future internet

MDPI

Article

UAV Control Method Combining Reptile Meta-Reinforcement Learning and Generative Adversarial Imitation Learning

Shui Jiang [1], Yanning Ge [1], Xu Yang [2,*], Wencheng Yang [3] and Hui Cui [4]

[1] College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350007, China; jiangshui87176@gmail.com (S.J.); ged_mail@163.com (Y.G.)
[2] College of Computer and Control Engineering, Minjiang University, Fuzhou 350108, China;
[3] School of Mathematics, Physics and Computing, University of Southern Queensland, Darling Heights, QLD 4350, Australia; wencheng.yang@unisq.edu.au
[4] Department of Software Systems & Cybersecurity, Monash University, Melbourne, VIC 3800, Australia; hui.cui@monash.edu
* Correspondence: xu.yang@mju.edu.cn

Abstract: Reinforcement learning (RL) is pivotal in empowering Unmanned Aerial Vehicles (UAVs) to navigate and make decisions efficiently and intelligently within complex and dynamic surroundings. Despite its significance, RL is hampered by inherent limitations such as low sample efficiency, restricted generalization capabilities, and a heavy reliance on the intricacies of reward function design. These challenges often render single-method RL approaches inadequate, particularly in the context of UAV operations where high costs and safety risks in real-world applications cannot be overlooked. To address these issues, this paper introduces a novel RL framework that synergistically integrates meta-learning and imitation learning. By leveraging the Reptile algorithm from meta-learning and Generative Adversarial Imitation Learning (GAIL), coupled with state normalization techniques for processing state data, this framework significantly enhances the model's adaptability. It achieves this by identifying and leveraging commonalities across various tasks, allowing for swift adaptation to new challenges without the need for complex reward function designs. To ascertain the efficacy of this integrated approach, we conducted simulation experiments within both two-dimensional environments. The empirical results clearly indicate that our GAIL-enhanced Reptile method surpasses conventional single-method RL algorithms in terms of training efficiency. This evidence underscores the potential of combining meta-learning and imitation learning to surmount the traditional barriers faced by reinforcement learning in UAV trajectory planning and decision-making processes.

Keywords: unmanned aerial vehicles (UAVs); meta-reinforcement learning; generative adversarial imitation learning

Citation: Jiang, S.; Ge, Y.; Yang, X.; Yang, W.; Cui, H. UAV Control Method Combining Reptile Meta-Reinforcement Learning and Generative Adversarial Imitation Learning. *Future Internet* **2024**, *16*, 105. https://doi.org/10.3390/fi16030105

Academic Editor: Michael Sheng

Received: 6 February 2024
Revised: 1 March 2024
Accepted: 15 March 2024
Published: 20 March 2024

1. Introduction

Reinforcement learning (RL) [1], a subset of machine learning, is predicated on the interaction between intelligent agents and their environments to facilitate decision making. This method has seen considerable advancements in areas such as systems control and gaming, and it holds significant promise for UAV trajectory planning and control [2,3]. However, the intricacies of UAV control and the dynamic nature of their operational environments necessitate rapid and precise decision making from these agents, challenging the capabilities of traditional RL methods.

A primary concern in this realm is sample efficiency, especially critical in UAV trajectory planning and control [4]. The substantial costs and safety risks associated with real-world UAV flights render the extensive sample data and environmental interactions required by conventional RL frameworks impractical for actual deployment,

as depicted in Figure 1. Thus, developing methods to enhance sample efficiency is of paramount importance.

Moreover, the generalization capabilities of RL models in the context of UAV trajectory planning and control are often found wanting. Due to the variability and complexity of real-world environments, models that perform well in training scenarios may experience significant performance declines when introduced to new, unfamiliar settings. This drop in performance is often linked to the models' strategies being too tailored to specific training environments, lacking the flexibility needed to adapt to new situations.

Additionally, traditional RL approaches are heavily reliant on the design of reward functions. Minor alterations in these functions can lead to vastly different flight strategies, making the task of crafting reward functions that accurately reflect the complex objectives of UAV missions challenging [5,6].

Figure 1. Illustration of a drone executing tasks in a simulated real-world environment.

