Reinforcement Learning Project Report

Aman Paliwal, Arbaaz Shafiq, Liza Wahi, Sri Ranga Deep Yarlagadda

Plaksha University, India
aman.paliwal@plaksha.edu.in, arbaaz.shafiq@plaksha.edu.in, liza.wahi@plaksha.edu.in, sri.yarlagadda@plaksha.edu.in

I. Problem Statement

Baba Is You is an award-winning puzzle game in which players manipulate rules in the game (e.g., "BABA IS YOU," "WALL IS STOP") by rearranging text blocks to alter the mechanics of the level and achieve goals. Unlike traditional games with static rules, Baba Is You requires players to dynamically reinterpret the environment, making it a unique testbed for AI agents. For example, if the rules are "BABA IS YOU, FLAG IS WIN, SKULL IS DEFEAT.", then the agent should learn to move the BABA object to the FLAG object and avoid the SKULL object. Now, if the rules change to "BABA IS YOU, FLAG IS LOSE, SKULL IS WIN", the previously learned policy would result in defeat, even though the game states look almost identical. The policy must be based on the roles of the objects rather than the objects themselves.

This project focuses on developing reinforcement learning agents capable of solving a range of Baba Is You levels. These levels, require agents to not only interpret complex rule sets but also navigate intricate environments and perform multi-step interactions like pushing boxes onto specific target locations while respecting object properties like STOP or SINK. While the simplest cases might involve navigating object A to target B while avoiding C based on rules like "A IS YOU", "B IS WIN", "C IS DEFEAT", many levels necessitate more sophisticated planning and handling of object physics.

Even without the added complexity of manipulating the rule blocks themselves, fixed-rule levels pose significant difficulties for current reinforcement learning (RL) methods. The challenges include:

- Combinatorial State Space: The interaction of objects and rules creates a vast state space.
- Sparse Rewards: Meaningful rewards (like placing a box or winning) are often infrequent and require long-term planning.
- Symbolic Reasoning: Standard RL architectures (like CNNs) struggle to inherently parse and reason over the symbolic, predicate logic-like nature of the game's rules.
- Generalization: Agents trained on one set of rules often fail to generalize to new levels with different, even slightly modified, rule combinations.

Existing RL approaches for Baba Is You often simplify the problem, struggle to generalize across diverse rule sets, or lack the scalability for complex interactions. Although the ultimate goal for a complete Baba Is You agent involves mastering rule manipulation, a critical and challenging prerequisite is developing agents that can robustly plan, reason, and generalize within the environments defined by diverse fixed rule sets. The limitations of current methods highlight a critical gap: the need for frameworks that effectively integrate perception, symbolic rule interpretation, planning, and generalization, which this work aims to address for fixed-rule scenarios.

II. Literature Review

The development of artificial intelligence capable of solving complex puzzle games has been a long-standing challenge. Games like Baba Is You, with its unique meta-puzzle mechanic of rule modification via text manipulation, present particularly difficult problems that push the boundaries of traditional planning and reinforcement learning (RL) approaches. The dynamic nature of the rule space requires agents not only to navigate a grid environment but also to reason about and manipulate the underlying game physics and objectives. Early exploration into AI for games with such dynamic mechanics was notably addressed in the Keke AI Competition (1). This competition specifically focused on agents solving puzzle levels within a dynamically changing mechanic space, closely mirroring the core challenge of Baba Is You. Resources and frameworks related to this competition are publicly available (2), providing a foundation for subsequent research. Several distinct approaches have been applied directly to

Baba Is You. Automated solvers, often leveraging sophisticated search algorithms, have been developed to find solutions to levels, demonstrating the feasibility of tackling the game through planning perspectives (3). Concurrently, Reinforcement Learning has been explored as a method for training agents to play the game, with introductory materials and frameworks being shared within the research community (4).

The inherent linguistic nature of the rule system in Baba Is You (“BABA IS YOU”, “WALL IS STOP”, etc.) also connects this research to broader work on state representation in AI. An overview of natural language state representations for RL is provided in (5), exploring how linguistic information can be integrated into agent learning, a concept highly relevant for interpreting and manipulating the text-based rules in Baba Is You. Furthermore, research on related grid-based puzzle games with complex planning requirements, such as Sokoban, offers valuable insights and potentially transferable techniques. Recent studies have investigated the use of recurrent neural networks capable of internal planning (6) and hierarchical reinforcement learning using landmarks to decompose the problem into manageable subgoals (7). These advancements in tackling combinatorial search and long-horizon planning in Sokoban may inform future approaches for developing more sophisticated AI agents for the even more complex, rule-shifting environment of Baba Is You. Collectively, this literature highlights the multifaceted challenge posed by Baba Is You and illustrates the diverse range of AI techniques—from dedicated competitions (1) and search-based solvers (3) to various RL paradigms (4) and novel rule-exploiting strategies (8)—being explored to master its unique gameplay.

