniGrid tasks, clustered using K-Means. Tasks with structurally distinct configurations—such as MultiRoom-N4-S5-v0, SimpleCrossing-S9N1 and UnlockPickup-v0, form clearly separated clusters, indicating that the encoder captures meaningful variations in spatial layout and task structure.

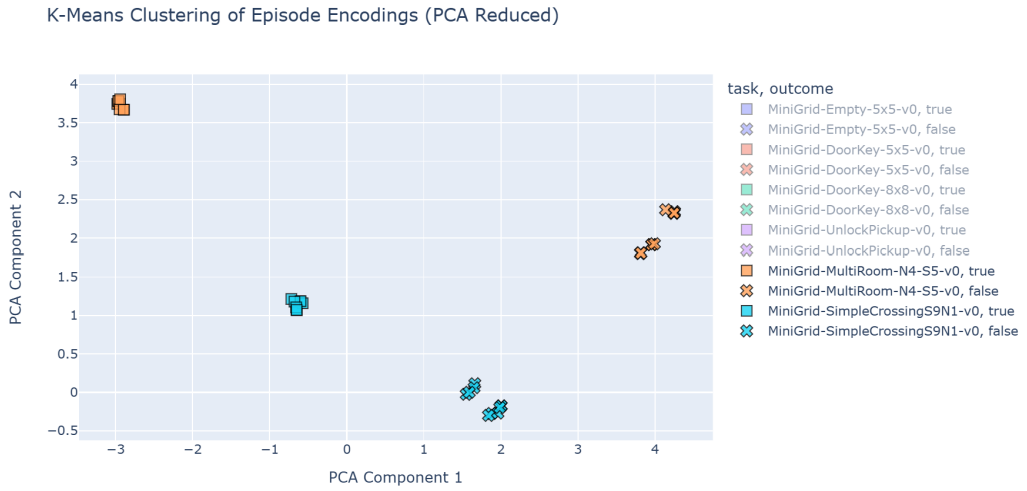


Figure 25: K-Means clustering of PCA-reduced episode embeddings across tasks: MultiRoom-N4-S5-v0 and SimpleCrossing-S9N1, including both successful and unsuccessful outcomes. While task identity still influences clustering, clear separation between positive and negative outcomes emerges, highlighting the encoder's sensitivity to behavioural results and trajectory success.

Interestingly, this separation is not universal: in simpler tasks such as "Empty-5x5-v0" and "DoorKey-5x5-v0", the embeddings overlap significantly, despite the fact that the latter includes a goal-directed key-door interaction. This overlap is further confirmed in Fig. 26, which focuses exclusively on positive outcomes from these two tasks. Notably, "Empty-5x5-v0" can be interpreted as a simplified form of "DoorKey-5x5-v0", where the spatial layout and goal location remain constant, but the constraint introduced by the door is removed. The encoder appears to capture this equivalence, treating both tasks as functionally similar due to their shared goal structure and minimal planning horizon. Crucially, this similarity persists even though the encoder was trained to maximise discriminability between episodes, indicating that SETLE's representations are not just outcome-aligned but also sensitive to deeper notions of task equivalence.

This alignment between simple tasks becomes even more apparent when extending the comparison to larger configurations. In particular, "DoorKey-8x8-v0", a more spatially demanding version of "DoorKey-5x5-v0", remains

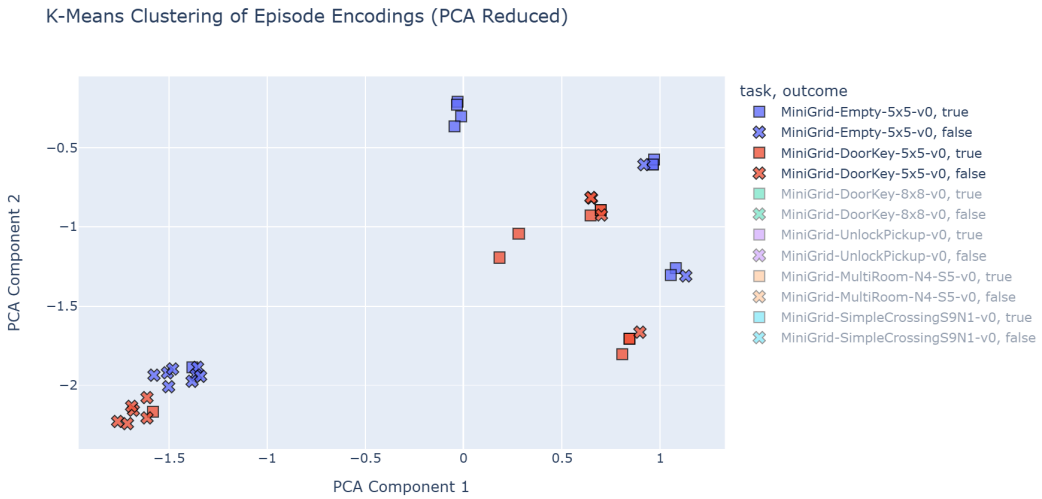


Figure 26: K-Means clustering of PCA-reduced episode encodings for "Empty-5x5-v0" and "DoorKey-5x5-v0", including both successful and unsuccessful outcomes. While outcome-specific clusters are clearly distinguishable within each task, the two tasks themselves show substantial representational overlap—indicating that the encoder treats them as structurally similar despite the presence of an additional key-door interaction in the latter.

tightly clustered alongside the simpler "Empty-5x5-v0" and "DoorKey-5x5-v0" tasks when considering only positive outcomes (Fig. 27). This reinforces the hypothesis that SETLE encodes task representations not purely based on grid size or visual complexity, but rather on the underlying goal structure and interaction demands. Despite being trained to distinguish episodes through a contrastive objective, the encoder learns to cluster trajectories that share similar causal and functional properties. At the same time, outcome-specific variation is still well represented: as shown in Fig. 28, positive and negative trajectories from the same task are consistently separated. This highlights SETLE's ability to encode both high-level structural similarity and fine-grained behavioural distinctions, making it well-suited for trajectory-level reasoning and generalisation.

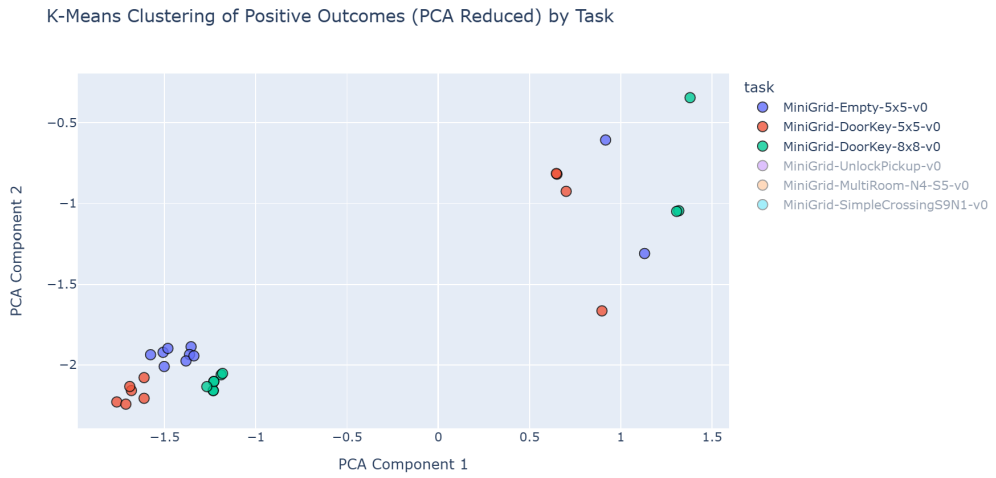


Figure 27: Clustering of positive outcomes across "Empty-5x5-v0", "DoorKey-5x5-v0", and "DoorKey-8x8-v0", visualised using PCA. The tight grouping across different grid sizes and levels of visual complexity suggests that SETLE encodes similarity based on functional structure and goal dynamics, rather than superficial spatial variation.

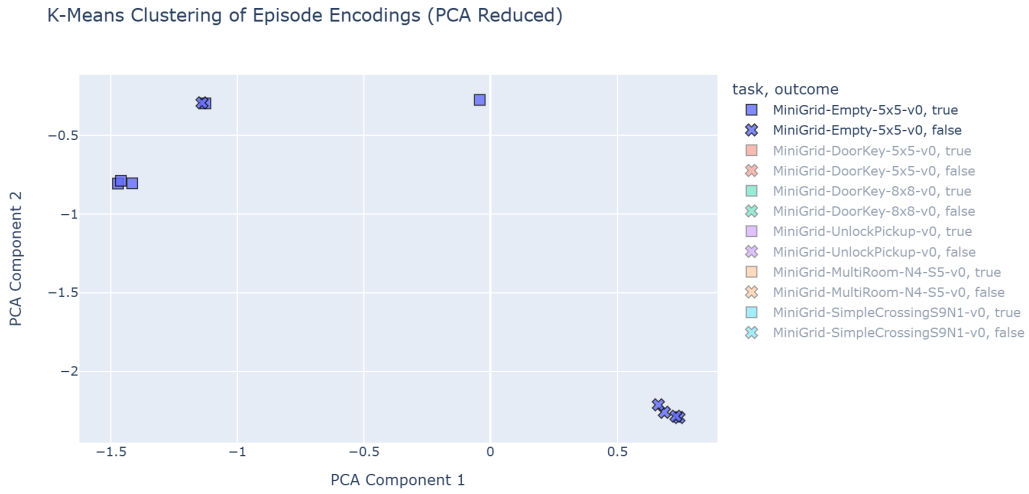


Figure 28: K-Means clustering of full episodes for "Empty-5x5-v0" reveals strong separation between successful and unsuccessful trajectories, even within a single task. This demonstrates that SETLE captures fine-grained behavioural differences and can distinguish between functionally meaningful outcomes.

The clustering results in MiniGrid confirm that SETLE effectively captures both structural task similarity and outcome-based distinctions across a variety of environments. The model generalises across tasks with shared goals and low interaction complexity—such as Empty and DoorKey—while clearly separating those with more complex layouts or affordance demands.

5.4.4 Future Experiments on Complex Environments

Future work will focus on evaluating SETLE in more complex, real-world environments that involve richer dynamics, higher-dimensional state representations, and more diverse task structures. Such environments could include robotics simulations. However, accessing these environments poses significant challenges. The computational complexity of encoding high-dimensional, multi-level graphs from real-world tasks can become prohibitive, requiring both advanced hardware and efficient learning algorithms.

Despite these challenges, applying SETLE in such scenarios could provide valuable insights into its scalability and robustness, further demonstrating

its ability to generalise across diverse and complex task settings. This would solidify its role as a framework capable of bridging the gap between dynamic, task-aware representations and real-world applications.

To explore how structured representations support learning in dynamic environments, in the next chapter we examine the integration of SETLE into the reinforcement learning loop. This stage operationalises the AIGenC framework's capacity for adaptive decision-making by using SETLE's enriched trajectory encodings to guide policy optimisation. Through this integration, we evaluate how reflective memory, structural abstraction, and context-aware retrieval improve the agent's ability to generalise, transfer, and act effectively in sparse and variable task settings.

6 AIGenC in the Loop: Integrating Structurally Enriched Trajectories into Reinforcement Learning

To move from architecture to application, this chapter details the integration of SETLE into the reinforcement learning training loop, thereby operationalising the full AIGenC framework within an interactive agent. While earlier chapters introduced AIGenC's three components—Concept Processing, Reflective Reasoning, and Blending—and described the design and function of SETLE as an instantiation of Reflective Reasoning, this section shows how these elements work together during the agent's learning process.

We describe how structured representations are built in real-time through Concept Processing, how episodic graphs are encoded and enriched with relevant past knowledge via SETLE, and how this enriched state supports generalisation and adaptation during action selection. We also show how memory is updated incrementally, enabling the agent to form increasingly abstract conceptual structures that support creative problem-solving over time.

This chapter thus completes the pipeline, from perceptual abstraction, through memory-guided reasoning, to adaptive behaviour—realising the AIGenC architecture in full within a functioning reinforcement learning agent.

6.1 Overview of the Baseline RL Architecture

To contextualise the performance of SETLE-enriched agents, we begin by outlining the baseline reinforcement learning (RL) setup used in our experiments. This architecture serves as the foundation upon which all enrichment mechanisms are evaluated.

The baseline agent is based on a Double Deep Q-Network (Double DQN) (Van Hasselt et al., 2016), selected for its stability in Q-value estimation and compatibility with mixed observation and action modalities. The agent uses two Q-heads: one for discrete action selection and one for continuous control, depending on the environment.

The baseline agent observes the raw state (for both environments), encodes it using a convolutional or linear encoder, and selects actions via an epsilon-greedy policy. The Q-network is updated using experience replay and periodic target network updates.

While simple, Double DQN provides a strong and interpretable baseline. Its lack of structured memory, graph-based reasoning, or conceptual abstraction makes it an ideal comparison point for assessing the contribution of the AIGenC framework.

We evaluate across two distinct domains:

CREATE Environment: A visually rich, physics-based environment designed for object manipulation and tool use. Tasks include pushing balls into containers, navigating obstacles, and achieving complex multi-step goals. CREATE is particularly suited for testing structured planning, affordance learning, and behavioural reuse.

MiniGrid Suite: A minimalist 2D gridworld composed of symbolic entities (e.g., keys, doors, goals) and agent-centric observations.

6.1.1 Training Procedure (CREATE Baseline)

The baseline agent for the CREATE environment is trained using a Deep Q-Network (DQN)-style architecture adapted for a **hybrid action space**. At each timestep, the agent selects a tool from its inventory (discrete action) and predicts a spatial coordinate (x, y) (continuous action) at which to apply the selected tool. Formally, each action is represented as:

$$a_t = \langle o_i, (x, y) \rangle$$

where o_i is a tool from the current inventory, and (x, y) denotes the interaction location in the environment.

To handle this mixed action structure, we adopt a **multi-headed Q-network** that outputs:

1. A categorical Q-value vector over all possible tools (up to 939), masked by the current inventory.
2. A continuous 2D vector representing the predicted coordinates (x, y) for tool application.

This follows the *multi-headed architecture* approach to hybrid action modelling, allowing independent training of discrete and continuous components while enabling joint decision-making.

Continuous Coordinate Prediction via Regression. A key design choice in our architecture concerns the handling of continuous spatial actions. In standard continuous-action reinforcement learning (Lillicrap et al., 2015), the agent must solve $a^* = \arg\max_a Q(s, a)$, which requires either iterative optimisation over the action space or a separate actor network that learns to output actions maximising expected return.

Our approach differs fundamentally: rather than treating (x, y) coordinates as actions to be optimised, we frame spatial prediction as a **supervised regression task** conditioned on the selected discrete tool. Specifically, once the discrete Q-head selects tool o_i via $\arg\max$, the continuous head outputs a deterministic coordinate prediction $(x, y) = f_\theta(s_t, o_i)$ trained to minimise the mean squared error against observed successful interaction locations.

This design sidesteps the need for continuous action-space maximisation entirely. The regression target is derived from environment feedback: when an action (o_i, x, y) yields positive reward, the coordinate (x, y) serves as a supervised signal for that tool-state pair. This is feasible in CREATE because:

1. **Spatial structure is learnable**: Successful interaction locations exhibit regularities tied to object positions and tool affordances, making regression a viable learning objective.
2. **Discrete selection dominates**: The primary decision, which tool to use, is handled by the discrete Q-head with standard $\arg\max$ selection. The continuous head refines *where* to apply the already-chosen tool.
3. **Exploration covers the space**: During ϵ-greedy exploration, random (x, y) samples provide diverse training data for the regression head, ensuring coverage of the coordinate space.

This hybrid approach trades off the theoretical guarantees of continuous action-value optimisation for practical simplicity and stability. While it assumes that coordinate prediction can be learned as a function of state and tool, rather than requiring explicit value-based selection, this assumption holds well in environments like CREATE where spatial affordances are visually grounded. We note that extending this approach to domains with less structured action-location relationships may require revisiting the action-value formulation, potentially incorporating actor-critic methods for the continuous component.

Action Selection Strategy. The agent follows an ϵ-greedy policy:

- With probability ϵ, it explores by randomly sampling a tool from the inventory and a random valid position.
- Otherwise, it exploits the learned Q-function by selecting the tool with the highest Q-value (subject to masking) and uses the network's predicted position.

The exploration rate decays exponentially over time:

$$\epsilon_t = \epsilon_{\text{end}} + (\epsilon_{\text{start}} - \epsilon_{\text{end}}) \cdot e^{-t/\text{decay\_rate}} \tag{33}$$

with parameters:

$$\epsilon_{\text{start}} = 1.0, \quad \epsilon_{\text{end}} = 0.1, \quad \text{decay\_rate} = 10^4$$

Learning. Both Q-heads are trained jointly using a combination of:

- Temporal-Difference (TD) loss for the discrete tool selection.
- Regression loss for coordinate prediction.

The discrete Q-head is initialised using *Xavier uniform weights*, and positive bias initialisation is applied to encourage early exploration. All training metrics—Q-values, rewards, loss breakdown, and ϵ decay—are logged using Weights & Biases (W&B) and will be discussed in the Results subsection.