To tackle these challenges, the research community is turning towards innovative approaches such as meta-learning and imitation learning [7–9]. Meta-learning, often described as 'learning to learn,' empowers models with the ability to swiftly acclimate to new tasks by discerning and assimilating the underlying similarities across diverse tasks. This approach fundamentally aims to equip models with the capability to identify and abstract the core attributes of tasks, thereby leveraging accumulated knowledge to adeptly modify strategies when confronted with novel scenarios. Such a methodology enhances models' adaptability and generalization skills, making them more versatile and effective in dynamic environments [10,11].

Imitation learning serves as a valuable complement to meta-learning, focusing on teaching agents to acquire strategies through the observation and replication of expert behaviors, instead of relying solely on reward signals. This learning paradigm is inspired by the human method of skill acquisition, which typically involves emulating the actions of individuals who have already achieved proficiency in a particular domain [12,13]. By adopting this approach, agents are able to circumvent the intricate process of reward function formulation, allowing them to directly assimilate the decision-making processes and critical maneuvers employed by experts in complex scenarios. Consequently, agents can attain or closely approximate the efficiency of experts in task execution, even in scenarios devoid of explicit reward indicators [14,15].

Models honed through meta-learning exhibit the remarkable ability to swiftly adjust to a variety of imitation learning tasks. Conversely, the tangible behavioral paradigms provided by imitation learning serve as invaluable practical guides for meta-learning. This synergistic blend not only mitigates the challenges associated with low sample efficiency and limited generalization but also equips models with the capacity to master intricate decision-making processes without the need for explicit reward signals.

In response to these challenges, we introduce an innovative meta-reinforcement learning algorithm that leverages the principles of Generative Adversarial Imitation Learning (GAIL) [16]. This algorithm harmoniously integrates the strengths of both meta-learning and imitation learning, tailoring its design to meet the specific demands of UAV trajectory planning. By augmenting the UAVs' perception of the state space with normalized methods and residual connections, we ensure consistent performance stability across diverse and challenging environments. These advanced techniques empower our algorithm to excel in UAV trajectory planning tasks, markedly enhancing both the sample efficiency and the ability to generalize across different scenarios.

In this paper, Section 2 introduces some relevant work in the field of drones, Section 3 provides a detailed explanation of meta-reinforcement learning and Generative Adversarial Imitation Learning, and Section 4 presents the specific methodology of this paper. Section 5 covers the experimental part.

2. Related Work

Researchers have actively adopted reinforcement learning methods to guide UAVs in autonomous decision making, which includes the application of basic control systems, path planning, obstacle avoidance, and advanced control techniques.

Application of Basic Control Systems: The fundamental aspects of UAV technology encompass attitude adjustment and basic flight control, with a key focus on ensuring the stability and controllability of UAVs under various flight conditions. Koch et al. [17] utilized Deep Reinforcement Learning (DRL) to optimize UAV attitude control, especially under extreme conditions, such as strong winds and turbulence. Their research demonstrates the potential application of DRL in UAV control systems.

Path Planning and Obstacle Avoidance: Path planning is a critical component of UAV navigation, involving the efficient selection of routes and avoidance of obstacles. Zhao et al. [18] employed the Q-learning algorithm to improve UAV path planning and obstacle avoidance capabilities, focusing on enhancing autonomous navigation technology, and confirmed the effectiveness of the algorithm in dealing with static and dynamic obstacles. Pham et al. [19] implemented autonomous navigation of UAVs using RL and the TEXPLORE algorithm, advancing sophisticated UAV control. Their research demonstrated the efficacy of this algorithm in navigating and avoiding obstacles in unknown environments. He et al. [20] improved the autonomous navigation capabilities of UAVs in unknown environments, particularly in path planning and obstacle avoidance, using DRL based on the Twin Delayed DDPG (TD3) algorithm. However, these methods also have drawbacks, requiring continuous trial-and-error training with low sample efficiency, and they struggle to generalize to new environments without prior examples.