III. RL Formulation

A. State Space

The state space S consists of:

1. **Grid Configuration**: An $n \times m$ grid where each cell contains game objects represented as one-hot encoded tensors. If there are k possible object types, each cell is represented by a k-dimensional vector.
2. **Rule State**: A representation of the current active rules in the form $(subject, relation, property)$ tuples. For example, “Baba is You” would be encoded as (BABA, IS, YOU). All the objects (BABA, IS, SKULL etc.) have a corresponding integer representation.

Formally, the state $s_t \in S$ at time t is represented as:

- Grid tensor $G_t \in \mathbb{R}^{n \times m \times k}$ (spatial configuration)
- Rules tensor $R_t \in \mathbb{N}^{r \times 3}$ representing active rules. r is the maximum number of rules.
- Aditionally, there is a rule mask tensor, $M_t \in \{0, 1\}^r$. It indicates valid (1) versus padded (0) rules.

B. Action Space

The action space A is discrete and consists of four directional movements:

$$A = \{\text{up}, \text{down}, \text{left}, \text{right}\} \tag{1}$$

This simple action space governs both navigation and rule manipulation, as moving the player character against objects can push them, potentially creating or breaking rules.

C. Transition Function

The transition function $P(s_{t+1}|s_t, a_t)$ determines the next state given the current state and action. This function is deterministic in Baba Is You. The original game has a lot of rules which would be too complex for us to consider, so we have restricted ourselves to the following rules:

1. If the action leads to movement into an empty space, the “YOU” object moves.
2. If an object tries to move into a “STOP” object, the movement fails.
3. If something tries to move into a “PUSH” object, it will try to move in the same direction to make room. It will act like a “STOP” object if it fails.
   (a) This includes other “PUSH” objects.

   (b) All TEXT objects are “PUSH” by default.
   (c) “PUSH” ignores “STOP”.
4. When an object moves into a block with the “SINK” property, the object and the block that had the “SINK” property both disapear.
5. If a “DEFEAT”-object intersects with or is a “YOU”-object, the “YOU”-object will be destroyed.
6. If rules are reconfigured, the game mechanics update accordingly.

D. *Reward Function*

The reward function $R(s_t, a_t, s_{t+1})$ is structured as follows:

1. **Win reward**: $+5$ when the “you” object reaches a “win” object or when a winning condition is satisfied
2. **Step penalty**: -0.01 per action to encourage efficiency
3. **Loss penalty**: -1 when the “you” object contacts a “lose” object or when loss conditions are met

While sparse rewards, providing feedback only upon winning or losing an episode, present a significant learning challenge due to the delayed signal, they possess the critical advantage of inherent generality. This structure directly reflects the true objective of the game – achieving the “WIN” state – without encoding potentially misleading human biases about how that state should be reached, thereby allowing for the discovery of novel solutions.

We experimented with denser reward structures with the aim of acclerating learning but we encountered significant hurdles. Level-specific rewards, such as incentivizing the formation of certain rules for specific levels, proved brittle and failed to generalize to other puzzles. Conversely, more general intermediate rewards, like providing positive feedback for any rule change or for specific interactions like sinking water with a rock, often led to reward hacking; the agent learned to exploit these proxy signals (e.g., repeatedly forming and breaking simple rules or unnecessarily sinking objects) rather than focusing on the ultimate goal of solving the level. Consequently, finding robust solutions often necessitates architectures and exploration strategies capable of tackling the fundamental sparsity of the original win/loss condition.

