On the impact of lowering temperature on learning with behavior clones

Anonymous authors
Paper under double-blind review

Abstract

In an environment requiring cooperation with unknown external agents, the agent will need to adapt and adjust their policies according to the external agent's behavior. We cannot simply adopt a self-optimal policy and assume the other agent to be similarly optimal. Even in cases where the external agent is highly adaptive, i.e., a human, it could still result in sub-optimal performance. Limited access to the external agent further compounds this challenge, rendering direct training implausible. To address this, a behavior clone (proxy) can be created from observations and the agent is subsequently trained offline with the behavior clone as partner. However, the accuracy of the behavior clone is often not guaranteed, constrained by limitations such as the amount of observations or the clone's inherent capacity. This inaccuracy of the behavior clone could lead to a decline in performance or even outright training failure. This paper will first demonstrate that learning from clones could result in a drop in the agent's performance. Following, it will show that lowering the temperature of the clone's behavior during training mitigates this drop in the agent's performance. These findings offer insights that could potentially contribute to improving learning from behavior clones.

1 Introduction

Carroll et al. (2019) have demonstrated that in cooperative environments, agents that are trained via self-play perform very well when paired with themselves but perform poorly when paired with agents that are humans or mimic human play through behavior cloning. They subsequently trained the agents with the clones and have improved performance (when paired with humans) for 3 out of 5 scenarios. The authors have attributed bad reinforcement learning as well as the clones being a poor human model, for the poor performance of the agent in some of the scenarios.

Our key idea is to investigate how a lack in clone's accuracy affects agent's performance, find ways to mitigate if so, and if scenarios in which the agent fails the training with the clone can be avoided.

This is not the first time temperature have been used in reinforcement learning. Usama & Chang (2021) have used temperature to adjust the exploration rate of the agent during training and Ha & Schmidhuber (2018) have used temperature to prevent exploitation of the deficiencies in the dynamics model. While some parallels can be drawn between our work and theirs, there are a few key differences: 1. Temperature is not applied to the clone, but the agent; instead of adjusting the exploration rate of the agent, the parameter is adjusting the stochasticity of the clone. 2. A dynamics model is not being used; the actual environment is used to train the agent.

Our contribution. In this paper we make the following primary contributions:

- **Analysis of the impact of behavior clones' temperature on agent's performance.** We investigated the impact of adjusting the temperature on agent's performance and demonstrated that lowering the temperature (decrease in entropy) may help to mitigate the inaccuracies of the behavior clones.

- **Analysis of the impact of behavior clones' temperature on agent's training.** We further investigate the impact of adjusting the temperature on the agent's training process and gained some insights into learning from behavior clones.

2 Related work

2.1 Model-based reinforcement learning

Dyna, an architecture introduced by Sutton (1991), includes a learned world model which takes in a state and an action and predicts the next state. This learned world model then is used to train (through reinforcement learning) the policy of the agent. Ha & Schmidhuber (2018) have demonstrated training an agent entirely inside a learned world model. While Ha & Schmidhuber (2018) uses a learned world model, *AlphaGo*, (Silver et al., 2018), uses a known world model.

2.2 Behavior cloning

Learning from demonstrations (Schaal, 1996), also called behavior cloning and imitation learning, learns a behavior, uses demonstrations from an expert to train an agent offline. It has been used to train agents to play video games (Spick et al., 2024) and robotics (Florence et al., 2021). In our work, the focus is on using behavior cloning to create a proxy to the partner; not to train the agent directly.

2.3 Human-AI cooperation

Inspired by the popular game *Overcooked*, Carroll et al. (2019) created a highly cooperative environment and demonstrated that self-play trained agents aren't good at cooperating with humans. Choudhury et al. (2020) compared 3 approaches, model-free, model-based (which is the approach this paper uses), and theory of mind, to human-AI interaction in the context of autonomous driving. Instead of using human data, Strouse et al. (2021) uses multiple self-play trained agents and their checkpoints, to increase the diversity of training partners. Tylkin et al. (2021) specifically train partners to increase robustness in performance against other agents and humans.

