An Evaluation of Deep-Q-Learning methods on the Hanabi Challenge

Kival R. Mahadew
221001688@stu.ukzn.ac.za
University of KwaZulu-Natal
Durban, KwaZulu-Natal, South Africa

ABSTRACT

The card game Hanabi presents a complex, cooperative, multi-agent problem that emphasizes communication and strategy within a partially observable environment. The challenges in developing Hanabi playing agents are similar to many real-world applications requiring efficient cooperation, from self-driving cars to virtual assistants. The importance of understanding how agents can effectively cooperate in such environments has become increasingly important. This paper evaluated the performance of deep-Q-learning techniques, focusing specifically on the Rainbow DQN and Simplified Action Decoder agents, in the cooperative multi-agent environment of Hanabi. The performance of the deep-Q-learning agents are compared in the two-player setting of Hanabi, focusing on their sample efficiency and generalization to unseen partners. Our results show that the Rainbow and the Simplified Action Decoder agents outperform the other agents in the two-player self-play version of Hanabi. Our findings highlight the importance of the Rainbow DQN components in achieving high performance in Hanabi and the importance of the Simplified Action Decoder in stabilizing the learning process and improving the performance of the agents. Additionally, we find that independently trained agents struggle to cooperate effectively with agents that they have not been trained with. This study contributes insights into the development of more robust and collaborative MARL systems, offering a foundation for future research in similar multi-agent cooperative domains.

KEYWORDS

Hanabi, Deep Q-Learning, Multi-Agent Systems, Reinforcement Learning

ACM Reference Format:
Kival R. Mahadew. 2024. An Evaluation of Deep-Q-Learning methods on the Hanabi Challenge. In . ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Cooperation is a fundamental aspect of human society. It is the foundation of social structures, economies, and even the evolution of species. Cooperation involves working together in teams to achieve a common goal and sharing resources and information with others. Cooperation is essential for solving complex problems that require the coordination of multiple agents, each with its own goals and objectives. Human beings are naturally adept at cooperating with others, even with challenges such as limited information, communication barriers, and conflicting interests.

Cooperation is a challenging problem in artificial intelligence and has been studied extensively [42]. The ability of agents to cooperate effectively can lead to significant improvements in performance and efficiency. However, cooperation is not always straightforward, especially when agents have limited information about their environment or the other agents they are interacting with. Despite these challenges, significant progress has been made in developing algorithms that enable agents to cooperate in a variety of settings [25, 29].

The game of Hanabi has emerged as a popular benchmark for evaluating the performance of cooperative multi-agent systems. Hanabi is a cooperative card game where players work together to build sets of cards in a specific order. Players have limited information about their own cards and must rely on communication and reasoning to infer their cards from other players' actions. Hanabi's combination of cooperative gameplay, dynamic environments, imperfect information, and reliance on theory of mind reasoning makes it a challenging benchmark for evaluating the performance of computational multi-agents in playing the game [5]. Human players have been shown to outperform AI agents by a large margin; even first-time players can easily understand the game and perform well [37, 2]. This makes Hanabi an interesting benchmark for automated game playing.

A large number of attempts have been made to develop automated players using deep reinforcement learning. Deep reinforcement learning is a sub-field of artificial intelligence that combines deep learning techniques with reinforcement learning to learn complex policies from high-dimensional input data. Deep reinforcement learning algorithms use neural networks to learn a mapping from environmental states to actions. The agent learns to attempt to maximize a reward signal that is provided by the environment. Deep reinforcement learning has been successful in learning to play complex games such as Go [1] and StarCraft 2 [19], control autonomous vehicles [8], and solve a variety of sequential decision-making problems [31].

Deep Multi-Agent Reinforcement Learning (MARL) has shown promise in addressing the challenges of cooperation in multi-agent systems. MARL combines deep learning techniques with reinforcement learning in multi-agent settings to enable agents to learn to cooperate effectively in complex and dynamic environments. Recent advances in MARL have led to significant improvements in the

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

performance of agents in cooperative settings, including Hanabi [27, 16]. However, there are still many open questions and challenges in the field of MARL, particularly in the context of cooperation with new partners, efficient exploration, and sample efficiency [15, 10, 5].

Deep-Q-learning reinforcement learning techniques, specifically, have shown state-of-the-art performance in Hanabi, with the Rainbow DQN and Simplified Action Decoder agents achieving superhuman performance in self-play [25, 5]. However, their performance in Hanabi has not yet been fully explored.

This project aims to evaluate the performance of deep-Q-learning techniques in Hanabi's cooperative multi-agent environment. We focus specifically on the Rainbow DQN and Simplified Action Decoder agents. We compare the performance of these agents to simple DQN agents in the two-player setting of Hanabi, focusing on their sample efficiency, exploration and generalization to unseen partners. Our goal is to identify the key factors that contribute to the success of these algorithms in Hanabi and to provide insights into how they can be further improved.

The paper is structured as follows: Section 2 provides a background on Hanabi and the challenges it presents for learning agents, as well as reinforcement learning and deep-Q-learning techniques. Section 3 provides an overview of literature surrounding Hanabi and cooperative MARL. Section 4 describes the methods used in this study, including the experimental setup, agents, training setup, and evaluation metrics. Section 5 presents the results of the experiments and discusses the performance of the agents. Section 6 provides a conclusion and discusses the implications of the findings for future research in cooperative MARL.

2 BACKGROUND

2.1 Hanabi

The Hanabi Challenge has become a popular benchmark in MARL due to its unique properties as a cooperative, partially observable game where players must deduce each other's intentions to succeed. Unlike many competitive MARL tasks, Hanabi requires long-term strategies and implicit communication, posing unique challenges for traditional and deep reinforcement learning algorithms.

Gameplay. Hanabi is a cooperative card game of two to five players working together to achieve a common goal. It is often described as a form of cooperative solitaire. The game is played with a deck of cards, each with a colour and a number. The game's goal is to play cards in ascending order for each colour, starting from 1 to 5. Players can only see the cards of other players but have to infer the cards in their hands from the actions of other players. Players can give hints to each other about the cards in their hands, but they are limited in the number of hints they can give, they can also play cards but lose a life if they play a card incorrectly, and they can discard cards to gain more hints. The game ends when the deck is empty, or the number of incorrect plays reaches a certain threshold. The players' score is determined by the number of cards played correctly by the end of the game, or 0 if the game ends due to too many incorrect plays.

Challenges. Hanabi was first presented as a challenge for AI research as early as 2015 by Osawa [34] to serve as a benchmark for evaluating the imitation of human intelligence with artificial intelligence. The work showcased strategies that attempted to recognize intent and infer hidden information to achieve high scores in Hanabi. The first rule-based approaches appeared later that year by Cox et al. [13]. Since then, several deep-reinforcement learning algorithms have been developed to play Hanabi, achieving superhuman performance [25, 11, 16]. Hanabi presents several challenges for MARL algorithms. The game is partially observable, meaning that players can only see the cards of other players and have to infer the cards in their hands from the actions of other players. This requires agents to develop theory-of-mind reasoning to infer the intentions of other players. The game also requires long-term planning and coordination between players, as the actions of one player affect the actions of other players. Finally, the game is stochastic, as the deck is shuffled at the beginning of each game, and the actions of other players can be unpredictable. Bard et al. [5] identified several key areas to focus on when developing agents for Hanabi. The need for sample efficient deep reinforcement learning algorithms that can learn to coordinate in Hanabi's complex environment, which, despite its relatively small state and action space, is challenging due to the partially observable and cooperative nature of the game. Additionally, the ability of agents to cooperate effectively with themselves, otherwise known as self-play, is also a key area of focus. Finally, the ability of agents to generalize and adapt to new partners is also a key area of focus, as agents need to adapt to different play styles and strategies of other agents in an ad-hoc manner. The Hanabi Learning Environment (HLE) Bard et al. [5] provides a standardized learning environment, the Hanabi, which provides an interface for developing and evaluating Hanabi agents. The HLE provides a simple API for interacting with the game but has since been deprecated in favour of the PettingZoo environment [35], making earlier work using HLE difficult to reproduce.

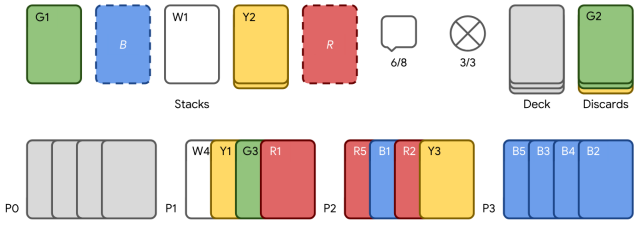

Figure 1: Four player Hanabi gameplay from the perspective of Player 0. The player can see the cards of other players but not their own, and the goal is to play cards in ascending order for each colour. This image is taken from the Hanabi Challenge paper [5]

2.2 Deep Q-Learning

Reinforcement Learning (RL) provides a powerful and scalable framework for modeling decision-making problem as Markov Decision Processes (MDPs). In an MDP, an agent interacts with an environment over a series of discrete time steps, selecting actions to maximize the expected cumulative reward. At each step, the agent observes the state of the environment, selects an action, and receives a reward and the next state. The goal of the agent is to

optimize a policy, a mapping from states to actions, that maximizes the expected cumulative reward. Value functions are used to estimate the expected cumulative reward of taking an action in a given state, helping the agent learn an effective policy.

In many real-world problems, the agents do not have access to the full state of the environment, making the problem partially observable, resulting in a Partially Observable Markov Decision Process (POMDP). In POMDPs, the agent has to infer the state of the environment from the observations, requiring the agent to develop a model of the environment and the other agents to make effective decisions.

In multi-agent settings we can generalize this to a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), where multiple agents interact with each other in a partially observable environment. Here agents recieve limited observations of the environment and the other agents, and independently selects actions based on its local information. The agents aim to learn a joint policy that maximizes the expected cumulative reward, requiring them to coordinate their actions and share information effectively to achieve the common goal. This setup is exemplified in games like Hanabi, where agents operate under limited information and communication constraints.

Deep Q-learning [31] has established itself as a powerful technique in reinforcement learning. Deep Q-Networks (DQN) use deep neural networks to approximate the Q-values of state-action pairs, enabling agents to learn complex policies from high-dimensional input data. The Q values represent the expected cumulative reward of taking an action in a given state, and the agent selects actions that maximize the Q-values to achieve the common goal. Despite originally being introduced for single-agent settings, Deep Q-Learning has been successfully extended to multi-agent settings [22, 25, 5, 11, 21], where multiple agents learn to cooperate effectively to achieve a common goal. Multi-agent deep Q-learning algorithms use centralized training with decentralized execution to learn a joint policy that maximizes the expected cumulative reward. Agents learn to coordinate their actions and share information effectively to achieve the common goal. However, the Hanabi Challenge [5] introduces an additional layer of complexity by requiring cooperative strategies among multiple agents, each with partial observability, limited communication capabilities, and balancing short-term and long-term rewards. The following paragraphs outline important techniques that have been used in solving Hanabi:

Rainbow DQN. Rainbow DQN [24] combines multiple improvements to the Deep Q-Learning algorithm to address sample efficiency, stability, and convergence speed.

