Swarm Behavior Cloning

Jonas Nüßlein, Maximilian Zorn, Philipp Altmann and Claudia Linnhoff-Popien
Institute of Computer Science, LMU Munich, Germany
fi

Keywords: Reinforcement Learning, Imitation Learning, Ensemble, Robustness.

Abstract: In sequential decision-making environments, the primary approaches for training agents are Reinforcement Learning (RL) and Imitation Learning (IL). Unlike RL, which relies on modeling a reward function, IL leverages expert demonstrations, where an expert policy π_e (e.g., a human) provides the desired behavior. Formally, a dataset D of state-action pairs is provided: $D = (s, a = \pi_e(s))$. A common technique within IL is Behavior Cloning (BC), where a policy $\pi(s) = a$ is learned through supervised learning on D. Further improvements can be achieved by using an ensemble of N individually trained BC policies, denoted as $E = \{\pi_i(s)\}_{1 \leq i \leq N}$. The ensemble's action a for a given state s is the aggregated output of the N actions: $a = \frac{1}{N}\sum_i \pi_i(s)$. This paper addresses the issue of increasing action differences—the observation that discrepancies between the N predicted actions grow in states that are underrepresented in the training data. Large action differences can result in suboptimal aggregated actions. To address this, we propose a method that fosters greater alignment among the policies while preserving the diversity of their computations. This approach reduces action differences and ensures that the ensemble retains its inherent strengths, such as robustness and varied decision-making. We evaluate our approach across eight diverse environments, demonstrating a notable decrease in action differences and significant improvements in overall performance, as measured by mean episode returns.

1 INTRODUCTION

Reinforcement Learning (RL) is a widely recognized approach for training agents to exhibit desired behaviors by interacting with an environment over time. In RL, an agent receives a state s, selects an action a, and receives feedback in the form of a reward, learning which behaviors are beneficial (high reward) and which are detrimental (low reward) based on the reward function provided by the environment (Sutton and Barto, 2018). While RL has shown great promise, designing an effective reward function can be highly challenging. A well-designed reward function must satisfy several criteria: (1) the optimal behavior should yield the maximum possible return R^* (the sum of all rewards in an episode); (2) suboptimal behaviors must be penalized, resulting in a return $R < R^*$, ensuring that shortcuts or unintended strategies are discouraged; (3) the reward function should be dense, providing informative feedback at every step of an episode rather than just at the end; (4) the reward should support gradual improvement, avoiding overly sparse rewards such as those that assign 1 to the optimal trajectory and 0 to all others, which can hinder exploration and learning (Eschmann, 2021; Knox et al., 2023).

Due to the inherent complexity of crafting such reward functions, the field of *Imitation Learning* (IL) has emerged as an alternative approach (Zheng et al., 2021; Torabi et al., 2019b). Instead of relying on an explicitly defined reward function, IL uses expert demonstrations to model the desired behavior. This paradigm has proven effective in various real-world applications, such as autonomous driving (Bojarski et al., 2016; Codevilla et al., 2019) and robotics (Giusti et al., 2015; Finn et al., 2016). A prominent method within IL is Behavior Cloning (BC), where supervised learning is applied to a dataset of state-action pairs $D = \{(s, a = \pi_e(s))\}$, provided by an expert policy π_e. Compared to other IL methods like Inverse Reinforcement Learning (Zhifei and Meng Joo, 2012; Nüßlein et al., 2022) or Adversarial Imitation Learning (Ho and Ermon, 2016; Torabi et al., 2019a), BC has the advantage of not requiring further interactions with the environment, making it particularly suitable for non-simulated, real-world scenarios.

A straightforward extension of BC is the use of an ensemble of N individually trained policies. In this approach, the *ensemble action* is computed by aggregating the N predicted actions as $a = \frac{1}{N}\sum_i \pi_i(s)$. Although ensemble methods often improve robustness, they can encounter challenges when states in the train-

Nüßlein, J., Zorn, M., Altmann, P. and Linnhoff-Popien, C.
Swarm Behavior Cloning.
DOI: 10.5220/0013086600003890

In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 1*, pages 23-32
ISBN: 978-989-758-737-5; ISSN: 2184-433X

ing data D are underrepresented. For these states, the N policies may predict actions $\{a_i\}_{1\leq i\leq N}$ that diverge significantly, leading to suboptimal aggregated actions.

In this paper, we address the problem of *increasing action differences* in such underrepresented states. Specifically, we propose a new loss function that encourages greater alignment among the N policies in the ensemble while preserving the diversity of their computations. This approach reduces action differences and ensures that the ensemble retains its inherent strengths, such as robustness and varied decision-making. As illustrated in Figure 1, this approach—termed *Swarm Behavior Cloning*—leads to more consistent predictions across the ensemble, with the N predicted actions (gray dots) clustered more closely together compared to standard Ensemble Behavior Cloning (middle plot). By minimizing action divergence, our approach improves the quality of the aggregated action and enhances performance in diverse environments.

2 BACKGROUND

2.1 Reinforcement Learning

Reinforcement Learning (RL) problems are often modeled as Markov Decision Processes (MDP). An MDP is represented as a tuple $M = \langle S, A, T, r, p_0, \gamma \rangle$ where S is a set of states, A is a set of actions, and $T(s_{t+1} \mid s_t, a_t)$ is the probability density function (pdf) for sampling the next state s_{t+1} after executing action a_t in state s_t. It fulfills the Markov property since this pdf solely depends on the current state s_t and not on a history of past states $s_{\tau<t}$. $r : S \times A \to \mathbb{R}$ is the reward function, p_0 is the start state distribution, and $\gamma \in [0;1)$ is a discount factor which weights later rewards less than earlier rewards (Phan et al., 2023).