Algorithm 3 Hybrid Action DQN Training Loop

Initialize Q-network Q_θ with:
    1. Discrete head for tool selection
    2. Continuous head for coordinate prediction
Initialize target network Q_{θ^-}, experience buffer $\mathcal{D}$
Set $\epsilon \leftarrow \epsilon_{\text{start}}$
for each episode **do**
    Reset environment and tool inventory
    for each timestep t **do**
        if random $< \epsilon$ **then**
            Sample tool $\hat{a}$ uniformly from inventory
            Sample $(\hat{x}, \hat{y})$ uniformly from valid space
        else
            $\hat{a} \leftarrow \arg\max Q_\theta(s_t, a)$ (masked by inventory)
            $(\hat{x}, \hat{y}) \leftarrow Q_\theta^{\text{coord}}(s_t, \hat{a})$
        Execute action $(\hat{a}, \hat{x}, \hat{y})$, observe r_t, s_{t+1}
        Store (s_t, a_t, r_t, s_{t+1}) in buffer $\mathcal{D}$
        Sample mini-batch from $\mathcal{D}$ and update Q_θ
        Update target network periodically: $Q_{\theta^-} \leftarrow Q_\theta$
        Decay ϵ using exponential schedule

6.1.2 Training Procedure (MiniGrid Baseline)

In contrast to the CREATE environment, where agents interact through a hybrid action space and inventory-based tool application, the MiniGrid setup features a **purely discrete action space**. The agent must learn to navigate, interact with objects, and solve tasks such as key-door unlocking and room traversal using a fixed set of primitive discrete actions: `move forward`, `turn left`, `turn right`, `pickup`, `drop`, and `toggle`.

To preserve consistency with the object-centric framing of the CREATE experiments while avoiding reliance on symbolic state input, the baseline MiniGrid agent receives only **pixel-level visual observations** via `env.render()`. It does not have access to internal simulator state or structured symbolic observations such as `obs["image"]`. This design choice aligns the setup with real-world perception constraints, where agents must extract actionable information from raw sensory input. Given MiniGrid's discrete and minimalistic action space, a number of structural and computational adapta-

tions were made to SETLE. Unlike in CREATE, where ActionRepresentation nodes captured rich object-tool interactions, the MiniGrid graph structure was simplified. It included only State and Affordance nodes, connected by `influences` and `outcome` edges. This simplification allowed direct testing of the thesis claim that the benefits of structured episodic memory do not stem from predefined semantics but from the flexible object-centric trajectory modelling enabled by SETLE.

To reduce computational overhead, the visual perception module was adapted. The Segment Anything Model (SAM) and CLIP were used only **once at the beginning of each episode**, leveraging the fact that MiniGrid environments are static within each episode. Extracted objects (e.g., walls, keys, doors) remain unchanged during interaction, and are thus linked to every State node throughout the episode graph without needing re-inference. This drastically reduced GPU usage and inference time, while still ensuring that the episodic graph maintained its conceptual integrity.

For the reinforcement learning architecture, a standard **Double DQN** agent is used. The Q-network consists of a convolutional encoder followed by a fully connected Q-head predicting discrete action values. Exploration follows an ϵ-greedy schedule:

$$\epsilon_t = \epsilon_{\text{end}} + (\epsilon_{\text{start}} - \epsilon_{\text{end}}) \cdot e^{-t/\text{decay\_rate}} \tag{34}$$

with:

$$\epsilon_{\text{start}} = 1.0, \quad \epsilon_{\text{end}} = 0.1, \quad \text{decay\_rate} = 10^4$$

The agent is trained using standard temporal-difference loss over mini-batches sampled from an experience replay buffer, and the target network is periodically updated. No memory augmentation or enrichment is used in this baseline.

Training is conducted across six standard MiniGrid tasks: `Empty 5x5`, `DoorKey 5x5`, `DoorKey 8x8`, `UnlockPickup`, `MultiRoom N4 S5`, and `SimpleCrossing S9N1`. These tasks collectively test basic navigation, symbolic object use, and multi-room spatial reasoning.

This baseline serves as the foundation for evaluating whether structured memory and enrichment mechanisms, introduced in subsequent sections, yield improvements in sample efficiency, stability, and generalisation across task variants.

6.2 SETLE Integration in the Reinforcement Learning Loop

To evaluate the impact of structured episodic memory on generalisation and learning efficiency, we integrate SETLE into the reinforcement learning (RL) training loop. SETLE implements the *Reflective Reasoning* component of the AIGenC framework, enabling the agent to supplement its current experience by matching recent trajectories against long-term memory (LTM), retrieving relevant structures, and enriching its working memory (WM) before selecting an action.

Algorithm 4 SETLE Matching and Enrichment at Timestep t

Require: Partial trajectory $G_{t-3:t}$, current *state_id*, *episode_id*, SETLE encoder, LTM

Ensure: Enriched working memory graph at time t

1: Encode partial trajectory: $z_{\text{query}} \leftarrow$ `encode_episode`$(G_{t-3:t})$
2: Retrieve LTM episode encodings $\{(z_i, \text{episode}_i)\}_{i=1}^{N}$
3: Compute similarity: $s_i = \cos(z_{\text{query}}, z_i)$
4: Select top-K most similar episodes: `top_episodes` $\leftarrow$ `argsort`$(s)[:K]$
5: **for** each matched `episode`$_j$ **do**
6: Extract candidate nodes (objects, affordances, actions) not in G_t
7: Apply relevance and attention filtering
8: Select top-N nodes by attention score
9: Inject selected nodes and edges into WM graph at *state_id* with `source=setle`

Functional Overview. Unlike standard RL agents that rely exclusively on immediate state observations, SETLE allows the agent to draw from structurally similar prior experiences. Rather than reusing entire episodes, the agent selectively retrieves and injects semantically aligned substructures, such as objects, affordances, and actions, into the current WM graph. This supplemented graph is then re-encoded and passed to the policy network, forming a cycle of perception, memory-guided reasoning, and decision-making.

The retrieval and injection mechanism in lines 18–24 of Algorithm 5 can operate in two modes, corresponding to Reflective Reasoning and Blending as defined in Sections 4.2 and 4.3:

- **Reflective Reasoning mode**: When the agent is performing ade-

quately (no impasse detected), retrieval uses a high similarity threshold, returning only closely matching episodes. Injected concepts are familiar and reinforce known strategies.

- **Blending mode**: When repeated failures trigger an impasse condition, the similarity threshold is relaxed, allowing retrieval of more distant episodes. This introduces diverse concepts that may enable novel solutions.

In both modes, the current implementation uses the same attention-weighted injection mechanism. The distinction lies in *what* is retrieved (similar vs. diverse) and *when* (continuous vs. impasse-triggered), not in how retrieved concepts are integrated. Full latent-space blending—where diverse concepts are fused into genuinely new representations—remains a direction for future work.

To support temporal abstraction and contextual matching, SETLE encodes a rolling window of the last $k = 4$ state graphs as a partial trajectory, denoted $G_{t-3:t}$. This representation captures both spatial and temporal dependencies, helping disambiguate states that may appear similar in isolation but differ in their broader interaction history. These richer representations improve the precision and utility of memory retrieval.

The core mechanism of SETLE follows a selective matching and supplement pattern. The agent compares the current partial trajectory $G_{t-3:t}$ to stored episodes in LTM, full encoded trajectories, and identifies structurally similar ones. From these, it extracts missing but relevant elements and injects them into the current WM graph. This interaction is formalised in Algorithm 4.

To identify what knowledge is most relevant, SETLE employs a trainable attention mechanism that scores candidate nodes based on their contextual alignment with the current partial trajectory. This targeted prioritisation enables fine-grained control over memory reuse, ensuring that only structurally and semantically pertinent components are integrated into the working memory graph.

The selected nodes and edges form a *supplement* to the current graph G_t, producing an enriched graph G_t^* that includes potentially useful but previously absent information. This enriched graph is re-encoded using the SETLE GNN encoder and passed through an adapter layer before being input to the policy network.

The integration of SETLE modifies the standard RL loop as follows:

1. Encode the last four states as a partial trajectory graph.
2. Retrieve the top-k most similar episodic embeddings from LTM.
3. Expand the episodes to be able to select the needed concepts.
4. Filter retrieved graph concepts by context and apply attention to rank and select the most relevant candidates.
5. Inject selected nodes and edges to supplement the current WM graph.
6. Re-encode the enriched graph and pass it to the policy head.

6.2.1 Two Approaches to Attention: Mackintosh vs. Transformer-Based Mechanisms

To investigate the impact of relevance-driven enrichment on learning and generalisation, we implemented and evaluated two distinct attention mechanisms for concepts selection:

1. Mackintosh-Based Associative Attention (Computational Attention). This biologically inspired mechanism is adapted from the Mackintosh model of associative learning. It treats attention as a scalar *associability value* that changes over time based on how reliably a concept predicts reward (for full implementation details and plots of the Mackintosh associative model, see Appendix B). Each concept's associative strength $V_A^{(n)}$ and associability $\alpha_A^{(n)}$ are updated at every timestep using the following rule:

$$\Delta V_A^{(n+1)} = \alpha_A^n \cdot \beta \cdot (1 - V_A^n + \overline{V_A^n}) \cdot |R^n|$$

$$\Delta \alpha_A^{(n+1)} = -\Theta \cdot \left(|\lambda^n - V_A^n + \overline{V_A^n}| - |\lambda^n - V_{\neg A}^n + \overline{V_{\neg A}^n}| \right)$$

Concepts that repeatedly co-occur with high-reward outcomes increase their associability and are more likely to be retained and reused. This attention model is computationally efficient and interpretable, but it is agnostic to structural embeddings or local task context at retrieval time. It is best suited for deciding what concepts should be stored long-term based on statistical regularities.

2. Transformer-Style Learned Attention. In contrast, this mechanism applies attention over node embeddings using a vector projection architecture inspired by the Transformer model. Given an encoded partial trajectory (query), and candidate node embeddings (keys), we compute similarity via:

$$\alpha_i = \frac{\exp\left(\langle W_q \cdot \text{query},\ W_k \cdot \text{candidate}_i \rangle / \tau\right)}{\sum_j \exp\left(\langle W_q \cdot \text{query},\ W_k \cdot \text{candidate}_j \rangle / \tau\right)}$$

This enables the model to learn a flexible, task-specific notion of relevance, allowing selective injection of nodes from retrieved episodes during runtime. It operates entirely at inference time and supports generalisation across diverse task structures by aligning latent concept representations.

In our architecture, the two attention systems are not redundant but complementary:

1. Associative strength determines which concepts are stored in LTM.
2. Transformer-based attention determines which concepts are retrieved and injected into WM.

Algorithm 5 AIGenC RL Loop with SETLE Enrichment, Mackintosh Associability, and Graph Supplementation

Initialize: LTM $\leftarrow \langle \texttt{keys}[], \{\}\rangle$, WM $\leftarrow \langle \texttt{keys}[], \{\}\rangle$
Parameters: Associability $\alpha_A = 0.5$, Learning rate β, Attention temperature τ, Decay rate Θ
for *episode* $= 1, 2, \ldots, N$ **do**
  $G_{episode} \leftarrow []$ ▷ *Initialize episode trace*
  env $\leftarrow$ Reset environment
  for $t = 1, \ldots, T$ **do**
    $s_t \leftarrow$ observe environment state
    // **Concept Extraction and Graph Construction**
    $G_t \leftarrow$ create_state_graph(s_t, inventory, env)
    Append G_t to $G_{episode}$
    // **Matching and Enrichment (SETLE)**
    if $t \geq 4$ **then**
      $G_{t-3:t} \leftarrow$ extract_partial_trajectory($G_{episode}$)
      $z_{query} \leftarrow \texttt{encode\_episode}(G_{t-3:t})$
      Retrieve $\{(z_i, \texttt{episode}_i)\}_{i=1}^{N}$ from LTM
      Compute similarities $s_i \leftarrow \cos(z_{query}, z_i)$
      topK $\leftarrow$ top-K episodes by s_i
      // — **Blending Operation (see Section 4.3)** —
      for each episode j in *topK* **do**
        Extract candidate nodes (affordances, objects, actions)
        Compute attention weights α_j using:

$$\alpha_j = \texttt{softmax}\left(\frac{(W_q z_{query}) \cdot (W_k z_j)}{\tau}\right)$$

        Select top-N candidates by attention score
        Inject candidates into G_t with tag `source="setle_attention"`
      // — **End Blending** —
    // **Policy Action**
    Encode enriched G_t^* using SETLE encoder
    $a_t \leftarrow \pi(G_t^*)$ using ϵ-greedy strategy
    $s_{t+1}, r_t \leftarrow$ take action a_t in environment
    // **Store Transition**
    Add (s_t, a_t, r_t, s_{t+1}) to Replay Buffer
    // **Associative Strength Update (Mackintosh)**
    for each concept A in G_t **do**
      Compute:

$$\Delta V_A^{(t+1)} = \alpha_A^{(t)} \cdot \beta \cdot (1 - V_A^{(t)} + \overline{V_A^{(t)}}) \cdot |R_t|$$

$$\Delta \alpha_A^{(t+1)} = -\Theta \cdot \left(|\lambda - V_A^{(t)} + \overline{V_A^{(t)}}| - |\lambda - V_{\neg A}^{(t)} + \overline{V_{\neg A}^{(t)}}|\right)$$

      Update V_A and α_A
  // **Supplement and Memory Consolidation**
  $WMclusters \leftarrow$ cluster($G_{episode}$) using concept similarity
  centroids $\leftarrow$ cluster centers
  for each centroid G_c **do**
    if $\texttt{match}(G_c, \texttt{LTM})$ is False and $V_A >$ threshold **then**
      Add G_c to LTM

This dual-system setup allows SETLE to balance global statistical regularities with local task relevance—enabling both robust concept consolidation and adaptive memory reuse (see Algorithm 5).

6.3 Training Regimes and Experimental Design

To evaluate the impact of SETLE-based memory enrichment on reinforcement learning performance, we design a controlled experimental framework that compares multiple agent configurations across diverse tasks. The goal is to assess whether structured memory reuse improves learning efficiency, generalisation, and policy stability.

Multi-Task Training Protocol. Importantly, all experiments employ a **random mixture training regime** rather than sequential or curriculum-based learning. At each training step, episodes are sampled uniformly at random from the full set of available tasks. The agent does not first master one task before encountering another; instead, it experiences interleaved episodes from all tasks throughout training. This design choice serves two purposes:

1. **Testing genuine generalisation**: By exposing the agent to all tasks simultaneously, we ensure that any observed transfer or knowledge reuse emerges from the agent's internal representations rather than from sequential skill accumulation or task-specific fine-tuning.

2. **Stress-testing memory mechanisms**: Random task mixing creates a challenging setting for memory-based systems, as the agent must maintain and retrieve relevant knowledge across interleaved experiences from structurally diverse tasks.

This protocol contrasts with curriculum learning approaches, where tasks are ordered by difficulty, or continual learning setups, where tasks are presented in sequence. Our random mixture design more closely reflects real-world deployment scenarios where agents cannot anticipate which task they will encounter next.

Experimental Conditions. We evaluate five distinct training configurations, each representing a different integration strategy for SETLE into the RL loop:

1. **Baseline RL:** A standard Double DQN agent trained without any graph memory or enrichment. This serves as the control condition.

2. **Action Selection Only:** SETLE enrichment is applied exclusively at inference time. The policy network receives graph-enhanced embeddings for action selection, but gradients do not propagate through the memory pathway.

3. **SETLE for Action and Optimisation:** Graph-enriched representations are used for both policy inference and backpropagation. The encoder is updated end-to-end, allowing memory reuse to influence learning directly.

4. **Adapter + Penalty:** A learnable adapter transforms SETLE embeddings before they are fed into the Q-network. Additionally, a penalty is applied when the same LTM episodes are repeatedly matched, encouraging more diverse memory access.

5. **Adapter + Penalty + Soft Update:** This final configuration adds a target encoder updated via soft updates (Polyak averaging) to increase temporal stability and reduce volatility during enrichment.

Each configuration is tested on five CREATE tasks and a suite of MiniGrid scenarios. Experiments are conducted across three random seeds per condition to ensure statistical robustness. All runs are logged using Weights & Biases (W&B), capturing reward trajectories, Q-value trends, enrichment frequency, and memory match statistics.

Performance is assessed using a combination of behavioural and representational metrics:

- **Success Rate:** Percentage of episodes in which the goal is successfully achieved.

- **Reward Frequency:** Average number of positive reward events per episode.[1]

- **Q-Value Stability:** Variance of predicted Q-values over time, used as a proxy for value estimation reliability.

[1]We report reward frequency (nonzero events per step) rather than average or cumulative reward because the extreme sparsity of rewards in this environment causes average reward to collapse near zero for all agents. Frequency provides a more interpretable measure of how reliably agents encounter positive outcomes. Cumulative reward (Fig. 43) confirms the same pattern.

- **Embedding Quality:** Silhouette scores and clustering metrics are used to evaluate how well SETLE embeddings separate tasks or structure trajectories.

Overview of Stabilisation Mechanisms. Integrating structured memory into the RL loop introduces challenges not present in standard architectures: distributional mismatch between retrieved and learned representations, over-reliance on frequently matched episodes, and training instability from varying enriched inputs. Through systematic experimentation, we identified three complementary mechanisms that address these challenges: (1) an *adapter layer* to align memory embeddings with the policy network's representation space, (2) a *matching penalty* to encourage retrieval diversity, and (3) *soft target updates* to stabilise learning under varying enriched inputs.

The following subsections present these components incrementally, reflecting the experimental process through which they were developed and validated. Together they form an integrated approach to a stable memory-augmented reinforcement learning.

6.3.1 Results in Physically Grounded Tasks (CREATE)

Baseline vs Action Selection only strategy To evaluate the impact of SETLE-based enrichment applied only at action selection time, we compare the reward trajectories of four independent training runs on the "Create Level Buckets" task. Figure 29 overlays the baseline and enriched configurations.

Baseline agents exhibit slow, consistent learning, with reward increasing modestly over time. In contrast, SETLE-enhanced agents demonstrate significantly higher initial rewards—particularly in one run (Run 3), which peaks above 1.8 early in training. However, this reward advantage tends to decay over time, suggesting that initial enrichment may bootstrap early performance but lacks sustained refinement. Baseline runs are tightly clustered, with minimal variation in performance. The action selection only condition introduces higher variance: while one run significantly outperforms the baseline, others perform comparably. This variance reflects the dependency of enrichment efficacy on the relevance and timing of matched episodes from long-term memory.