Double Q-Learning. Double Deep Q-Learning [23] uses two separate neural networks to estimate the Q values. The target Q-network, used to update the Q-values, is updated less frequently than the online Q-network, which is used to predict and stabilize the learning process.

Prioritized Experience Replay. Prioritized Experience Replay [36] assigns priorities to experiences based on their temporal difference error. The agent samples experiences with higher priorities more frequently, leading to more efficient learning and better performance.

Dueling Network Architecture. A Dueling DQN [41] separates the value function into two streams: the value stream and the advantage stream. This separation enables the agent to learn the value and advantage functions separately, improving the performance of the Deep Q-learning algorithm. The value stream estimates the expected value of being in a given state, while the advantage stream estimates the advantage of taking an action in a given state. The two streams are combined to estimate the Q-values for more effective learning.

Multi-Step Learning. Multi-step learning [3] algorithms use several time steps to update the Q-values instead of only using one. This helps agents learn more effective long-term strategies and improve the efficiency of the sample of the learning process.

Distributional RL (Categorical DQN). Distributional Reinforcement Learning [7] is a framework for modelling the distribution over the expected cumulative reward of the state-action pairs instead of a single value. Categorical DQN (C51) is a specific implementation of Distributional RL that uses a categorical distribution to estimate the Q-values. over a set of discrete bins or "atoms". This approach provides a richer, more nuanced representation of the expected cumulative reward, enabling the agent to learn more complex policies.

Noisy Networks. Noisy Networks [17] add noise to the weights of the neural network to encourage exploration. Noise is sampled from a factorized Gaussian distribution and added to the weights of the neural network during training. This stochasticity helps the agent explore new actions and learn more effective policies and has shown improved performance over traditional exploration strategies such as ϵ -greedy.

Deep Recurrent Q-Networks. DRQNs have shown exceptional performance in learning to play complex games such as Hanabi but are computationally expensive due to the recurrent nature of the network. We avoid using DRQNs in this study due to the computational complexity of network training and the reliance on extremely large amounts of data to learn a policy (30B+ steps) [5, 25, 16].

Simplified Action Decoder (SAD). The Simplified Action Decoder (SAD) [25] is a cooperative multi-agent reinforcement learning technique that aims to assist agents in learning to cooperate effectively in multi-agent environments where implicit communication is required. Reinforcement Learning requires agents to explore the environment to learn effective policies; however, this random exploration can lead to suboptimal policies by making the actions less expressive of the agent's intentions. The SAD technique takes advantage of the centralized training process. It provides agents with both the observed actions and the intended actions of their partners, enabling them to learn more effective policies by providing access to a part of the other policy during training. SAD has established itself as the state-of-the-art technique for Hanabi, achieving superhuman performance in the 2-5 player settings.

3 LITERATURE REVIEW

Since Bard et al. [5] introduced the Hanabi Learning Environment (Hanabi-LE) alongside the Hanabi Challenge, the game has become

a popular testbed for evaluating multi-agent reinforcement learning algorithms. Bard et al. [5] notes that Hanabi is a challenging environment due to its cooperative nature, limited information, and reliance on communication and reasoning.

Previous efforts for solving Hanabi have used rule-based [13, 40] or imitation learning [34] based approaches, which have had middling success. Some notable rule-based approaches include Flawed, IGGI and Piers, as introduced by Walton-Rivers et al. [40]. Flawed is an agent that is designed to deliberately play sub-optimally with players that are unable to adapt to its behaviour [10]. IGGI and Piers follow a similar strategy to each other, where they aim to maximize the expected score of the game by playing the card that is most likely to be playable. The work by Osawa [34] and Walton-Rivers et al. [40] both outline the value of recognizing intent and inferring hidden information to achieve high scores in Hanabi by comparing agents that follow fixed rules to those that adapt their strategies based on the actions of other players. Additionally, it has been noted that applying Monte Carlo Tree Search (MCTS) to Hanabi Agents can improve their performance [5]. Despite the relative success of rule-based approaches, they all follow strategies that are designed by humans, which makes them intuitive and fairly interpretable, but they rely on established conventions and strategies between players, which inherently limits their ability to adapt to new partners.

Deep reinforcement learning has also been applied to Hanabi due to its ability to be modelled as a decentralized partially observable Markov decision process (Dec-POMDP) [5, 33]. The first technique involved applying the Actor-Critic algorithm to Hanabi, which performed well in the two-player version but struggled to learn in larger games [5].

The Rainbow DQN algorithm, introduced by Hessel et al. [24], is a deep reinforcement learning algorithm that combines several improvements to the DQN algorithm, such as prioritized experience replay and distributional reinforcement learning. Rainbow DQN has been shown to achieve state-of-the-art performance in various tasks, including playing Atari games and controlling autonomous vehicles. It has shown impressive performance in the two-player self-play version of Hanabi. However, it struggles to play with an unseen partner[12].

Recent work has shown that Rainbow DQN can be extended to learn a more general policy when playing with rule-based agents by training with a diverse population of agents [10]. The literature notes that better performance could be achieved with a larger population of diverse agents. Interestingly, the vanilla Rainbow DQN algorithm always converges to the same policy through independent training [10, 5].

The current state-of-the-art in Hanabi is the Simplified Action Decoder (SAD) algorithm, introduced by Hu and Foerster [25]. SAD is a deep reinforcement learning augmentation that aims to stabilize the performance of epsilon-greedy exploration. It was developed as an extension of the Bayesian Action decoder(BAD) developed by Foerster et al. [16] and has been shown to achieve superhuman performance in the game when coupled with large-scale RL frameworks like R2D2 [28]. SAD has been shown to outperform other deep reinforcement learning algorithms, such as Rainbow DQN, in the two-player self-play version of Hanabi [25].

The addition of auxiliary learning tasks has been shown to improve agents' performance in Hanabi by optimising the learning process to focus on the better prediction of unobserved states [25].

Despite the success of SAD, it is not designed to handle the AD-Hoc Teamplay challenge. SAD learns to cooperate effectively with itself or with fixed partners, but it struggles to generalize to new partners [27] due to its tendency to learn arbitrary conventions that are specific to that independent train.

Additionally, agents that rely on Theory of Mind models have been developed. Fuchs et al. [18] introduced an extension to the Dec-POMDP framework that allows agents to model the beliefs of other agents by providing nested reasoning over their beliefs and other agents' decision-making processes. This Interactive-POMDP (I-POMDP) is effective in learning to cooperate with other agents in Hanabi, with a particular focus on efficient hinting and communication strategies.

Recent work has since shifted away from better, more efficient Reinforcement Learning (RL) algorithms to focus on algorithms that can learn to generalize and adapt to new partners[26, 27, 30, 38, 43]. These algorithms aim to address the Zero-shot Coordination Problem, where agents need to learn to cooperate with new partners without prior exposure. These algorithms do this by augmenting the self-play training process with additional information and exploiting the symmetries of the MDP, such as Any-Play Intrinsic Augmentation (APIA) [30] and Other-Play [27], or techniques that can change policies during the game, such as On-the-Fly Coordination [43]. These techniques allow the agents to better generalize to new partners.

Other approaches to train agents to have more 'human-like' reasoning abilities have been introduced. K-Level-Reasoning [14] is a method that utilizes Cognitive Hierarchies, a concept from game theory, to train agents that can reason at multiple levels of reasoning. This is done by training agents to be the best response to agents that reason at a lower level of reasoning. This is similar to Off-Belief Learning [26], which also aims to introduce multi-level cognitive reasoning in a controlled manner. These techniques are effective at training agents that can generalize to new partners.

This, however, requires an incredible amount of computational resources, as the agents need diverse training to learn a general policy on top of the pre-existing sample inefficiency of current RL methods. This is a significant limitation of these algorithms, raising the importance of finding sample-efficient algorithms for Hanabi.

4 METHODS

For this study, we train 7 agents in the Hanabi environment using the PettingZoo wrapper for the Hanabi Learning Environment (HLE). We evaluate the agents in a two-player self-play setting and an AD-Hoc Teamplay setting. We train the agents to understand the impact of the various DQN improvements on the agents' performance in the game in terms of sample efficiency, effective exploration, and generalization to unseen partners.

Experimental Setup

Environment. The choice of Hanabi as the testbed for evaluating multi-agent reinforcement learning algorithms is motivated by its complexity and the challenges it presents for learning agents. The

experiments will be conducted using a PettingZoo[35] Wrapper based on the Hanabi Learning Environment (HLE)[20] developed by Bard et al. [5].

PettingZoo provides a standardized interface for multi-agent reinforcement learning environments. Due to Hanabi's complexity, the experiments will focus on a smaller version of the game, allowing for faster training and evaluation of the agents. We used the 'Hanabi-Small' environment by Bard et al. [5], which is a two-player version of Hanabi, where each player has a hand of 2 cards, and the number of colours is reduced to 2. The maximum score in this environment is 10, and the game ends when the deck is empty or the players have made one mistake.

Agents. As a baseline for the study, we consider simple Deep Q-Networks (DQN), with only Double DQNs and Dueling networks, to evaluate the self-play performance of the agents. We compare 3 Rainbow agents with varying history lengths of 1, 3, and 5 steps. We also train a Rainbow agent with ϵ-greedy exploration instead of Noisy Networks to evaluate the value of Noisy Network exploration. Additionally we train 2 Simplified Action Decoder (SAD) agents to evaluate the performance of the SAD algorithm in the Hanabi environment. We will compare the performance and sample efficiency of the agents with the various improvements included in the Rainbow and SAD algorithms.

All learning agents will be trained using the same DQN framework programmed in Pytorch and TorchRL [9], with the same hyperparameters, to ensure a fair comparison. The agents will be trained using the Adam optimizer with a learning rate of 0.0001 and a batch size of 256. The agents will be trained using a replay buffer of size 10000 and a soft target network update frequency 1 with $\tau = 0.005$. The agents will be trained using a linear ϵ-greedy policy(besides Rainbow) from 1.0 to 0.01 over training. For all algorithms, we burn 1000 steps into the replay buffer before starting training. The agents utilize Independent Q-Learning, where each agent maintains its Q-function and policy to learn a joint policy.

Training Setup

This study considers sample efficiency and generalization of agents to unseen partners as the primary evaluation criteria. Agents will be trained in a sample-limited setting, where the number of experiences seen by the agent is limited to 30M steps. This number is decided based on the paper by Bard et al. [5], which used 100M as a limit for the full game. Hanabi-small has a state space of 30 cards, which is 1/3 of the full game, so we will use 1/3 of the training steps.

Evaluation Metrics

The agents' performance will be evaluated based on their ability to score a high score in the game. We will also evaluate the training curves to determine the sample efficiency of the agents. We will compare the agents' performance with the various improvements to the DQN algorithm, such as Noisy Networks, Distributional Deep-Q-Networks, and Multi-Step Learning.

We will also allow the agents to play together in a two-player self-play version of Hanabi to evaluate their performance in a cooperative setting with the other learning agents in an AD-Hoc Teamplay or cross-play(XP) setting.