Advanced Control: Advanced control in UAV technology not only includes basic flight operations but also encompasses the ability to execute complex tasks, such as formation control and advanced navigation techniques. Yang et al. [21] significantly enhanced the formation control capabilities of UAVs using GAIL. Their method, by imitating peer actions and combining historical data, effectively improved the recognition of real-world environmental states. Additionally, Wang et al. [22] significantly enhanced the navigation and control efficiency of racing drones by combining imitation learning with modular strategies and utilizing Convolutional Neural Networks (CNNs). Hu et al. [23] employed meta-reinforcement learning to improve trajectory design for energy-constrained UAVs in dynamic network environments, focusing on enhancing the adaptability and response speed of UAVs as mobile base stations in unfamiliar environments.

While the applications of meta-reinforcement learning and imitation learning have achieved significant results in improving the efficiency and adaptability of UAV training, they each have their limitations. Meta-reinforcement learning (Meta-RL) often requires the manual design of reward functions to describe new tasks, which can be both cumbersome and impractical for non-experts. A more natural way to describe tasks is through demonstration, as performed in imitation learning (IL). However, IL is limited in its abil-

ity to continue improving policies and, compared to reinforcement learning methods, is constrained by the similarity between new tasks and the training dataset, leading to deficiencies in generalization capabilities [24]. Therefore, this paper is dedicated to combining meta-reinforcement learning and imitation learning to address the aforementioned issues.Table 1 compares our work with previous studies.

Table 1. Comparison of related work.

Work	Koch et al. [17]	Zhao et al. [18]	Pham et al. [19]	He et al. [20]	Ours
Learning Method	DRL	Q-learning	RL + TEXPLORE	DRL (TD3)	Meta-RL + IL
Key Contribution	UAV attitude stability in extreme conditions	Efficient obstacle avoidance	Navigation and obstacle avoidance in unknown environments	Improved navigation in unknown environments	Combined adaptability and generalization
Drawbacks	Limited to attitude control	Low sample efficiency, poor generalization	Low sample efficiency	Low sample efficiency, poor generalization	The training time is relatively long

3. Preliminary

RL is typically conceptualized within the framework of a Markov Decision Process (MDP) [25], formalized as $M \equiv (S, A, P, R, \gamma)$. Within this framework, S delineates the set of possible states, while A encapsulates the available actions. The function $P(s'|s, a)$ defines the likelihood of transitioning to a subsequent state s' upon the execution of an action a in the current state s. The reward function $R(s, a)$ quantifies the immediate benefit received upon performing action a in state s. Lastly, the discount factor γ balances the agent's valuation of immediate rewards against future returns, with values approaching 0 indicating a preference for immediate rewards and values nearing 1 signifying a longer-term outlook.

3.1. Meta-RL

Unlike conventional reinforcement learning approaches that are tailored for singular tasks, meta-reinforcement learning (Meta-RL) aspires to craft policies that are capable of rapid adaptation or optimization when faced with a sparse influx of data from new tasks [26]. The prevailing methodologies in Meta-RL typically presuppose a task framework where all constituent subtasks are unified by identical state and action spaces. The distinctions among these tasks often emerge through variations in the reward functions, the probabilities of state transitions, or the distributions of initial states. For instance, in the context of robotic object-grasping experiments, diverse tasks may entail the manipulation of different objects within a consistent physical environment. Despite the variability in tasks, the robot's action space (such as arm movements) and state space (encompassing joint angles and object locations) remain unchanged. Tasks are derived from the distribution $\mathcal{T}_i \sim p(\mathcal{T})$, with each task $\mathcal{T}_i$ characterized by $(\mathcal{S}, \mathcal{A}, p_i, r_i, \gamma)$. Given the shared framework of state and action spaces across tasks, the task distribution $p(\mathcal{T})$ is delineated by the reward function and the initial state distribution $p(s_0)$, thus formulated as $p(\mathcal{T}) = p(R) \times p(s_0)$.