A deterministic *policy* $\pi : S \to A$ is a mapping from states to actions. Return $R = \sum_{t=0}^{\infty} \gamma^t \cdot r(s_t, a_t)$ is the (discounted) sum of all rewards within an episode. The task of RL is to learn a policy such that the expected cumulative return is maximized:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, J_{p_0}(\pi, M) = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot r(s_t, a_t) \mid \pi\right]$$

Actions a_t are selected following policy π. In Deep Reinforcement Learning the policy π is represented by a neural network $\hat{f}_\phi(s)$ with a set of trainable parameters ϕ (Sutton and Barto, 2018).

2.2 Imitation Learning

Imitation Learning (IL) operates within the framework of Markov Decision Processes, similar to Reinforcement Learning (RL). However, unlike RL, IL does not rely on a predefined reward function. Instead, the agent learns from a dataset of expert demonstrations consisting of state-action pairs:

$$D = \{(s_i, a_i = \pi_e(s_i))\}_i$$

where each a_i represents the expert's action $\pi_e(s_i)$ in state s_i. IL is particularly useful in situations where demonstrating the desired behavior is easier than designing a corresponding reward function.

IL can be broadly divided into two main categories: Behavior Cloning (BC) and Inverse Reinforcement Learning (IRL). Behavior Cloning focuses on directly mimicking the expert's actions by training a policy through supervised learning on the provided dataset D (Torabi et al., 2018). In contrast, Inverse Reinforcement Learning seeks to infer the underlying reward function $r_e(s,a)$ that would make the expert's behavior optimal, using the same dataset D.

The key advantage of BC over IRL is that BC does not require further interactions with the environment during training. This makes BC more applicable to real-world (non-simulated) scenarios, where collecting new trajectories can be costly, time-consuming, or even dangerous due to the exploratory actions involved (Zheng et al., 2021; Torabi et al., 2019b).

In addition to BC and IRL, there are adversarial approaches to IL that do not neatly fit into these two categories. These methods also necessitate environment rollouts for training. The core idea behind adversarial methods is to frame the learning process as a zero-sum game between the agent and a discriminator. The discriminator's objective is to distinguish between state-action pairs generated by the agent and those produced by the expert, while the agent tries to generate actions that fool the discriminator (Ho and Ermon, 2016; Torabi et al., 2019a).

3 PROBLEM ANALYSIS: ACTION DIFFERENCE IN ENSEMBLE BEHAVIOR CLONING

When training an ensemble of N policies, denoted as $\{\pi_i\}_{1\leq i\leq N}$, on a given dataset $D = \{(s_t, a_t)\}_t$ consisting of state-action pairs, each policy π_i is trained independently to predict actions based on the input states. Due to differences in the training paths and the inherent variability in the learning process, the ensemble members typically produce different action pre-

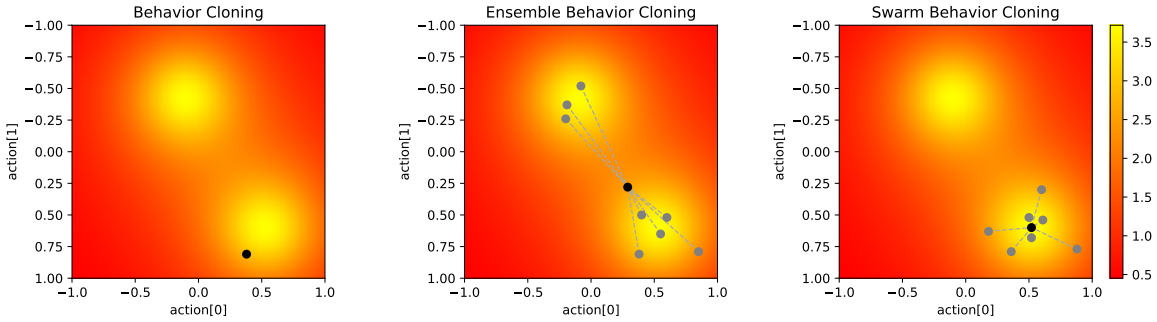

Figure 1: This figure visualizes schematically the predicted actions of three different Behavior Cloning approaches, represented as black dots, in a 2-dim action space for some state s_t. The heatmap represents the Q-values $Q(a_t, s_t)$. (Left) the left plot shows plain *Behavior Cloning*. A policy π was trained using supervised learning on some training data D. The black dot is the predicted action $a_t = \pi(s_t)$. (Middle) in *Ensemble Behavior Cloning* an ensemble of N policies is trained individually on D. The N predicted actions $\{a_i = \pi_i(s_t)\}$ (gray dots) are then aggregated to the ensemble action (black dot). (Right) in our approach *Swarm Behavior Cloning* an ensemble of N policies is trained as well. However, they are not trained individually but using a modified loss function, see formula (2). The effect is a smaller difference between the N predicted actions, resembling a swarm behavior. Similar to *Ensemble Behavior Cloning* the ensemble action (black dot) is then aggregated from the N predicted actions (gray dots).

dictions for the same state s. This divergence can be quantified through the concept of *mean action difference*, which we formally introduce in Definition 3.1.