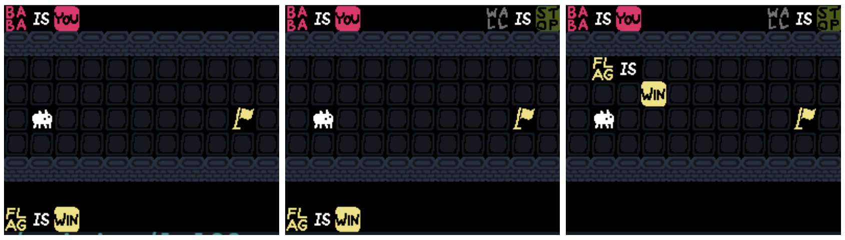
IV. Methodology

A. *DQN*

The DQN algorithm implemented here uses a CNN to approximate the Q-function for the Baba Is You environment. The network consists of several convolutional layers followed by batch normalization and a fully connected layer to output action values. Experience replay is employed to stabilize training by storing past experiences in a `ReplayMemory` buffer and sampling randomly from it during updates. The agent explores the environment using an epsilon-greedy strategy, gradually decreasing exploration over time to prioritize exploitation. The training process minimizes the mean squared error between predicted Q-values and the TD-target using smooth L1 loss. (3)

B. *Curriculum Learning*

We first replicate and test existing DQN and REINFORCE models, implemented for level specific training. We also test an attention based PPO model on the same levels. Better performance on single instances motivates training on multiple levels. All models generalise poorly with built in game levels, we attribute this to the steep learning curve of the game and rapid addition of new objects and rules at each stage. To counter this, we develop new training and testing levels constructed with gradual difficulty increase. Each stage adds one new object or rule at a time, or modifies an existing rule in a slight manner, encouraging gradual learning (See Figure 1). These levels are then arranged and passed to each model before testing, forming a curriculum learning approach. Our results further show that this approach is well suited for the task as demonstrated by the performance of the PPO model.



(a) Level 1 **(b)** Level 2 **(c)** Level 3

Figure 1: A Sample Curriculum of 3 levels

C. *PPO*

We utilize the Proximal Policy Optimization (PPO) algorithm for training our agent (4). The agent employs an Actor-Critic architecture designed to process the game's unique observation space, consisting of both the grid layout and the active symbolic rules. The model features two parallel input streams:

- A Convolutional Neural Network (CNN) encodes the spatial grid information.
- An Embedding layer followed by a Transformer Encoder processes the variable-length list of active rules, attending to the provided rule mask to handle padding.

Features from both streams are fused through linear layers before splitting into two output heads: an Actor head producing action logits and a Critic head predicting the state value.

Training proceeds via standard PPO. The agent collects trajectories over a fixed number of environment steps (NUMSTEPS). Generalized Advantage Estimation (GAE) with parameters (GAMMA) and (LAMBDA) is used to calculate advantage targets. The agent's policy and value functions are updated over multiple epochs using mini-batches. The update uses the characteristic PPO clipped surrogate objective for the policy loss, a mean squared error loss for the value function (scaled by VALUELOSS-COEF), and includes an entropy bonus (scaled by ENTROPYCOEF) to encourage exploration. We use the Adam optimizer with learning rate annealing and gradient clipping for stable training.

This PPO process is embedded within the curriculum learning strategy, allowing the agent to learn progressively complex levels. We also experiment with introducing forced NOOP actions (–thinking-steps) at the episode start to investigate the impact of additional decision time, inspired by studies on recurrent architectures (6). However, this performs similar to the usual PPO algorithm and we do not report separate results.

D. *Hierarchical Reinforcement Learning*

Traditional algorithms face significant challenges in the complex and rule-based world of Baba Is You due to its sparse rewards and the requirement for long-term, abstract planning. Standard RL agents often struggle with effective exploration and discovering the multi-step strategies needed to solve puzzles. The combination of Curriculum Learning and Hierarchical Reinforcement Learning offers a candidate architecture by decomposing the decision-making process into multiple levels. A high-level "Manager" policy learns to set abstract goals or strategies, while a low-level "Worker" policy executes primitive actions to achieve those goals.

The project implements a 2 level HRL approach using state-based subgoals. We also experiment with abstract manager actions but find that subgoal based managerial actions work better. The Manager generates a target state that the Worker aims to reach within a specified timeframe. The neural network architecture comprises a shared feature extractor with a CNN for the grid layout and a Transformer-based Rule Encoder for interpreting rules. The Manager outputs a value estimate and a target subgoal state, while the Worker outputs a policy to reach the subgoal and a value estimate for the state-subgoal pair. The training follows a PPO-based procedure, with separate updates for the Worker's actor and critic, and the Manager's critic. Gradients from the Manager's value loss implicitly train the Generator network to produce valuable subgoals.