In the meta-training phase, tasks are drawn from the task distribution $p(\mathcal{T})$ to facilitate the training process. The fundamental goal during this stage is to equip the model with the ability to swiftly acclimate to novel tasks. Following this, in the meta-testing phase, the model is presented with a series of new tasks that, while related, are distinct from those encountered during training. This phase is critical for evaluating the model's capacity for adaptability and its proficiency in assimilating new tasks. Throughout the meta-training period, the agent utilizes specific meta-learning parameters or employs update mechanisms designed to foster this rapid adaptation, which is then put to the test in the subsequent meta-testing phase. The intricacies of the meta-training process are

illustrated in Figure 2, highlighting the structured approach to achieving adaptability and flexibility in task learning.

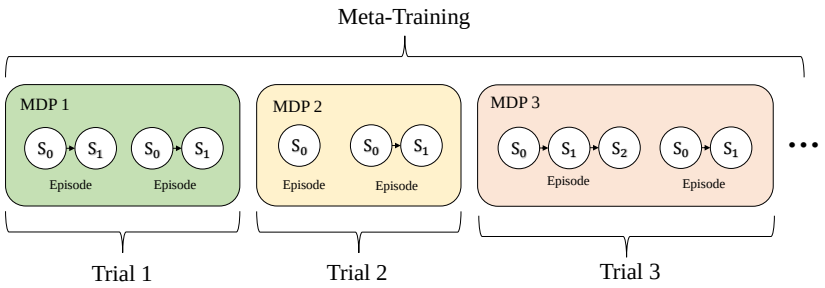

Figure 2. Meta-training process.

The process of meta-reinforcement learning can be subdivided into two critical cycles: the inner loop and the outer loop.

1. **Inner Loop:**
   - During the inner loop phase, the agent leverages meta-knowledge provided by the outer loop, which is crucial for application.
   - The main goal of this phase is to efficiently use meta-knowledge for quick adaptation to new tasks or environments.
   - The effectiveness of the meta-knowledge is gauged by the agent's performance in this segment, serving as a vital evaluation criterion.
2. **Outer Loop:**
   - The outer loop's primary role is to evolve and fine-tune the meta-knowledge itself.
   - This stage involves a thorough examination of similarities and differences across tasks to enhance the meta-knowledge that facilitates learning.
   - It also updates the meta-knowledge based on feedback from the inner loop, ensuring its efficacy across diverse tasks.

The concept of 'meta-knowledge' encapsulates the extensive knowledge framework an agent amasses while navigating through diverse tasks or challenges, serving as a pivotal element in its ability to swiftly adapt to novel tasks or environments [27]. The primary aim of meta-reinforcement learning is to develop reinforcement learning algorithms endowed with the agility to adapt to new situations, with meta-knowledge at the heart of this endeavor. This knowledge domain includes, but is not limited to, learning methodologies, task-specific nuances, accumulated experiences, and environmental dynamics. The intrinsic value of meta-knowledge is rooted in its capacity for generalization—the extent to which insights gained from one task can be leveraged to enhance performance in other, ostensibly disparate tasks.

In the meta-reinforcement learning framework (shown in Figure 3), the meta-knowledge accumulated in the outer loop significantly guides the inner loop, helping it adapt more quickly to new tasks. The essence of this learning mode lies in how effectively meta-knowledge can be extracted and utilized from multiple tasks, and how it can be adjusted according to the evolution of tasks, thus enabling the agent to rapidly adapt when facing new tasks. This approach is particularly suitable for scenarios in which the task environment frequently changes or the agent needs to handle a variety of different tasks, enhancing the model's generalization capabilities.

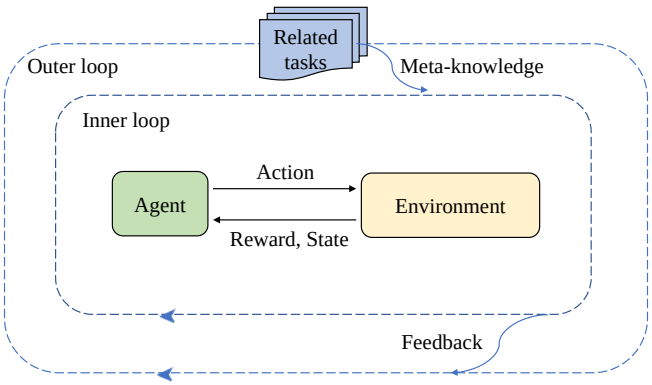

Figure 3. The framework of Meta-RL.