2.4 Temperature

In reinforcement learning, Usama & Chang (2021) and Ha & Schmidhuber (2018) have used temperature to adjust the stochasticity of the agent and the environment respectively. Hinton et al. (2015) uses temperature to soften the probability distribution for model distillation. In large language models, temperature is a parameter that can be adjusted to improve performance (Zhu et al., 2023).

3 Preliminaries

Throughout the paper, the agent that is being trained will be referred to as the *agent*, the original partner, human or otherwise, will be referred to as the *partner*, the behavior clone that is meant as a proxy to the partner will be referred to as the *clone*, the agent, the partner and the clone will be collectively referred to as the *players*.

3.1 Markov decision process

The Markov decision process (MDP) is then denoted by a tuple, $(\mathcal{S}, \mathcal{A}, P, \mathcal{R}, R, \gamma)$, π where we use $t \in \mathbb{N}_{\geq 0}$ to denotes the time step, where $\mathbb{N}_{\geq 0}$ denotes the natural numbers *including zero*. $\mathcal{S}$ is the set of possible states that the agent can be in. Similarly, $\mathcal{A}$ is the set of possible actions the agent can perform. $\mathcal{R}$ is the set of possible rewards. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longmapsto [0, 1]$ is called the *transition function*. For all $(s, a, s', t) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathbb{N}_{\geq 0}$, let $P(s, a, s') \coloneqq \Pr(S_{t+1} = s' \mid S_t = s, A_t = a)$. That is, P characterizes the distribution over states at time $t + 1$ given the state and action at time

t. The reward function is denoted by R. γ is the discount factor defined as $\gamma \in [0, 1)$. π is the policy which is defined as $\pi : \mathcal{S} \times \mathcal{A} \longmapsto [0, 1]$.

The agent's objective is to maximize the expected sum of discounted rewards, G_t at any time-step t: $G_t = \sum_{k=0}^{H} \gamma^k r_{t+k+1}$

4 Problem formulation

4.1 Environment

We will be adopting the Overcooked AI environment from Carroll et al. (2019). The environment consist of 5 environments with varying levels of coordination challenges. The goal is to place three onions in a pot (dark gray), which takes 20 time steps to cook, take out the resulting soup on a plate (white) and deliver it (light gray), as many times as possible within the time limit. In each of the scenario as shown in 1, the agent is represented by the one in blue hat while the partner is represented by the one in green hat.

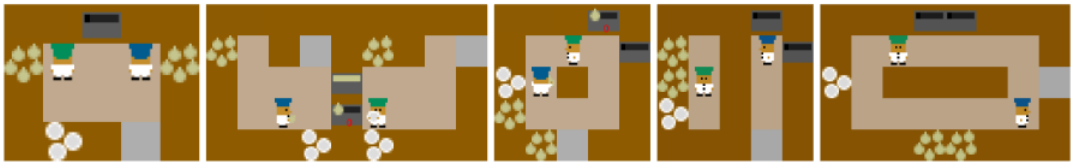


Figure 1: 5 scenarios of Overcooked AI (Carroll et al., 2019). From left to right: *Cramped Room*, *Asymmetric Advantages*, *Coordination Ring*, *Forced Coordination* and *Counter Circuit*

We model the above environment as a Markov decision process (MDP).

4.1.1 State space

The state space consist of 1. the positions and facing direction of the 2 players, 2. the positions of the onions, plates, pot and serving stations (which are static), 3. the positions of any plates/onions/soups that is placed on the ground/counter and if they are held by any players (these items can be moved around by the players), 4. the number of onions in the pot, 5. the countdown timer for the cooking of the soup and if it is ready when the timer is done.

4.1.2 Action space

There are 6 discrete actions in total: up, down, left, right, idle, and interact. Interact action is used to pick up or put down onions, plates or soup.

4.1.3 Reward function

The players are rewarded 20 points every time a soup is delivered. Due to the sparse rewards, to speed up training, Carroll et al. (2019) introduced reward shaping – 3 points is rewarded if a player places an onion into a pot or if it takes a dish while the soup is cooking, and 5 points if it picks up the soup with a dish.

5 Experimental set-up

5.1 Players

5.1.1 Agent

The architecture of the agent consist of 3 parts: An encoder, in which latent representations of the observations of the agent is extracted, a LSTM, which serves as an episodic memory where the

trajectories' information is being stored, and a MLP, which outputs the action probabilities of the agent. Inspired by Ha & Schmidhuber (2018), only the MLP layer is trained online. The encoder and the LSTM are pre-trained using observation data offline. The agent's architecture is further detailed in the Appendix.