We evaluated all agents over 1000 episodes and considered the average and standard deviation of the final score as the primary evaluation metric.

5 RESULTS AND DISCUSSION

This section presents the results of the experiments conducted in this study. We evaluate the performance of the learning agents in the two-player self-play version of Hanabi and the AD-Hoc Teamplay setting.

Self-Play Performance

Table 1 shows the performance of the learning agents in the two-player self-play version of Hanabi. The agents are evaluated based on their ability to achieve a high score in the game. The final score is an average of the score achieved over 1000 episodes. The standard deviation of the final score is also included to show the stability of the learning process. See Appendix A for the full results.

It is important to note that the rule-based approaches generally outperform the learning agents when considering sample-limited training, with only large-scale RL frameworks like R2D2+SAD [25] and ACHA [5] outperforming rule-based approaches.

We observe that none of the agents could achieve an average high score in the game. While not a focus of this study, we can attribute this to the sample's limited training setting and the higher complexity of the small version compared to the full game.

The simple Dueling Double DQN fails to learn the game effectively and struggles to learn an optimal policy for the game. With a score of 0.502, we can conclude that the agent cannot play 1 card correctly on average. It is important to note that, at best, the agent can reach a score of 2.19 earlier in the training process, which shows some promise in the agent's ability to learn the game. However, the agent fails to maintain this performance and quickly regresses to a lower score. Without a clear trend in performance, we can conclude that the agent cannot effectively learn a policy for the game. This can be observed in Figure 2.

The Rainbow DQN shows an impressive improvement in performance over the simple DQN agent, and we can identify a general improvement over time in its performance. However, we observe a cycle of regression and correction in the training, which we will discuss later.

When using a 1 step history, the Rainbow agent quickly learns a policy but then struggles to improve over time, likely due to the agent learning a suboptimal policy early in the game that it cannot recover from.

When using a 3 step history, we can identify an almost linear improvement in performance over time, with the agent quickly learning a policy and then steadily improving its performance. It performs slightly worse on average than the 1 step history but with a more stable learning curve. This is likely due to more samples being needed when looking at a longer history, which is a limitation of the sample-limited training setting. The final score of the agent is lower due to the aforementioned exploration regression cycle, but with more training, the agent would likely be able to achieve a higher score.

With a 5-step history, the results are similar to the 1-step history, with the agent quickly learning a policy but then struggling to

Table 1: Performance of learning agents in Self-Play

Agent	Exploration	n Step	Distributional	SAD	Score(std.dev)
Simple DQN	ϵ-Greedy	1	No	No	0.502(0.7)
Rainbow	Noisy Networks	1	Yes	No	3.26(1.07)
Rainbow	Noisy Networks	3	Yes	No	1.01(1.48)
Rainbow	Noisy Networks	5	Yes	No	3.17(1.47)
Distributed	ϵ-Greedy	1	Yes	No	2.78(1.05)
SAD Agent	ϵ-Greedy	1	Yes	Yes	2.69(0.98)
SAD Agent	ϵ-Greedy	3	Yes	Yes	2.64(0.72)

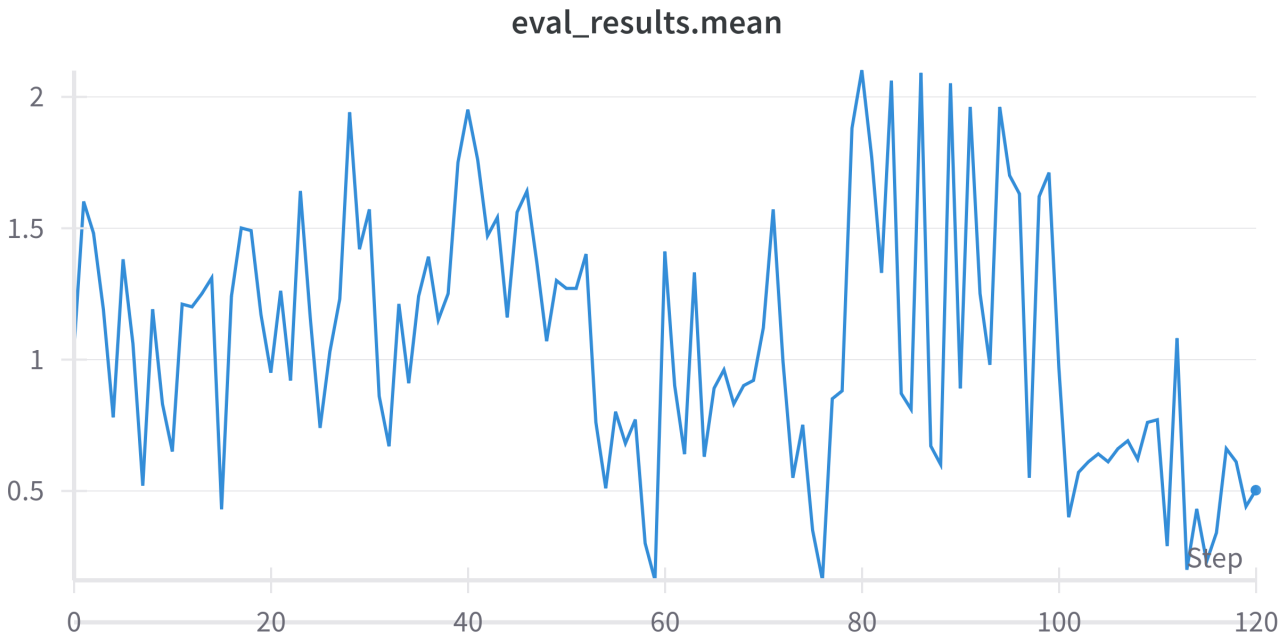

Figure 2: Training curve for Simple DQN Agent

Table 2: Performance of learning agents in Cross-Play

Agent 1\2	Simple DQN	Rainbow	Rainbow-3	Rainbow-5	Distributed	SAD	SAD-3	Avg XP
Simple DQN	-	1.2(0.91)	0.78(0.92)	1.16(0.8)	0.53(0.8)	N/A	N/A	0.92(0.86)
Rainbow	1.1(0.98)	-	1.49(1.6)	2.5(1.4)	1.36(1.36)	N/A	N/A	1.61(1.33)
Rainbow-3	0.91(0.95)	1.56(1.59)	-	1.87(1.4)	1.07(1.3)	N/A	N/A	1.35(1.31)
Rainbow-5	0.96(0.99)	2.39(1.43)	1.6(1.37)	-	1.18(1)	N/A	N/A	1.53(1.2)
Distributed	1.25(1.02)	1.61(1.40)	1.26(1.31)	1.28(1.01)	-	N/A	N/A	1.35(1.19)
SAD	N/A	N/A	N/A	N/A	N/A	-	2.4(1)	2.4(1)
SAD-3	N/A	N/A	N/A	N/A	N/A	2.11(0.8)	-	2.11(0.8)
Avg XP	1.01(0.99)	1.69(1.33)	1.28(1.3)	1.7(1.15)	1.04(1.12)	2.11(0.8)	2.4(1)	

improve over time. The agent receives a slightly lower score than the 1-step history, but due to the sample requirements of the longer history, the agent is likely to perform better with more training.

We compare these Rainbow results with a baseline Rainbow Agent with ϵ-greedy exploration instead of Noisy Networks to evaluate the value of Noisy Network exploration. We find that