3.2. GAIL

GAIL represents a sophisticated approach that synergizes the principles of Generative Adversarial Networks (GANs) with IL. This technique is particularly efficacious in complex settings where articulating a precise reward function poses significant challenges. GAIL acquires the capability to execute specific tasks by mimicking the behaviors of experts, circumventing the need for explicit reward signals in the learning process. Within this paradigm, the expert's strategy is denoted by π_E, and the learner's strategy is signified by π. The formulation for the learner's strategy π encompasses the generation of expected trajectories, illustrating the process of learning through observation and emulation of expert actions.

$$\mathbb{E}_{\pi}[f(s,a)] \triangleq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t f(s_t,a_t)\right] \tag{1}$$

Here, $s_0 \sim \rho_0$, $a_t \sim \pi(a_t|s_t)$, and f is an arbitrary function.

At the heart of GAIL lies the emulation of the expert policy's state–action occupancy measure. This measure is a critical probability distribution that captures the frequency of an agent executing certain actions in given states under a specific policy. Unlike behavior cloning, which directly replicates expert trajectories without necessitating environmental interaction and consequently faces the challenge of compounding errors—where slight inaccuracies in imitation accumulate over time—GAIL necessitates active engagement with the environment. This interaction is pivotal for GAIL as it enables the learning policy to refine its actions based on the dynamic feedback from the environment, thereby mitigating the risk of error propagation inherent in behavior cloning.

The GAIL algorithm includes a discriminator D and a policy π, where policy π acts as the generator in the Generative Adversarial Network. The discriminator D takes state–action pairs (s,a) as input and outputs a probability between 0 and 1, indicating how likely it is that the pair (s,a) originates from the agent's policy rather than the expert's policy. The goal of the discriminator is to make the output for expert data as close to 0 as possible and the imitator's output close to 1, thus clearly distinguishing between the two. Based on the above, we can define the loss function for the discriminator D as:

$$L(\phi) = -E_{\rho_\pi}[\log D_\phi(s,a)] - E_{\rho_E}[\log(1 - D_\phi(s,a))] \tag{2}$$

where ϕ represents the parameters of the discriminator. The core idea of this algorithm is to make the trajectories generated by the policy π be misidentified by the discriminator as coming from the expert. Using this approach, we do not need to preset a specific reward function; instead, we can define the reward $r(s,a)$ as $-\log(D(s,a))$. In this setting, if the discriminator D judges the state–action pair (s,a) to be extremely similar to the expert's behavior, the value of $D(s,a)$ is close to 1, and then $-\log(D(s,a))$ will be a smaller negative value, equivalent to a higher reward. Conversely, if the value of $D(s,a)$ is close to 0, implying that the discriminator considers the state–action pair unlike the expert's

behavior, then the reward $-\log(D(s,a))$ becomes a larger negative value, representing a lower reward.

Building on the theoretical framework described, we embarked on a series of comparative analyses involving three distinct algorithms, applied to the Cartpole task. The initial step entailed the generation of expert data, for which we opted for the Proximal Policy Optimization (PPO) algorithm, given the relative simplicity of the Cartpole task. From the generated trajectory, we sampled 30 state–action pairs (s,a). Our comparative analysis revealed that the Generative Adversarial Imitation Learning (GAIL) algorithm required merely 80 training epochs to attain the maximum reward threshold. In contrast, the traditional Deep Q-Network (DQN) algorithm necessitated a more extensive trial-and-error approach, taking approximately 200 epochs to reach a comparable performance level. Notably, the training trajectory of the model underpinned by GAIL exhibited greater stability than its DQN counterpart. The outcomes of these experimental endeavors are encapsulated in Figure 4.