The comparison reveals that even lightweight enrichment strategies, such as injecting prior knowledge only during policy inference, can substantially

accelerate early learning. However, these benefits are inconsistent unless coupled with mechanisms for continual memory refinement or adaptation.

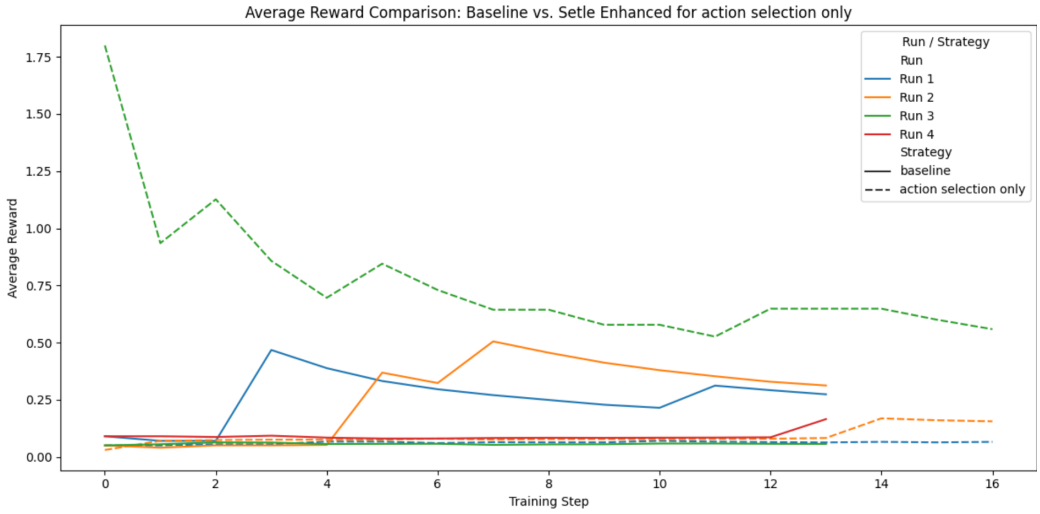


Figure 29: Average reward over time on `Create Level Buckets` for baseline and SETLE-enhanced (action selection only) agents. Solid lines: baseline; dashed lines: enriched strategy. Each *training step* corresponds to one episode of interaction with the environment, after which the policy network is updated using sampled transitions from the replay buffer.

Action selection only strategy vs Action Selection and Optimisation strategy The performance comparison of the inference vs inference and optimisation strategy Fig.30 reveals a clear advantage of the full SETLE enrichment strategy that integrates both action selection and optimisation over the action-selection-only baseline [2]. While both configurations benefit from retrieving relevant past experiences, applying enrichment only at inference time limits its long-term impact. In contrast, feeding enriched state representations into the training loss allows the agent to adjust its value func-

[2]The periodic drops in reward observed in the optimisation strategy (dashed line) result from target network updates. These updates cause a sudden shift in the learning objective, creating a temporary misalignment between the policy's learned representations and the new target values. The agent recovers and typically surpasses its previous performance level as the optimisation realigns with the updated targets, producing the characteristic staircase pattern. This instability motivates the introduction of soft updates, discussed later

tion based on structurally informed information. This leads to more stable learning, faster convergence, and improved generalisation. The Basket task, which involves precise object placements and temporal dependencies, particularly benefits from this deeper integration. By using enrichment during both forward and backward passes, the agent reinforces not just immediate decisions, but also the internal representations that govern its learning dynamics. This illustrates the key claim of the thesis: memory-based structural generalisation is most effective when it influences both decision and optimisation processes.

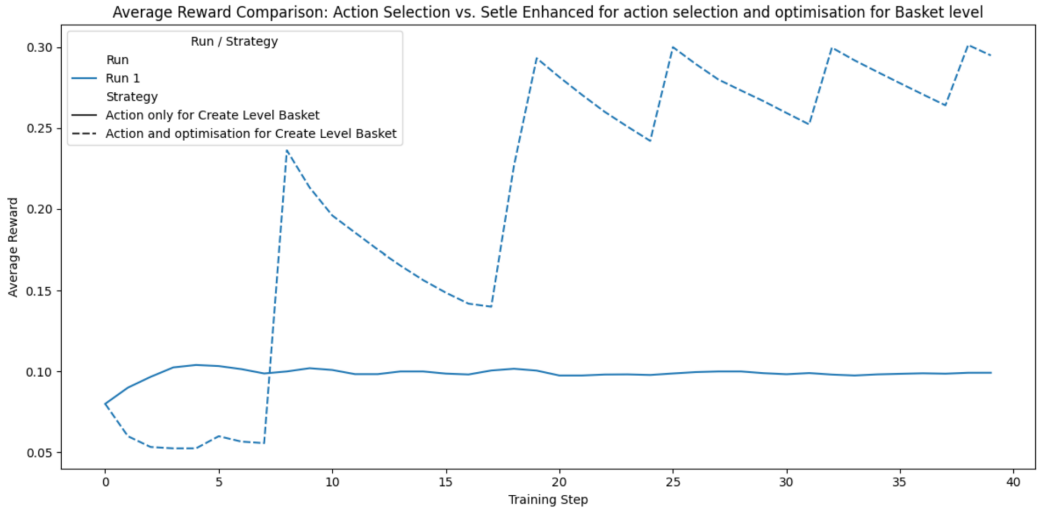


Figure 30: Comparison of average reward over time for the CreateLevelBasket task using two SETLE configurations—Action Selection Only vs. Action + Optimisation. The full optimisation strategy yields higher and more stable rewards, demonstrating the benefit of incorporating enriched memory representations during both inference and training

Further improvements to the RL loop: adapter and penalty To further stabilise and enhance the integration of memory into the reinforcement learning loop, we introduced two complementary mechanisms within the SETLE-enhanced agent: an adapter layer and a matching penalty. The adapter is a learnable projection module (implemented as a two-layer MLP) applied after graph-based enrichment. Its purpose is to map enriched state embeddings—composed of objects, affordances, and contextual relations re-

trieved from long-term memory—into a representation space better aligned with the Q-network's expectations. This addresses a critical issue in memory-based reinforcement learning: the potential distributional shift between learned embeddings and externally injected representations. Empirically, we found that the adapter significantly reduced the continuous loss (Fig. 31), and also improved the Q-value stability (Fig. 32), indicating that the enriched information became more usable for the continuous action head.

Figure 31: Comparison of training loss for the continuous (left) and categorical (right) heads under two strategies: standard SETLE enrichment and SETLE with an adapter. The adapter-based strategy achieves consistently lower loss in the continuous head and comparable or improved loss in the categorical head, indicating that the enriched embeddings are easier to optimise when mapped through a learnable projection layer.

Figure 32: Comparison of mean Q-values for the continuous (left) and categorical (right) action heads, with and without the adapter layer. The agent using the adapter (with_adapter_set_enrichment_and_matching) exhibits smoother and more stable Q-value evolution in both heads. This suggests that the adapter helps reconcile the enriched embeddings with the policy network's expectations, leading to more reliable value estimation.

The matching penalty, on the other hand, addresses a common failure mode in memory-based systems: repeated reliance on familiar or frequently matched trajectories. In earlier experiments, we observed that the agent would frequently retrieve the same high-scoring past episodes during enrichment, leading to repeated reinforcement of narrow behavioural patterns. To counteract this, we implemented a dynamic penalisation scheme that down-weights the similarity scores of frequently matched episodes. This forced the retrieval mechanism to consider a broader set of candidates, encouraging exploration and preventing overfitting to a small subset of trajectories. This penalty was particularly effective in later stages of training, where diversity in retrieved episodes was critical to maintain learning momentum and avoid premature convergence.

Together, these strategies aim to balance effective reuse of prior knowledge with the flexibility needed for robust adaptation across tasks.

To evaluate the downstream behavioural impact of the adapter and reuse penalty, we executed the full SETLE-enriched reinforcement learning loop across multiple tasks. Figure 33 focuses on the `Create Level Push` task, where agents must manipulate tools to move objects toward target zones, a task requiring spatial precision and multi-step affordance chaining.

Among the configurations tested, the adapter-based strategy consistently achieved higher average rewards over time, indicating improved sample efficiency and generalisation. This sustained performance suggests that the adapter module effectively aligns enriched graph representations with the policy network's internal feature space, allowing the agent to better exploit retrieved knowledge without destabilising learning.

While all SETLE-augmented variants, including action selection only and action+optimisation, outperformed the early baseline, the adapter + penalty strategy was unique in maintaining reward gains after the initial peak. This was particularly notable given the challenge of delayed rewards in the Push task. The adapter appeared to dampen the mismatch between static memory embeddings and dynamic task features, while the penalty term discouraged overfitting to frequently matched episodes.

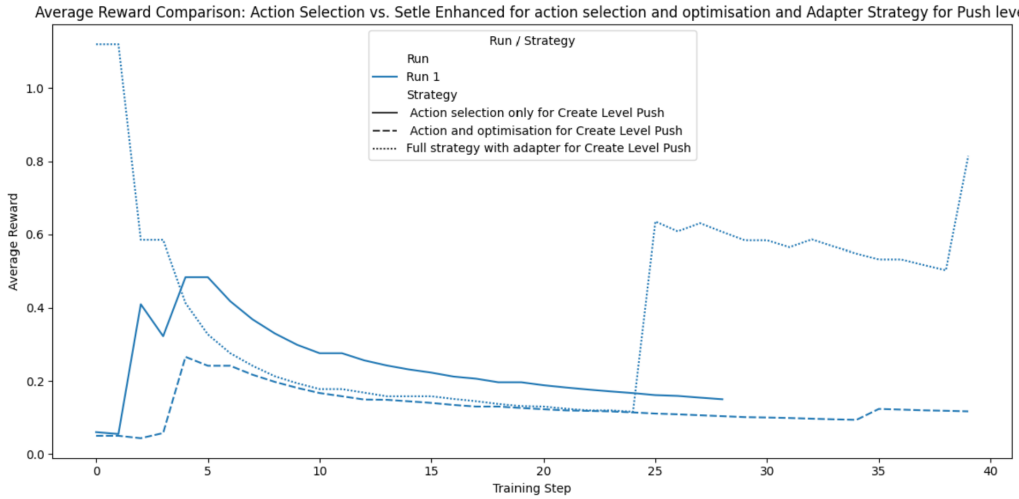


Figure 33: Average episode reward on CreateLevelPush across three SETLE integration strategies. The adapter-enhanced model (dotted line) consistently outperforms simpler strategies, demonstrating improved reward acquisition and training robustness.

Further improvements to the RL loop: soft update While the adapter-enhanced SETLE model showed improved policy learning and reward acquisition, the Q-value and loss plots (Figures 32, 31) reveal training instabilities, particularly in the form of sharp spikes in both continuous and categorical loss. These fluctuations suggest that although enriched embeddings carry

useful information, their integration introduces representational shifts that are difficult for the policy network to absorb consistently in a fully synchronous update scheme. To mitigate this, we introduce a soft target network update mechanism, which blends the parameters of the policy and target networks more gradually. By slowing the rate of change in Q-value targets, soft updates reduce oscillations and promote convergence when learning from enriched state representations that vary over time.

Figure 34: Loss curves for the adapter-enhanced SETLE agent with and without soft updates. Sharp spikes in both continuous and categorical loss indicate unstable value updates caused by abrupt changes in enriched state representations.

As shown in Figure 34, without soft updates, both continuous and categorical losses display frequent high-magnitude spikes. These indicate unstable value updates, especially when the enriched state representation changes abruptly between timesteps or episodes. By contrast, the introduction of the soft update mechanism leads to noticeably smoother training dynamics (Figure 36). All three loss components: total, continuous, and categorical, show reduced variance, lower peaks, and more consistent downward trends over time. Similarly, in the Q-value plots (Figure 35), the mean Q-values remain more stable and within narrower ranges, especially for the continuous branch. This indicates that the soft update effectively damps abrupt value shifts caused by mismatch between new enriched information and the existing value function.

Figure 35: Q-value statistics for the soft update strategy. The mean Q-values remain bounded and relatively stable over time, indicating more consistent learning dynamics when enriched state information is gradually integrated.

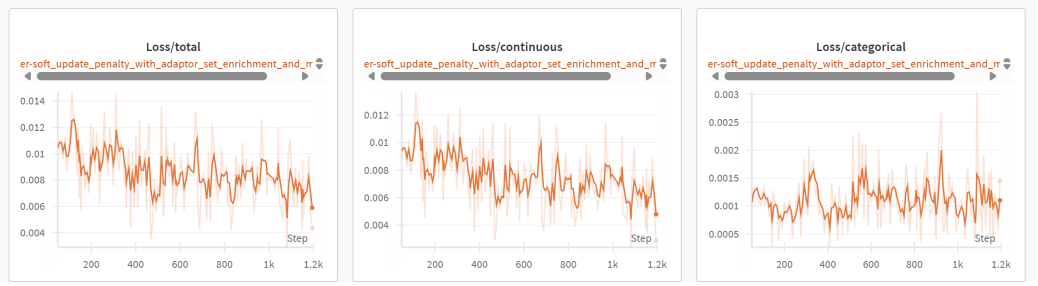


Figure 36: Loss curves for the full strategy with adapter and soft target network updates. The soft update stabilises training by reducing oscillations and suppressing large loss spikes, enabling smoother convergence.

Together, these results support the conclusion that soft updates are critical for stabilising learning when using adaptive memory systems like SETLE.

Challenges of Generalisation in the CREATE Environment The CREATE environment presents a particularly challenging setting for structured reinforcement learning due to its requirements for high interactivity, sparse rewards, and multi-step problem-solving. Unlike symbolic environments such as MiniGrid, where object interactions and transitions are discretised and low-dimensional, CREATE demands fine-grained motor control

(e.g., tool-object manipulation) and sequential reasoning over long temporal horizons (e.g., positioning a ball onto a ramp to launch it into a target).

In the preceding sections, we introduced several architectural components designed to address these challenges: SETLE-based memory enrichment, an adapter module for aligning latent spaces, a matching penalty to discourage repetitive retrieval, and a soft update mechanism to enhance stability during policy learning. While each of these mechanisms showed empirical benefits, their success depends heavily on the interplay between memory retrieval, enrichment quality, and policy plasticity. Not all runs achieve generalisation, as shown in Fig. 37.

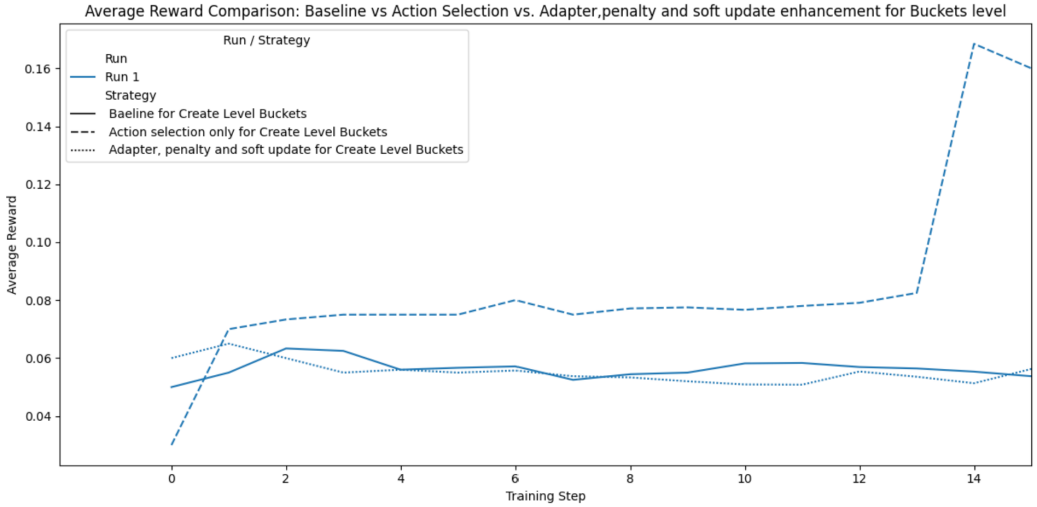


Figure 37: Some CREATE tasks remain sub-optimal even with SETLE enrichment, highlighting the difficulty of achieving consistent generalisation across runs.

Nevertheless, when memory matching and integration mechanisms align successfully, particularly under the full adapter and soft update strategy, the agent demonstrates substantial improvements in reward acquisition and trajectory quality, significantly outperforming the baseline (Fig. 38).

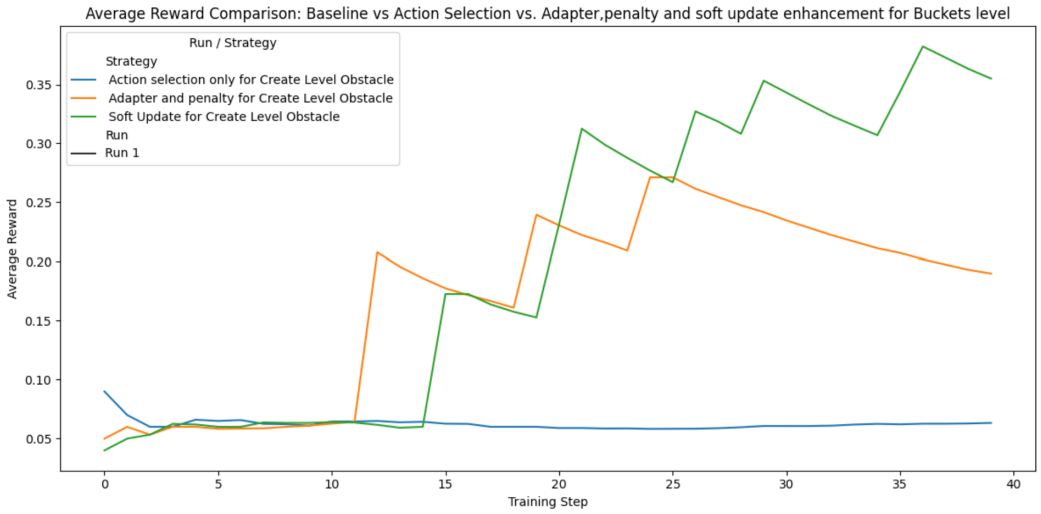


Figure 38: Performance comparison illustrating a case where full SETLE integration leads to dramatically improved reward outcomes.