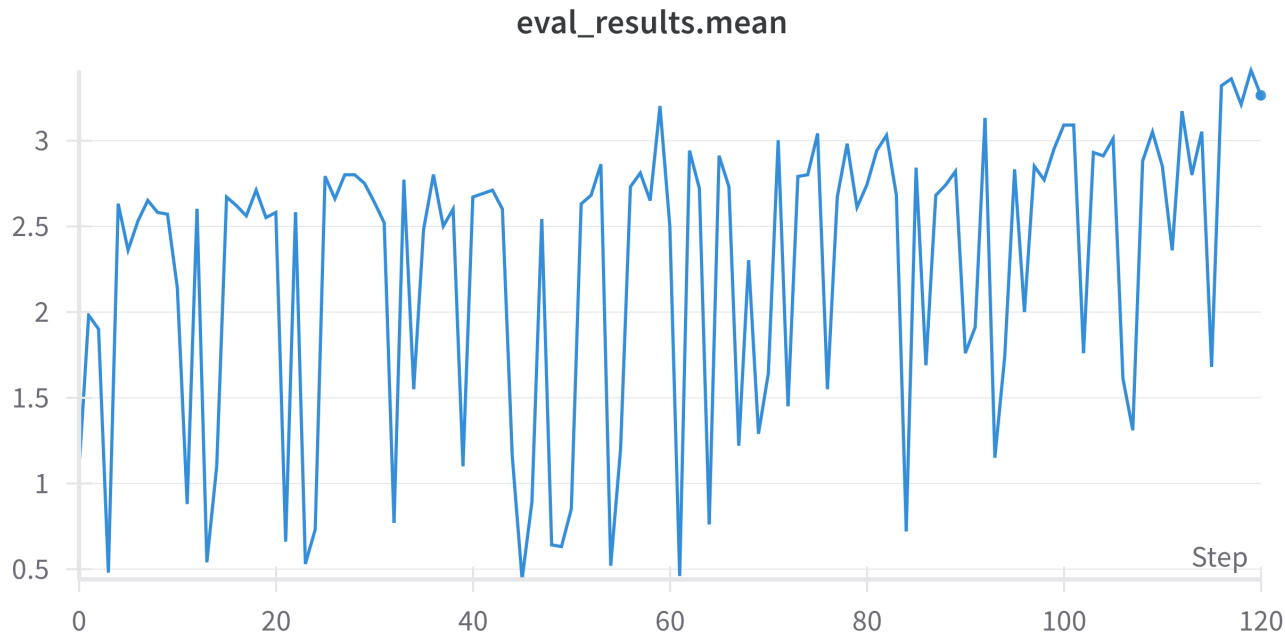

Figure 3: Training curve for Rainbow Agent

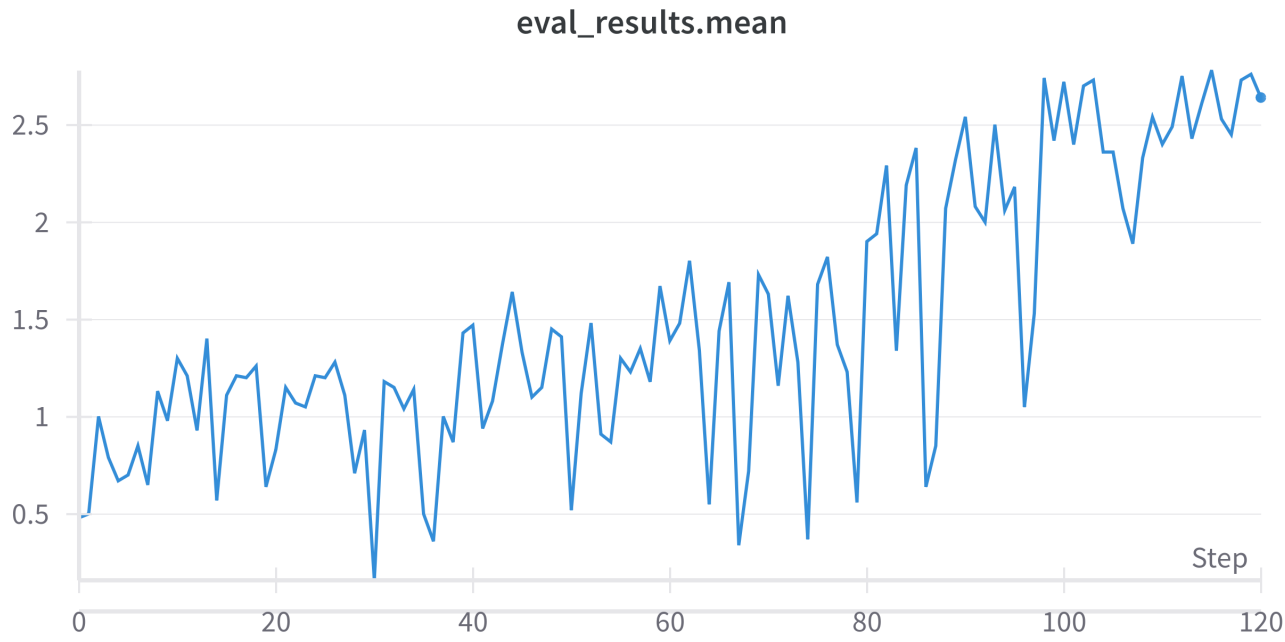

Figure 4: Training curve for SAD-3 Agent

the Noisy Network exploration is more sample efficient than ϵ-greedy exploration, and the Rainbow agent can quickly achieve a significantly higher score without too much impact on the standard deviation. This is likely due to the Noisy Network exploration being

more effective at exploring the state space of the game, and the agent can learn a more effective policy more quickly. However, we observe a cycle of regression and correction, observed in Figure 3, in the noisy network exploration agent. This is likely due to the agent learning a suboptimal policy and correcting itself in a cycle. This is a limitation of the exploration function of the agent, and the agent would likely be able to achieve a higher score and more stable learning curve with more training.

The simplified action decoder shows more stable learning and slightly lower performance than the Rainbow DQN agent. However, it showcases a nearly linear improvement in performance over time with a stable and consistent learning curve. When looking at a 3-step history, observed in Figure 4, this improvement is more pronounced. SAD, however, slightly decreases the performance of the agent when compared to the baseline Rainbow DQN agent. This is likely due to the additional information provided to the agent during training, which increases the complexity of the learning process.

The stable training curve and lower standard deviation in the final score outline the value of the SAD algorithm in learning the partnership dynamics of the game more effectively. The SAD algorithm can model the other policy effectively and learn more efficiently from it. This is important as in a game like Hanabi, the partnership dynamics are crucial to achieving a high score.

In the literature, the SAD algorithm has been shown to outperform the Rainbow DQN algorithm in the two-player self-play version of Hanabi [25]. However, in this study, the Rainbow DQN algorithm can outperform the SAD algorithm. In the literature, the SAD algorithm is often trained in an unlimited sample setting, with a combination of distributed reinforcement learning and recurrent experience replay. This is likely the reason for the discrepancy in performance between the two algorithms. The SAD algorithm is likely to perform better with more training but, in a sample-limited setting, is less effective than the Rainbow DQN algorithm.

AD-Hoc Teamplay Performance

Table 2 shows the performance of learning agents in the AD-Hoc Teamplay setting.

By observing the decrease in the average score (Avg XP) and increase in the standard deviation of the final score, we find that all the agents struggle to cooperate effectively in cross-play (XP) settings, except for the simple DQN, which seems to improve, which can be attributed to the skill of its partners. The general decrease in the score and increase in the standard deviation is likely due to the agents learning arbitrary conventions that are specific to their partners during training, or "idiosyncratic conventions" as described by Lucas and Allen [30]. This is a significant limitation of the current state-of-the-art in Hanabi, and it raises the importance of finding sample-efficient algorithms for Hanabi that can generalize to new partners.

6 CONCLUSIONS AND FUTURE WORK

This work presented an evaluation of the impact of Deep-Q-Learning techniques on a smaller version of the Hanabi Challenge. We have shown that the Rainbow DQN algorithm improves the sample efficiency of the agents over the simple DQN algorithm. We have also shown that the Simplified Action Decoder (SAD) algorithm significantly stabilizes the self-play learning process, at a slight cost to performance. We have also outlined the value of efficient exploration on sample efficiency with Noisy Networks as an exploration strategy over ϵ-greedy.

While both techniques showcase promising results in the self-play setting, the agents struggle to cooperate effectively in the AD-Hoc Teamplay setting. This highlights the importance of developing sample-efficient algorithms that can generalize to new partners effectively.

These results are limited by the computational cost of training Hanabi agents and the sample-limited setting of the experiments. Future work can extend these experiments by training a larger population of diverse agents to better showcase the performance capabilities of the agents. Additionally, the impact of hierarchical reinforcement learning [39] on the performance of agents in Hanabi can be explored to help agents learn to navigate the action space more effectively by breaking it down into smaller sub-tasks, with frameworks like Option-Critic architecture [4]. Techniques like Meta-Reinforcement [6] Learning can pave the way for more adaptable agents, and more research in the direction of Few-Shot coordination [32] can help agents learn to cooperate effectively with new partners with limited exposure.

REFERENCES

[1] *AlphaGo - Google DeepMind*. URL: https://deepmind.google/technologies/alphago/ (visited on 06/02/2024).

[2] *Artificial Intelligence Is Smart, but Does It Play Well with Others?* MIT News | Massachusetts Institute of Technology. Oct. 4, 2021. URL: https://news.mit.edu/2021/does-artificial-intelligence-play-well-others-1004 (visited on 10/29/2024).

[3] Kristopher De Asis et al. *Multi-Step Reinforcement Learning: A Unifying Algorithm*. June 11, 2018. arXiv: 1703.01327 [cs]. URL: http://arxiv.org/abs/1703.01327 (visited on 10/31/2024). Pre-published.

[4] Pierre-Luc Bacon, Jean Harb, and Doina Precup. *The Option-Critic Architecture*. Dec. 3, 2016. DOI: 10.48550/arXiv.1609.05140. arXiv: 1609.05140. URL: http://arxiv.org/abs/1609.05140 (visited on 10/25/2024). Pre-published.

[5] Nolan Bard et al. "The Hanabi Challenge: A New Frontier for AI Research". In: *Artificial Intelligence* 280 (Mar. 1, 2020), p. 103216. ISSN: 0004-3702. DOI: 10.1016/j.artint.2019.103216. URL: https://www.sciencedirect.com/science/article/pii/S0004370219300116 (visited on 04/16/2024).

[6] Jacob Beck et al. *A Survey of Meta-Reinforcement Learning*. Aug. 16, 2024. DOI: 10.48550/arXiv.2301.08028. arXiv: 2301.08028. URL: http://arxiv.org/abs/2301.08028 (visited on 10/30/2024). Pre-published.

[7] Marc G. Bellemare, Will Dabney, and Rémi Munos. *A Distributional Perspective on Reinforcement Learning*. July 21, 2017. arXiv: 1707.06887 [cs]. URL: http://arxiv.org/abs/1707.06887 (visited on 10/28/2024). Pre-published.

[8] Sushrut Bhalla, Sriram Ganapathi Subramanian, and Mark Crowley. "Deep Multi Agent Reinforcement Learning for Autonomous Driving". In: *Advances in Artificial Intelligence*

(2020), pp. 67–78. URL: https://cir.nii.ac.jp/crid/1363107370885035008 (visited on 06/02/2024).

[9] Albert Bou et al. *TorchRL: A Data-Driven Decision-Making Library for PyTorch*. Nov. 27, 2023. DOI: 10.48550/arXiv.2306.00577. arXiv: 2306.00577. URL: http://arxiv.org/abs/2306.00577 (visited on 10/29/2024). Pre-published.

[10] Rodrigo Canaan et al. "Diverse Agents for Ad-Hoc Cooperation in Hanabi". In: *2019 IEEE Conference on Games (CoG)*. 2019 IEEE Conference on Games (CoG). London, United Kingdom: IEEE, Aug. 2019, pp. 1–8. ISBN: 978-1-72811-884-0. DOI: 10.1109/CIG.2019.8847944. URL: https://ieeexplore.ieee.org/document/8847944/ (visited on 04/16/2024).

[11] Rodrigo Canaan et al. "Evaluating Rl Agents in Hanabi with Unseen Partners". In: *AAAI'20 Reinforcement Learning in Games Workshop*. 2020. URL: https://www.honda-ri.de/pubs/pdf/4260.pdf (visited on 06/24/2024).

[12] Rodrigo Canaan et al. *Evaluating the Rainbow DQN Agent in Hanabi with Unseen Partners*. Apr. 28, 2020. arXiv: 2004.13291 [cs]. URL: http://arxiv.org/abs/2004.13291 (visited on 06/24/2024). Pre-published.

[13] Christopher Cox et al. "How to Make the Perfect Fireworks Display: Two Strategies for Hanabi". In: *Mathematics Magazine* 88.5 (Dec. 1, 2015), pp. 323–336. ISSN: 0025-570X. DOI: 10.4169/math.mag.88.5.323. URL: https://doi.org/10.4169/math.mag.88.5.323 (visited on 05/14/2024).

[14] Brandon Cui et al. "K-Level Reasoning for Zero-Shot Coordination in Hanabi". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 8215–8228. URL: https://proceedings.neurips.cc/paper_files/paper/2021/hash/4547dff5fd7604f18c8ee32cf3da41d7-Abstract.html (visited on 04/16/2024).

[15] Allan Dafoe et al. *Open Problems in Cooperative AI*. Dec. 15, 2020. DOI: 10.48550/arXiv.2012.08630. arXiv: 2012.08630 [cs]. URL: http://arxiv.org/abs/2012.08630 (visited on 06/02/2024). Pre-published.

[16] Jakob Foerster et al. "Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, May 24, 2019, pp. 1942–1951. URL: https://proceedings.mlr.press/v97/foerster19a.html (visited on 04/16/2024).

[17] Meire Fortunato et al. *Noisy Networks for Exploration*. July 9, 2019. DOI: 10.48550/arXiv.1706.10295. arXiv: 1706.10295. URL: http://arxiv.org/abs/1706.10295 (visited on 10/28/2024). Pre-published.

[18] Andrew Fuchs et al. *Theory of Mind for Deep Reinforcement Learning in Hanabi*. Jan. 22, 2021. DOI: 10.48550/arXiv.2101.09328. arXiv: 2101.09328 [cs]. URL: http://arxiv.org/abs/2101.09328 (visited on 04/16/2024). Pre-published.

[19] Deepmind Google. *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*. Google DeepMind. Jan. 24, 2019. URL: https://deepmind.google/discover/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii/ (visited on 06/02/2024).

