MINDSTORES: MEMORY-INFORMED NEURAL DECISION SYNTHESIS FOR TASK-ORIENTED REINFORCEMENT IN EMBODIED SYSTEMS

Anirudh Chari[1] **Suraj Reddy**[1] **Aditya Tiwari**[2] **Richard Lian**[1] **Brian Zhou**[3]
[1]Massachusetts Institute of Technology, Cambridge, MA
[2]Illinois Mathematics and Science Academy, Aurora, IL
[3]Harvard University, Cambridge, MA
{anichari,surajrdy,rlian}@mit.edu; atiwari@imsa.edu,
brianzhou@college.harvard.edu

ABSTRACT

While large language models (LLMs) have shown promising capabilities as zero-shot planners for embodied agents, their inability to learn from experience and build persistent mental models limits their robustness in complex open-world environments like Minecraft. We introduce MINDSTORES, an experience-augmented planning framework that enables embodied agents to build and leverage *mental models* through natural interaction with their environment. Drawing inspiration from how humans construct and refine cognitive mental models, our approach extends existing zero-shot LLM planning by maintaining a database of past experiences that informs future planning iterations. The key innovation is representing accumulated experiences as natural language embeddings of (state, task, plan, outcome) tuples, which can then be efficiently retrieved and reasoned over by an LLM planner to generate insights and guide plan refinement for novel states and tasks. Through extensive experiments in the MineDojo environment, a simulation environment for agents in Minecraft that provides low-level controls for Minecraft, we find that MINDSTORES learns and applies its knowledge significantly better than existing memory-based LLM planners while maintaining the flexibility and generalization benefits of zero-shot approaches, representing an important step toward more capable embodied AI systems that can learn continuously through natural experience.

1 INTRODUCTION

Recent advances in large language models (LLMs) have demonstrated enhanced capabilities in reasoning (Plaat et al., 2024; Huang & Chang, 2023), planning (Sel et al., 2025), and decision-making (Huang et al., 2024) through methods that strengthen analytical depth. Among the numerous domains of active innovation, the success of AI agents serves as a critical benchmark for assessing our progress toward generally capable artificial intelligence (Brown et al., 2020).

Building *embodied* agents—AI systems with physical form—that learn continuously from real-world interactions through persistent memory and adaptive reasoning remains a fundamental challenge in the future of artificial intelligence. Classical approaches, such as reinforcement learning (Dulac-Arnold et al., 2021) and symbolic planning (Zheng et al., 2025), struggle with scalability, irreversible errors, and rigid assumptions in complex environments.

A promising paradigm for such agents leverages LLMs as high-level planners (Jeurissen et al., 2024): the LLM decomposes abstract goals into step-by-step plans (e.g., "mine wood → craft tools → smelt iron"), while a low-level controller translates these plans into environment-specific actions (e.g., movement, object interaction). This "brain and body" architecture capitalizes on the LLM's capacity for structured reasoning while grounding its outputs in the dynamics of the physical world—a critical capability for real-world applications like robotic manipulation (Shentu et al., 2024; Bhat et al., 2024; Wang et al., 2024b), autonomous navigation (Zawalski et al., 2024), and adaptive disaster response.

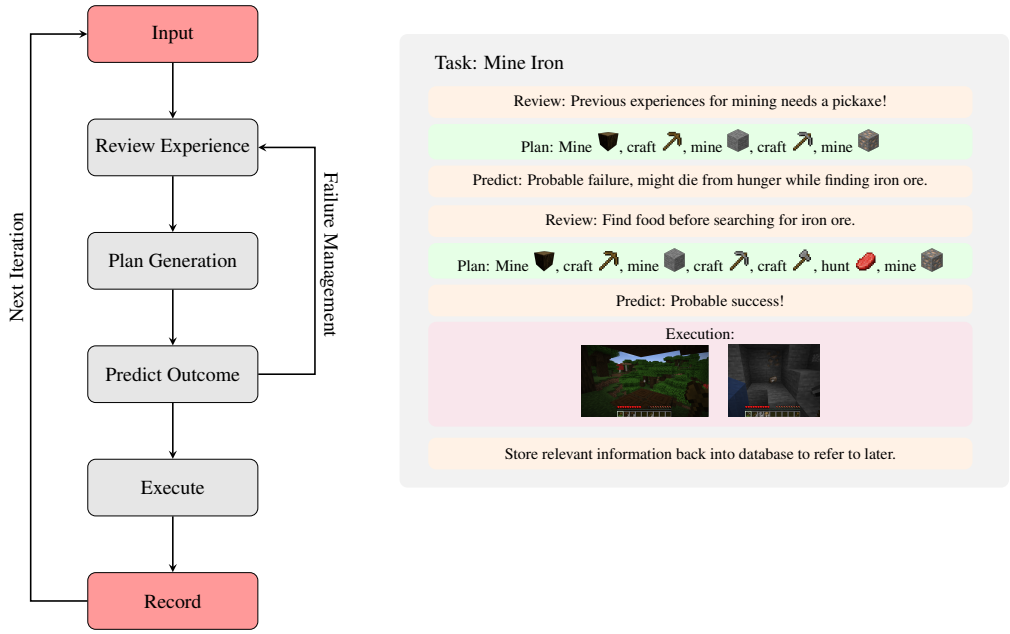

Figure 1: Overview of the MINDSTORES planning architecture. The left shows the iterative experiential learning pipeline leveraging the experience database. Database-related methods are in orange, planning steps are in green, and Minecraft steps are in red. The right shows an example applying this pipeline to an example task in Minecraft.

While recent LLM-based agents show promise in generating action plans for embodied tasks, many lack *experiential* learning, i.e., the ability to apply insights from past experiences to planning for future tasks. Unlike humans—who build mental models to generalize insights, avoid errors, and reason counterfactually (e.g., "Crafting a stone pickaxe first would enable iron mining")—existing agents cannot synthesize persistent representations of past interactions. This gap hinders their ability to tackle long-horizon tasks in open worlds like Minecraft, where success requires inferring objectives, recovering from failures, and transferring insights across scenarios.