Out of over 40 independent runs across different tasks and configurations, only those using both the adapter and soft update enhancements managed to solve the environment at least once (Fig. 39). These results highlight not only the potential of structured trajectory enrichment, but also the fragility of the approach in complex domains. Importantly, these experiments were conducted under deliberately constrained training regimes: episode lengths were capped at 9 steps, and overall training time was limited. We deliberately constrained the training setup—limiting the number of episodes and capping each trajectory at 9 steps—to focus on evaluating whether structured memory and enrichment mechanisms can accelerate learning and enable generalisation under low-data, few-step conditions. This design tests the efficiency of SETLE-enhanced agents in challenging settings where standard reinforcement learning strategies would typically struggle to succeed without extensive trial-and-error or curriculum-based shaping.

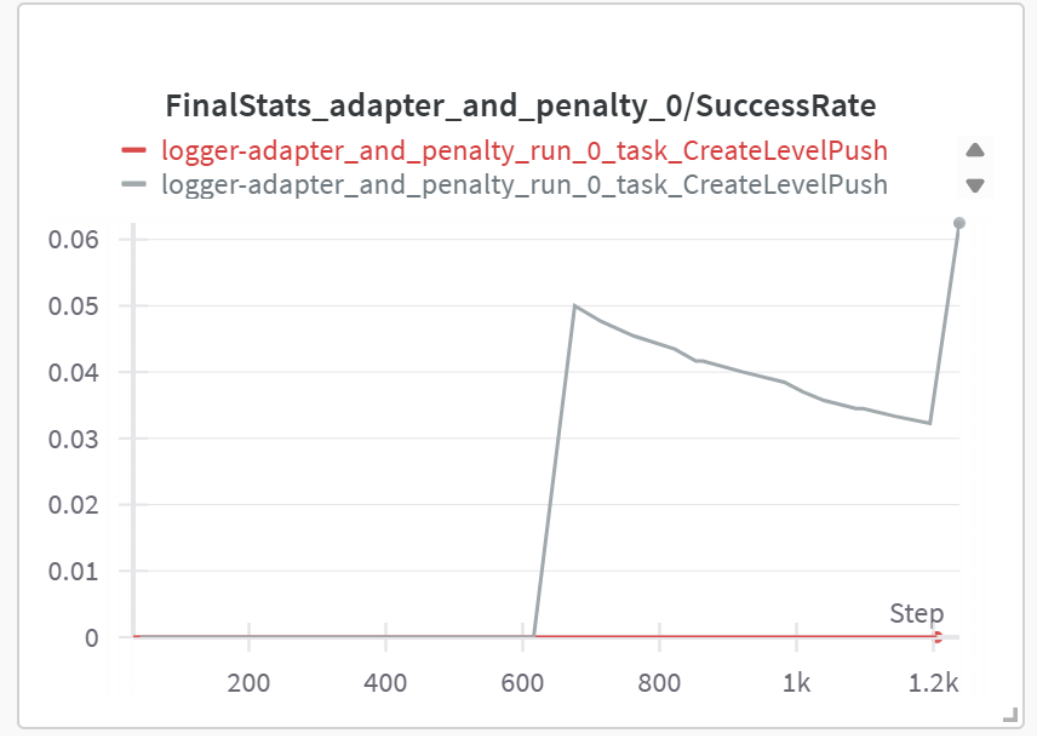


Figure 39: Only runs using the full enrichment strategy achieved success in at least one complex CREATE task.

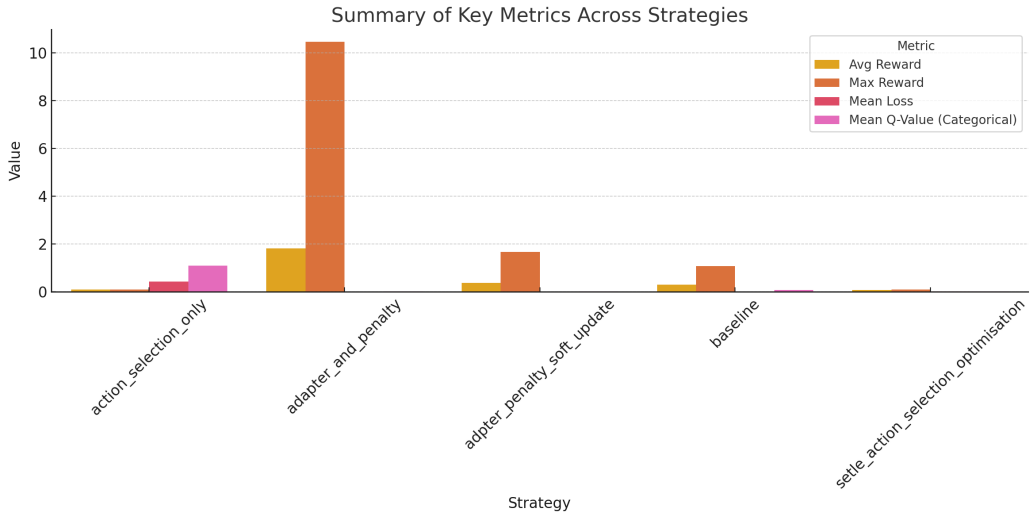


Figure 40: The bar chart compares average reward, maximum reward, mean loss, and mean Q-value (categorical) across all strategy variants for 5 different runs for each of the task and strategy. The adapter and penalty strategy exhibits the highest average and peak rewards, while the action selection only strategy shows higher Q-values but also greater loss. Baseline and other non-optimised configurations remain lower across all axes, highlighting the benefits of integrated enrichment and adaptive mechanisms.

Summary of key metrics across strategies The bar plot (Fig.40) summarises key performance metrics across different reinforcement learning strategies enhanced with SETLE. Here are the main insights:

1. Adapter and Penalty is the most effective strategy overall, achieving the highest average reward (around 1.8) and maximum reward (> 10), with the lowest overall loss and modest Q-values. This supports earlier findings that combining knowledge integration (via SETLE) with adaptation and exploration encourages meaningful generalisation.

2. Adapter + Penalty + Soft Update comes second in performance, with lower but still non-trivial rewards and stable losses, suggesting that soft updates help maintain training stability but may slow aggressive reward gains.

3. Action Selection Only and SETLE for Action Selection and Optimisation yield the lowest average rewards, indicating that access to memory

alone is not sufficient—adaptation layers and retrieval regulation are essential. While the Action and Optimisation strategy does outperform the baseline in several runs as seen previously, its performance is inconsistent across tasks. This suggests that optimisation over enriched states can lead to better outcomes, but without additional mechanisms like the adapter or penalty, the agent may struggle to stabilise learning or generalise across different task configurations.

4. Baseline performs slightly better than random SETLE variants without adaptation but lacks the reward stability and peak performance of more structured approaches.

Overall, these results confirm that structured enrichment must be accompanied by mechanisms for adaptation (adapter layers) and memory regulation (penalty, soft update) to produce consistent and transferable learning in environments like CREATE. Other logging, as well as how attention and tool reuse has been tracked, can be found in Appendix C.

6.3.2 Results in Symbolic Reasoning Tasks (MiniGrid)

To evaluate the behaviour of the SETLE-enhanced reinforcement learning agent in reward-sparse settings, we first conducted experiments in MiniGrid on the Empty-5x5 task. This environment is intentionally minimalist, containing no explicit goals or rewards, and thus provides no extrinsic learning signals during interaction. The aim of this experiment is not to assess task completion but to observe the **value dynamics and learning behaviour** of agents when operating under severe epistemic uncertainty and absence of reinforcement. Both the baseline and SETLE-enhanced agents were trained using Double DQN designed to reduce overestimation bias by decoupling action selection from evaluation.

Despite this bias correction, the baseline agent displayed a rapid and sustained increase in Q-values over time, even in the complete absence of reward (see Fig. 41).

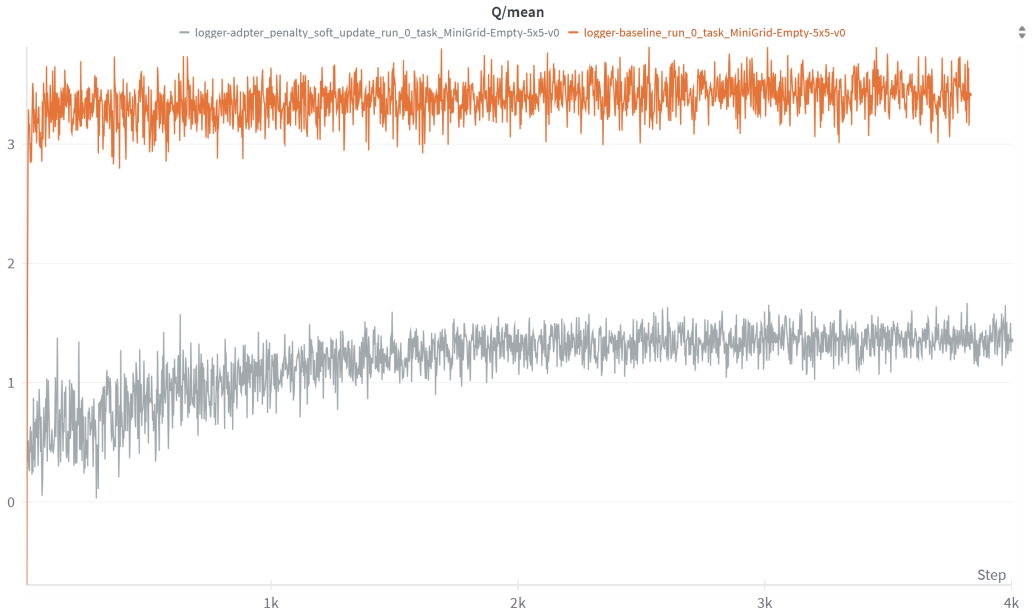


Figure 41: Comparison of Q-value evolution for Baseline and SETLE-enhanced agents in `MiniGrid-Empty-5x5`. The baseline agent exhibits sustained value inflation despite receiving no reward, with Q-values rising above 3.0. The SETLE-enhanced agent maintains more conservative Q-values around 1.0, reflecting structurally grounded value estimation under sparse feedback.

On average, baseline Q-values stabilized around 3.3 after a few hundred steps. This behaviour indicates a form of **residual value inflation**, wherein Q-values are recursively bootstrapped from their own overestimated targets. Since no extrinsic signal exists to anchor these predictions, the value function drifts into **ungrounded optimism**, assigning high expected returns to sequences of actions that are never positively reinforced. Such overconfidence, though syntactically correct in terms of Bellman updates, poses a threat to generalisation, especially when agents are transferred to environments where structural conditions change or delayed rewards appear.

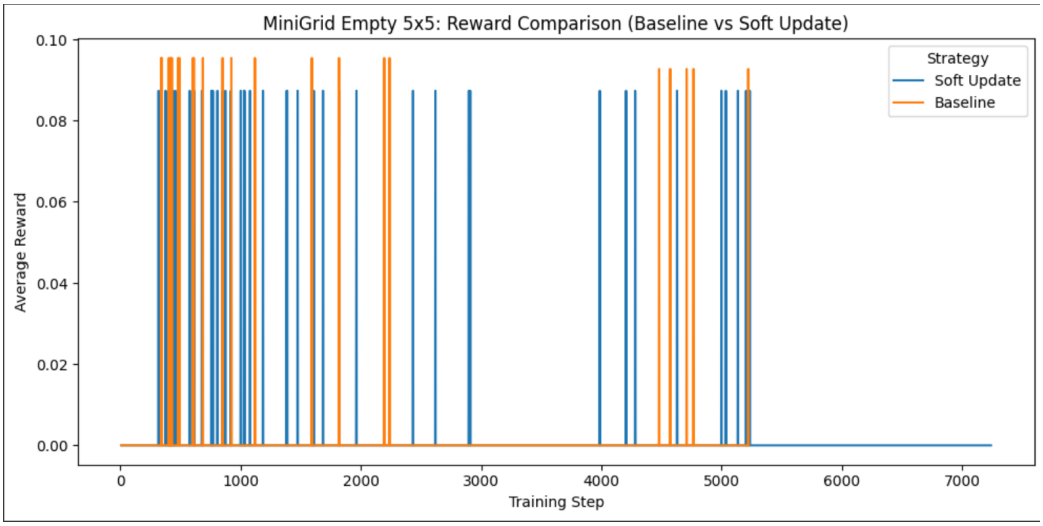


Figure 42: Average reward per step for Baseline (orange) and SETLE-enhanced agents (blue) in `MiniGrid-Empty-5x5`. Although overall rewards are low due to the sparse-reward nature of the environment, the SETLE-enhanced agent achieves nearly twice the reward frequency of the baseline, indicating improved exploration and task-relevant behaviour despite the absence of explicit supervision.

By contrast, the SETLE-enhanced agent maintained Q-values near 1.0 throughout training, less than one-third of the baseline level, indicating a more conservative and context-sensitive learning dynamic. At first glance, this may appear as an indication of stagnation or failure to learn. However, closer inspection of the reward logs (Fig.42) reveals that the SETLE-enhanced agent achieved **nearly double the reward frequency per step** compared to the baseline (0.0084 vs. 0.0048) (Fig.43). Although absolute rewards remained low due to the task's sparsity, the increased reward density and stabilised Q-values suggest that SETLE's structured memory support fosters more **calibrated value estimation** and meaningful exploration.

Figure 43: Cumulative reward comparison between the baseline agent and the SETLE-enhanced agent in the `MiniGrid-Empty-5x5` environment. The SETLE-enhanced agent consistently collects reward more frequently and achieves a higher overall total, despite the sparse and minimal nature of the environment. This demonstrates the benefit of structured trajectory enrichment in guiding exploration and sustaining learning under low-signal conditions.

Rather than hallucinating reward where none exists, SETLE retrieves structured prior experiences—such as object co-occurrences and contextual affordances, that allow the agent to infer when sequences are not worth propagating optimistic returns. In doing so, the agent suppresses speculative Q-value growth and aligns its internal value function with the actual utility of its behaviours. This form of value realism is increasingly recognised as a desirable property in deep RL. Recent work (Fujimoto et al., 2018; Kumar et al., 2020) has highlighted the risks of overestimation in Q-learning and proposed conservative or uncertainty-aware alternatives to promote trustworthy generalisation.

In summary, while the baseline agent appears to "learn" faster in numerical terms, its Q-values are **ungrounded and misleading** in the context of a reward-sparse environment. The SETLE-enhanced agent, by contrast, produces Q-value trajectories that more accurately reflect the lack of positive reinforcement, and in doing so, demonstrates **stronger potential for generalisation** across structurally similar tasks.

Behavioural Analysis in the Hardest Mini-Grid Task: SimpleCrossingS9N1 In contrast to the `Empty-5x5` task, which served as a diagnostic environment for reward-free value estimation, the `SimpleCrossingS9N1` task represents the most complex environment in our Mini-Grid evaluation suite. It requires the agent to navigate through multiple rooms separated by walls

and doors, some of which are partially obstructed, requiring careful planning and multi-step exploration. The sparse reward signal is delivered only upon successful navigation to a distant goal, making this task particularly challenging for unstructured exploration.

The results reveal significant divergence in agent behaviour across strategies. In terms of raw task success, the **adapter + penalty + soft update** strategy achieves the highest number of successful episodes (see Fig.44), while the **adapter + penalty** strategy performs second, even though still low (Fig.44). Both significantly outperform the baseline in terms of final performance, for which no success if obtained.

Figure 44: Success and failure counts across strategies for the `SimpleCrossingS9N1` task. All agents were trained for the same number of episodes. The **adapter + penalty + soft update** strategy achieved the most successful completions. In contrast, the baseline agent recorded zero successes, underscoring the importance of structured memory for navigating long-horizon, sparse-reward environments.

However, as shown in Fig. 44, these success gains come at the cost of increased trial failures. The method all though comes with an improvements it is yet to be satisfactory.

The Q-value trajectories in Fig. 45 illustrate again the q-value overestimation. The baseline agent quickly settles into a narrow range of inflated Q-values (2.0–2.2), reflecting premature convergence to suboptimal policies. In contrast, SETLE-enhanced agents—particularly those using soft update and adapter integration—exhibit high-frequency but bounded Q-value fluctuations near 0.0. These fluctuations reflect richer but more cautious value estimation, as agents continuously reassess the utility of action paths in the absence of consistent external reinforcement.