[20] *Google-Deepmind/Hanabi-Learning-Environment*. Google DeepMind. URL: https://github.com/google-deepmind/hanabi-learning-environment (visited on 05/09/2024).

[21] Bram Grooten. "Deep Reinforcement Learning for the Cooperative Card Game Hanabi". In: (2021). URL: https://research.tue.nl/files/190273227/Grooten_B.pdf (visited on 06/24/2024).

[22] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. *Deep Q-Network Based Multi-agent Reinforcement Learning with Binary Action Agents*. Aug. 6, 2020. DOI: 10.48550/arXiv.2008.04109. arXiv: 2008.04109. URL: http://arxiv.org/abs/2008.04109 (visited on 10/31/2024). Pre-published.

[23] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 30.1 (1 Mar. 2, 2016). ISSN: 2374-3468. DOI: 10.1609/aaai.v30i1.10295. URL: https://ojs.aaai.org/index.php/AAAI/article/view/10295 (visited on 10/28/2024).

[24] Matteo Hessel et al. *Rainbow: Combining Improvements in Deep Reinforcement Learning*. Oct. 6, 2017. DOI: 10.48550/arXiv.1710.02298. arXiv: 1710.02298 [cs]. URL: http://arxiv.org/abs/1710.02298 (visited on 05/16/2024). Pre-published.

[25] Hengyuan Hu and Jakob N. Foerster. *Simplified Action Decoder for Deep Multi-Agent Reinforcement Learning*. May 12, 2021. arXiv: 1912.02288 [cs]. URL: http://arxiv.org/abs/1912.02288 (visited on 04/16/2024). Pre-published.

[26] Hengyuan Hu et al. *Off-Belief Learning*. Aug. 17, 2021. DOI: 10.48550/arXiv.2103.04000. arXiv: 2103.04000 [cs]. URL: http://arxiv.org/abs/2103.04000 (visited on 05/16/2024). Pre-published.

[27] Hengyuan Hu et al. ""Other-Play" for Zero-Shot Coordination". In: *International Conference on Machine Learning*. PMLR, 2020, pp. 4399–4410. URL: http://proceedings.mlr.press/v119/hu20a.html (visited on 06/24/2024).

[28] Steven Kapturowski et al. "Recurrent Experience Replay in Distributed Reinforcement Learning". In: International Conference on Learning Representations. Sept. 27, 2018. URL: https://openreview.net/forum?id=r1lyTjAqYX (visited on 10/29/2024).

[29] Chuming Li et al. *ACE: Cooperative Multi-agent Q-learning with Bidirectional Action-Dependency*. Version 2. Dec. 2, 2022. DOI: 10.48550/arXiv.2211.16068. arXiv: 2211.16068 [cs]. URL: http://arxiv.org/abs/2211.16068 (visited on 06/02/2024). Pre-published.

[30] Keane Lucas and Ross E. Allen. *Any-Play: An Intrinsic Augmentation for Zero-Shot Coordination*. Jan. 28, 2022. DOI: 10.48550/arXiv.2201.12436. arXiv: 2201.12436 [cs]. URL: http://arxiv.org/abs/2201.12436 (visited on 04/17/2024). Pre-published.

[31] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. Dec. 19, 2013. DOI: 10.48550/arXiv.1312.5602. arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602 (visited on 10/24/2024). Pre-published.

[32] Hadi Nekoei et al. "Towards Few-shot Coordination: Revisiting Ad-hoc Teamplay Challenge In the Game of Hanabi". In: *Proceedings of The 2nd Conference on Lifelong Learning Agents*. Conference on Lifelong Learning Agents. PMLR, Nov. 20, 2023, pp. 861–877. URL: https://proceedings.mlr.press/v232/nekoei23b.html (visited on 04/16/2024).

[33] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. 1st ed. Springer Publishing

Company, Incorporated, May 2016. 134 pp. ISBN: 978-3-319-28927-4.

[34] Hirotaka Osawa. "Solving Hanabi: Estimating Hands by Opponent's Actions in Cooperative Game with Incomplete Information". In: *Computer Poker and Imperfect Information - Papers Presented at the 29th AAAI Conference on Artificial Intelligence, Technical Report*. 29th AAAI Conference on Artificial Intelligence, AAAI 2015. AI Access Foundation, 2015, pp. 37–43. URL: https://keio.elsevierpure.com/en/publications/solving-hanabi-estimating-hands-by-opponents-actions-in-cooperati (visited on 05/14/2024).

[35] *PettingZoo Documentation*. URL: https://pettingzoo.farama.org/index.html (visited on 10/29/2024).

[36] Tom Schaul et al. *Prioritized Experience Replay*. Feb. 25, 2016. DOI: 10.48550/arXiv.1511.05952. arXiv: 1511.05952. URL: http://arxiv.org/abs/1511.05952 (visited on 10/28/2024). Pre-published.

[37] Matthew Sidji, Wally Smith, and Melissa J. Rogerson. "The Hidden Rules of Hanabi: How Humans Outperform AI Agents". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI '23. New York, NY, USA: Association for Computing Machinery, Apr. 19, 2023, pp. 1–16. ISBN: 978-1-4503-9421-5. DOI: 10.1145/3544548.3581550. URL: https://doi.org/10.1145/3544548.3581550 (visited on 10/29/2024).

[38] Ho Chit Siu et al. "Evaluation of Human-AI Teams for Learned and Rule-Based Agents in Hanabi". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 16183–16195. URL: https://proceedings.neurips.cc/paper/2021/hash/86e8f7ab32cfd12577bc2619bc635690-Abstract.html (visited on 04/16/2024).

[39] Alexander Sasha Vezhnevets et al. *FeUdal Networks for Hierarchical Reinforcement Learning*. Mar. 6, 2017. DOI: 10.48550/arXiv.1703.01161. arXiv: 1703.01161. URL: http://arxiv.org/abs/1703.01161 (visited on 10/25/2024). Pre-published.

[40] Joseph Walton-Rivers et al. *Evaluating and Modelling Hanabi-Playing Agents*. Apr. 24, 2017. DOI: 10.48550/arXiv.1704.07069. arXiv: 1704.07069 [cs]. URL: http://arxiv.org/abs/1704.07069 (visited on 05/16/2024). Pre-published.

[41] Ziyu Wang et al. "Dueling Network Architectures for Deep Reinforcement Learning". In: *Proceedings of The 33rd International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, June 11, 2016, pp. 1995–2003. URL: https://proceedings.mlr.press/v48/wangf16.html (visited on 10/28/2024).

[42] Lei Yuan et al. *A Survey of Progress on Cooperative Multi-agent Reinforcement Learning in Open Environment*. Dec. 2, 2023. DOI: 10.48550/arXiv.2312.01058. arXiv: 2312.01058 [cs]. URL: http://arxiv.org/abs/2312.01058 (visited on 05/09/2024). Pre-published.

[43] Jaleh Zand, Jack Parker-Holder, and Stephen J. Roberts. *On-the-Fly Strategy Adaptation for Ad-Hoc Agent Coordination*. Mar. 7, 2022. DOI: 10.48550/arXiv.2203.08015. arXiv: 2203.08015 [cs, stat]. URL: http://arxiv.org/abs/2203.08015 (visited on 04/16/2024). Pre-published.

A EXTENDED RESULTS AND DISCUSSION

This section provides a full overview of the training and evaluation results of the agents in the two-player self-play setting. The training and evaluation results are presented in the form of the average score and loss curves for each agent during training. The training and loss curves provide insights into the learning process of the agents and their performance in the game and are presented in Figures 5 - 18.

We observe that the simple DQN agent struggles to learn the game effectively and fails to achieve a high score in the game. The agent's performance is unstable, and it fails to maintain a high score over time, despite the loss curve showing a decreasing trend. The agent's performance is likely limited by the simple DQN architecture and the sample-limited training setting. The agent's performance is presented in Figures 5 and 6.

All of the Rainbow agents show the highest performance in the game, with the base Rainbow agent achieving the highest average score. The Rainbow agents show a cyclic pattern of improvement and regression in their performance, discussed earlier in Section 5. Interestingly, the Rainbow agents with a 3-step and 5-step history show no clear trend in their loss curves, despite showing a general improvement in their performance over time. This is only present in the Rainbow agents, and it is likely due to the added noise from the Noisy Networks exploration strategy in combination with the larger history lengths causing some instability in the learning process. It is also likely linked to the cyclic pattern of improvement and regression in the Rainbow agents' performance. The Rainbow agents' performance is presented in Figures 7 - 12.

Both the SAD and Distributed agents show a very pronounced rapid increase in loss for the first 10-20 episodes, followed by a rapid decrease in loss. This can likely be explained by the lack of meaningful experiences in the replay buffer at the start of training, leading to a high TD error and thus a high loss. As the agents gain more experience, the loss decreases rapidly, indicating that the agents are learning effectively from the experiences. Additionally, the target network would only begin to stabilize after a few episodes, leading to a decrease in the loss. This is also likely exacerbated by the initial fully random exploration coupled with Hanabi's partially observable nature. The SAD and Distributed agents' performance is presented in Figures 13 - 18.