The training process incorporates curriculum learning, gradually increasing the difficulty based on the agent's win rate. Experience is collected separately for the Manager and Worker, with Generalized Advantage Estimation (GAE) used to compute advantages for both levels. In the PPO update phase, the Worker's actor and critic are updated using the standard objective and value loss function, while the Manager's critic is updated independently. The Manager's Generator network is trained indirectly through the backpropagation of gradients from the Manager's value loss, optimizing it to generate subgoals that lead to high-value states. (7)

V. Contributions

Our project makes some key contributions to reinforcement learning for symbolic, combinatorially complex environments (like Baba is You). As demonstrated by the levels as part of our Priming folder, we've developed a curriculum learning framework that incrementally introduces objects and rules, enabling agents to generalize across increasingly difficult levels. Our adaptations of PPO with attention mechanisms significantly outperformed baseline DQN models. We also experiment with other architectures including HRL and show potential for good performance in these. Through systematic experimentation, we demonstrated the limitations of reward shaping in symbolic tasks, emphasizing the importance of working with sparse, goal-aligned rewards and building suitable models to work with them.

VI. Results

A. DQN

The DQN implementation struggled with generalization and wasn't able to solve some of the harder levels that we designed. Given in Figure 2 is the reward curve for the curriculum learning that we did using the DQN. The curriculum had 28 levels of gradually increasing difficulty. The DQN wasn't able to solve levels beyond level 24, as can be seen by the flattened reward curve.

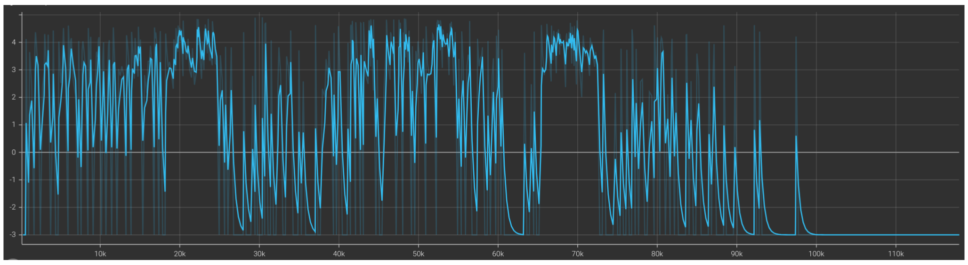
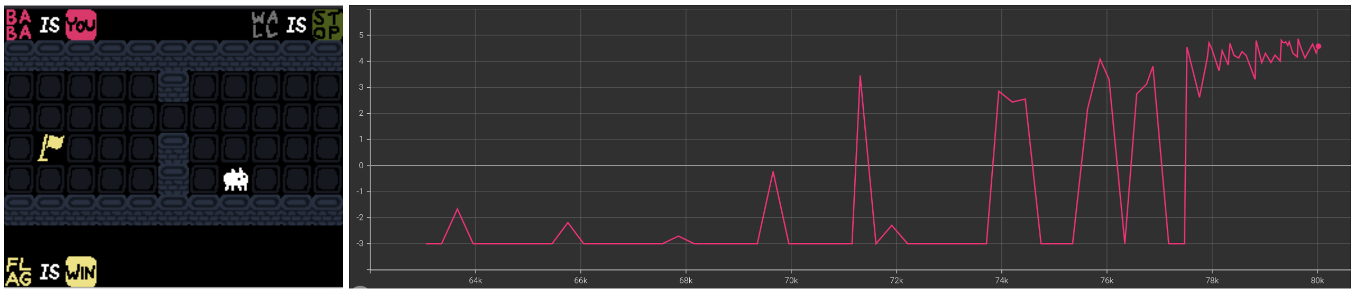


Figure 2: Y-axis: Reward, X-axis: Global Step

B. PPO

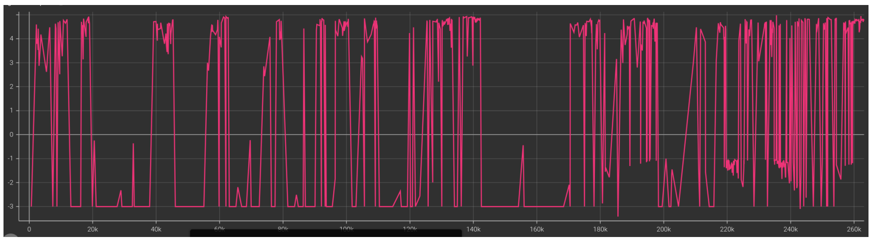
The PPO + Attention implementation worked the best out of all the algorithms that we tried. It was able to learn decently well and was able to solve all 28 levels of the curriculum.

Figure 3 shows the performance of ppo on a sample level. Figure 4 shows the reward curve for curriculum learning.