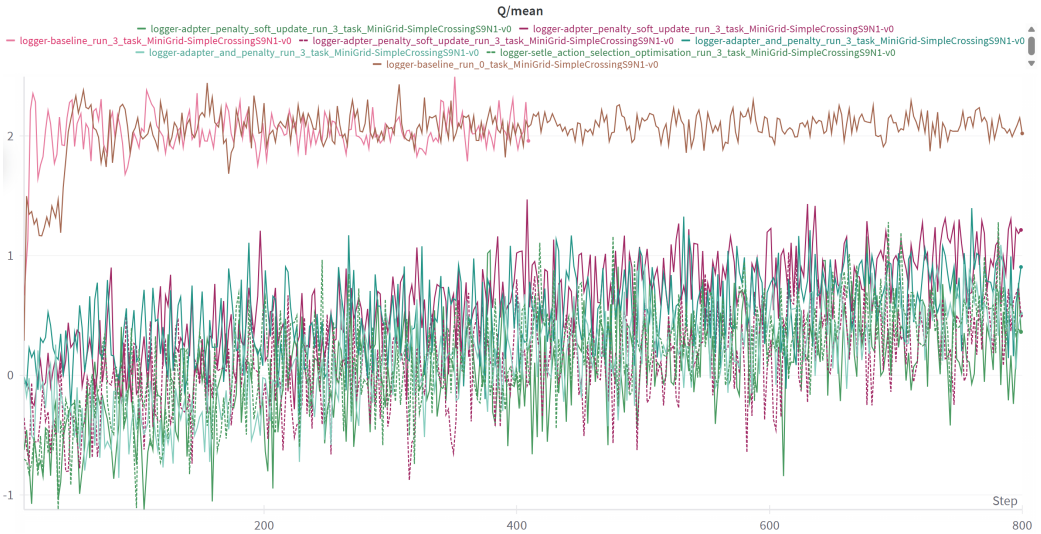


Figure 45: Evolution of average Q-values during training in `SimpleCrossingS9N1`. The baseline agent exhibits elevated and persistent Q-value inflation (around 2.0), which does not correlate with actual reward acquisition. SETLE-enhanced agents, especially those using adapters and soft updates, maintain lower, noisier Q-values—reflecting a more grounded and cautious value estimation process suitable for sparse-reward environments.

Overall, these findings underscore the strength of structured memory enrichment in complex symbolic environments. SETLE-enhanced strategies not only improve final success rates, but also exhibit more realistic and adaptive learning dynamics under sparse-reward conditions. In the hardest MiniGrid setting, this manifests as a trade-off between exploratory depth and training volatility—one that SETLE manages better than traditional baselines.

6.3.3 Cross-task episode embedding comparison

While traditional object-centric representations offer semantic interpretability, they often fall short in enabling cross-domain generalisation. Our analysis reveals this limitation clearly. The violin plot in Figure 46 shows the distribution of top-k cosine similarities between MiniGrid objects (e.g., *a golden key*, *a red triangle*) and their closest counterparts in CREATE. Even for visually simple or conceptually similar categories like *a red key*, the distribution remains narrow and shifted towards low similarity scores. This indicates that despite having similar names or roles, the embeddings of objects trained in distinct environments remain poorly aligned in practice. Such discrepancies stem from differences in rendering style, visual resolution, and context-specific usage—highlighting the fragility of object-level transfer across domains.

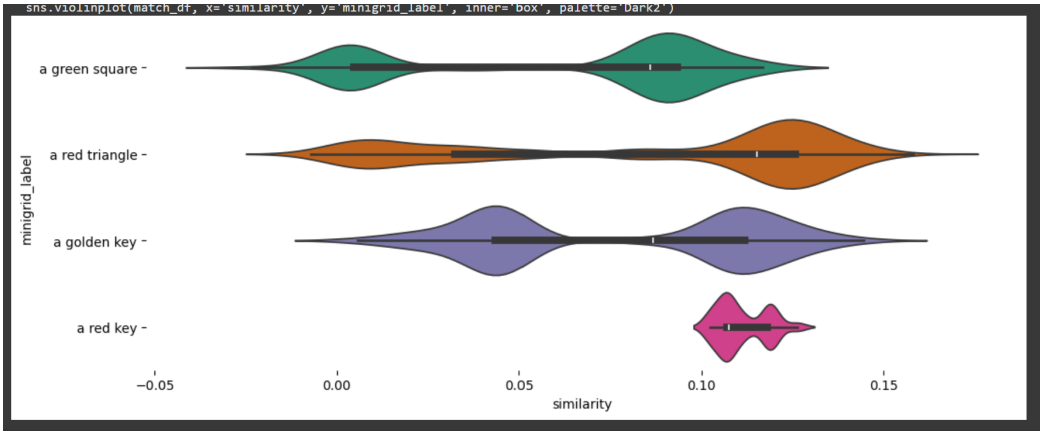


Figure 46: Distribution of top-k cosine similarities for MiniGrid object embeddings matched to CREATE objects. Even semantically aligned objects exhibit narrow similarity distributions, indicating poor object-level alignment across domains. This supports the claim that functional generalisation must operate above raw perceptual encoding.

In contrast, when comparing entire trajectory embeddings—capturing not just objects but also actions, affordances, and their interdependencies, a more encouraging picture emerges. As illustrated in the heatmap of Figure 47, several CREATE and MiniGrid task pairs show non-trivial cosine similarities at the episode level. For example, `CreateLevelPush-v0` and

`CreateLevelBelt-v0` exhibit meaningful structural overlap with "MiniGrid-DoorKey-5x5 and `MiniGrid-Empty-5x5`, suggesting that the agent learns shared procedural structures such as navigation, obstacle manipulation, and goal seeking.

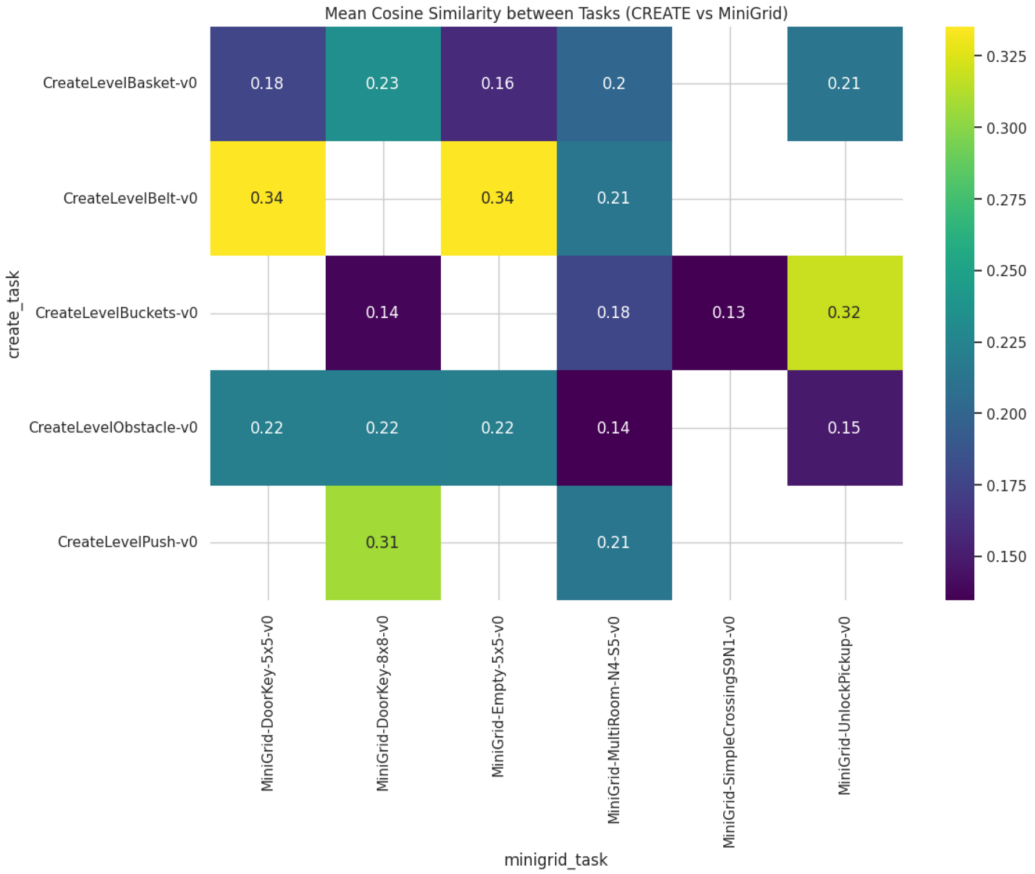


Figure 47: Mean cosine similarity between episode-level embeddings across CREATE and MiniGrid tasks. Higher values along diagonals suggest structural alignment between tasks like `CreateLevelPush` and `MiniGrid-DoorKey-5x5`, despite differences in appearance and control schemes. This highlights SETLE's ability to extract relational and sequential regularities shared across environments.

The clustermap in Figure 48 reinforces this trajectory-level alignment. Tasks from the two domains are not segregated by origin but grouped accord-

ing to their functional and temporal similarities. For instance, "CreateLevelBuckets" and `MiniGrid-UnlockPickup` are clustered together, despite vast differences in visual and mechanical design. This pattern reflects the agent's ability to learn and compare episodes based on high-level interaction graphs, rather than low-level appearance features.

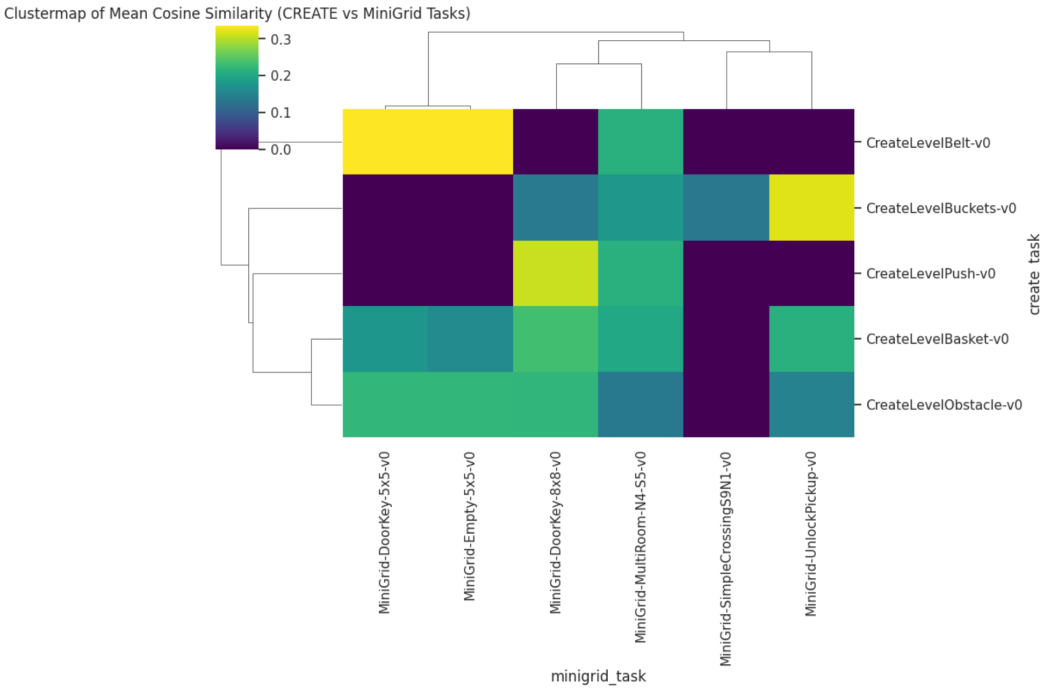


Figure 48: Hierarchical clustering of average episode embeddings from CREATE and MiniGrid tasks. Tasks from both environments are grouped based on shared interaction structure and relational dependencies, not domain origin.

Together, these results highlight the core advantage of SETLE's approach: rather than relying on static embeddings of isolated objects, it encodes episodes as structured graphs capturing action-effect relationships and contextual dependencies. This abstraction enables meaningful comparison across environments and supports the agent's ability to retrieve and reuse experience even in domains with drastically different visual or symbolic structure. These findings strengthen the thesis argument that creative generalisation in reinforcement learning hinges on structured, relational memory—not

raw perceptual similarity.

7 General discussion

A central aim of this thesis has been to investigate whether agents can generalise across tasks and domains not merely through memorisation, but by structurally encoding and reusing experience in conceptually meaningful ways. To this end, we operationalised the AIGenC framework as a model of creative problem-solving and knowledge transfer, instantiated through the SETLE architecture. This framework posits that generalisation emerges not from scale or pattern interpolation alone, but from an agent's capacity to build and navigate a hierarchical concept space—one in which prior knowledge can be retrieved through matching and flexibly adapted through blending to solve novel tasks.

This work does not attempt to resolve long-standing questions about the nature of concepts or the formal limits of representation (Piantadosi, 2021; Frixione and Lieto, 2012; Lieto et al., 2017). Nor does it aim to model human cognition per se. Instead, it takes a pragmatic stance: if generalisation in natural cognition often involves abstracting and recombining experience, can artificial agents be designed to do the same, without supervision, and without task-specific engineering? This thesis has explored this hypothesis empirically, using AIGenC as an architecture and SETLE as an instantiation to build agents that learn structured memory from interaction and apply it creatively during training.

Yet even as empirical progress continues, a deeper limitation remains. Contemporary AI systems, particularly large-scale generative models, often perform well not because they autonomously discover abstract structure, but because human designers have already done much of the conceptual work. For instance, task formulations, such as chain-of-thought prompting (Wei et al., 2022), that embed reasoning steps directly into inputs, and alignment procedures, like Reinforcement Learning from Human Feedback (RLHF), that shape models' outputs toward socially and semantically coherent responses. These interventions embed human reasoning and abstraction into the model's learning environment, effectively outsourcing much of the conceptual burden to the design process itself. Roitblat (2025) characterises this as anthropogenic debt: the hidden scaffolding of problem formulation, representation, and evaluation that humans supply. These systems typically

reframe tasks as language modelling problems and rely on dense pattern interpolation over massive datasets. What appears as general intelligence may instead be the byproduct of solving narrowly structured tasks with vast supervision. Piantadosi and Hill (2022) similarly argue that genuine generalisation depends not on surface pattern matching, but on the ability to represent and manipulate conceptual roles—internal structures that support inference, abstraction, and reuse. From this perspective, the challenge of AGI is not scaling models further, but endowing agents with the capacity to construct and navigate structured knowledge autonomously. Without this, intelligence remains bounded by the assumptions encoded in the data and tasks we design.

This critique suggests not just a performance gap, but a structural one: the absence of systems capable of forming, organising, and reusing internal representations in flexible, task-independent ways. To explore this gap, we drew from cognitive science to design agents equipped with mechanisms for low-level concept formation and hierarchical memory organisation, while remaining agnostic about whether these mechanisms resemble human cognition. While debate continues over whether AI can ultimately achieve human-like generalisation (Fjelland, 2020; Shevlin and Halina, 2019), there is broad consensus that its current absence remains a critical obstacle to AGI (Zhang et al., 2021).

To evaluate whether structured memory and compositional abstraction can help address this limitation, we tested agents equipped with these capacities in a range of reinforcement learning tasks. Our findings provide partial support for the hypothesis. SETLE agents reused prior concepts across tasks more effectively than baseline models, displayed more stable and calibrated value estimation, and revealed meaningful clustering of structurally similar episodes, even across different environments. In some cases, knowledge transfer occurred across symbolic and perceptual boundaries. Still, generalisation remained incomplete. On more complex tasks, success rates were limited, and agents often required task-specific data to converge.

Across both domains, SETLE-enhanced agents outperformed baseline agents in several ways:

1. Increased reward acquisition: In CREATE tasks such as CreateLevelBuckets and CreateLevelObstacle, agents using enrichment (especially with the adapter and soft update enhancements) consistently achieved higher average rewards and exhibited greater stability in training.

2. Improved value estimation in sparse settings: In MiniGrid's Empty 5x5 environment, where no rewards are available, baseline agents demonstrated value inflation, with Q-values increasing unrealistically over time. SETLE agents, by contrast, maintained grounded estimates near zero, avoiding spurious optimism.

3. Cross-task retrieval and structural similarity: Episode-level embeddings revealed meaningful structural similarity across different environments, even when object-level embeddings failed to align. For instance, clustering analyses showed that MiniGrid and CREATE episodes involving navigation or tool usage were grouped together, despite domain differences.

However, these improvements came with significant caveats:

Fragility of enrichment mechanisms, attention-based retrieval and graph supplementation, proved highly sensitive to architectural and procedural choices. Without components such as the adapter or soft target updates, the policy network often struggled to integrate enriched representations, resulting in unstable Q-values and high loss variance. This was particularly evident in MiniGrid tasks with high compositional complexity (MultiRoom, DoorKey 8x8), where naive enrichment degraded performance.

Limited Success Rates despite better value calibration and more efficient learning, success on more challenging MiniGrid tasks remained low. In MultiRoom, agents with enrichment (e.g. adapter and penalty) achieved some success (up to 5/250), while baseline agents failed entirely. Yet the overall absolute success rate remained modest. This underscores that structural representations help learning, but are not sufficient for strong generalisation without additional components such as curriculum learning, environment modelling, or planning.

Enrichment $\neq$ Generalisation (Yet). One of the thesis's hypotheses was that structured memory can enable generalisation via knowledge transfer. The results partially support this: retrieval of relevant subgraphs from past episodes improved early learning and provided inductive biases. However, complete generalisation, adapting to tasks with unseen goals, object combinations, or spatial layouts, remained elusive. SETLE reduces the sample complexity of learning by providing more meaningful inputs; however, the learned policies still heavily depend on observed data. In this sense, SETLE enables more informed memorisation, but not yet true extrapolation.

The Illusion of Learning: One of the more surprising findings in this thesis was the consistent overestimation of value in baseline agents trained in sparse environments like Empty 5x5. Despite no rewards, the Q-function "learned" to assign high expected returns. This illusion of progress—learning unanchored in meaningful feedback—is not unlike the hallucinations of LLMs: fluent yet disconnected from ground truth. In this sense, SETLE's conservative value estimates, while slower to rise, are a sign of epistemic humility. By grounding its predictions in structural similarity, SETLE avoids the "hallucination trap". However, it also exposes a difficult tradeoff: slower, structurally aware learning versus faster, yet brittle, convergence. This mirrors broader tensions in AI research between sample efficiency and reliability, and raises questions about how to properly evaluate learning systems, not only by reward curves, but by the trustworthiness and interpretability of their behaviours.