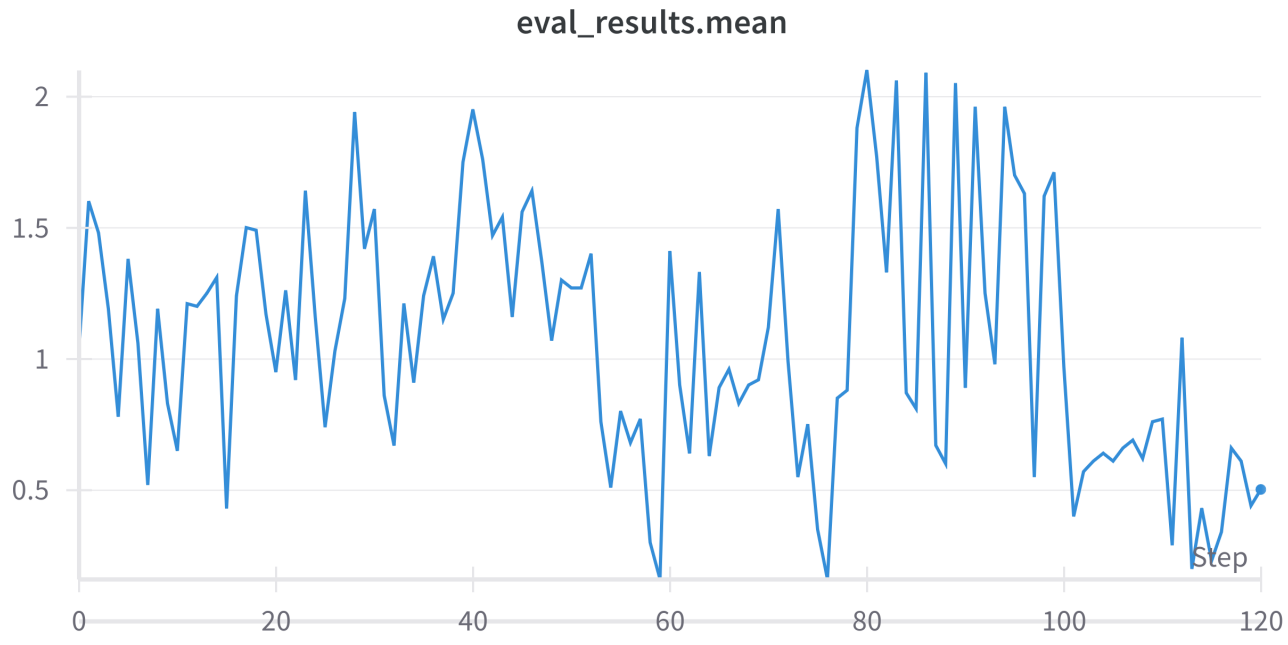

Figure 5: Training curve for Simple DQN Agent

Figure 6: Loss curve for Simple DQN Agent

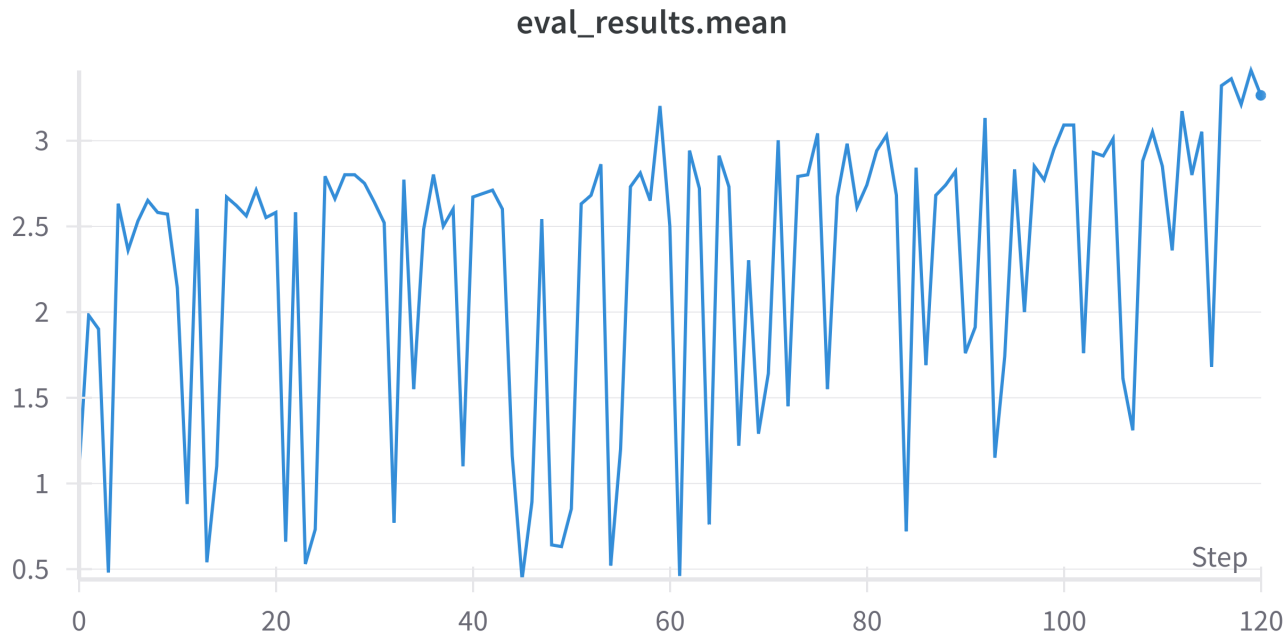

Figure 7: Training curve for Rainbow Agent

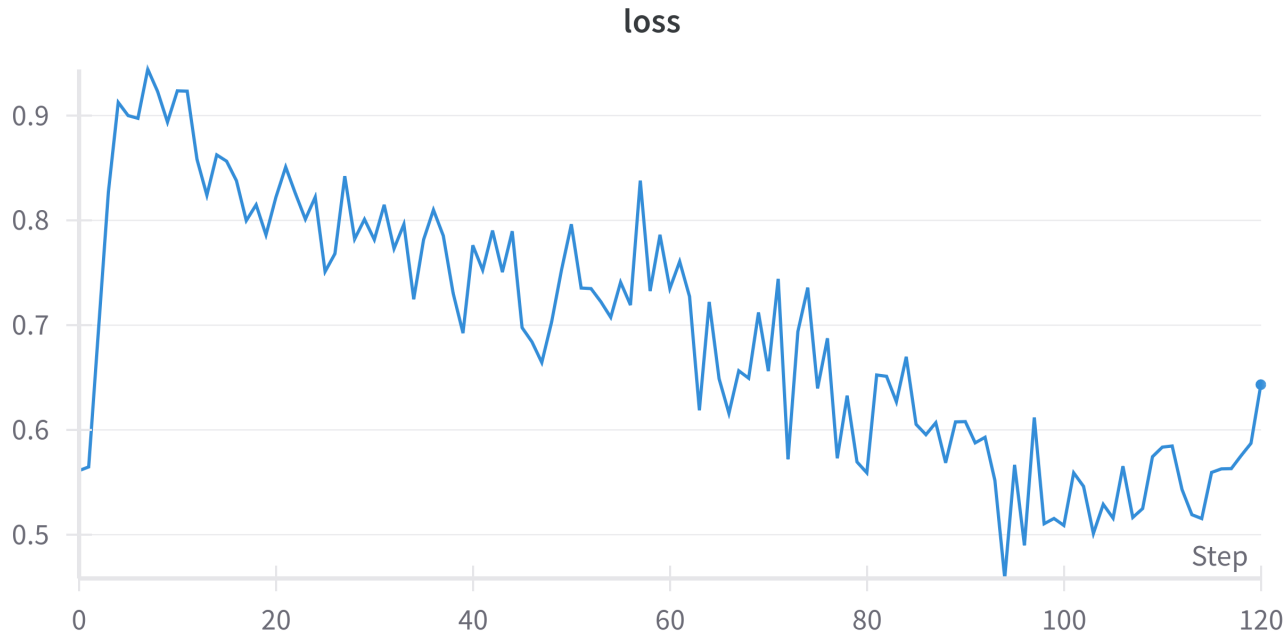

Figure 8: Loss curve for Rainbow Agent

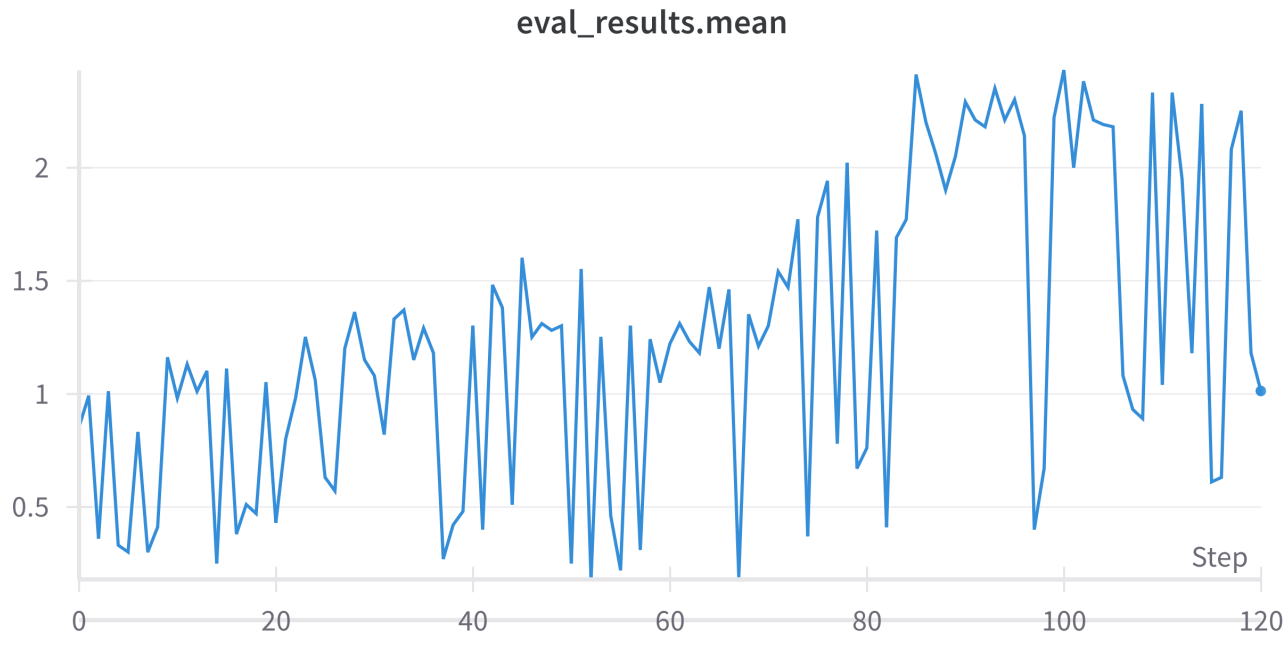

Figure 9: Training curve for Rainbow-3 Agent