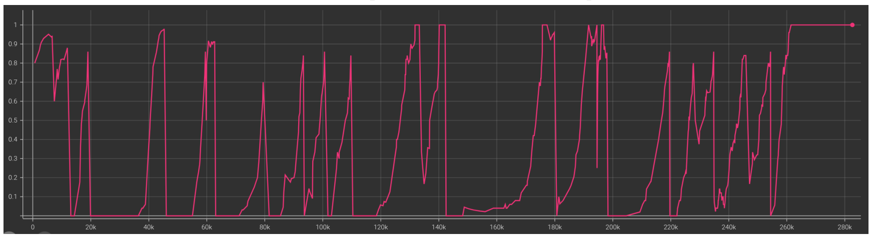


(a) Example Level **(b)** Y-axis: Reward, X-axis: Steps

Figure 3: PPO Performance on a sample level



(a) Y-Axis: Episode Rewards, X-Axis: Steps

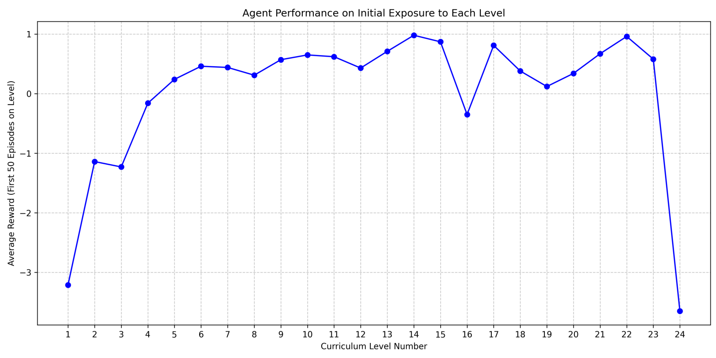


(b) Win Rate (window size = 50)

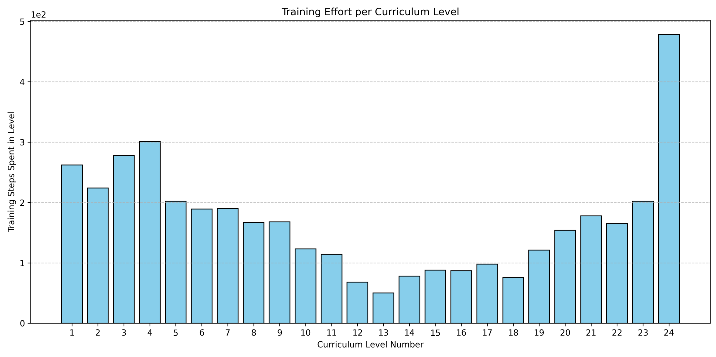
Figure 4: PPO - Curriculum Learning

C. HRL

HRL training curves demonstrate the effectiveness of curriculum learning. Initial levels train over time, slowly learning the appropriate actions and subgoals to generate. Later levels, though gradually harder, learn faster and also accumulate higher rewards over first k episodes. An observation is that the HRL model requires longer training time and steps as compared to the PPO algorithm. This can be attributed to the more complex architecture as well as simultaneous optimization of the Worker and Manager losses and models. Though results do support the idea of curriculum learning and offer promise, the model demands a higher number of levels and training data to solve unseen levels at test time.



(a) Y-Axis: Avg. Rewards Over First 50 Eps , X-Axis: Level



(b) Y-Axis: Training Steps To Advance , X-Axis: Level

Figure 5: HRL : Training Results

VII. GitHub Link

Link to our GitHub repository: `https://github.com/baba-is-rl/baba-is-rl`

References

[1] M. Charity and J. Togelius, "Keke ai competition: Solving puzzle levels in a dynamically changing mechanic space," in *2022 IEEE Conference on Games (CoG)*, (Beijing, China), pp. 570–575, 2022.

[2] MasterMilkX, "Keke competition," 2022. GitHub repository.

[3] utilForever, "Baba is auto," 2023. GitHub repository.

[4] U. Lab, "Baba rl intro," 2023. GitHub repository.

[5] B. Madureira and D. Schlangen, "An overview of natural language state representation for reinforcement learning," *arXiv preprint arXiv:2007.09774*, 2020.

[6] M. Taufeeque, P. Quirke, M. Li, C. Cundy, A. D. Tucker, A. Gleave, and A. Garriga-Alonso, "Planning in a recurrent neural network that plays sokoban," 2024.

[7] S. Pastukhov, "Solving sokoban using hierarchical reinforcement learning with landmarks," 2025.

[8] C. et al., "Baba is ai: Break the rules to beat the benchmark," *arXiv preprint arXiv:2407.13729*, 2024.