Minecraft exemplifies these challenges: agents must explore procedural terrains, infer task dependencies (e.g., stone tools before iron mining), and adapt to unforeseen challenges. Current LLM planners, namely zero-shot architectures like DEPS (Wang et al., 2024c), exhibit critical flaws: (1) they lack persistent mental models, causing repetitive errors (e.g., using wooden pickaxes for iron mining); and (2) they underutilize LLMs' reasoning to synthesize experiential insights, producing brittle plans.

To address these limitations, we propose MINDSTORES, a framework that leverages LLMs to construct dynamic mental models—internal representations guiding reasoning and decision-making, inspired by human cognition. Just as humans build simplified models of reality to anticipate events and solve problems, our approach equips agents to actively interpret experiences through structured reasoning. By analyzing failures (e.g., "Wooden pickaxes break mining iron"), inferring causal rules (e.g., "Stone tools are prerequisites"), and predicting outcomes, the LLM transforms raw interaction data into adaptive principles.

MINDSTORES augments planners with an experience database storing natural language tuples (state, task, plan, outcome) and operates cyclically: observe, retrieve relevant experiences, synthesize context-aware plans, act, and log outcomes. This closed-loop process enables semantic analysis of memories, iterative strategy refinement, and outcome prediction, bridging the gap between static planning and experiential learning while grounding agent reasoning in human-like cognitive foundations.

Hence, our key contributions are as follows:

- A cognitive-inspired formulation of artificial mental models to enable natural-language memory accumulation and transfer learning.

- **MINDSTORES**, a novel open-world LLM planner leveraging the above formulation to develop lifelong learning embodied agents.
- Extensive evaluation of MINDSTORES in Minecraft, demonstrating a **9.4%** mean improvement in open-world planning tasks over existing methods.

In the remainder of this paper, we detail the theoretical foundations of mental models in Section 2, present the MINDSTORES architecture in Section 3, and validate its performance through experiments in Sections 4 and 5. Our findings underscore the critical role of memory-informed reasoning in developing lifelong learning agents for open-world environments.

2 BACKGROUND

2.1 OPEN-WORLD PLANNING FOR EMBODIED AGENTS

Planning for embodied agents in open-world environments presents unique challenges due to the unbounded action space, long-horizon dependencies, and complex environmental dynamics. In environments like Minecraft, agents must reason about sequences of actions that may span dozens of steps, where early mistakes can render entire trajectories infeasible (Fan et al., 2022). Traditional planning approaches that rely on explicit state representations and value functions struggle in such domains due to the combinatorial explosion of possible states and actions.

The key challenges in open-world planning stem from two main factors. First, the need for accurate multi-step reasoning due to long-term dependencies between actions presents a significant hurdle. Second, the requirement to consider the agent's current state and capabilities when ordering parallel sub-goals within a plan poses additional complexity. Consider the example of crafting a diamond pickaxe in Minecraft: the process requires first obtaining wood, then crafting planks and sticks, mining stone with a wooden pickaxe, crafting a stone pickaxe, mining iron ore, smelting iron ingots, and finally crafting the iron pickaxe—a sequence that can easily fail if any intermediate step is incorrectly executed or ordered.

2.2 ZERO-SHOT LLM PLANNING WITH DEPS

Recent work has shown that large language models can serve as effective zero-shot planners for embodied agents through their ability to decompose high-level tasks into sequences of executable actions (Huang et al., 2022a). The DEPS (Describe, Explain, Plan and Select) framework leverages this capability through an iterative planning process that combines several key components (Wang et al., 2024c). The framework utilizes a descriptor that summarizes the current state and execution outcomes, an explainer that analyzes plan failures and suggests corrections, a planner that generates and refines action sequences, and a selector that ranks parallel candidate sub-goals based on estimated completion steps.

The key innovation of DEPS is its ability to improve plans through verbal feedback and explanation. When a plan fails, the descriptor summarizes the failure state, the explainer analyzes what went wrong, and the planner incorporates this feedback to generate an improved plan. This creates a form of zero-shot learning through natural language interaction.

However, DEPS and similar approaches maintain no persistent memory across episodes. Each new planning attempt starts fresh, unable to leverage insights gained from previous successes and failures in similar situations. This limitation motivates our work on experience-augmented planning.

2.3 MENTAL MODELS

Mental models are cognitive representations of how systems and environments work, enabling humans to understand, predict, and interact with the world around them. Originally proposed by Craik (1952), mental models theory suggests that people construct small-scale internal models of reality that they use to reason, anticipate events, and guide behavior. These models are built through experience and observation, continuously updated as new information becomes available, and help reduce cognitive load by providing ready-made frameworks for understanding novel situations.