Definition 1 (Mean Action Difference)**.** *Let $E = \{\pi_i\}_{1 \le i \le N}$ represent an ensemble of N policies, and let s be a given state. For the ensemble E, we can compute N action predictions, denoted as $A = \{a_i = \pi_i(s)\}_{1 \le i \le N}$. The mean action difference, d, is defined as the average pairwise L2-norm between the actions in A. Formally, the mean action difference d is given by:*

$$d = \frac{2}{N(N-1)} \sum_i \sum_{j>i} \|a_i - a_j\|$$

This measure quantifies the average difference between the action predictions of the ensemble members for a particular state s. A higher value of d indicates greater divergence in the actions predicted by the policies, whereas a lower value suggests that the ensemble members are more consistent in their predictions.

In Figure 2 (left), we illustrate the *mean action difference* across an entire episode within the *LunarLander-continuous* environment. The ensemble in this experiment consisted of six policies trained on a dataset D derived from a single expert demonstration. The expert used for this demonstration was a fully-trained Soft-Actor-Critic (SAC) model from Stable-Baselines 3 (Raffin et al., 2021). The figure highlights areas with both high and low mean action differences. In regions with high *mean action difference*, the ensemble members produce actions that are quite divergent, whereas regions with low *mean action difference* show greater agreement among the predictions.

To explore this phenomenon further, we analyzed two specific timesteps, $t = 120$ and $t = 225$, where we visualized the actions predicted by the $N = 6$ policies (represented by gray dots) alongside the aggregated action (black dot) on the 2D action space of this environment. The Q-value heatmap, provided by the expert SAC critic network, is also displayed. Notably, in the upper-right plot, we observe a scenario where the Q-value of the aggregated action, $Q(a,s)$, is lower than the Q-values of all individual actions, i.e., $Q(a,s) < Q(a_i,s)$ for all i. This phenomenon is more prevalent in states where the *mean action difference* is large, leading to suboptimal aggregated actions due to the inconsistency among the policies.

In Chapter 5, we present a modified training loss designed to tackle the problem of divergent action predictions within the ensemble. This new loss function encourages the individual policies in the ensemble to learn more similar hidden feature representations, effectively reducing the *mean action difference*. **By fostering greater alignment among the policies while preserving the diversity of their computations, the ensemble retains its inherent strengths—such as robustness and varied decision-making—while producing more consistent actions**. This reduction in action divergence improves the quality of the aggregated actions, ultimately leading to enhanced overall performance.

4 RELATED WORK

Imitation Learning is broadly divided into Behavior Cloning (BC) and Inverse Reinforcement Learning

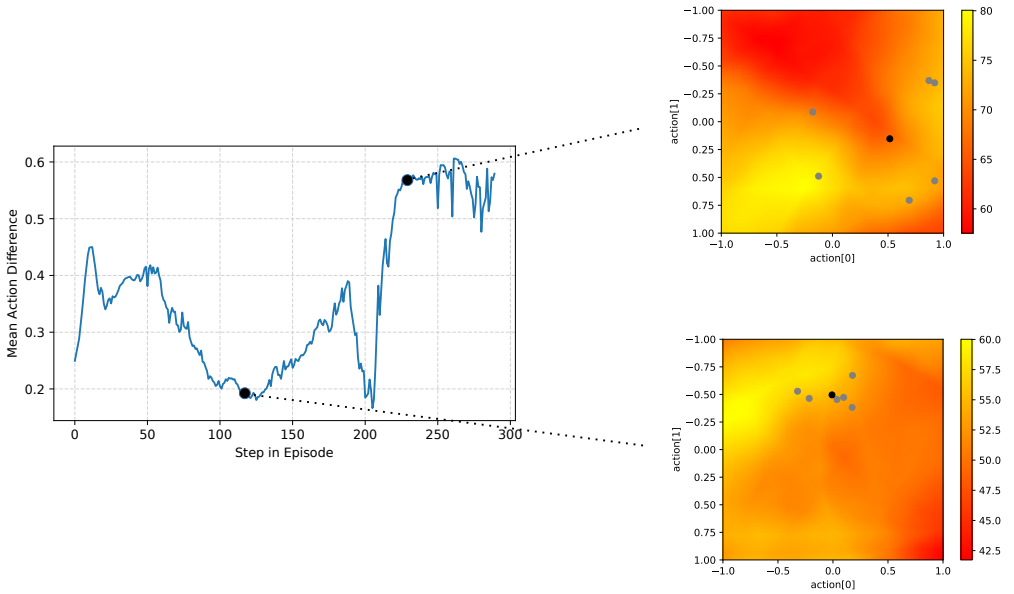

Figure 2: This figure visualizes exemplarily the *mean action difference* for an entire episode of an ensemble containing $N = 6$ policies. We used the *LunarLander-continuous* environment since it has a 2-dim action space that can be easily visualized. The x-axis in the left plot represents the timestep in the episode. For two interesting timesteps, we have visualized the predicted actions of the N policies $\{a_t^i = \pi_i(s_t)\}$ (gray dots) as well as the aggregated action (black dot) on a 2-dim map (the complete action space). The underlying heatmap represents the Q-values from the expert critic (a fully-trained SAC model from Stable-Baselines 3).

(IRL) (Zheng et al., 2021; Torabi et al., 2019b). While in Behavior Cloning the goal is to learn a direct mapping from states to actions (Bain and Sammut, 1995; Torabi et al., 2018; Florence et al., 2022), IRL is a two-step process. First, the missing Markov Decision Process (MDP) reward function is reconstructed from expert data, and then a policy is learned with it using classical reinforcement learning (Arora and Doshi, 2021; Ng et al., 2000). Besides BC and IRL, there are also adversarial methods such as GAIL (Ho and Ermon, 2016) or GAIfO (Torabi et al., 2019a). BC approaches cannot be adequately compared to IRL or adversarial methods, since the latter two require access to the environment for sampling additional episodes. Therefore, we compare our approach only against other Behavior Cloning approaches: BC (Bain and Sammut, 1995) and Ensemble BC (Yang et al., 2022).