5.1.2 Partner

Due to the impracticality of using humans as partners to train agents, we will be using a trained agent as the partner. We will adopt the self-play trained agent from Carroll et al. (2019). We demonstrate that even with non-adaptive partners, a drop in performance is observed when the agent is trained with the clone.

5.1.3 Clones

The clones will be trained in a supervised manner using 10 sets of partner-partner observations using the pre-trained modules – the encoder and the LSTM.

5.2 Training details

In Carroll et al. (2019), The agent is trained using PPO, (Schulman et al., 2017), for 500 episodes, each episodes comprises of 400 environment time steps. Both Carroll et al. (2019) and Ha & Schmidhuber (2018) uses on-policy methods with Carroll et al. (2019) using PPO as one of it's training algorithm as well.

The temperature softmax function, $q_t = \frac{\exp(z_t/T)}{\sum_j \exp(z_i/T)}$, is used to transform the continuous action space of the cloned PPO agents into a discrete action space via the logits. q_t and z_t represents the probability and logits of action A_t respectively, T represents the temperature parameter. A higher temperature increases the entropy of the probability distribution, lowering the confidence of the agent's actions. A lower temperature decreases the entropy of the probability distribution, increasing the confidence of the agent's actions.

Due to varying difficulty levels of the scenarios, there will be training differences as stated in Table 1.

Reward shaping refers to whether intermediate rewards from picking up ingredients/dishes/soup is used. *Reward shaping* helps with the agent's training as the actual reward of the environment, serving a soup, is very sparse and requires multiple steps. *Pre-trained* refers to whether the agent's policy is randomly initiated and trained from scratch, or pre-trained with imitation learning. Baselines, which are trained with the partners instead of clones, retains the *reward shaping* configuration but all policies are randomly initiated and trained from scratch.

Scenarios without *pre-trained* configuration will be trained 5 times, each with a different random seed, while scenarios with *pre-trained* configuration will be trained 4 times, each with a different random seed.

SCENARIO	REWARD SHAPING	PRE-TRAINED
Cramped Room	False	False
Asymmetric Advantages	True	False
Coordination Ring	True	False
Forced coordination	True	True
Counter circuit	True	True

Table 1: Scenarios' training configurations

5.3 Evaluation details

The agent is check-pointed after every 4000 environment time steps of training. For each of the last 10 checkpoints, the agent is evaluated based on mean-rewards obtained over 10 episodes. During the evaluation, the partner's actions remain probabilistic while the agent's actions are not.

6 Results

To study the impact of lowering the clone's temperature on agent's performance, each scenario will be trained 3 times: clone's temperature = 1.0, clone's temperature = 0.1 and the baseline – where the agent is trained with the partner itself (partner's temperature = 1.0) We present our results in Figure 2.

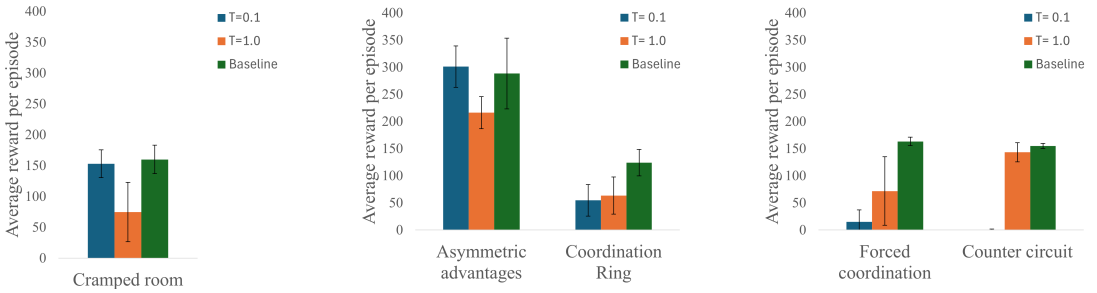
6.1 Clone vs partner

First of all, the experiments demonstrated that training with behavior clones can indeed results in performance degradation as compared to training with the partner itself. In *Cramped Room* and *Asymmetric Advantages*, the performance drops when training with behavior clones in their respective settings. In *Coordination Ring*, *Forced Coordination* and *Counter Circuit*, without *pre-training*, training with behavior clones have all failed. Even with *pre-training*, performance degradation is still observed in scenarios *Coordination Ring* and *Forced Coordination*.