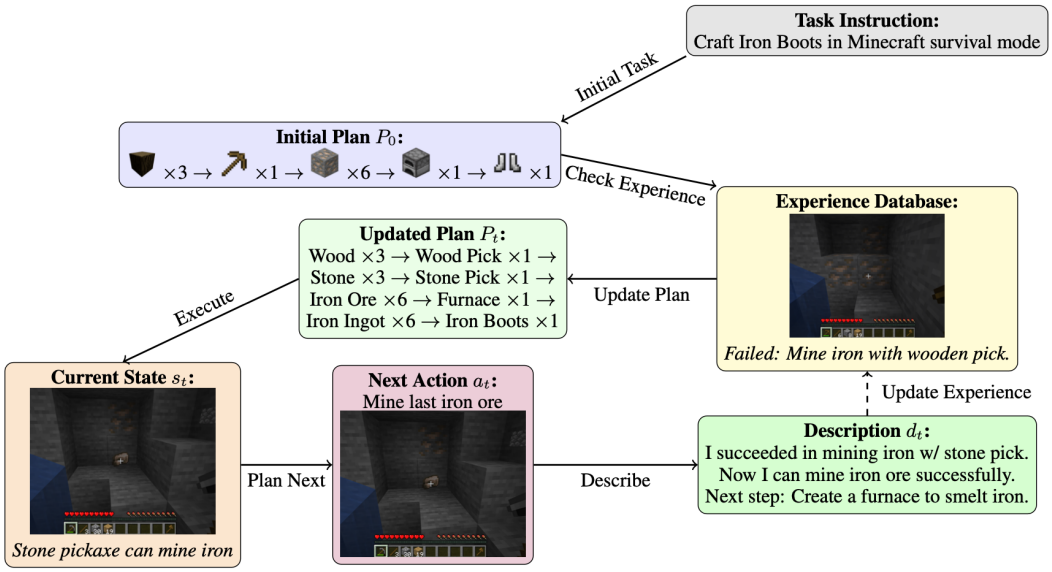

Figure 2: Interactive planning process for crafting iron boots in Minecraft. The system initially plans to mine iron with a wooden pickaxe but learns from past experience that this will fail. It then updates the plan to include creating a stone pickaxe first, leading to successful iron ore mining.

A key insight from psychological research on mental models is their role in transfer learning and generalization (Canini et al.). When faced with new scenarios, humans naturally draw upon their existing mental models to make informed decisions, even in previously unseen contexts. This ability to leverage past experiences through abstract representations is particularly relevant for embodied agents operating in open-world environments, where they must constantly adapt to novel situations while maintaining coherent, generalizable knowledge about environmental dynamics.

3 METHODS

3.1 OVERVIEW

We propose an experience-augmented planning framework that maintains a similar foundation to DEPS but advances by maintaining a persistent mental model of the environment through natural language experiences. Our approach integrates several key components into a cohesive system. The framework maintains a database $\mathcal{D}$ of experience tuples (s, t, p, o) containing state descriptions s, tasks t, plans p, and outcomes o. This is complemented by a semantic retrieval system for finding relevant past experiences, an LLM planner that generates insights and plans informed by retrieved experiences, and a prediction mechanism that estimates plan outcomes before execution.

3.2 EXPERIENCE DATABASE

Each experience tuple $(s, t, p, o) \in \mathcal{D}$ consists of natural language paragraphs describing the environmental context. The state s captures the environmental context and agent's condition. The task t represents the high-level goal to be achieved. The plan p contains the sequence of actions generated by the planner. Finally, the outcome o describes the execution result and failure description if applicable.

For each component, we compute a dense vector embedding $e(x) \in \mathbb{R}^d$ using a pretrained sentence transformer, where x represents any of s, t, p, or o. This allows efficient similarity-based retrieval using cosine distance:

$$\text{sim}(x_1, x_2) = \frac{e(x_1) \cdot e(x_2)}{\|e(x_1)\| \|e(x_2)\|} \tag{1}$$

3.3 EXPERIENCE-GUIDED PLANNING

Given a new state s_t and task t_t, our algorithm proceeds through several stages. Initially, it retrieves the k most similar past experiences based on state and task similarity:

$$\mathcal{N}_k(\mathcal{D}, s_t, t_t) = \text{top-k}_{(s,t,p,o)\in\mathcal{D}} \left[\sum_{x\in\{s,t\}} \lambda_x \text{sim}(x, x_t) \right] \tag{2}$$

The LLM is then prompted to analyze these experiences and generate insights about common failure modes to avoid, successful strategies to adapt, and environmental dynamics to consider. Following this analysis, it generates an initial plan p_t conditioned on the state, task, experiences, and insights.

The system then predicts the likely outcome by retrieving similar past plans:

$$\mathcal{N}_k(s_t, t_t, p_t) = \text{top-k}_{(s,t,p,o)\in\mathcal{D}} \left[\sum_{x\in\{s,t,p\}} \lambda_x \text{sim}(x, x_t) \right] \tag{3}$$

If predicted outcomes suggest likely failure, the system returns to the plan generation stage to revise the plan. Finally, it executes the plan and stores the new experience tuple in $\mathcal{D}$. The complete process is formalized in Algorithm 1.

Algorithm 1 Experience-Augmented Planning

Require: State s_t, Task t_t, Database $\mathcal{D}$, LLM M, k neighbors
Ensure: Plan p_t

```
N_k ← retrieve_top_k(D, s_t, t_t, k)
insights ← M.analyze_experiences(N_k)
p_t ← M.generate_plan(s_t, t_t, N_k, insights)
while true do
    similar_plans ← get_similar_plans(D, s_t, t_t, p_t)
    pred_outcome ← analyze_outcomes(similar_plans)
    if pred_outcome is success then
        break
    end if
    p_t ← M.revise_plan(p_t, pred_outcome)
end while
outcome ← execute_plan(p_t)
D.add((s_t, t_t, p_t, outcome))
return p_t
```

3.4 DESIGN JUSTIFICATION

Our approach incorporates several carefully considered design elements that work together to create an effective planning system. The use of natural language experiences, rather than vectors or symbolic representations, leverages the LLM's ability to perform flexible reasoning over arbitrary descriptions. The semantic retrieval system employs dense embeddings to enable efficient similarity search while capturing semantic relationships between experiences beyond exact matches. The two-stage retrieval process first retrieves experiences based on state/task similarity to inform plan generation, then retrieves similar plans to predict outcomes, allowing the planner to both learn from past experiences and validate new plans. Finally, the iterative refinement capability enables the planner to revise plans based on predicted outcomes before execution, reducing the cost of actual failures.