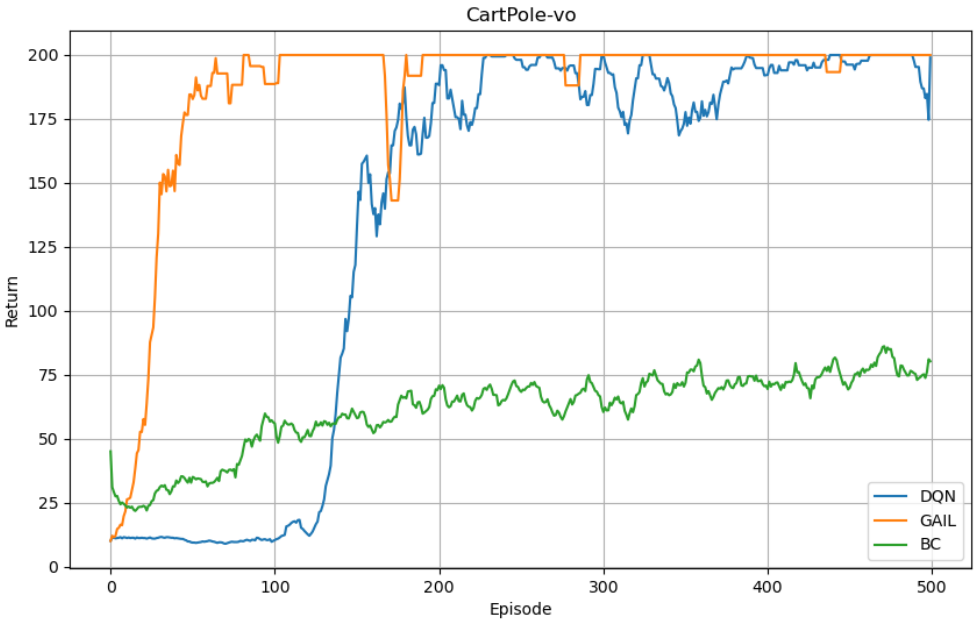

Figure 4. The comparison of three algorithms under the CartPole task.

Nevertheless, given the constrained dataset of merely 30 data points, the performance of the Behavior Cloning (BC) algorithm was found to be subpar in comparison to the traditional DQN algorithm, failing to deduce the optimal strategy. This deficiency was predominantly attributed to the model's propensity for overfitting when faced with such a scant data volume. In essence, the GAIL algorithm demonstrated its efficacy by propelling the model to attain superior performance levels more expediently, utilizing fewer data points and training epochs. Consequently, this underscores the potential merit of integrating the GAIL algorithm within the meta-reinforcement learning framework, especially for UAV tasks where articulating a precise reward function is challenging.

4. Methods

Building upon the experiments previously mentioned, we introduced the Reptile algorithm [28] as our chosen method for meta-learning. Reptile is a gradient-based meta-learning algorithm aimed at identifying a set of initial parameters that enable rapid adaptation to new tasks with a minimal number of gradient updates. Its main strategy involves applying gradient updates across various tasks, with the goal of achieving a set of initial parameters that ensures good initial performance across a range of tasks. This approach allows the Reptile algorithm to significantly enhance the model's adaptability and flexibility when encountering new tasks. The structure of this methodology is depicted in Figure 5.

The Reptile algorithm offers a computational advantage over the MAML approach [29], particularly due to its avoidance of second-order derivative computations. MAML seeks to establish a foundational model that, through minor parameter modifications across a spectrum of tasks, can adeptly adapt to new challenges. This methodology necessitates the refinement of model parameters for one task, followed by the application of these tweaked parameters to subsequent tasks, with additional adjustments required for each new task. Such a process demands intricate calculations of second-order derivatives, leading to considerable computational demands and extended processing times.