6.2 Impact of temperature on performance

In scenarios *Cramped Room* and *Asymmetric Advantages*, there is evidence to show that lowering the temperature of the clone does help to mitigate the drop in agent's performance when training with an imperfect clone. By lowering the temperature of the clone, the agent's performance have even matched the baseline performance. However, in other scenarios, not only does lowering the temperature not increase performance, but it is also observed to be detrimental, i.e., *Counter Circuit*

We speculate that in these scenarios, where a high-level of coordination is required, the self-played trained partners might be conforming to very specific coordination policies. Since the behavior clone is trained using partner-partner trajectories, the clone will learn only this specify policy and the partner's behavior in trajectories other than that which might be encountered during the agent's training might be misrepresented – When training from scratch, the agent might fail to invoke any meaningful reactions from the clone. Even when pre-trained with imitation learning, the agent might venture into situations which are unknown to the clone.



(a) *Reward shaping*: False, *Pre-trained*: False

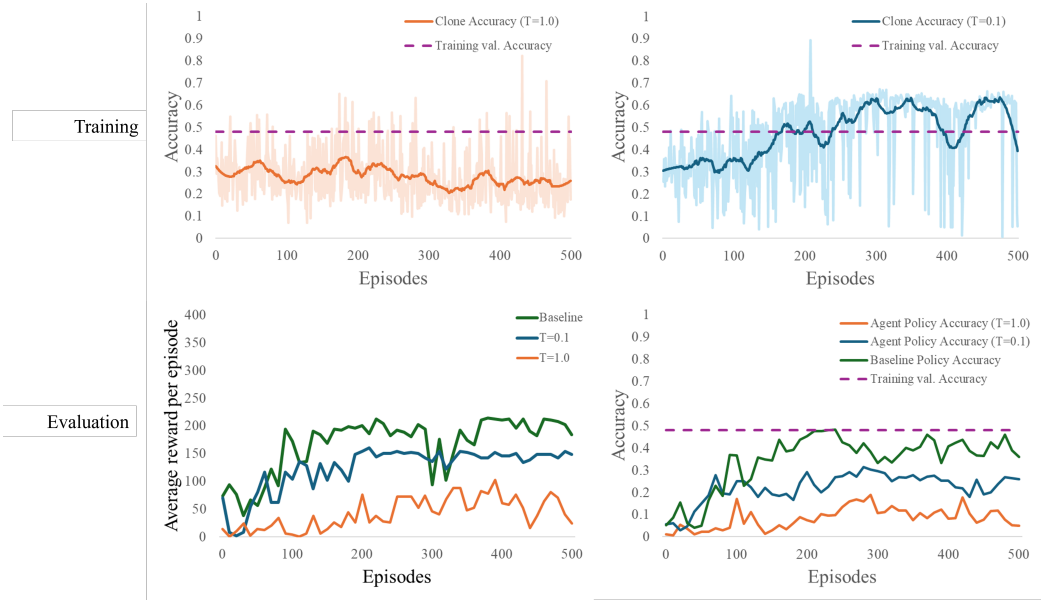
(b) *Reward shaping*: True, *Pre-trained*: False