This design maintains the benefits of zero-shot LLM planning while enabling continual learning through natural experience.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate our experience-augmented planning approach in MineDojo using 8 tiers of task complexity (MT1–MT8) (Fan et al., 2022). The observation space includes RGB view, GPS coordinates, and inventory state, with 42 discrete actions mapped from MineDojo's action space (Fan et al., 2022). All experiments utilize the behavior cloning controller trained on human demonstrations, following similar methodology to DEPS and Voyager. Due to software version constraints, our implementation of the controller achieves lower baseline performance than the original DEPS controller. Therefore, we use our implementation of DEPS without the experience database as the primary baseline for fair comparison. Each task is evaluated over 30 trials with randomized initial states and a fixed random seed of 42.

Our experience database uses Sentence-BERT embeddings (768-dim) stored in FAISS for efficient search. Key parameters were determined through ablation studies:

- Optimal $k = 5$ neighbors (tested $k = 1, 3, 5, 10, 20$)
- Weighted similarity: $\lambda_s = 0.4$ (state), $\lambda_t = 0.4$ (task), $\lambda_p = 0.2$ (plan)

For the complete agent algorithm and associated LLM prompts, see Appendix A, and for detailed implementation aspects including environment integration and neural component configurations, see Appendix B.

4.2 EVALUATION TASKS

We evaluate on 53 Minecraft tasks grouped into 3 complexity tiers:

- **Basic (MT1–MT2)**: Fundamental tasks (wood/stone tools, basic blocks)
- **Intermediate (MT3–MT5)**: Progressive tasks (food, mining, armor crafting)
- **Advanced (MT6–MT8)**: Complex tasks (iron tools, minecart, diamond)

Episode lengths range from 3,000 steps (Basic) to 12,000 steps (Challenge tasks).

For additional task details and performance statistics, see Appendix C.

4.3 BASELINES

We compare MINDSTORES to the following existing state-of-the-art approaches:

- **DEPS**: State-of-the-art zero-shot LLM planner (Wang et al., 2024c)
- **Voyager**: Automated curriculum learning agent (Wang et al., 2023)
- **Reflexion**: LLM planner with environmental feedback (Shinn et al., 2023)

4.4 ABLATIONS

To analyze the function of each individual component of the MINDSTORES framework, we perform the following ablations:

- **No Experience**: Remove retrieval component
- **Fixed k Values**: Test $k = 1, 3, 5, 10, 20$ retrieval contexts
- **Single-Shot**: Disable iterative plan refinement (DEPS)

4.5 METRICS

To quantify each method's performance in open-world planning in the Minecraft environment, we measure:

- **Success Rate**: Completion percentage across trials
- **Learning Efficiency**: Iterations required for skill mastery
- **Complexity Scaling**: Performance vs. task complexity tiers
- **Retrieval Impact**: Success rate vs. context size (k)
- **Continuous Learning**: Effect of non-discrete experience database for each task progression

5 Results and Analysis

Our experiments reveal significant performance differences between MINDSTORES and DEPS across task categories, highlighting key insights into their scalability and effectiveness.

5.1 Performance Metrics

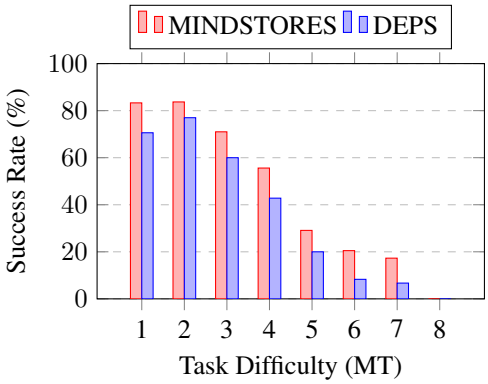

Figure 3: Performance comparison: MINDSTORES consistently outperforms DEPS across tasks. Both systems show declining success rates with increasing complexity (MT1–MT8), with MT8 resulting in 0% success for both. Mean difference: 9.4%.

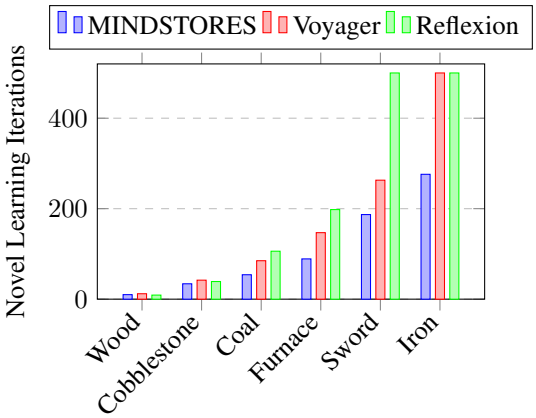

Figure 4: Novel learning iterations across different Minecraft tasks. MINDSTORES demonstrates superior efficiency in complex tasks. (Note: Iteration counts for Reflexion are capped at 500 in later tasks.)

As we analyze Figure 3 in comparison to our version of DEPS, we see an all-around improvement with the addition of the experience database.

Fundamental Tasks (MT1–MT2)

Both systems achieve strong performance in fundamental crafting tasks, with DEPS achieving success rates of 70.6–77.0% and MINDSTORES performing slightly better at 83.3–83.7%. Notably, there is near-parity in Wooden Axe crafting, with both systems achieving a 96.7% success rate. However, the largest performance gap in MT1 occurs in Stick production, where MINDSTORES outperforms DEPS by 6.3%. In MT2, MINDSTORES maintains a consistent advantage, with an average performance improvement of 6.7% across tasks.

Intermediate Tasks (MT3–MT5)

The maximum disparity between the two systems occurs in MT3 painting, where MINDSTORES achieves a 96.7% success rate compared to DEPS's 76.7%, resulting in a 20.0% performance gap. In cooked meat tasks, MINDSTORES maintains a 6.7–16.7% advantage over DEPS. For MT5 armor challenges, the performance gaps are particularly pronounced, with Leather Helmet showing a 20.0% difference and Iron Boots a 10.3% difference. Overall, MINDSTORES maintains an average advantage of +11.0% across intermediate tasks, demonstrating significant divergence in system performance.