Besides these approaches, other Behavior Cloning algorithms exist which, however, require additional inputs or assumptions. In (Brantley et al., 2019) the algorithm *Disagreement-regularized imitation learning* is presented that first learns a standard BC ensemble. In the second phase, another policy is learned by using a combination of BC and RL, in which the reward function is to not drift into states where the variance of the ensemble action predictions is large. The policy therefore has two goals: (1) it should act similarly to the expert (2) the policy should only perform actions that ensure the agent doesn't leave the expert's state distribution. However, this approach also requires further interactions with the environment making it inappropriate to compare against a pure Behavior Cloning algorithm. The similarity to our approach *Swarm BC* is that both algorithms try to learn a policy that has a low *mean action difference*. Our algorithm can therefore be understood as a completely offline version of *Disagreement-regularized imitation learning*.

In (Smith et al., 2023) the proposed algorithm *IL-BRL* requires an additional exploration dataset beyond the expert dataset and subsequently uses any Offline RL algorithm. In (Hussein et al., 2021) a data-cleaning mechanism is presented to remove suboptimal (adversarial) demonstrations from D before applying BC. (Torabi et al., 2018) proposes a state-only BC approach using a learned inverse dynamics model for inferring the executed action.

In (Shafiullah et al., 2022) Behavior Transformers are introduced for learning offline from multi-modal data. The authors in (Wen et al., 2020) present a BC adaption for combating the "copycat problem" that emerges if the policy has access to a sliding window of past observations. In this paper, we assume a Markov policy that only receives the last state as input. Therefore the "copycat problem" does not apply here.

There is much literature on ensemble methods

(Dong et al., 2020; Sagi and Rokach, 2018; Dietterich et al., 2002; Zhou and Zhou, 2021; Webb and Zheng, 2004). The main difference of current ensemble methods to our approach is that we encourage the ensemble members to reduce the output diversity, while current methods try to increase the output diversity. We show that in Markov Decision Problem environments, ensembles with large action diversities can lead to poor aggregated actions. In the next chapter, we therefore present an algorithm that reduces the *mean action difference*.

5 SWARM BEHAVIOR CLONING

In this section, we introduce our proposed approach, *Swarm Behavior Cloning* (Swarm BC), which aims to reduce the divergence in action predictions among ensemble policies by encouraging them to learn similar hidden feature representations.

We assume that each of the N policies in the ensemble $E = \{\pi_i\}_{1 \leq i \leq N}$ is modeled as a standard Multilayer Perceptron (MLP). The hidden feature activations of policy π_i at hidden layer k, given input state s, are represented as $h_{ik}(s) \in \mathbb{R}^m$, where m is the number of neurons in that hidden layer. These hidden activations form the basis of the ensemble's predictions.

Consider a training data point $(s,a) \in D$, where s is the state and a is the expert's action. In standard Behavior Cloning (BC), each policy in the ensemble is trained individually using a supervised learning loss function. The goal is to minimize the difference between each policy's predicted action $\pi_i(s)$ and the corresponding expert action a. The standard loss for training a BC ensemble is given by:

$$L(s,a) = \sum_i (\pi_i(s) - a)^2 \tag{1}$$

This formulation treats each policy independently, which can lead to divergence in their predicted actions, especially in underrepresented states, resulting in a high *mean action difference*.

The core idea behind *Swarm BC* is to introduce an additional mechanism that encourages the policies to learn more similar hidden feature activations, which in turn reduces the variance in their predicted actions. This is achieved by modifying the standard loss function to include a regularization term that penalizes large differences in hidden feature activations between policies. The adjusted loss function is:

$$L(s,a) = \sum_i (\pi_i(s) - a)^2 + \tau \sum_k \sum_{i<j} \left(h_{ik}(s) - h_{jk}(s)\right)^2 \tag{2}$$

Algorithm 1: Swarm Behavior Cloning.

Input: expert data $D = \{(s, a = \pi_e(s))\}$
Parameters:
τ (regularization coefficient)
N (number of policies in the ensemble)
Output: trained ensemble $E = \{\pi_i\}_{1 \leq i \leq N}$. Predict an action a for a state s using formula (3)

1: initialize N policies $E = \{\pi_i\}_{1 \leq i \leq N}$
2: train ensemble E on D using loss (2)
3: **return** trained ensemble E

The first term is the standard supervised learning loss, which minimizes the difference between the predicted action $\pi_i(s)$ and the expert action a. The second term introduces a penalty for dissimilarity between the hidden feature activations of different policies at each hidden layer k. The hyperparameter τ controls the balance between these two objectives: reducing action divergence and maintaining accuracy in reproducing the expert's behavior.

By incorporating this regularization term, the individual policies in the ensemble are encouraged to align their internal representations of the state space, thereby reducing the *mean action difference*. At the same time, the diversity of the ensemble is preserved to some extent, allowing the individual policies to explore different solution paths while producing more consistent outputs.

The final action of the ensemble, known as the *ensemble action*, is computed as the average of the actions predicted by the N policies, following the standard approach in ensemble BC:

$$a = \frac{1}{N} \sum_i \pi_i(s) \tag{3}$$

This averaging mechanism allows the ensemble to benefit from the collective knowledge of all policies, while the regularization ensures that the predictions remain aligned.