(c) *Reward shaping*: True, *Pre-trained*: True

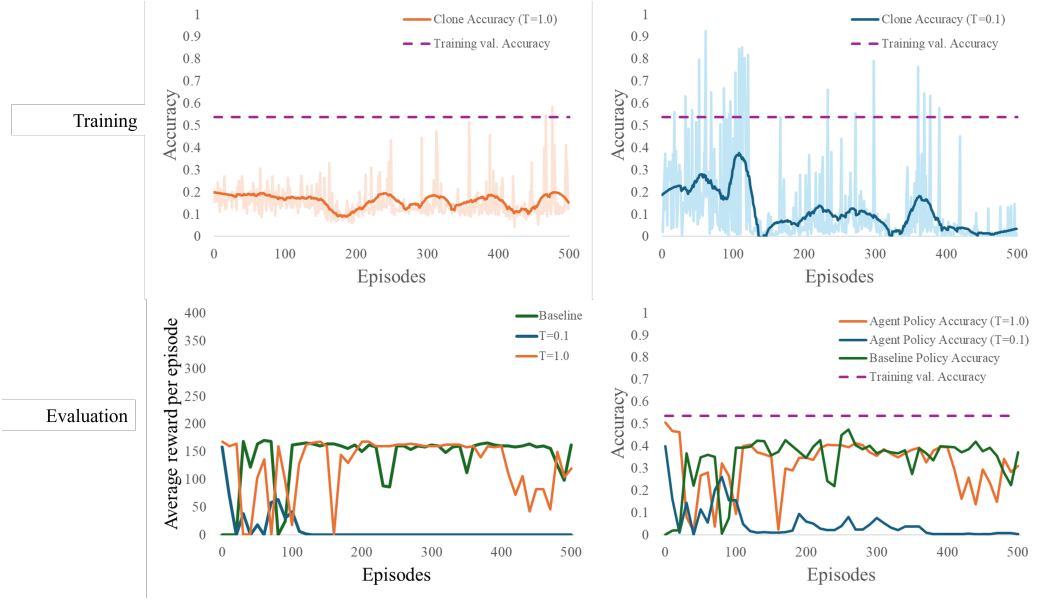
Figure 2: Average rewards per episodes of agents trained with: 1. Behavior clone with temperature=0.1. 2. Behavior clone with temperature=1.0. 3. Actual partner (baseline) with temperature=1.0. Mean and standard deviation is calculated from results of 5 training runs each with a different random seed.

6.2.1 Analysis

To investigate, we look at: 1. The clone's accuracy (w.r.t. the partner) during training. 2. The accuracy of the learned agent's policy (w.r.t. the partner) during evaluation. Figure 3a and Figure 3b shows the above information for a single training for scenario *Cramped Room* and *Counter Circuit* respectively.



(a) *Cramped Room*



(b) *Counter Circuit*

Figure 3: Top left: Clone accuracy at temperature=1.0. Top right:Clone accuracy at temperature=0.1. Bottom left: Evaluation results for each training checkpoint. Bottom right: Agent policy accuracy during evaluation. Clone accuracy and agent policy accuracy refers to the action with the maximum probability, regardless of actual action taken. Training val. accuracy refers to the accuracy of the validation dataset during behavior cloning.

The results from both the scenarios show that the behavior clone do indeed only learn behavior for very specific situations. The clone's accuracy during agent's training is consistently lower than the validation accuracy during the behavior cloning process (which is based on partner-partner trajectories). Only during the training for scenario *Cramped Room* with temperature=0.1 did the clone's accuracy during rose to the validation accuracy level. This might also shine some light on how lowering the clone's temperature affects training. At temperature=1.0, the clone's accuracy remain relatively similar throughout the training process. This suggests that the agent remained in a the unknown space of the clone. At temperature=0.1 though, we observed that the variation of the clone's accuracy is much higher and often matches or exceeds the validation accuracy. This suggests that through a lower clone's temperature, actions more true to the partner are being invoked and in turn, steers the agent's training towards a more familiar space of the clone. For scenario *Counter Circuit*, it seems to agree in the first 100 episodes of training. However, the training crashes and even though there are spikes of high accuracy, the clone could have been stuck in a space where actions are meaningless. This seems to agree with the previous speculation that the clone (being trained with partner-partner trajectories) misrepresents behavior in non-optimal trajectories.

In both scenarios, there's seems to be more steering power with lower temperatures, which increases the chance of the agent navigating towards a more familiar space of the clone. However, if missed, could have an even more detrimental effect on the agent training. This might explain why the results from Counter Circuit disagrees with the results from Cramped Room and Asymmetric Advantages.

6.3 Further experiments

The following experiments aim to expand the study on the impact of temperature of behavior clone on agent performance. To do so, we first expand the range of temperature. Additional, we run a deterministic as well as a random clone to represent the lower and upper limits of temperature respectively. As we change the temperature, the mean probability of the most probable action (i.e., entropy) changes. We present the impact of the clone's temperature on the performance of the agent as a plot of average reward per episode against the mean probability of the most probable action for the clone. Additionally we plot an adjusted mean probability for the clone (the probability is multiplied by the accuracy of the behavior clone plus the remaining probability divide by 5). This will provide us with a measure of the probability that the clone will execute an action that is true to the partner.