ADVANCED TASKS (MT6–MT8)

In MT6 iron tool crafting, MINDSTORES achieves an average performance improvement of 12.2% over DEPS, with the Iron Axe task showing a particularly large gap (23.3% vs. 6.7%). MT7 highlights another standout difference, with Tripwire Hook success rates at 43.3% for MINDSTORES compared to 20.0% for DEPS. However, both systems experience a performance collapse in advanced tasks, with MT6–MT7 success rates dropping below 21% (DEPS: 6.7–8.3%, MINDSTORES: 17.3–20.5%). Notably, neither system can solve the MT8 diamond crafting challenge, with both achieving a 0% success rate.

5.2 LEARNING EFFICIENCY ANALYSIS

MINDSTORES demonstrates superior learning efficiency, particularly for complex tasks. For basic tasks like mining wood and cobblestone, all systems perform comparably (9 to 42 iterations) (see Figure 4). However, as complexity increases, MINDSTORES requires fewer iterations (54 to 276) compared to Voyager and Reflexion, which show exponential increases in required iterations.

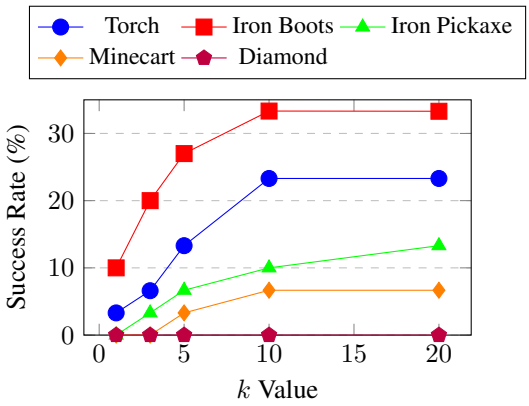

Figure 5: Success rates vs. retrieval context size k for different tasks. Simple tasks improve steadily with k, while more complex tasks require larger k values. Advanced tasks remain unachievable regardless of k.

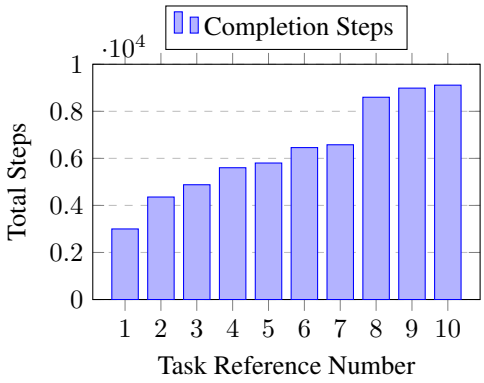

Figure 6: Steps required for task completion with continuous building of the experience database. (See Appendix Table 1 for corresponding tasks.)

5.3 SCALABILITY WITH TASK COMPLEXITY

Performance divergence becomes pronounced with increasing task complexity. MINDSTORES maintains efficient novel learning iterations for tasks like crafting a stone sword and mining iron, while Voyager and Reflexion require significantly more iterations, even reaching the max range (500+) for a relatively simple Mine Iron task (see Figure 4).

5.4 CONTINUOUS EXPERIENCE BUILDING ANALYSIS

Figure 6 shows an experiment in which the experience database is not reset between tasks but is built continuously across multiple tasks. We observe that the entire process of completing the **Minecart** task takes only 9112 steps including the previous 9 tasks, compared to the 6000 steps needed in a fresh environment. This indicates that only approximately 200 new steps were required. The number of new task completion steps decreases non-linearly even as task complexity grows:

- Basic crafting (Wooden Door): 3000 steps
- Mid-tier crafting (Furnace): 4879 steps
- Advanced crafting (Iron Pickaxe): 8598 steps

The system maintains a 100% success rate across all tasks, indicating robust skill transfer and knowledge utilization from the growing experience database, which expands from 26 entries for Wooden Door to 355 entries for Minecart (see Figure 6 and Appendix Table 1).

6 Related Works

6.1 Embodied Planning & Classical Methods

Early approaches used hierarchical reinforcement learning (Sutton et al., 1999) and symbolic planning (Kaelbling & Lozano-Perez, 2011) but struggled with scalability in open-world domains like Minecraft. Hybrid methods like PDDLStream (Garrett et al., 2020) combined symbolic planning with procedural samplers, while DreamerV3 Hafner et al. (2024) employed latent world models. However, these methods depend on rigid priors, lack causal reasoning, and fail to recover from irreversible errors. Reinforcement learning frameworks (e.g., DQN (Mnih et al., 2015), PPO (Schulman et al., 2017)) and LLM-RL hybrids like Eureka (Ma et al., 2023) also falter in dynamic, long-horizon tasks due to static reward mechanisms and error propagation.

6.2 Zero-Shot LLM Planners

DEPS (Wang et al., 2024c) pioneered zero-shot LLM planning through iterative verbal feedback, enabling dynamic plan refinement. Subsequent works like Voyager (Wang et al., 2023) (skill libraries), ProgPrompt (Singh et al., 2023) (code generation), and Reflexion (Shinn et al., 2023) (feedback loops) advance LLM-based planning but share critical flaws. Namely, they suffer from brittle execution due to dependency on hardcoded assumptions (e.g., ProgPrompt's code templates), opaque memory due to non-interpretable representations (e.g., Voyager's code snippets, PaLM-E's latent vectors (Driess et al., 2023)), and the inability to learn from failed task executions (e.g., Inner Monologue (Huang et al., 2022b) lacks persistent memory).