The overall procedure for *Swarm BC* is summarized in Algorithm 1. The algorithm takes as input the expert dataset $D = (s, a = \pi_e(s))$, the number of ensemble members N, and the hyperparameter τ. It outputs a trained ensemble $E = \{\pi_i\}_{1 \leq i \leq N}$ capable of making robust action predictions by computing the ensemble action as shown in Equation (3).

6 EXPERIMENTS

In this experiments section, we want to verify the following two hypotheses:

- Using our algorithm *Swarm Behavior Cloning* we can reduce the *mean action difference* as defined in **Definition 3.1** compared to standard Ensemble Behavior Cloning.
- *Swarm Behavior Cloning* shows a better performance compared to baseline algorithms in terms of *mean episode return*.

For testing these hypotheses we used a large set of eight different OpenAI Gym environments (Brockman et al., 2016): *HalfCheetah, BipedalWalker, LunarLander-continuous, CartPole, Walker2D, Hopper, Acrobot and Ant*. They resemble a large and diverse set of environments containing discrete and continuous action spaces and observation space sizes ranging from 4-dim to 27-dim.

To examine if *Swarm BC* improves the test performance of the agent we used a similar setting as in (Ho and Ermon, 2016). We used trained SAC- (for continuous action spaces) and PPO- (for discrete action spaces) models from Stable-Baselines 3 (Raffin et al., 2021) as experts and used them to create datasets D containing $x \in [1,8]$ episodes. Then we trained our approach and two baseline approaches until convergence. We repeated this procedure for 5 seeds. The result is presented in Figure 3.

For easier comparison between environments, we scaled the return. For this, we first determined the mean episode return following the expert policy R^{expert} and the random policy R^{random}. Then we used the formula $R^{scaled} = (R - R^{random})/(R^{expert} - R^{random})$ for scaling the return into the interval [0,1]. 0 represents the performance of the random policy and 1 of the expert policy.

The solid lines in Figure 3 show the mean test performance for 20 episodes and 5 seeds. The shaded areas represent the standard deviation.

The main conclusion we can draw from this experiment is that *Swarm BC* was nearly never worse than *BC* or *Ensemble BC* and in larger environments significantly better. In *HalfCheetah*, for example, the agent achieved a mean scaled episode return of 0.72 using *Swarm BC* for datasets D containing 8 expert episodes and just 0.17 using *Ensemble BC*. *Ensemble BC* still performed better than *single BC* in most environments.

To test whether our approach does reduce the *mean action difference* as introduced in **Definition 3.1** we used the same trained models from the previous experiment and calculated the *mean action difference* for each timestep in the test episodes. The x-axes in Figure 4 represent the timestep and the y-axes represent the *mean action difference*. The plots show the average for 20 episodes and 5 seeds. *Swarm Behavior Cloning* did reduce the *mean action difference*, but not always to the same extend. In the *BipedalWalker* environment it was reduced by almost 44% while in *Ant* it was only reduced by 11%.

Nevertheless, we can verify the hypothesis that *Swarm BC* does reduce the *mean action difference*.

Swarm BC shows significantly better performance compared to baseline algorithms with nearly no computational overhead. The main disadvantage however is the introduction of another hyperparameter τ. To test the sensitivity of this parameter we conducted an ablation study regarding τ and also about N (the number of policies in the ensemble E). The results are plotted in Figure 5. For this ablation study, we used the *Walker2D* environment and again scaled the return for better comparison. For τ we tested values within $\{0.0, 0.25, 0.5, 0.75, 1.0\}$. The main conclusion for the ablation in τ is that too large values can decrease the performance. $\tau = 0.25$ worked best so we used this value for all other experiments in this paper.

For the ablation on N we tested values in the set $\{2,4,6,8\}$. For $N = 2$ the test performance was significantly below the test performance for $N = 4$. For larger ensembles ($N = 6$ and $N = 8$) the performance did not increase significantly anymore. But since the training time scales linearly with the number of policies N we chose $N = 4$ for all experiments.

As a conclusion of this experiments section we can verify both hypotheses that *Swarm BC* increases test performance and decreases the *mean action difference*. In the next chapter, we provide a theoretical analysis of our approach.

7 THEORETICAL ANALYSIS

Let $(s,a) \in D$ be a random but fixed state-action tuple out of the training dataset D. Let $E = \{\pi_i\}_{1 \le i \le N}$ be an ensemble of N policies, each represented as an MLP containing K hidden layers. For state s the ensemble E produces N hidden feature activation $h_{i,k}$ for each layer $k \in [1,K]$.

The basic idea of our approach *Swarm BC* is to train an ensemble E that produces similar hidden features:

$$\forall\, (i,j) \in [1,N]^2, k \in [1,K]: \;\; h_{i,k} \approx h_{j,k}$$

By doing so the ensemble tries to find features $h_{i,k}$ that can be transformed by at least N different transformations to the desired action a since we have no restriction regarding the weights W, b:

$$h_{i,k+1} = \sigma(W_{i,k} \cdot h_{i,k} + b_{i,k})$$

Training a single neural net f_ϕ with parameters ϕ on D with some fixed hyperparameters Q corresponds to