These observations align closely with the critique offered by Morad et al. (2023), who argue that memory is only beneficial when it is both selective and structurally aligned with task demands. In their analysis, large, unstructured memory systems can degrade performance by reinforcing spurious patterns, leading to worse generalisation. Their proposed "fast and forgetful" memory system echoes the principle we adopt with AIGenC: that reuse should be conditional and relevance-driven, not blind replay. Together, these insights suggest that generalisation in RL cannot be achieved through architectural scale or data accumulation alone. Without structurally grounded representations that encode not only what was rewarded but also why those outcomes were possible, agents remain prone to brittle reuse, spurious generalisation, and illusory learning.

The findings in this thesis reflect a broader shift in reinforcement learning and beyond: generalisation is increasingly understood not as a byproduct of scale or memorisation, but as a function of how experience is represented and reused. Structured episodic memory contributes to this shift by enabling the retrieval and recomposition of past experiences based on functional and relational similarity, rather than surface-level correlation

This view aligns with a broader movement in RL toward architectures that treat memory as more than a buffer for replay. Recent work in dual-memory systems, such as Two-Memory Reinforcement Learning (Yang et al., 2023b), explores how fast, non-parametric recall can complement slower, gradient-based policy updates. Similarly, research into abstract episodic control—such as NECSA (Li et al., 2023)—emphasises the importance of struc-

ture in stored experiences, showing that compositional patterns over multiple steps can support more effective transfer than isolated transitions. These approaches differ in implementation, but converge on a common principle: that memory becomes useful for generalisation when it captures higher-order regularities.

The direction taken in this thesis reflects this same principle. In SETLE, episodes are not stored as flat vectors or sequences but as graph—structured representations that encode affordances, object-action relations, and causal dependencies. This enables retrieval to operate based on conceptual similarity, rather than just spatial or temporal proximity. The result is a form of generalisation grounded not in parameter interpolation but in structural analogy, where trajectories with similar internal logic can guide new behaviour.

This structural view of memory is increasingly mirrored in trends outside RL as well. In large language models, retrieval-augmented generation (RAG) methods (Lewis et al., 2020) enhance generalisation by incorporating external data at inference time. Rather than relying solely on parametric knowledge, these systems retrieve relevant information from structured sources, such as documents, knowledge graphs, or application programming interfaces (APIs), and integrate it into their predictions. Crucially, this augmentation is not arbitrary—retrieval is guided by contextual signals in the input, ensuring that the retrieved knowledge is functionally aligned with the task at hand. Similarly, hybrid LLM architectures are exploring the integration of symbolic planners, relational memories, or program-like reasoning modules to support generalisation through structure rather than scale.

While these systems operate in different domains compared to our model, they share a conceptual thread: the move from passive memory to active retrieval, from monolithic models to modular systems where structure informs inference. AIGenC and SETLE represent a reinforcement learning analogue of this design trend. It shows that agents benefit not just from storing experience, but from organising it in ways that support flexible recomposition and adaptive reasoning.

This perspective carries implications for how we think about general intelligence. While SETLE does not claim to approach AGI in capability, it contributes to a growing research direction focused on functional generalisation through conceptual mechanisms, such as abstraction, analogy, and recombination, rather than brute-force pattern learning. This aligns with calls from across machine learning and cognitive science to focus on the in-

ternal organisation of knowledge, and on architectures that support reuse and recomposition across tasks (Lake et al., 2017; Chollet, 2019; Marcus, 2020).

The contribution here is deliberately modest. As Roitblat (2025) points out, fluency and scale are often mistaken for intelligence, when what is needed are systems that can construct and apply internal structure autonomously. SETLE takes a small but significant step in this direction—supporting agents that can not only act, but also interpret and organise what they've learned in a reusable form. In doing so, the AIGenC framework shifts the framing of AGI from a fixed capability threshold to a set of capacities related to generative reuse, compositional inference, and relational learning—capacities that can be directly tested and measured.

Defining AGI remains an open and often contested problem (Goertzel, 2014; Shevlin and Halina, 2019). There is little consensus on what constitutes generalisation or how to assess it meaningfully. AIGenC sidesteps the problem of definition by offering a more operational framing: intelligence is not about universality, but about the generative flexibility to adapt experience across novel configurations. This reframing aligns with work in cognitive systems and neurosymbolic reasoning, where generalisation is defined not by the breadth of input coverage but by the system's ability to recombine internal representations to solve new problems.

This shift also has practical implications. As argued in strategic policy discussions (e.g., Mitre and Joel (2025)), even limited generalisation capabilities can be consequential when applied in high-stakes contexts—from decision-making systems to autonomous agents. Architectures that explicitly model the internal structure of experience offer more than improved sample efficiency—they provide a basis of interpretability and auditability. Unlike purely parametric systems, which often collapse reasoning into opaque activations, systems like SETLE provide explicit traces of what was retrieved, why it was relevant, and how it informed behaviour.

Nonetheless, AIGenC and SETLE remain early-stage models with several limitations. While its relational memory enables improved transfer, it is sensitive to design choices, such as attention mechanisms and adapter modules, and currently lacks mechanisms for long-term concept consolidation or symbolic abstraction. Additionally, the current form depends on domain-specific processing (e.g., affordance extraction) that may not generalise easily.

Future work may explore the integration of large language models as dynamic concept generators, utilising their latent semantics to create or re-

fine the graph-based representations used by SETLE. LLMs could serve as external guides for shaping conceptual spaces in ways that reduce human intervention—for example, automatically identifying which types of nodes are relevant for a given context (e.g., whether to instantiate ActionRepr nodes, which affordances apply, or how to distinguish real-world from simulated objects and effects). Such models could help adapt the structure of the concept space on the fly, tailoring it to the task, environment, or embodiment modality. Similarly, deployment in robotic settings could further ground affordance learning in physical interaction, enabling generalisation to emerge through embodied experience rather than predefined schemas.

Ultimately, this thesis does not propose a blueprint for AGI—but it does offer one path forward: by treating experience not just as data to compress, but as relational knowledge to organise, abstract, and adapt. In this light, generalisation becomes not a matter of scale, but a matter of structure and reuse, a challenge that demands both architectural clarity and conceptual flexibility.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alvarez-Melis, D. and Jaakkola, T. S. (2018). Gromov-wasserstein alignment of word embedding spaces. *arXiv preprint arXiv:1809.00013*.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., and Blundell, C. (2020). Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pages 507–517. PMLR.

Baer, J. (2020). The consensual assessment technique. In *Handbook of research methods on creativity*, pages 166–177. Edward Elgar Publishing.

Bahdanau, D. (2020). On sample efficiency and systematic generalization of grounded language understanding with deep learning.

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., and Mordatch, I. (2019). Emergent tool use from multi-agent autocurricula. In *International conference on learning representations*.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Bayer, M., Kaufhold, M.-A., Buchhold, B., Keller, M., Dallmeyer, J., and Reuter, C. (2023). Data augmentation in natural language processing: a novel text generation approach for long and short text classifiers. *International journal of machine learning and cybernetics*, 14(1):135–150.

Bechberger, L. and Kühnberger, K.-U. (2017). A comprehensive implementation of conceptual spaces. *arXiv preprint arXiv:1707.05165*.

Bender, E. M. and Koller, A. (2020). Climbing towards nlu: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 5185–5198.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bengio, Y., Lecun, Y., and Hinton, G. (2021). Deep learning for ai. *Communications of the ACM*, 64(7):58–65.

Boden, M. A. (1996). *Artificial intelligence*. Elsevier.

Boden, M. A. (2009). Computer models of creativity. *Ai Magazine*, 30(3):23–23.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Bowman, S. R. and Dahl, G. E. (2021). What will it take to fix benchmarking in natural language understanding? *arXiv preprint arXiv:2104.02145*.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*.

Catarau-Cotutiu, C., Mondragon, E., and Alonso, E. (2023). Aigenc: Ai generalisation via creativity. In *EPIA Conference on Artificial Intelligence*, pages 38–51. Springer.

Cătărău-Cotuţiu, C., Mondragón, E., and Alonso, E. (2023). Aigenc: Ai generalisation via creativity. In *EPIA Conference on Artificial Intelligence*, pages 38–51. Springer.

Catarau-Cotutiu, C., Mondragon, E., and Alonso, E. (2025). A representational framework for learning and encoding structurally enriched trajectories in complex agent environments. *arXiv preprint arXiv:2503.13194*.

Chemero, A., Klein, C., and Cordeiro, W. (2003). Events as changes in the layout of affordances. *Ecological Psychology*, 15(1):19–28.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information

maximizing generative adversarial nets. *Advances in neural information processing systems*, 29.

Chevalier-Boisvert, M., Dai, B., Towers, M., Perez-Vicente, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., and Terry, J. (2023). Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems*, 36:73383–73394.

Chollet, F. (2019). On the measure of intelligence. *arXiv preprint arXiv:1911.01547*.

Clark, S., Lerchner, A., von Glehn, T., Tieleman, O., Tanburn, R., Dashevskiy, M., and Bosnjak, M. (2021). Formalising concepts as grounded abstractions. *arXiv preprint arXiv:2101.05125*.

Co-Reyes, J., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. (2018). Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International conference on machine learning*, pages 1009–1018. PMLR.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. (2020). Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR.

Colin, T. R., Belpaeme, T., Cangelosi, A., and Hemion, N. (2016). Hierarchical reinforcement learning as creative problem solving. *Robotics and Autonomous Systems*, 86:196–206.

Collins, A. M. and Quillian, M. R. (1969). Retrieval time from semantic memory. *Journal of verbal learning and verbal behavior*, 8(2):240–247.

Crosby, M., Beyret, B., Shanahan, M., Hernández-Orallo, J., Cheke, L., and Halina, M. (2020). The animal-ai testbed and competition. In *Neurips 2019 competition and demonstration track*, pages 164–176. PMLR.

Dasgupta, I., Guo, D., Gershman, S. J., and Goodman, N. D. (2020). Analyzing machine-learned representations: A natural language case study. *Cognitive Science*, 44(12):e12925.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Doumas, L. A., Hummel, J. E., and Sandhofer, C. M. (2008). A theory of the discovery and predication of relational concepts. *Psychological review*, 115(1):1.

Doumas, L. A., Puebla, G., Martin, A. E., and Hummel, J. E. (2022). A theory of relation learning and cross-domain generalization. *Psychological review*.

Edwards, H. and Storkey, A. (2016). Towards a neural statistician. *arXiv preprint arXiv:1606.02185*.

Efroni, Y., Merlis, N., and Mannor, S. (2021). Reinforcement learning with trajectory feedback. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 7288–7295.

Eilifsen, C. and Arntzen, E. (2021). Mediated generalization and stimulus equivalence. *Perspectives on Behavior Science*, 44(1):1–27.

Fauconnier, G. and Turner, M. (1998). Conceptual integration networks. *Cognitive science*, 22(2):133–187.

Fjelland, R. (2020). Why general artificial intelligence will not be realized. *Humanities and Social Sciences Communications*, 7(1):1–9.

Freire, I. T., Amil, A. F., and Verschure, P. F. (2024). Sequential memory improves sample and memory efficiency in episodic control. *Nature Machine Intelligence*, pages 1–13.

Frith, E., Elbich, D. B., Christensen, A. P., Rosenberg, M. D., Chen, Q., Kane, M. J., Silvia, P. J., Seli, P., and Beaty, R. E. (2021). Intelligence and creativity share a common cognitive and neural basis. *Journal of Experimental Psychology: General*, 150(4):609.

Frixione, M. and Lieto, A. (2012). Representing concepts in formal ontologies. compositionality vs. typicality effects. *Logic and Logical Philosophy*, 21(4):391–414.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR.

Gabora, L., Rosch, E., and Aerts, D. (2008). Toward an ecological theory of concepts. *Ecological Psychology*, 20(1):84–116.

Garcez, A. S., Lamb, L. C., and Gabbay, D. M. (2008). *Neural-symbolic cognitive reasoning*. Springer Science & Business Media.

Gardenfors, P. (2004). *Conceptual spaces: The geometry of thought*. MIT press.

Garnelo, M. and Shanahan, M. (2019). Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29:17–23.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

Gibson, J. J. (1977). The theory of affordances. hilldale, usa. *Hilldale USA*, 1(2).

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.

Gizzi, E., Nair, L., Sinapov, J., and Chernova, S. (2020). From computational creativity to creative problem solving agents. In *ICCC*, pages 370–373.

Goertzel, B. (2014). Artificial general intelligence: concept, state of the art, and future prospects. *Journal of Artificial General Intelligence*, 5(1):1.

Gordon, J., Lopez-Paz, D., Baroni, M., and Bouchacourt, D. (2019). Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*.

Goyal, A. and Bengio, Y. (2022). Inductive biases for deep learning of higher-level cognition. *Proceedings of the Royal Society A*, 478(2266):20210068.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Greeno, J. G. (1994). Gibson's affordances.

Haines, N., Vassileva, J., and Ahn, W.-Y. (2018). The outcome-representation learning model: A novel reinforcement learning model of the iowa gambling task. *Cognitive science*, 42(8):2534–2561.

Haskell, R. E. (2000). *Transfer of learning: Cognition and instruction*. Elsevier.

Hayman, G. and Huebner, B. (2019). Temporal updating, behavioral learning, and the phenomenology of time-consciousness. *Behavioral and Brain Sciences*, 42.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heit, E. (1996). The instantiation principle in natural categories. *Memory*, 4(4):413–452.

Higgins, I., Amos, D., Pfau, D., Racaniere, S., Matthey, L., Rezende, D., and Lerchner, A. (2018). Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*.

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., et al. (2021). Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37.

Hupkes, D., Dankers, V., Mul, M., and Bruni, E. (2020). Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*.

Jain, A., Szot, A., and Lim, J. J. (2020). Generalization to new actions in reinforcement learning. *arXiv preprint arXiv:2011.01928*.

James, W. (1890). The principles of psychology. *Henry Holt*.

Jawahar, G., Sagot, B., and Seddah, D. (2019). What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

Jin, L., Zhao, F., Yan, C., and Gui, X. (2024). Causal graph representation learning for outcome-oriented link prediction. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. (2017). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910.

Kaufman, J. C. and Beghetto, R. A. (2009). Beyond big and little: The four c model of creativity. *Review of general psychology*, 13(1):1–12.

Keysers, D., Schärli, N., Scales, N., Buisman, H., Furrer, D., Kashubin, S., Momchev, N., Sinopalnikov, D., Stafiniak, L., Tihon, T., et al. (2019). Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*.

Kingma, D. P., Welling, M., et al. (2013). Auto-encoding variational bayes.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., and Pérez, P. (2021). Deep reinforcement learning for autonomous driving: A survey. *IEEE transactions on intelligent transportation systems*, 23(6):4909–4926.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. (2023). Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026.

Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T. (2023). A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Koestler, A. and Burt, C. (1964). The act of creation.

Kokkola, N. H., Mondragón, E., and Alonso, E. (2019). A double error dynamic asymptote model of associative learning. *Psychological review*, 126(4):506.

Konidaris, G., Kuindersma, S., Grupen, R., and Barto, A. (2011). Cst: Constructing skill trees by demonstration. *Doctoral Dissertations, University of Massachuts, Ahmrest*.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. (2020). Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191.

Laird, J. E., Derbinsky, N., and Tinkerhess, M. (2012). Online determination of value-function structure and action-value estimates for reinforcement learning in a cognitive architecture. *Advances in Cognitive Systems*, 2:221–238.

Lake, B. and Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Lake, B. M. and Baroni, M. (2023). Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253.

Lal, M. (2015). *Neo4j graph data modeling*. Packt Publishing Ltd.

Lesort, T., Díaz-Rodríguez, N., Goudou, J.-F., and Filliat, D. (2018). State representation learning for control: An overview. *Neural Networks*, 108:379–392.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Li, Z., Zhu, D., Hu, Y., Xie, X., Ma, L., Zheng, Y., Song, Y., Chen, Y., and Zhao, J. (2023). Neural episodic control with state abstraction. *arXiv preprint arXiv:2301.11490*.

Lieto, A., Chella, A., and Frixione, M. (2017). Conceptual spaces for cognitive architectures: A lingua franca for different levels of representation. *Biologically inspired cognitive architectures*, 19:1–9.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, Z., Azaman, H., Kumar, M. G., and Tan, C. (2023). Compositional learning of visually-grounded concepts using reinforcement. *arXiv preprint arXiv:2309.04504*.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538.

Loynd, R., Fernandez, R., Celikyilmaz, A., Swaminathan, A., and Hausknecht, M. (2020). Working memory graphs. In *International conference on machine learning*, pages 6404–6414. PMLR.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Lyre, H. (2020). The state space of artificial intelligence. *Minds and Machines*, 30(3):325–347.

Malik, D., Li, Y., and Ravikumar, P. (2021). When is generalizable reinforcement learning tractable? *Advances in Neural Information Processing Systems*, 34:8032–8045.

Marcus, G. (2020). The next decade in ai: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*.

Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial neural networks and machine learning–ICANN 2011: 21st international conference on artificial neural networks, espoo, Finland, June 14-17, 2011, proceedings, part i 21*, pages 52–59. Springer.

Medin, D. L., Altom, M. W., and Murphy, T. D. (1984). Given versus induced category representations: use of prototype and exemplar information in classification. *Journal of experimental psychology: Learning, memory, and cognition*, 10(3):333.

Mednick, S. (1962). The associative basis of the creative process. *Psychological review*, 69(3):220.

Mezghan, L., Sukhbaatar, S., Lavril, T., Maksymets, O., Batra, D., Bojanowski, P., and Alahari, K. (2022). Memory-augmented reinforcement learning for image-goal navigation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3316–3323. IEEE.

Michaels, C. F. (2003). Affordances: Four points of debate. *Ecological psychology*, 15(2):135–148.

Minsky, M. (1974). A framework for representing knowledge.

Mitchell, M. (2021). Abstraction and analogy-making in artificial intelligence. *Annals of the New York Academy of Sciences*, 1505(1):79–101.