6.3 Memory-Based Planners

Recent memory-augmented systems like E^2CL (Wang et al., 2024a), ExpeL (Zhao et al., 2024), and AdaPlanner (Sun et al., 2023) store experiences but face key limitations. Namely, they suffer from shallow reasoning capabilities due to lack of environmental context (ExpeL) or causal analysis (ReAct (Yao et al., 2023)), especially of failure modes (Voyager). Above all, these systems are often only evaluated on narrow, controlled-environment benchmarks (e.g., ALFRED), not open-world tasks.

6.4 Mental Models in AI

While cognitive-inspired architectures like predictive coding (Rao & Ballard, 1999) and world models (Ha & Schmidhuber, 2018) encode environmental dynamics, they rely on latent vectors (PIGLeT (Zellers et al., 2021)) or symbolic logic (RAP (Hao et al., 2023)), sacrificing interpretability and adaptability. Neuro-symbolic methods (Garcez & Lamb, 2023) and tree-search frameworks (LATS (Zhou et al., 2024)) further struggle with scalability and causal reasoning.

7 Conclusion

In this paper we presented MINDSTORES, an experience-augmented planning framework that enables embodied agents to build and leverage mental models through natural interaction with their environment. Our approach extends zero-shot LLM planning by maintaining a database of natural language experiences that inform future planning iterations. Through extensive experiments in MineDojo, MINDSTORES demonstrates significant improvements over baseline approaches, particularly in intermediate-complexity tasks, while maintaining the flexibility of zero-shot approaches. The success of our "artificial mental model" approach, which represents experiences as retrievable natural language tuples and enables LLMs to reason over past experiences, demonstrates that incorporating principles from human cognition can substantially improve complex reasoning and experiential learning capabilities in AI systems.

However, several limitations remain. Performance degrades significantly for advanced tasks, and computational overhead scales with database size. Future work should explore more sophisticated experience pruning mechanisms, hierarchical memory architectures for managing larger experience databases, and improved methods for transferring insights across related tasks. Additionally, investigating ways to combine our experience-based approach with traditional reinforcement learning could help address the challenge of long-horizon planning in complex environments.

REFERENCES

Vineet Bhat, Ali Umut Kaypak, Prashanth Krishnamurthy, Ramesh Karri, and Farshad Khorrami. Grounding llms for robot task planning using closed-loop state feedback, 2024. URL https://arxiv.org/abs/2402.08546.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Kevin R Canini, Mikhail M Shashkov, and Thomas L Griffiths. Modeling Transfer Learning in Human Categorization with the Hierarchical Dirichlet Process.

Kenneth James Williams Craik. *The nature of explanation*. Cambridge : University Press, 1952. URL http://archive.org/details/natureofexplanat0000crai.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An Embodied Multimodal Language Model, March 2023. URL http://arxiv.org/abs/2303.03378. arXiv:2303.03378 [cs].

G. Dulac-Arnold, N. Levine, D.J. Mankowitz, et al. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110:2419–2468, 2021. doi: 10.1007/s10994-021-05961-4. URL https://doi.org/10.1007/s10994-021-05961-4.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, December 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/74a67268c5cc5910f64938cac4526a90-Abstract-Datasets_and_Benchmarks.html.

Artur d'Avila Garcez and Luís C. Lamb. Neurosymbolic AI: the 3rd wave. *Artif. Intell. Rev.*, 56 (11):12387–12406, March 2023. ISSN 0269-2821. doi: 10.1007/s10462-023-10448-w. URL https://doi.org/10.1007/s10462-023-10448-w.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 30:440–448, June 2020. ISSN 2334-0843. doi: 10.1609/icaps.v30i1.6739. URL https://ojs.aaai.org/index.php/ICAPS/article/view/6739.

David Ha and Jürgen Schmidhuber. World Models. March 2018. doi: 10.5281/zenodo.1207631. URL http://arxiv.org/abs/1803.10122. arXiv:1803.10122 [cs].

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

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with Language Model is Planning with World Model. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.507. URL https://aclanthology.org/2023.emnlp-main.507/.

Jie Huang and Kevin Chen-Chuan Chang. Towards Reasoning in Large Language Models: A Survey. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 1049–1065, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.67. URL https://aclanthology.org/2023.findings-acl.67/.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents, March 2022a. URL http://arxiv.org/abs/2201.07207. arXiv:2201.07207 [cs].

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied Reasoning through Planning with Language Models, July 2022b. URL http://arxiv.org/abs/2207.05608. arXiv:2207.05608 [cs].

Zhijian Huang, Tao Tang, Shaoxiang Chen, Sihao Lin, Zequn Jie, Lin Ma, Guangrun Wang, and Xiaodan Liang. Making Large Language Models Better Planners with Reasoning-Decision Alignment, August 2024. URL http://arxiv.org/abs/2408.13890. arXiv:2408.13890 [cs].

Dominik Jeurissen, Diego Perez-Liebana, Jeremy Gow, Duygu Cakmak, and James Kwan. Playing nethack with llms: Potential & limitations as zero-shot agents, 2024. URL https://arxiv.org/abs/2403.00690.

Leslie Pack Kaelbling and Tomas Lozano-Perez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pp. 1470–1477, Shanghai, China, May 2011. IEEE. ISBN 978-1-61284-386-5. doi: 10.1109/ICRA.2011.5980391. URL http://ieeexplore.ieee.org/document/5980391/.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236. URL https://www.nature.com/articles/nature14236. Publisher: Nature Publishing Group.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. Reasoning with Large Language Models, a Survey, July 2024. URL http://arxiv.org/abs/2407.11511. arXiv:2407.11511 [cs].

Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, January 1999. ISSN 1546-1726. doi: 10.1038/4580. URL https://www.nature.com/articles/nn0199_79. Publisher: Nature Publishing Group.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. URL http://arxiv.org/abs/1707.06347. arXiv:1707.06347 [cs].