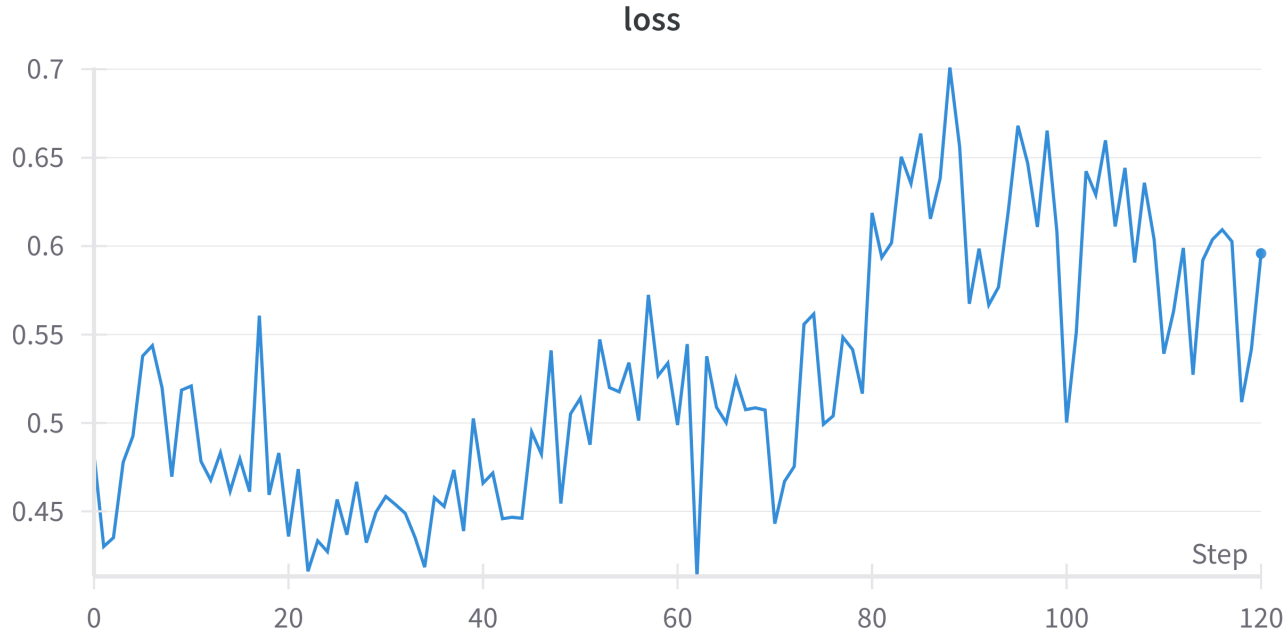

Figure 10: Loss curve for Rainbow-3 Agent

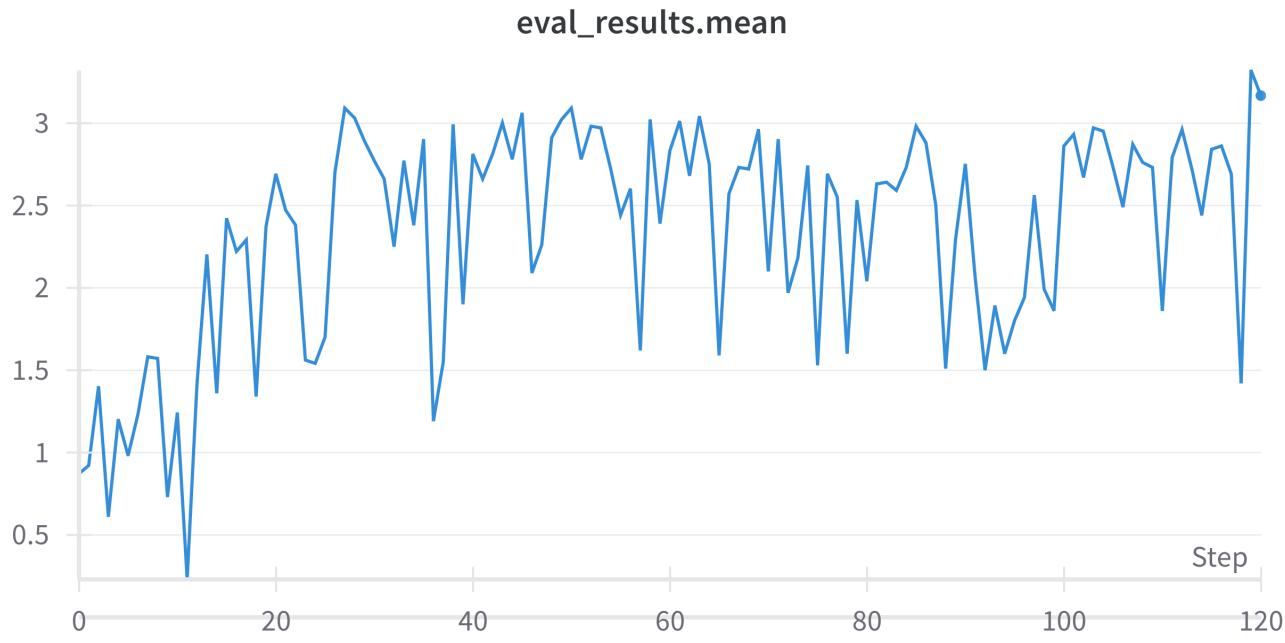

Figure 11: Training curve for Rainbow-5 Agent

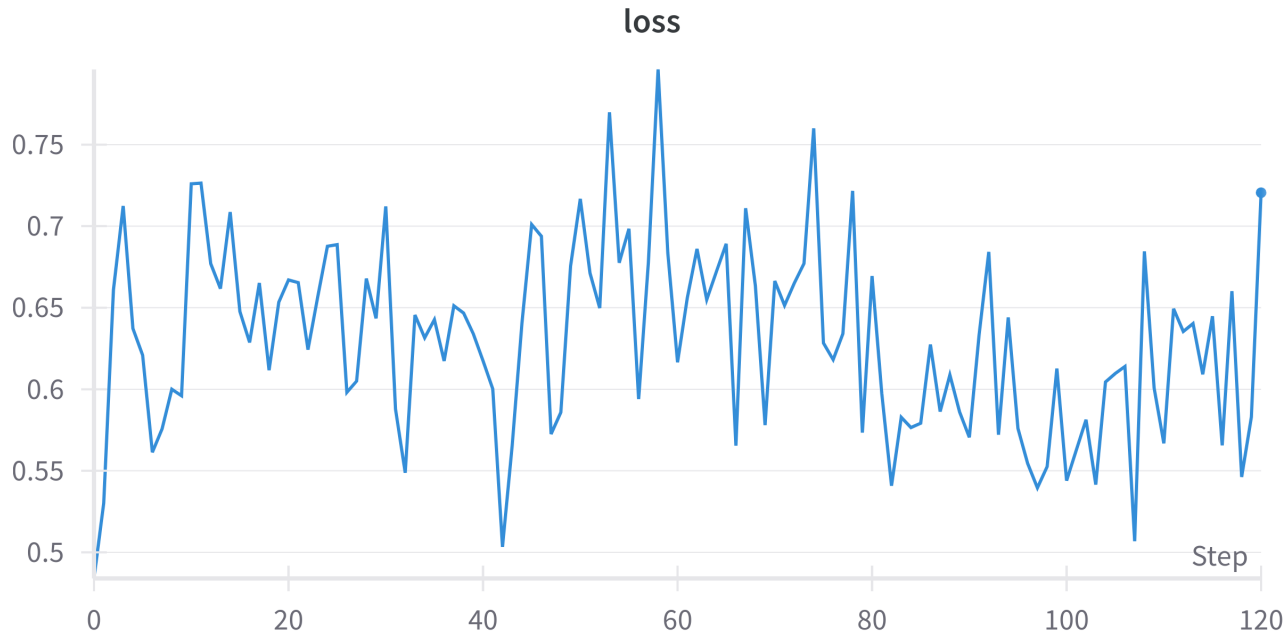

Figure 12: Loss curve for Rainbow-5 Agent

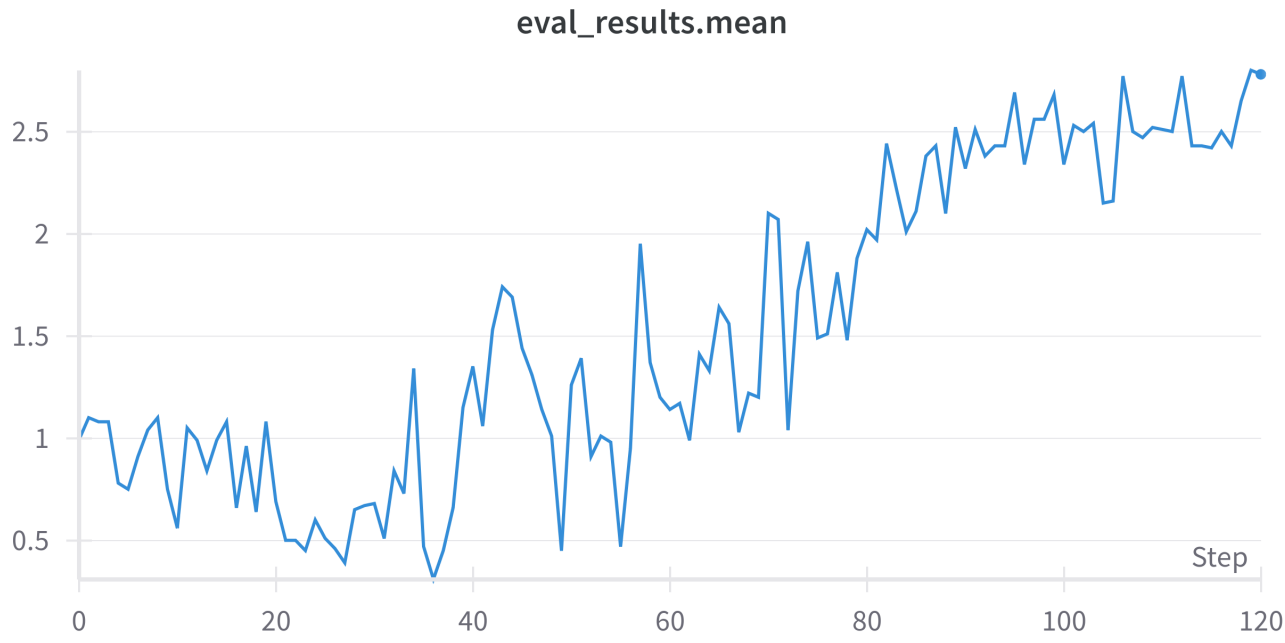

Figure 13: Training curve for Distributed Agent

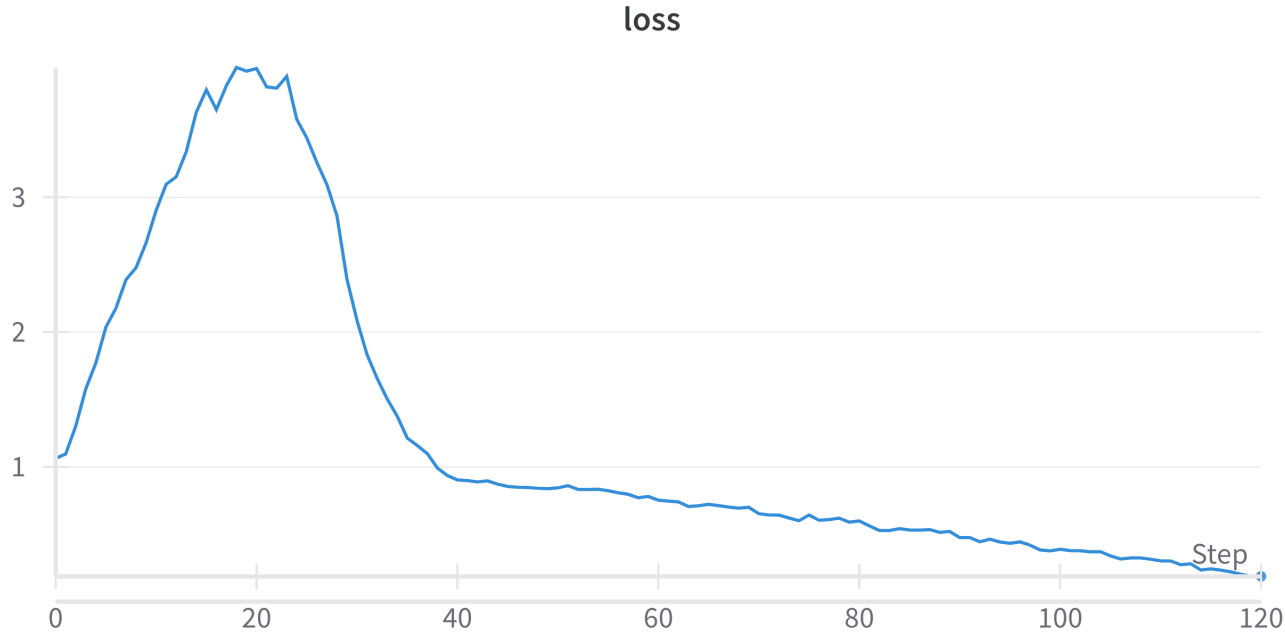