In figure 4, both baseline and clone curves shows that an overly confident clone/partner is as detrimental to the agent's training as a random one. An agent which is trained with an overly confident clone/partner, learns a policy in response to a deterministic (or very close to one) behavior. This results in a rigid and specialized policy which does not perform well under evaluation where the partner's actions are probabilistic. This is exacerbated by the fact that the clone's action is only partially representative of the partner's. By introducing stochasticity (increasing temperature from 0) into the clone/partner's actions, we exposes the agent to a wider variety of situations which is a better model of the partner's action during evaluation. However, there will come a point where the benefits will cease when the clone/partner's actions approach total randomness (when temperature is very high).

The adjusted clone curve shows that they, (the clone and the baseline) had a similar threshold before performance starts to deteriorate. And this threshold is based on the measure of the probability that the clone will execute an action that is true to the partner. This suggests a few things: 1. The impact of temperature is linked to the accuracy of the clone's behavior. 2. The impact of temperature is linked to the initial entropy of the clone's behavior; in our experiments, the trained policies have relatively high entropy at default temperature (1.0), shown by the low mean probability of the most probable action. Therefore, there can be an increase in performance with lower temperature. 3. There is a limitation to the improvements that lowering the clone's temperature can have. If the clone's accuracy is high to begin with, most probably there wouldn't be much of an improvement. Conversely, if the clone's accuracy is too low, while there might be improvements by lowering the

clone's temperature, the clone might approach deterministic actions (which is detrimental) before approach performance plateau; the baseline performance.

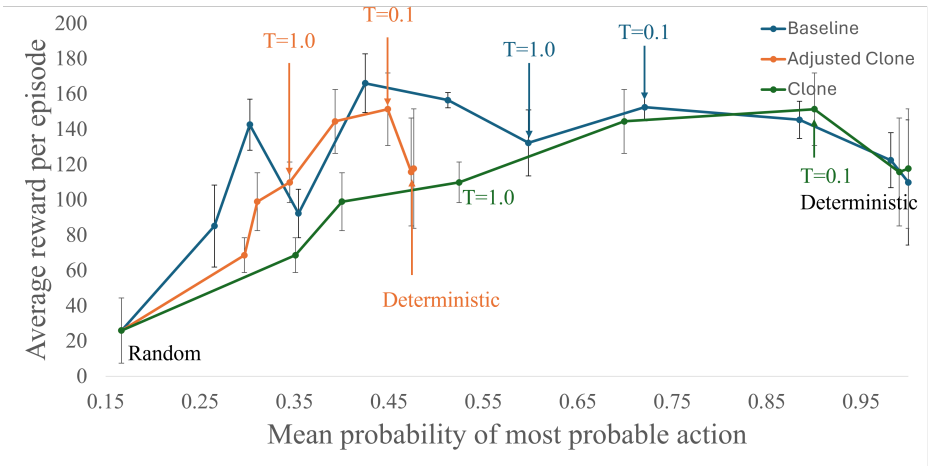


Figure 4: Performance against mean probability of the most probable action

7 Conclusion

Training a behavior clone as a proxy have been demonstrated to be difficult. Not only does the accuracy of the clone suffers from the training of the clone itself, the observations/trajectories used to train the clone also suffers from non-coverage of all situations the agent might encounter. However, experiments demonstrated that lowering the temperature might assist and mitigate the lack in accuracy of the clone in agent's training. Due to the inherent fragility of reinforcement learning, some of the scenarios used in this paper might not display and prove to be strong evidence. Nevertheless, temperature can be a potential and viable consideration during training with clones.

Limitations and future work.

- **Generalizability** The current agent's architecture consist of various components and each component might affect the agent's training in a variety of ways. While there are 5 different scenarios with varying level of cooperation needed, the environment's rewards, action space are fixed. Therefore, the results presented in this paper is only valid for this particular setting and environment and might not be generalized to other settings/environment as well as other training algorithms.