Mitchell, M. and Krakauer, D. C. (2023). The debate over understanding in ai's large language models. *Proceedings of the National Academy of Sciences*, 120(13):e2215907120.

Mitre, J. and Joel, B. P. (2025). Artificial general intelligence's five hard national security problems. *Santa Monica, CA: RAND Corporation*.

Momennejad, I. (2020). Learning structures: predictive representations, replay, and generalization. *Current Opinion in Behavioral Sciences*, 32:155–166.

Mondragón, E. and Murphy, R. A. (2010). Perceptual learning in an appetitive pavlovian procedure: Analysis of the effectiveness of the common element. *Behavioural Processes*, 83(3):247–256.

Monnier, T., Vincent, E., Ponce, J., and Aubry, M. (2021). Unsupervised layered image decomposition into object prototypes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8640–8650.

Morad, S., Kortvelesy, R., Liwicki, S., and Prorok, A. (2023). Reinforcement learning with fast and forgetful memory. *Advances in Neural Information Processing Systems*, 36:72008–72029.

Mukherjee, S., Asnani, H., Lin, E., and Kannan, S. (2019). Clustergan: Latent space clustering in generative adversarial networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4610–4617.

Murphy, G. L. and Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological review*, 92(3):289.

Nosofsky, R. M. (1988). Exemplar-based accounts of relations between classification, recognition, and typicality. *Journal of Experimental Psychology: learning, memory, and cognition*, 14(4):700.

Olteţeanu, A.-M. (2020). *Cognition and the Creative Machine: Cognitive AI for Creative Problem Solving*. Springer Nature.

O'Quin, K. and Besemer, S. P. (1989). The development, reliability, and validity of the revised creative product semantic scale. *Creativity Research Journal*, 2(4):267–278.

Piantadosi, S. T. (2021). The computational origin of representation. *Minds and machines*, 31:1–58.

Piantadosi, S. T. and Hill, F. (2022). Meaning without reference in large language models. *arXiv preprint arXiv:2208.02957*.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural networks*, 6(3):623–641.

Precup, D. (2000). *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst.

Quillian, M. R. (1967). Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral science*, 12(5):410–430.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. (2021). Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Raileanu, R., Goldstein, M., Yarats, D., Kostrikov, I., and Fergus, R. (2021). Automatic data augmentation for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 34:5402–5415.

Raji, I. D., Bender, E. M., Paullada, A., Denton, E., and Hanna, A. (2021). Ai and the everything in the whole wide world benchmark. *arXiv preprint arXiv:2111.15366*.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. (2022). A generalist agent. *arXiv preprint arXiv:2205.06175*.

Rhodes, M. (1961). An analysis of creativity. *The Phi delta kappan*, 42(7):305–310.

Roitblat, H. (2025). Some things to know about achieving artificial general intelligence. *arXiv preprint arXiv:2502.07828*.

Rosch, E. (1975). Cognitive representations of semantic categories. *Journal of experimental psychology: General*, 104(3):192.

Rosch, E. (2024). Principles of categorization. In *Cognition and categorization*, pages 27–48. Routledge.

Rosch, E. H. (1973). Natural categories. *Cognitive psychology*, 4(3):328–350.

Rudner, T. G., Pong, V., McAllister, R., Gal, Y., and Levine, S. (2021). Outcome-driven reinforcement learning via variational inference. *Advances in Neural Information Processing Systems*, 34:13045–13058.

Rumelhart, D. E. and Norman, D. A. (1983). Representation in memory.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. pearson.

Şahin, E., Cakmak, M., Doğar, M. R., Uğur, E., and Üçoluk, G. (2007). To afford or not to afford: A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, 15(4):447–472.

Sanders, J. T. (1997). An ontology of affordances. *Ecological psychology*, 9(1):97–112.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and brain sciences*, 3(3):417–424.

Shanahan, M., Crosby, M., Beyret, B., and Cheke, L. (2020). Artificial intelligence and the common sense of animals. *Trends in Cognitive Sciences*, 24(11):862–872.

Shanahan, M. and Mitchell, M. (2022). Abstraction for deep reinforcement learning. *arXiv preprint arXiv:2202.05839*.

Shevlin, H. and Halina, M. (2019). Apply rich psychological terms in ai with care. *Nature Machine Intelligence*, 1(4):165–167.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216.

Stoffregen, T. A. (2018). Affordances as properties of the animal-environment system. In *How Shall Affordances Be Refined?*, pages 115–134. Routledge.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. (2021). Decoupling representation learning from reinforcement learning. In *International conference on machine learning*, pages 9870–9879. PMLR.

Sun, Y., Zhu, D., Wang, Y., Fu, Y., and Tian, Z. (2025). Gtc: gnn-transformer co-contrastive learning for self-supervised heterogeneous graph representation. *Neural Networks*, 181:106645.

Sutton, R. S., Barto, A. G., et al. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Turvey, M. T. (1992). Affordances and prospective control: An outline of the ontology. *Ecological psychology*, 4(3):173–187.

Vahdat, A. and Kautz, J. (2020). Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354.

Wallas, S. (1970). The art of thought. harcourt brace & world inc. 1926. *Creativity*.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2019). Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wang, X., Liu, N., Han, H., and Shi, C. (2021). Self-supervised heterogeneous graph neural network with co-contrastive learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 1726–1736.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Wells, A. J. (2002). Gibson's affordances and turing's theory of computation. *Ecological psychology*, 14(3):140–180.

Wiggins, G. A. (2006). A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-based systems*, 19(7):449–458.

Wu, C. M., Schulz, E., Speekenbrink, M., Nelson, J. D., and Meder, B. (2018). Generalization guides human exploration in vast decision spaces. *Nature human behaviour*, 2(12):915–924.

Xie, S., Ma, X., Yu, P., Zhu, Y., Wu, Y. N., and Zhu, S.-C. (2021). Halma: Humanlike abstraction learning meets affordance in rapid problem solving. *arXiv preprint arXiv:2102.11344*.

Yang, X., Yan, M., Pan, S., Ye, X., and Fan, D. (2023a). Simple and efficient heterogeneous graph neural network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 10816–10824.

Yang, Y., Cao, J., Wen, Y., and Zhang, P. (2021). Table to text generation with accurate content copying. *Scientific reports*, 11(1):22750.

Yang, Z., Moerland, T. M., Preuss, M., and Plaat, A. (2023b). Two-memory reinforcement learning. In *2023 IEEE Conference on Games (CoG)*, pages 1–9. IEEE.

Yao, L., Mao, C., and Luo, Y. (2019). Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. (2021). Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 11920–11931. PMLR.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. (2020). Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., et al. (2018). Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*.

Zhai, X. (2023). Chatgpt and ai: The game changer for education. *Zhai, X.(2023). ChatGPT: Reforming Education on Five Aspects. Shanghai Education*, pages 16–17.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115.

Zhang, W., GX-Chen, A., Sobal, V., LeCun, Y., and Carion, N. (2022). Light-weight probing of unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2208.12345*.

Zhao, L., Kong, L., Walters, R., and Wong, L. L. (2022). Toward compositional generalization in object-oriented world modeling. In *International Conference on Machine Learning*, pages 26841–26864. PMLR.

Zhao, S., Song, J., and Ermon, S. (2017). Learning hierarchical features from deep generative models. In *International Conference on Machine Learning*, pages 4091–4099. PMLR.

Zhou, K., Yang, J., Loy, C. C., and Liu, Z. (2022). Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348.

A Appendix: Slot-attention trainning and fine-tuning for object discovery

This appendix documents the experimental process and outcomes of training the Slot Attention model (Locatello et al., 2020) for unsupervised object discovery. The goal was to assess the model's capacity to learn disentangled object representations in controlled synthetic settings and then evaluate its transferability to new, visually distinct environments.

The Slot Attention model was first trained using the CLEVR dataset, which provides high-contrast, compositional scenes with diverse 3D objects. Training was performed with a batch size of 32 over 300 epochs. As shown in Figure 47, the model successfully learned to disentangle and segment multiple objects within each frame, assigning distinct slot vectors to individual shapes. The output included both reconstructed input frames and visualisations of slot-wise object masks, demonstrating accurate object discovery in the source domain.

To test generalisation, the pretrained model was evaluated on frames from the CREATE environment—an interactive, physics-based 2D environment with distinct object dynamics. Without fine-tuning, the model failed to generalise: both reconstructions and slot activations appeared noisy, as illustrated in Figure 48. Notably, despite the simplicity of CREATE scenes, the model attempted to apply 3D priors learned from CLEVR, misrepresenting flat 2D objects as volumetric shapes. This behaviour suggests that the inductive biases learned during training were overfitted to the 3D rendering style and spatial composition of CLEVR.

Subsequent experiments extended training on CREATE-specific data with contrast-enhanced frames. While some improvements in segmentation fidelity were observed, the learned object representations remained fragile. This highlights an important limitation in current slot-based models: their generalisation is highly sensitive to the visual domain and rendering assumptions of the source data. For robust application in agent-centric environments like CREATE or MiniGrid, domain-specific training or meta-adaptation strategies will likely be necessary To address this bias, the model underwent training on the NVLRM dataset, a 2D dataset of shapes Fig.51. The training results, similar to those of the CLVR dataset, indicated proficient slot allocation and reconstruction. Nevertheless, the model still struggled to generalise across shapes, as evidenced by the lack of meaningful results

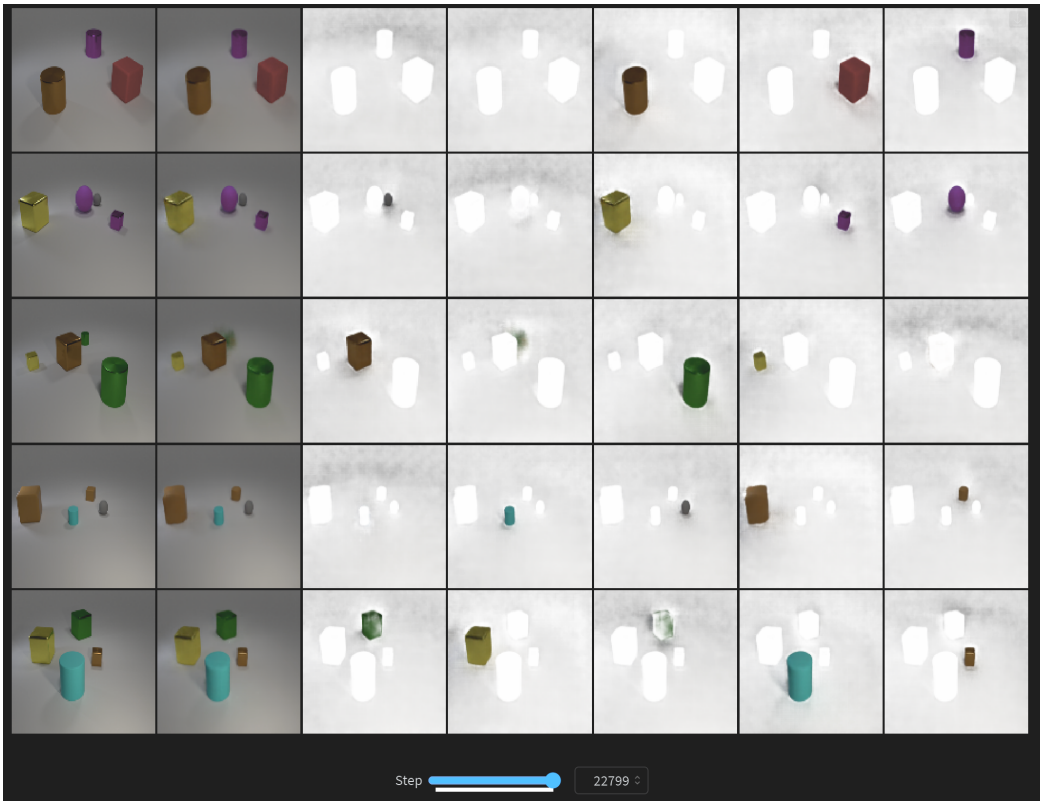


Figure 49: Slot Attention model trained on CLEVR: Input frames, object reconstructions, and slot masks show consistent disentanglement and identification of multiple objects.

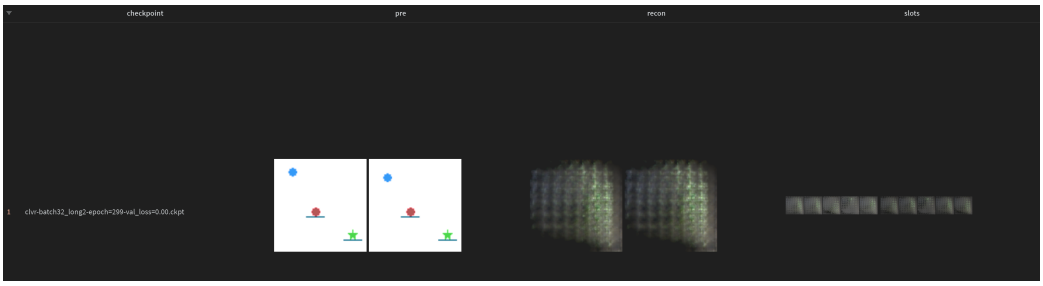


Figure 50: Transfer failure: CLEVR-trained Slot Attention applied to CREATE. Reconstructions and slot activations are poor, revealing overfitting to CLEVR's 3D priors.

in both reconstruction and slots.

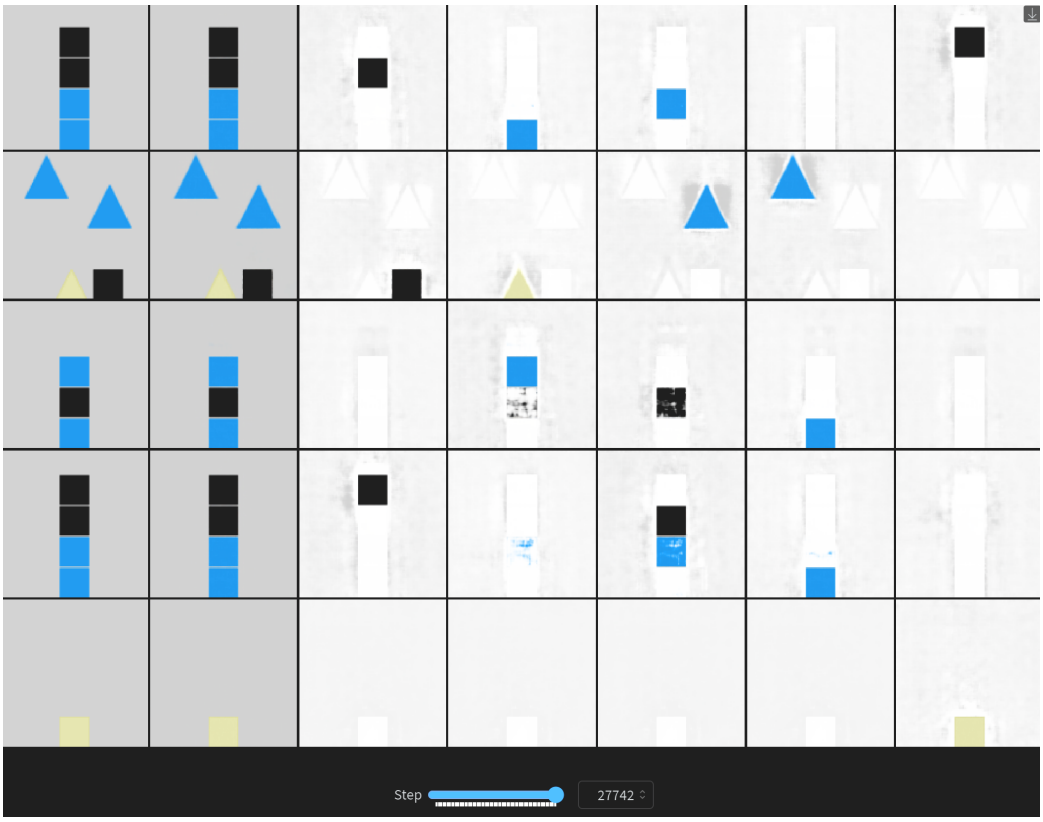


Figure 51: Slot attention segmentation on NVLRM after training directly on the same dataset. While the model exhibited stable reconstruction during training, transfer to other settings remained limited.

Additional experiments were conducted to improve the performance of slot-based object discovery, including preprocessing CREATE frames to artificially enlarge objects and training with a noisy autoencoder to encourage robustness. However, these strategies failed to yield significant improvements in either slot quality or reconstruction accuracy.

Fine-tuning was also explored, using the best-performing checkpoints pretrained on CLVR or NVLRM, and exposing them to data from the CREATE and AnimalAI environments. Randomised frames from both environments were used to fine-tune the autoencoder for up to five epochs. Although fine-tuning improved reconstruction fidelity to some extent, the model continued

to struggle with meaningful object segmentation, especially in scenes where objects were partially occluded, off-centre, or low in contrast. In CREATE, where objects appear in low resolution and often blend into a white background, both reconstruction and slot separation remained unreliable.

To counteract dataset-specific biases, we also trained the model from scratch on frames from CREATE. However, the results were even less promising. The network frequently defaulted to producing blank or all-white reconstructions, likely overwhelmed by the visual uniformity and low contrast of the background. Although extending the training time produced some marginal improvements—such as slight slot divergence from the reconstruction—these gains were insufficient. The slots often remained too similar to one another, failing to capture distinct object-level features or provide useful semantic separation.

Moreover, the need to train or fine-tune concept extraction models individually for each environment is itself undesirable. Such an approach contradicts the broader goal of generalisable perception and concept formation, as it undermines scalability and reusability across domains. In the context of lifelong learning and cross-environment generalisation, reliance on environment-specific training pipelines would introduce unacceptable fragility and manual overhead.