Figure 14: Loss curve for Distributed Agent

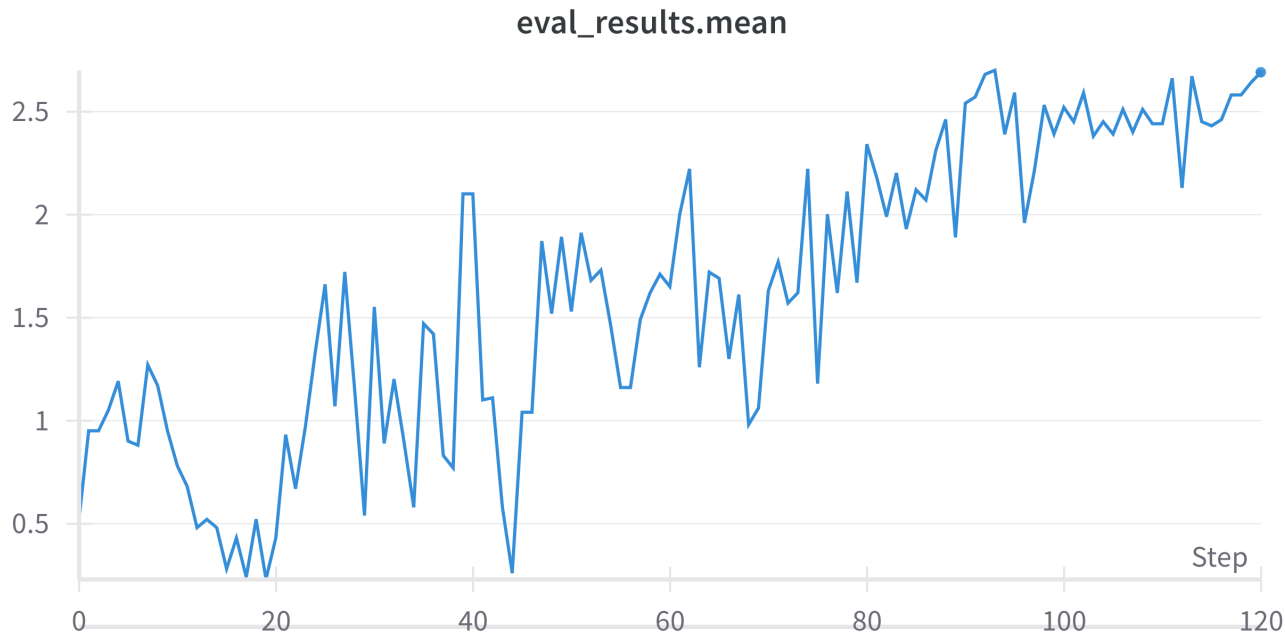

Figure 15: Training curve for SAD Agent

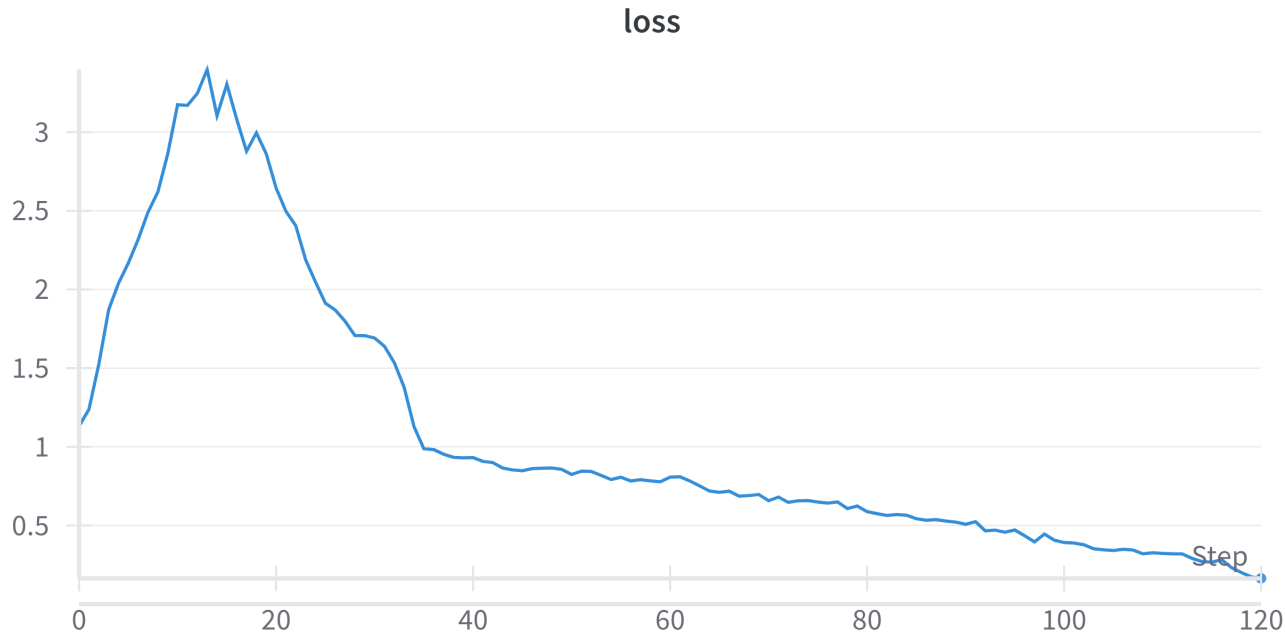

Figure 16: Loss curve for SAD Agent

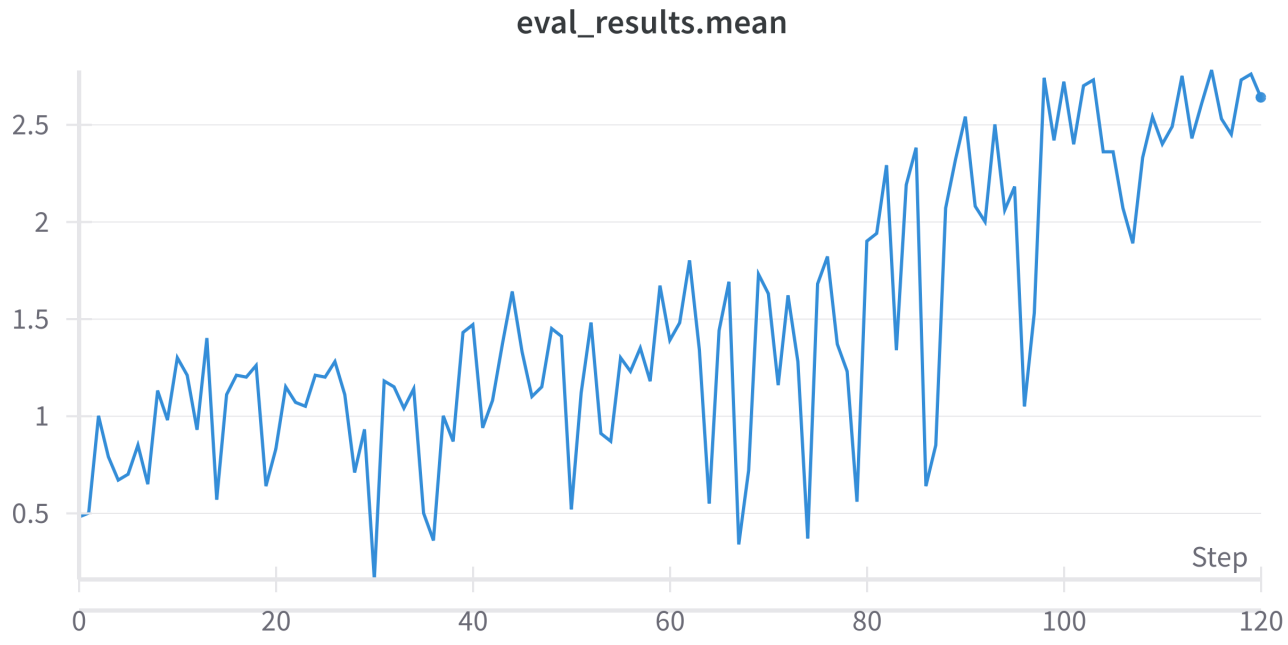

Figure 17: Training curve for SAD-3 Agent

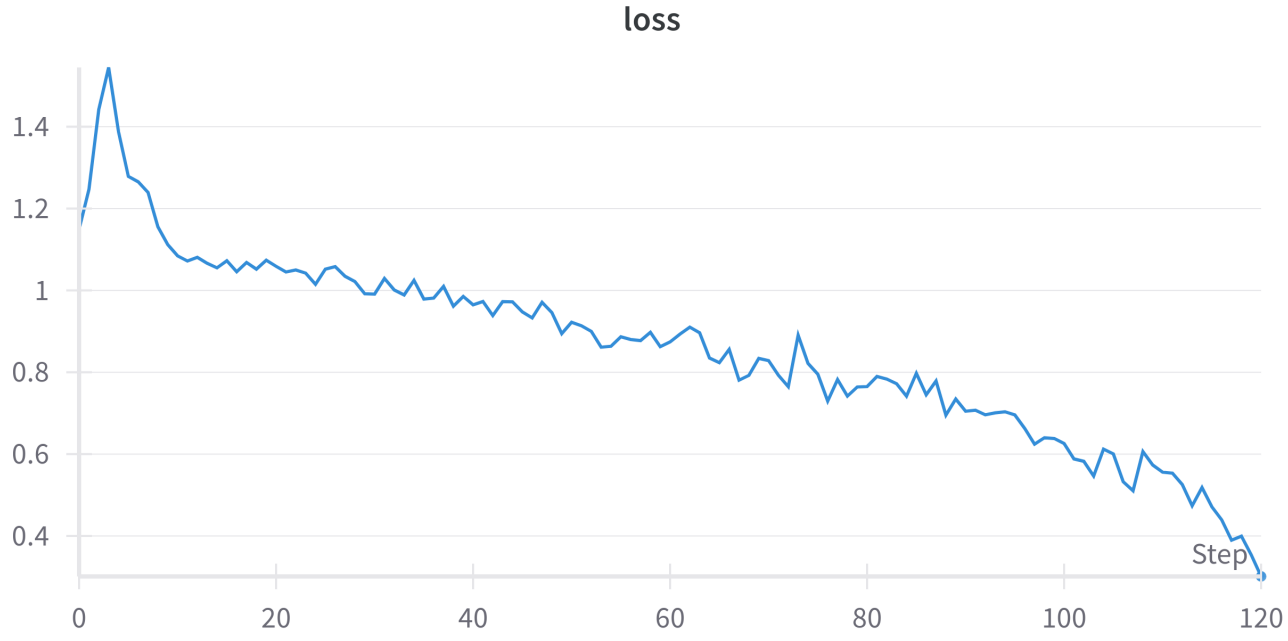

Figure 18: Loss curve for SAD-3 Agent