Bilgehan Sel, Ruoxi Jia, and Ming Jin. LLMs Can Plan Only If We Tell Them, January 2025. URL http://arxiv.org/abs/2501.13545. arXiv:2501.13545 [cs].

Yide Shentu, Philipp Wu, Aravind Rajeswaran, and Pieter Abbeel. From llms to actions: Latent codes as bridges in hierarchical robot control, 2024. URL https://arxiv.org/abs/2405.04798.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, December 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. AdaPlanner: Adaptive Planning from Feedback with Language Models. *Advances in Neural Information Processing Systems*, 36:58202–58245, December 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/b5c8c1c117618267944b2617add0a766-Abstract-Conference.html.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, August 1999. ISSN 0004-3702. doi: 10.1016/S0004-3702(99)00052-1. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models, October 2023. URL http://arxiv.org/abs/2305.16291. arXiv:2305.16291 [cs].

Hanlin Wang, Chak Tou Leong, Jian Wang, and Wenjie Li. E^2CL: Exploration-based Error Correction Learning for Embodied Agents. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7626–7639, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.448. URL https://aclanthology.org/2024.findings-emnlp.448/.

Jiaqi Wang, Enze Shi, Huawen Hu, Chong Ma, Yiheng Liu, Xuhui Wang, Yincheng Yao, Xuan Liu, Bao Ge, and Shu Zhang. Large language models for robotics: Opportunities, challenges, and perspectives. *Journal of Automation and Intelligence*, 2024b. ISSN 2949-8554. doi: https://doi.org/10.1016/j.jai.2024.12.003. URL https://www.sciencedirect.com/science/article/pii/S2949855424000613.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents, July 2024c. URL http://arxiv.org/abs/2302.01560. arXiv:2302.01560 [cs].

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, March 2023. URL http://arxiv.org/abs/2210.03629. arXiv:2210.03629 [cs].

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning, 2024. URL https://arxiv.org/abs/2407.08693.

Rowan Zellers, Ari Holtzman, Matthew Peters, Roozbeh Mottaghi, Aniruddha Kembhavi, Ali Farhadi, and Yejin Choi. PIGLeT: Language Grounding Through Neuro-Symbolic Interaction in a 3D World. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2040–2050, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.159. URL https://aclanthology.org/2021.acl-long.159/.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM Agents Are Experiential Learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, March 2024. ISSN 2374-3468. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936. Number: 17.

Wenqing Zheng, S. P. Sharan, Zhiwen Fan, Kevin Wang, Yihan Xi, and Zhangyang Wang. Symbolic visual reinforcement learning: A scalable framework with object-level abstraction and differentiable expression search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(1):400–412, 2025. doi: 10.1109/TPAMI.2024.3469053.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language Agent Tree Search Unifies Reasoning Acting and Planning in Language Models, June 2024. URL http://arxiv.org/abs/2310.04406. arXiv:2310.04406 [cs].

APPENDIX A: AGENT ALGORITHM AND LLM PROMPTS

A.1 AGENT ALGORITHM

```
def run_agent(
    environment,    # MineDojo environment
    max_steps=1000, # Maximum steps to run
    goal_input=""   # Optional high-level goal
):
    # Initialize metrics and experience tracking
    metrics_logger = MetricsLogger()
    experience_store = ExperienceStore()

    # Initial environment reset
    obs, _, _, info = environment.step(environment.action_space.no_op())

    step = 0
    while step < max_steps:
        # 1. Create structured state description
        state_json = get_state_description(obs, info)

        # 2. Get next immediate task
        sub_task = get_next_immediate_task(state_json)
        metrics_logger.start_subtask()

        # 3. Plan action sequence
        actions = plan_action(state_json, info["inventory"], sub_task)

        # 4. Execute actions and track experience
        obs, reward, done, info = execute_action_sequence(actions)

        # 5. Store experience and update metrics
        if done:
            store_experience(state_json, reward, done)
            break

        step += len(actions)

    environment.close()
    metrics_logger.print_summary()
```

A.2 LLM PROMPTS

A.2.1 Environment Description Prompt

```
You are an expert Minecraft observer. Describe the current environment state focusing on:
1. The agent's immediate surroundings (blocks, entities, tools)
2. Environmental conditions (weather, light, temperature)
3. Agent's physical state (health, food, equipment)
4. Notable resources or dangers

Current state:
${state_json_str}

Provide a clear, concise description that would be useful for planning actions.
```

A.2.2 Situation Analysis Prompt

```
You are an expert Minecraft strategist. Given the current state and environment description:
1. Analyze available resources and their potential uses
2. Identify immediate opportunities or threats
3. Consider crafting possibilities based on inventory
4. Evaluate progress towards goals

Environment description:
${description}

Current state:
${state_json_str}

Provide strategic insights about the current situation.
```

A.2.3 Strategy Planning Prompt

```
You are an expert Minecraft planner. Create a strategic plan considering:
1. The current goal: ${goal}
2. Available resources and tools
3. Environmental conditions
4. Potential obstacles or requirements
5. Do not assume intermediate tasks can be achieved without running another agent loop
6. Specify quantities and required actions

Environment description:
${description}

Situation analysis:
${explanation}

Current state:
${state_json_str}

Create a specific, actionable plan that moves towards the goal.
```

A.2.4 Action Selection Prompt

```
You are an expert Minecraft action selector. Convert the plan into specific actions:
1. Use only valid Minecraft actions (move_forward, move_backward, jump, craft, etc.)
2. Consider the current state and available resources
3. Break down complex tasks into simple action sequences
4. Ensure actions are feasible given agent capabilities
5. Make actions incremental and build progressively

Available actions:
- forward [N]: Move forward N steps (default 1)
- backward [N]: Move backward N steps (default 1)
- move_left
- move_right
- jump
- sneak
- sprint
- attack [N]
- use
- drop
- craft
- equip [item]
```

```
- place [block]
- destroy
- look_horizontal +/-X
- no_op

Strategic plan:
${plan}

Current state:
${state_json_str}

Return ONLY a list of actions, one per line, that can be directly
↪  executed.
```