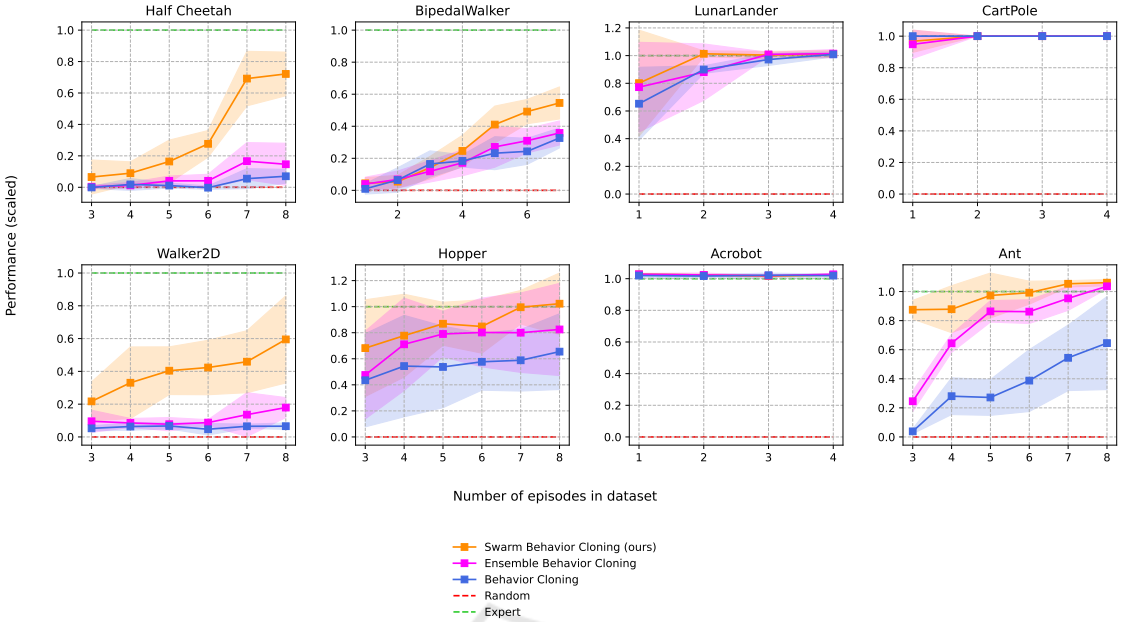

Figure 3: These plots show the mean normalized test returns of our approach *Swarm BC* and two baseline algorithms on eight different OpenAI Gym environments. The graphs represent the mean over 20 episodes and 5 seeds. The x-axes represent the number of expert episodes in the training data D. The results show a significant performance improvement in environments with larger observation- and action spaces.

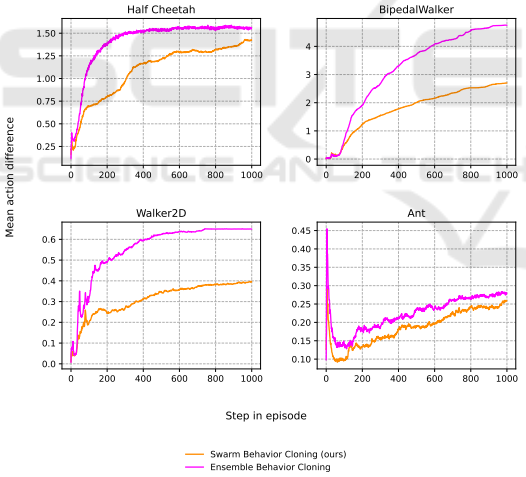

Figure 4: In this figure we are evaluating whether *Swarm BC* can reduce the *mean action difference* as defined in Definition 3.1, which is the difference between the N predicted actions $\{a_i = \pi_i(s)\}_{1 \le i \le N}$ of an ensemble E containing N policies. The results show that our approach does indeed reduce it but depending on the environment not always to the same extent. The x-axes in these plots represent the timestep in the test episodes and the y-axes represent the *mean action difference*. The graphs are the mean over 20 episodes and 5 seeds.

sampling from the probability density function (pdf):

$$\phi \sim p(\phi \mid D, Q)$$

Since D and Q are fixed we can shorten this expression to $p(\phi)$. The pdf for hidden features h_k for state s corresponds to the integral over all weights that produce the same feature activations:

$$p(h_k \mid s) = \int_{\phi} \mathbb{1}[\hat{f}^k_{\phi}(s) = h_k] \cdot p(\phi)$$

For a fixed state s we can shorten this expression to $p(h_k)$.

We now show that training an ensemble with similar feature activations corresponds to finding the *global mode* of the pdf $p(h_k)$.

If we are training a standard ensemble, we are sampling N times independently from the pdf $p(h_k)$. But

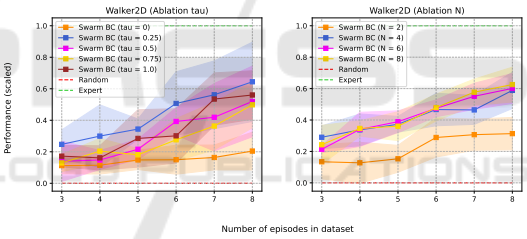

Figure 5: To examine the sensitivity of the two hyperparameters τ and N we did an ablation study. (Left) choosing τ too large or too small can reduce the test performance in terms of *mean episode return*. For Walker2D the best value was $\tau = 0.25$. Thus, we chose this value for all experiments in this paper. (Right) the conclusion of the ablation on the ensemble size N is that larger N are better, but this comes at the expense of longer runtime. For $N > 4$, however, the performance does not increase significantly anymore. Thus we chose $N = 4$ for all experiments in this paper.