To mitigate the model's tendency to favour reconstruction quality over slot disentanglement, we modified the slot loss to incorporate a novelty term based on pairwise slot dissimilarity. This novelty coefficient penalised slots that were too similar, encouraging the model to spread attention across distinct features. While this modification led to modest improvements in slot diversity, it also introduced instability in training and failed to boost object-level segmentation or downstream usability significantly.

These findings highlight the limitations of directly applying slot-attention models to real-world or semi-naturalistic environments like CREATE and AnimalAI. Despite promising results on synthetic datasets, transfer remained elusive—underscoring that high-quality object discovery in reinforcement learning contexts requires not only robust architectures but carefully designed perception pipelines.

As a result, the final implementation adopted a pragmatic approach: a pre-processing pipeline using the Segment Anything Model (SAM) and CLIP for object extraction, described in detail in the Methodology Chapters.

The novelty coefficient, defined as:

$$novelty = mean(p2 - distance(s_i, s_j))$$

where s_i and s_j was integrated into the loss function as follows:

$$loss = mse\_loss(recon\_combined, input) + \frac{1}{novelty}$$

B Appendix: Associative Strength Clustering for Long-Term Memory Selection

B.1 Mackintosh-Based Associative Attention Mechanism

As an alternative to trainable transformer-based attention, this thesis also explored a cognitively inspired, interpretable mechanism for long-term memory (LTM) selection based on associative learning theory. Specifically, we implemented a variation of the Mackintosh model (1975), which modulates concept salience based on its historical correlation with reward outcomes. This model provides a symbolic and lightweight means of deciding which object concepts in Working Memory (WM) should be promoted to LTM.

The core principle is that concepts which consistently help predict reward should become more salient (i.e., receive higher attention), while concepts that are consistently irrelevant should be downweighted or forgotten.

The update rule for the associative value $V_A^{(n)}$ at time n is:

$$\Delta V_A^{(n+1)} = \alpha_A^n \cdot \beta \cdot (1 - V_A^n + \overline{V_A^n}) \cdot |R^n| \tag{35}$$

The associability of concept A, denoted α_A, is then updated based on the comparative prediction error:

$$\Delta \alpha_A^{(n+1)} = -\Theta \cdot \left(|\lambda^n - V_A^n + \overline{V_A^n}| - |\lambda^n - V_{\neg A}^n + \overline{V_{\neg A}^n}| \right) \tag{36}$$

Concepts that consistently correlate with reward will see their α increase, while concepts that are uninformative will gradually decay in associability.

Over time, this mechanism converges toward a sparse set of high-attention concepts that are reliably associated with task success. These are then selected for promotion to LTM. Concepts that appear frequently but without predictive value decay naturally in associability, filtering out distractors.

By using this symbolic mechanism alongside the learnable self-attention approach described in the main chapters, this thesis offers two complementary strategies for determining relevance in memory systems: one grounded in cognitive theory and interpretability, the other in gradient-based optimisation and scalability.

B.2 Python Implementation

The associative attention mechanism described above was implemented using Python and Neo4j. The following function updates attention values for object concepts in Working Memory (WM) based on their historical association with reward. Attention values and associabilities are stored in a Neo4j database and updated online during training.

B.2.1 Modified Mackintosh Attention Update

```
def compute_attention(self, time, episode_id, omega=0.1, beta=0.6, default_alpha=0.1):
    ep_id = int(episode_id.split(':')[2])
    reinforcer_and_sum = self.concept_space.objects_attention_and_reinforcer(time, ep_id)
    reinforcer_time_t = abs(reinforcer_and_sum['reinforcer'][0])
    sum_all_obj = reinforcer_and_sum['sum(o.att)'][0]
    reward_current_time = reinforcer_and_sum['s.reward'][0]

    df_all_obj_att = self.concept_space.get_obj_att_at_time_t(time, ep_id)
    df_all_obj_att['not_obj_i'] = sum_all_obj - df_all_obj_att['o.att']

    obj_values_prev_time = self.concept_space.get_obj_att_values_prev_time(time, ep_id)
    if not obj_values_prev_time.empty:
        obj_values_prev_time['last_value_obj_i'] = obj_values_prev_time['prev_att']
        obj_values_prev_time['sum_val_obj_not_i'] = (
            obj_values_prev_time['last_value_obj_i'].sum() - obj_values_prev_time['last_value_obj_i']
        )
        obj_values_prev_time['delta_alpha_obj_i'] = -omega * (
            reward_current_time - obj_values_prev_time['last_value_obj_i']
        ) - (
            reward_current_time - obj_values_prev_time['sum_val_obj_not_i']
        )
        obj_values_prev_time['alpha_obj_i_temp'] = (
```

```
            obj_values_prev_time['alpha'] +
    obj_values_prev_time['delta_alpha_obj_i']
        )
        obj_values_prev_time.loc[obj_values_prev_time["
    alpha_obj_i_temp"] <= 0.05, 'alpha_obj_i'] = 0.05
        obj_values_prev_time.loc[obj_values_prev_time["
    alpha_obj_i_temp"] >= 1, 'alpha_obj_i'] = 1

        obj_to_update = pd.merge(df_all_obj_att,
    obj_values_prev_time, on="id_o")
        obj_to_update['delta_obj_i'] = (
            obj_to_update['alpha_obj_i'] * beta * (1 -
    obj_to_update['o.att']) * reinforcer_time_t
        )
        obj_to_update['new_value_obj_i'] = obj_to_update['o.
    att'] + obj_to_update['delta_obj_i']

    else:
        obj_to_update = df_all_obj_att
        obj_to_update['delta_obj_i'] = (
            default_alpha * beta * (1 - obj_to_update['o.att'
    ]) * reinforcer_time_t
        )
        obj_to_update['new_value_obj_i'] = obj_to_update['o.
    att'] + obj_to_update['delta_obj_i']
        obj_to_update['alpha_obj_i'] = obj_to_update['alpha']

    # Update Neo4j memory with new attention values
    dict_values_to_update = list(obj_to_update[[
        'id_o', 'new_value_obj_i', 'att_values', 'alpha_obj_i
    ']
    ]].to_dict('index').values())
    self.concept_space.update_objects_attention(
    dict_values_to_update)
```

B.3 Visualisation of Associative Attention

To better understand how the associative attention mechanism evolves during training, we visualise both the dynamic changes in attention values over time and the final attention scores after interaction with the environment.

Figure 52 presents the trajectory of attention weights computed using the Mackintosh model for several object concepts across timesteps. Each colored line corresponds to a distinct object ID encountered during training. The

trends clearly demonstrate how some objects rapidly increase in attention due to strong predictive associations with reward, while others remain low, indicating a lack of reward relevance. For instance, the orange and red lines represent concepts whose attention sharply increases and stabilises near the maximum, reflecting high utility in predicting outcomes. In contrast, flatter or noisier lines represent objects that were encountered often but did not prove predictive.

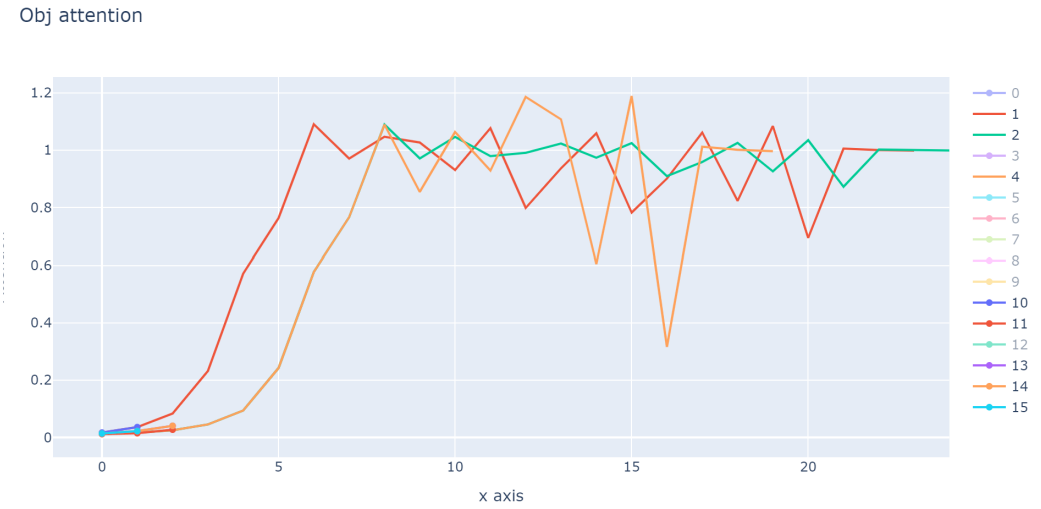


Figure 52: Evolution of object attention values over training timesteps. Line color represents object ID.

Figure 53 offers a static snapshot of the final attention values, concept frequencies, and raw association strengths after a full episode. Here, we observe the outcome of the associative learning process: object concepts that appeared frequently and accrued positive reward influence display higher associative weights and attention scores (values near 1.0), while others that were rare or non-predictive maintain low attention. The use of a frequency-normalised relevance metric (e.g., vobj/freq) helps highlight whether concepts were predictive due to their reward correlation or simply frequent co-occurrence.

```
match(e:Episode)-[:has_state]-(s)-[r:has_object]→(o) where
elementId(e)="4:35c6f93f-aedf-40a2-ba92-c88bc937e420:387"
with elementId(o) as id,count(r) as freq, o.att as vobj, o.alpha
as att
return id,freq, vobj, att, vobj/freq as vobj2
```

	freq	vobj	att	vobj2
93f-aedf-40a2-ba92-c88bc937e420:436"	11	1.571084551262026	1.0	0.1428258682965478
93f-aedf-40a2-ba92-c88bc937e420:395"	25	0.9985579261334732	1.0	0.039942317045338925
93f-aedf-40a2-ba92-c88bc937e420:390"	25	0.9993993007810059	1.0	0.03997597203124024
93f-aedf-40a2-ba92-c88bc937e420:453"	1	0.01	0.1	0.01
93f-aedf-40a2-ba92-c88bc937e420:419"	13	0.9598239081105198	1.0	0.07383260831619383
93f-aedf-40a2-ba92-c88bc937e420:389"	20	0.9967742674953202	1.0	0.04983871337476601

Figure 53: Final associative values for each object concept based on accumulated reward history.

Together, these figures illustrate how the Mackintosh-based associative clustering mechanism provides a grounded and interpretable means of selecting memory candidates for long-term storage. By prioritising concepts that repeatedly co-occur with reward and suppressing uninformative distractors, the system builds a semantically meaningful long-term memory. This process serves as a cognitively inspired alternative to black-box trainable attention layers and enables agents to curate their own symbolic concept space incrementally.

C Appendix: Additional Logs and Visualisations

To support the experimental findings presented in the main body of the thesis, we include below supplementary visualisations and logs from training runs involving SETLE-enhanced agents. These logs offer further insight into the internal behaviour of the agent's graph memory system and enrichment mechanisms.

C.1 Trajectory-Level Behaviour

Figure 54 shows five key trajectory-level metrics recorded throughout training in the CREATE environment using the Soft update with adapter strategy. These include:

- **UniqueTools:** Number of distinct tools used per episode.
- **ToolSwitches:** Number of tool transitions per episode.
- **Steps:** Total interaction steps before success/failure.
- **Reward:** Episode reward signal.
- **RedundantActions:** Number of ineffective or repeated actions.

These metrics help visualise agent behaviour, tool use diversity, and learning progression over time.

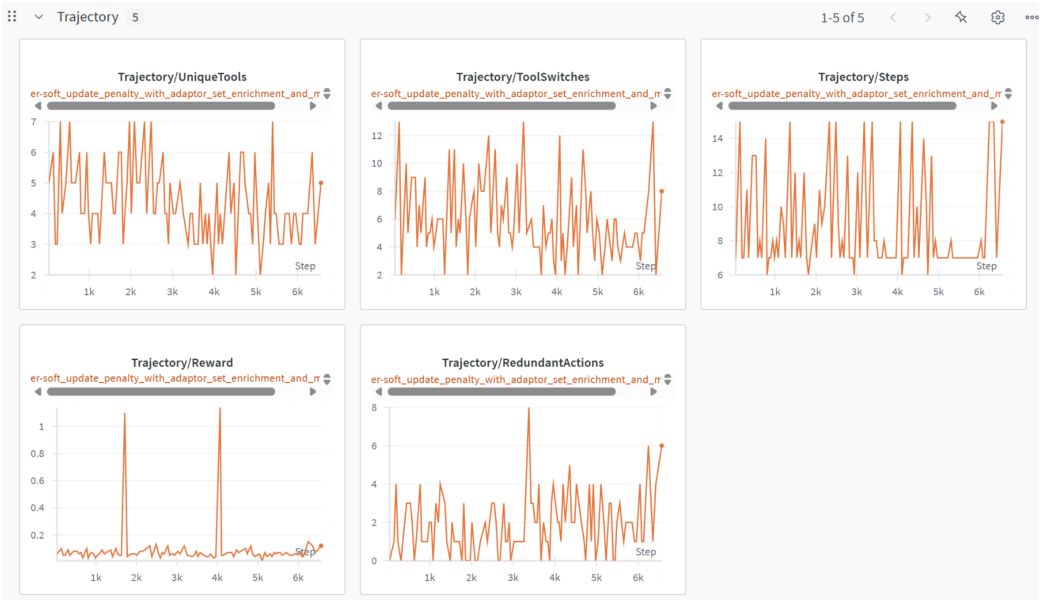


Figure 54: Trajectory-level statistics during training: unique tools, tool switches, steps per episode, reward progression, and redundant actions.

C.2 Attention Weight Visualisation

Figure 55 shows logs of attention weights assigned to `ObjectConcept` nodes at different timesteps. The top-weighted nodes correspond to object concepts most relevant for contextual enrichment. Consistent attention on specific concepts suggests stable and interpretable memory reuse.

Figure 55: Attention weights over ObjectConcept nodes across enrichment steps. Each panel shows the top-weighted retrieved nodes and their scores.

C.3 Episode Matching Frequency and Similarity

Figures 56 and 57 depict the episode retrieval behaviour of the system. Figure 56 shows the top-3 matched episodes at a selected timestep (e.g., $t = 3$), based on cosine similarity to the current partial trace. Figure 57 presents the histogram of most frequently matched episode IDs, indicating which trajectories were repeatedly reused as templates during enrichment.

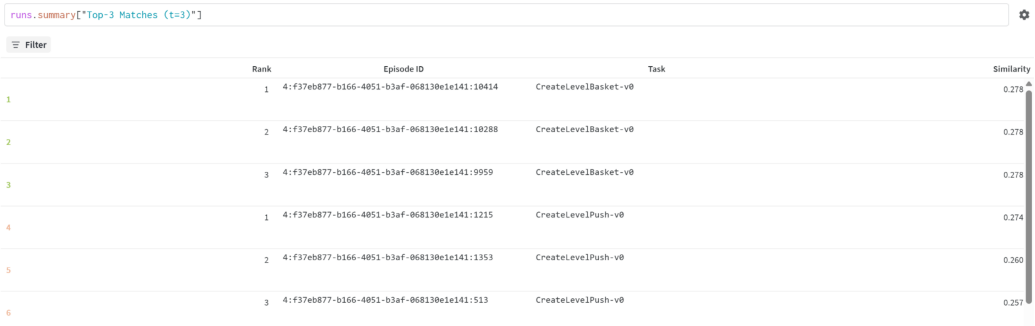


Figure 56: Episodic matches with their similarities retrieved by SETLE at timestep $t = 3$.

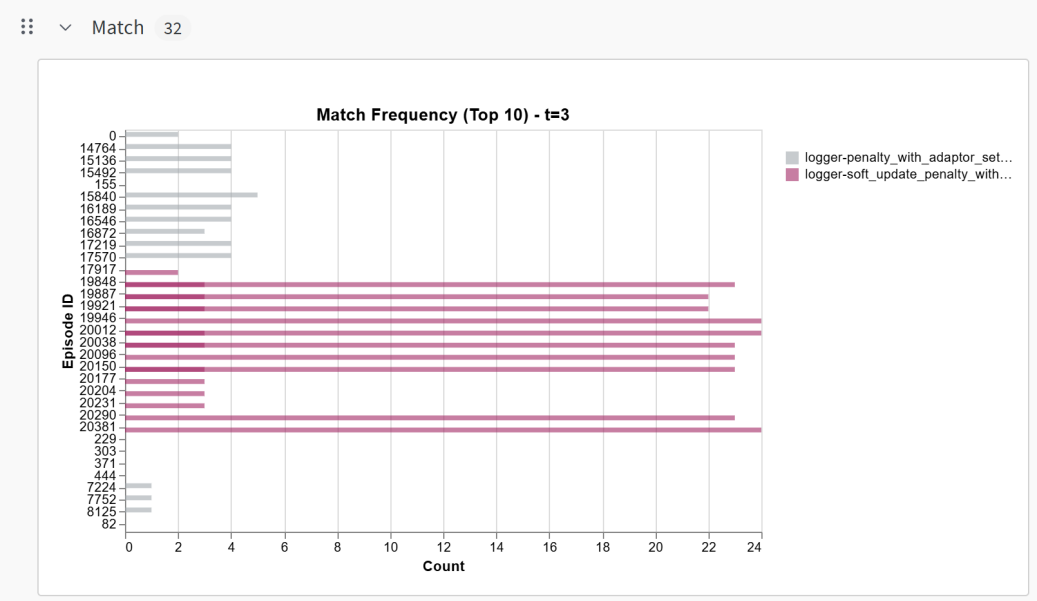


Figure 57: Histogram showing episode retrieval frequency for top-10 matched episodes across runs and strategies. Higher counts reflect consistent structural similarity.

These visualisations collectively demonstrate that the SETLE framework not only retrieves past experience selectively and consistently, but also enables dynamic adaptation of attention to support sample-efficient, structured generalisation.