A.2.5 Outcome Evaluation Prompt

```
Evaluate the outcome of a Minecraft action sequence in brief.

Initial state (JSON): ${initial_state}
Final state (JSON): ${final_state}
Reward: ${reward}
Done: ${done}
GPT Plan: ${gpt_plan}
Executed Actions: ${executed_actions}

Format response as: outcome|success|explanation
```

APPENDIX B: IMPLEMENTATION DETAILS

B.1 CORE COMPONENTS

Our implementation leverages:

- MineDojo environment for Minecraft interaction
- OpenAI GPT-4 API for planning and reasoning
- SBERT for semantic embeddings
- FAISS for efficient similarity search
- Custom logging system for experiment tracking

The codebase is structured into modules for state processing, experience management, action planning, metrics collection, and environment interaction.

B.2 ENVIRONMENT INTEGRATION

```
env = minedojo.make(
    task_id="survival",
    image_size=(480, 768),
    seed=40,
    initial_inventory=[
        InventoryItem(slot=0, name="wooden_axe", quantity=1),
    ]
)
```

The action space includes movement (forward, backward, left, right, jump, sneak, sprint), interaction (attack, use, drop, craft, equip, place, destroy), camera control (look_horizontal, look_vertical), and special (no_op).

B.3 Neural Components

Embedding configuration:

- Model: SBERT 'all-MiniLM-L6-v2'
- Output dimension: 768
- Normalization: L2
- Distance metric: cosine similarity

FAISS index parameters:

- Index type: IndexFlatL2
- Dimension: 768
- Metric: L2 distance

Appendix C: Additional Tables

Table 1: Task Details

Meta	Name	Number	Example	Steps	Given Tool
MT1	Basic	14	Make a wooden door	3000	Axe
MT2	Tool	12	Make a stone pickaxe	3000	Axe
MT3	Hunt and Food	7	Cook the beef	6000	Axe
MT4	Dig-down	6	Mine Coal	6000	Axe
MT5	Equipment	9	Equip the leather helmet	3000	Axe
MT6	Tool (Complex)	7	Make shears and bucket	6000	Axe
MT7	IronStage	13	Obtain an iron	6000	Axe
MT8	Challenge	1	Obtain a diamond!	12000	Axe

Table 2: Task Details with MINDSTORES and DEPS Percentages

Category	Task Name	MINDSTORES (%)	DEPS (%)
MT1	Wooden Door	83.3	66.7
MT1	Stick	90.0	83.7
MT1	Wooden Slab	83.3	73.7
MT1	Planks	80.0	73.3
MT1	Fence	80.0	66.7
MT1	Sign	86.7	73.3
MT1	Trapdoor	80.0	56.7
MT2	Furnace	70.0	56.67
MT2	Crafting Table	93.3	83.3
MT2	Wooden Axe	96.7	96.7
MT2	Wooden Sword	90.0	86.7
MT2	Wooden Hoe	86.7	86.7
MT2	Stone Pickaxe	76.7	73.3
MT2	Stone Sword	83.3	80.0
MT2	Stone Shovel	70.0	66.7
MT2	Wooden Shovel	86.7	63.3
MT3	Cooked Beef	60.0	43.3
MT3	Bed	50.0	43.3
MT3	Item Frame	86.7	83.3
MT3	Cooked beef	76.7	63.3
MT3	Cooked Mutton	73.3	66.7
MT3	Painting	96.7	76.67
MT3	Cooked Porkchop	53.3	43.3
MT4	Torch	13.3	3.3
MT4	Cobblestone wall	66.7	53.3
MT4	Lever	86.7	73.3
MT4	Coal	23.3	10.0
MT4	Stone Slab	70.0	53.33
MT4	Stone Stairs	73.3	63.33
MT5	Iron Boots	27.0	16.67
MT5	Iron Helmet	10.0	0.0
MT5	Shield	23.3	13.3
MT5	Iron Chestplate	10.0	0.0
MT5	Leather boots	63.3	60.0
MT5	Iron leggings	3.3	3.3
MT5	Leather Helmet	66.7	46.67
MT6	Iron pickaxe	6.67	0.0
MT6	Bucket	13.3	6.7
MT6	Iron Sword	23.3	6.7
MT6	Iron Hoe	23.3	13.3
MT6	Iron Axe	23.3	6.67
MT6	Shears	33.3	16.67
MT7	Minecart	13.3	0.0
MT7	Iron Nugget	36.7	20.0
MT7	Furnace Minecart	6.7	3.3
MT7	Rail	13.3	6.7
MT7	Cauldron	10.0	3.3
MT7	Iron Bars	13.3	6.7
MT7	Iron Door	13.3	3.3
MT7	Tripwire Hook	43.3	20.0
MT7	Iron trap door	16.7	3.3
MT7	Hopper	6.7	0.0
MT8	Diamond	0.0	0.0

Task	MINDSTORES	Voyager	Reflexion
Mine Wood	10	12	9
Mine Cobblestone	34	42	39
Mine Coal	54	85	106
Make Furnace	89	147	198
Make Stone Sword	187	263	500
Mine Iron	276	500	500

Table 3: Time steps required to complete different Minecraft tasks across three systems. (Values for Voyager and Reflexion are capped at 500 in some tasks.)

Task	MINDSTORES (Predicted)	DEPS (No Prediction)
MT1	83.3%	70.6%
MT2	83.7%	77.0%
MT3	71.0%	60.0%
MT4	55.6%	42.8%
MT5	29.1%	20.0%
MT6	20.5%	8.3%
MT7	17.3%	6.7%
MT8	0.0%	0.0%

Table 4: Success rate comparison with outcome prediction (MINDSTORES) vs. without (DEPS).