- **Temperatures are fixed** Future work can include using adaptive temperature scaling (Zhu et al., 2023) to vary temperature during training.

- **Behavior clones are static; they do not undergo training themselves.** Also mentioned by Carroll et al. (2019), humans are highly adaptive and a static clone will not be able to represent the behavior of a human. Future work can include engaging real human players as partners or using a non-static clone to train the agent; clones that are able to mimic the way humans learn and adapt to different behaviors of the agent.

- **Environment consist only of 2 players** The study done in this study is limited to only 2 players; the clone/partner and the agent. Future work can include a study of how clones-partners, clones-clones interact with each other in environments where cooperation is needed with more than one external agent.

References

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/f5b1b89d98b7286673128a5fb112cb9a-Paper.pdf.

Rohan Choudhury, Gokul Swamy, Dylan Hadfield-Menell, and Anca D. Dragan. On the utility of model learning in hri. In *Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction*, HRI '19, pp. 317–325. IEEE Press, 2020. ISBN 9781538685556.

Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. *Conference on Robot Learning (CoRL)*, 2021.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

Stefan Schaal. Learning from demonstration. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, pp. 1040–1046, Cambridge, MA, USA, 1996. MIT Press.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404. URL https://www.science.org/doi/abs/10.1126/science.aar6404.

Ryan Spick, Timothy Bradley, Ayush Raina, Pierluigi Vito Amadori, and Guy Moss. Behavioural Cloning in VizDoom. *arXiv e-prints*, art. arXiv:2401.03993, January 2024. doi: 10.48550/arXiv.2401.03993.

DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 14502–14515. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/797134c3e42371bb4979a462eb2f042a-Paper.pdf.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, jul 1991. ISSN 0163-5719. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

Paul Tylkin, Goran Radanovic, and David C. Parkes. Learning robust helpful behaviors in two-player cooperative atari environments. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, pp. 1686–1688, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450383073.

Muhammad Usama and Dong Eui Chang. Learning-driven exploration for reinforcement learning. In *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, pp. 1146–1151, 2021. doi: 10.23919/ICCAS52745.2021.9649810.

Yuqi Zhu, Jia Li, Ge Li, YunFei Zhao, Jia Li, Zhi Jin, and Hong Mei. Hot or cold? adaptive temperature sampling for code generation with large language models, 2023.

A Agent's architecture

The architecture of the agent consist of 3 parts: An encoder, in which latent representations of the observations of the agent is extracted, a LSTM, which serves as an episodic memory where the trajectories' information is being stored, and a MLP, which outputs the action probabilities of the agent. Inspired by Ha & Schmidhuber (2018), only the MLP layer is trained online. The encoder and the LSTM are pre-trained using observation data offline.

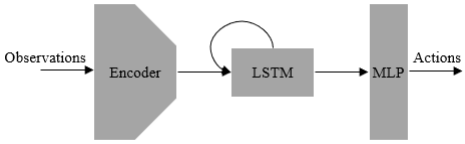
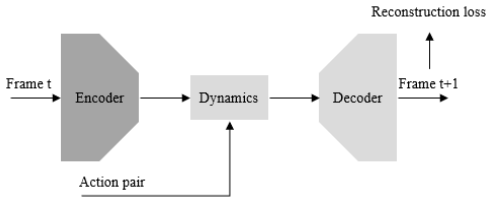


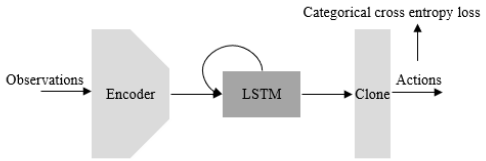
Figure 5: Agent's architecture.

The encoder is pre-trained by training a encoder-decoder pair to predict the next frame with the input actions. By doing so, we are training the encoder to emphasize on the features that are mostly dynamic; not the features that are static/unimportant. Only the encoder is kept, the rest are discarded.

The LSTM is pre-trained by training with the pre-trained encoder (frozen) above and a MLP to predict the actions of the players. This is the same method that is used to train the clones, however, during behavior cloning, both pre-trained encoder and pre-trained LSTM are frozen.



(a) Pre-training for the encoder



(b) Pre-training for the LSTM

Figure 6: Pre-training of the encoder and the LSTM in the agent's model