the pdf for sampling N times the same hidden feature activation corresponds to:

$$p^N(h_k) = \frac{p(h_k)^N}{\int_\varphi p(\varphi)^N}$$

with

$$p(h_k)^N = \prod_{i=1}^{N} p(h_k)$$

For $N \to \infty$ the pdf $p^N(h_k)$ corresponds to the Dirac delta function being $p^N(h_k) = +\infty$ for the mode h_k^+ of $p(h_k)$ and 0 elsewhere (if there is just one mode). So we just need to sample once from $p^N(h_k)$ to get the mode h_k^+. Note that the *probability density* $p(h_k)$ is not the *probability* for sampling h_k. The probability for sampling a specific h_k is always 0. The probability can just be inferred by integrating the density over some space. We use as a space the hypercube T of length τ around activation h_k:

$$P_\tau^N(h_k) = \int_T p^N(h_k)$$

Proposition: For $\tau \to 0$ and $N \to \infty$, the probability for sampling the global mode with maximum error τ from $p^N(h_k)$ is $P_\tau^N(h_k^+) = 1$ if $p(h_k)$ is continuously differentiable, there is just a single mode h_k^+ and the activation space $H_k \ni h_k$ is a bounded hypercube.

Proof: By assumption we know that the activation space H_k is a bounded hypercube of edge length l and number of dimensions m. We further know that h_k^+ is the only mode of $p(h_k)$ and $p(h_k)$ is continuously differentiable (i.e. $p(h_k)$ is differentiable and its deviation is continuous which implies that there is a maximum absolute gradient).

For a given $\tau \in (0;\infty)$ we split the hypercube H_k in each dimension into $\lceil \frac{l}{\tau} \rceil$ parts. Thus H_k is split into $\lceil \frac{l}{\tau} \rceil^m$ sub-hypercubes. Each of them has maximum volume τ^m. If the mode h_k^+ lies on the edge between two sub-hypercubes we move all sub-hypercubes by $\tau/2$. So we need a maximum of $\lceil \frac{l}{\tau} + 1 \rceil^m$ sub-hypercubes to ensure that h_k^+ lies in exactly one sub-hypercube. We name it H^+ and all other sub-hypercubes are labeled $\{H_i^-\}_{1 \le i < \lceil \frac{l}{\tau}+1 \rceil^m}$.

We can calculate the mode for $p(h_k)$ in the remaining space of H_k without H^+ as follows:

$$h_k^\# = \underset{h_k \in H_k \setminus H^+}{\operatorname{argmax}} \; p(h_k)$$

Now we can calculate the upper bound for the probability mass in each H^- sub-hypercube by integrating the maximal possible density $p(h_k^\#)$ over the maximal possible volume τ^m:

$$P(H^-) \le \tau^m \cdot p(h_k^\#)$$

Since h_k^+ is the only mode, H_k is bounded and $p(h_k)$ is continuously differentiable there is a $\beta \in \mathbb{R}^+$ such that: $p(h_k^+) = p(h_k^\#) + \beta$. This implies that there is a sub-hypercube H^* inside of H^+ with edge length $\tilde{\tau} < \tau$ such that:

$$\forall\, h_k \in H^* : p(h_k) > p(h_k^\#) + \frac{1}{2} \cdot \Big(p(h_k^+) - p(h_k^\#) \Big)$$

Thus we can calculate a lower bound for the probability mass in H^+:

$$P(H^+) \ge \tilde{\tau}^m \cdot \left[p(h_k^\#) + \frac{1}{2} \cdot \Big(p(h_k^+) - p(h_k^\#) \Big) \right] =$$

$$= \frac{\tilde{\tau}^m}{2} \Big(p(h_k^\#) + p(h_k^+) \Big)$$

We can generalize both bounds to $P^N(H)$:

$$P^N(H^-) \le \frac{\tau^m \cdot p(h_k^\#)^N}{Z}$$

$$P^N(H^+) \ge \frac{\tilde{\tau}^m \cdot \Big(p(h_k^\#)^N + p(h_k^+)^N \Big)}{2Z}$$

with Z being the normalization constant:

$$Z = \int_{h_k \in H_k} p(h_k)^N$$

Let $\alpha \in [0;\infty)$ be a threshold. To proof that $P_\tau^N(h_k)$ approximates the global mode of $p(h_k)$ for $\tau \to 0$ and $N \to \infty$ we need to show that for any α and any $\tau \in (0;\infty)$ we can choose $N \in \mathbb{N}$ such that:

$$\frac{P^N(H^+)}{\lceil \frac{l}{\tau} + 1 \rceil^m \cdot P^N(H^-)} \ge \alpha$$

Because this would mean we can shift arbitrarily much probability mass into the sub-hypercube H^+ by increasing N. For that let's consider the ratio:

$$\frac{P^N(H^+)}{P^N(H^-)} \ge \frac{\tilde{\tau}^m \cdot p(h_k^\#)^N + \tilde{\tau}^m \cdot p(h_k^+)^N}{2\tau^m \cdot p(h_k^\#)^N} =$$

$$= \underbrace{\frac{\tilde{\tau}^m}{2\tau^m}}_{\equiv c} \cdot \left(1 + \left(\frac{p(h_k^+)}{p(h_k^\#)} \right)^N \right)$$

Since h_k^+ is the only mode the density $p(h_k^+)$ is larger than $p(h_k^\#)$. We can therefore see that the ratio gets arbitrarily large for $N \to \infty$. So we can choose N according to:

$$\frac{P^N(H^+)}{\lceil \frac{l}{\tau} + 1 \rceil^m \cdot P^N(H^-)} \ge \frac{c}{\lceil \frac{l}{\tau} + 1 \rceil^m} \cdot \left[1 + \left(\frac{p(h_k^+)}{p(h_k^\#)} \right)^N \right] \overset{!}{\ge} \alpha$$

$$\Rightarrow N = \left\lceil \frac{ln\Big(\frac{\lceil \frac{l}{\tau}+1 \rceil^m \cdot \alpha}{c} - 1 \Big)}{ln\Big(\frac{p(\phi^+)}{p(\phi^\#)} \Big)} \right\rceil \qquad \square$$

8 CONCLUSION

Behavior Cloning (BC) is a crucial method within Imitation Learning, enabling agents to be trained safely using a dataset of pre-collected state-action pairs provided by an expert. However, when applied in an ensemble framework, BC can suffer from the issue of increasing action differences, particularly in states that are underrepresented in the training data $D = (s_t, a_t)_t$. These large *mean action differences* among the ensemble policies can lead to suboptimal aggregated actions, which degrade the overall performance of the agent.

In this paper, we proposed *Swarm Behavior Cloning* (Swarm BC) to address this challenge. By fostering greater alignment among the policies while preserving the diversity of their computations, our approach encourages the ensemble to learn more similar hidden feature representations. This adjustment effectively reduces action prediction divergence, allowing the ensemble to retain its inherent strengths—such as robustness and varied decision-making—while producing more consistent and reliable actions.

We evaluated Swarm BC across eight diverse OpenAI Gym environments, demonstrating that it effectively reduces *mean action differences* and significantly improves the agent's test performance, measured by episode returns.

Finally, we provided a theoretical analysis showing that our method approximates the hidden feature activations with the highest probability density, effectively learning the global mode $h_k^* = \underset{h_k}{\text{argmax}}; p(h_k; |; D)$ based on the training data D. This theoretical insight further supports the practical performance gains observed in our experiments.

REFERENCES

Arora, S. and Doshi, P. (2021). A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500.

Bain, M. and Sammut, C. (1995). A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Brantley, K., Sun, W., and Henaff, M. (2019). Disagreement-regularized imitation learning. In *International Conference on Learning Representations*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Codevilla, F., Santana, E., López, A. M., and Gaidon, A. (2019). Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338.

Dietterich, T. G. et al. (2002). Ensemble learning. *The handbook of brain theory and neural networks*, 2(1):110–125.

Dong, X., Yu, Z., Cao, W., Shi, Y., and Ma, Q. (2020). A survey on ensemble learning. *Frontiers of Computer Science*, 14:241–258.

Eschmann, J. (2021). Reward function design in reinforcement learning. *Reinforcement Learning Algorithms: Analysis and Applications*, pages 25–33.

Finn, C., Levine, S., and Abbeel, P. (2016). Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR.

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. (2022). Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR.

Giusti, A., Guzzi, J., Cireşan, D. C., He, F.-L., Rodríguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., et al. (2015). A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667.

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. *Advances in neural information processing systems*, 29.

Hussein, M., Crowe, B., Petrik, M., and Begum, M. (2021). Robust maximum entropy behavior cloning. *arXiv preprint arXiv:2101.01251*.

Knox, W. B., Allievi, A., Banzhaf, H., Schmitt, F., and Stone, P. (2023). Reward (mis) design for autonomous driving. *Artificial Intelligence*, 316:103829.

Ng, A. Y., Russell, S., et al. (2000). Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2.

Nüßlein, J., Illium, S., Müller, R., Gabor, T., and Linnhoff-Popien, C. (2022). Case-based inverse reinforcement learning using temporal coherence. In *International Conference on Case-Based Reasoning*, pages 304–317. Springer.

Phan, T., Ritz, F., Altmann, P., Zorn, M., Nüßlein, J., Kölle, M., Gabor, T., and Linnhoff-Popien, C. (2023). Attention-based recurrence for multi-agent reinforcement learning under stochastic partial observability. In *International Conference on Machine Learning*, pages 27840–27853. PMLR.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8.

Sagi, O. and Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249.

Shafiullah, N. M., Cui, Z., Altanzaya, A. A., and Pinto, L. (2022). Behavior transformers: Cloning *k* modes with one stone. *Advances in neural information processing systems*, 35:22955–22968.

Smith, M., Maystre, L., Dai, Z., and Ciosek, K. (2023). A strong baseline for batch imitation learning. *arXiv preprint arXiv:2302.02788*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Torabi, F., Warnell, G., and Stone, P. (2018). Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*.

Torabi, F., Warnell, G., and Stone, P. (2019a). Adversarial imitation learning from state-only demonstrations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2229–2231.

Torabi, F., Warnell, G., and Stone, P. (2019b). Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*.

Webb, G. I. and Zheng, Z. (2004). Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques. *IEEE Transactions on Knowledge and Data Engineering*, 16(8):980–991.

Wen, C., Lin, J., Darrell, T., Jayaraman, D., and Gao, Y. (2020). Fighting copycat agents in behavioral cloning from observation histories. *Advances in Neural Information Processing Systems*, 33:2564–2575.

Yang, Z., Ren, K., Luo, X., Liu, M., Liu, W., Bian, J., Zhang, W., and Li, D. (2022). Towards applicable reinforcement learning: Improving the generalization and sample efficiency with policy ensemble. *arXiv preprint arXiv:2205.09284*.

Zheng, B., Verma, S., Zhou, J., Tsang, I., and Chen, F. (2021). Imitation learning: Progress, taxonomies and challenges. *arXiv preprint arXiv:2106.12177*.

Zhifei, S. and Meng Joo, E. (2012). A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, 5(3):293–311.

Zhou, Z.-H. and Zhou, Z.-H. (2021). *Ensemble learning*. Springer.