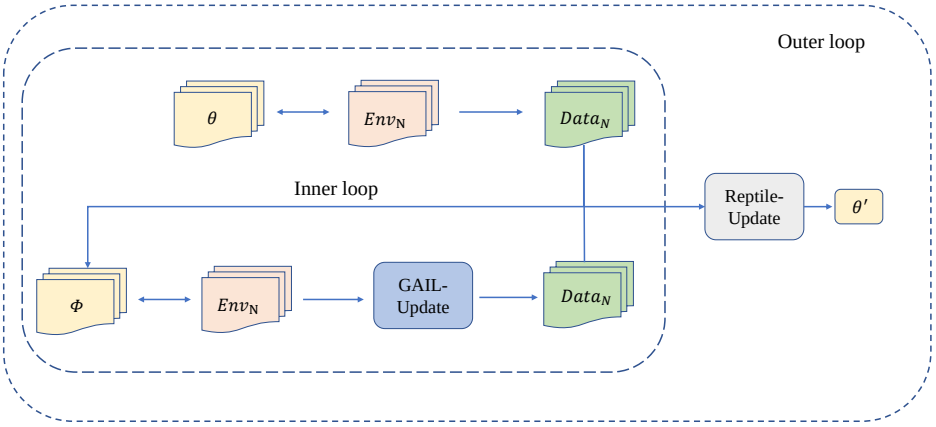

Figure 5. This figure depicts the global architecture of the model, demonstrating how to optimize the parameters of the strategy network through a multi-step iterative process, enabling rapid adaptation to various tasks. The main steps include: (1) Outer Loop: This phase is responsible for updating the meta-parameters θ. Throughout a series of training cycles, these meta-parameters are used to generate initial strategy network parameters adapted to various tasks. (2) Inner Loop: For each subtask, starting from the meta-parameters θ, data are generated through interaction with the environment, and the parameters Φ of the strategy network are iteratively updated using the GAIL algorithm. (3) Parameter Update: After the inner loop training, the task-specific parameters Φ obtained are used to update the meta-parameters θ, thus achieving rapid adaptation to new tasks.

Reptile, on the other hand, adopts a strategy of executing iterative parameter updates within individual tasks, eschewing MAML's approach of multiple parameter fine-tunings across various tasks. Reptile's primary aim is to extract more generalizable features by repetitively adjusting parameters over a collection of tasks, thereby facilitating swifter adaptation to novel tasks. Hence, while MAML's strength lies in crafting an initial model that can be fine-tuned for diverse tasks, Reptile streamlines the adaptation process to new tasks through a restricted series of iterative parameter modifications across tasks.

This diagram (Figure 6) illustrates how a model updates its parameters progressively after training on multiple tasks, starting from its initial parameters, to enhance its generalization ability for new tasks. The key elements include:

- **Initial Model Parameters** ϕ^0: Represent the model's parameter settings before training begins.
- **Paths in Different Colors**: Illustrate the optimization process of the model on various training tasks. For instance, a blue path indicates training on task n, while a light yellow path signifies training on task m.
- **Arrows**: Indicate the direction of the model parameters' movement in the parameter space during training.
- **Model Parameters** $\phi^1, \phi^2, \ldots$: Denote the model parameters after a series of training steps.
- **Final Model Parameters** $\hat{\theta}_m, \hat{\theta}_n$: Represent the final state of the model parameters on their respective tasks.

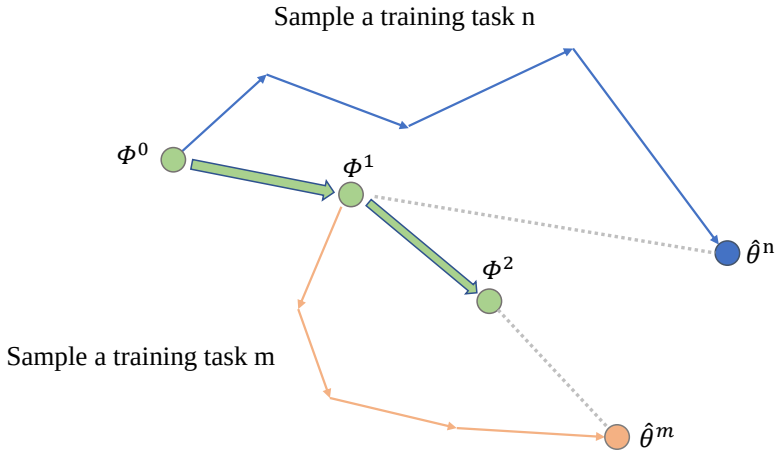

Figure 6. Principle of Reptile.

The essence of the Reptile algorithm lies in iteratively updating the model parameters to swiftly adapt to new tasks. This involves executing multiple steps of gradient descent on each task and then updating the global parameters.

Assume the model parameters are θ, and for a randomly selected task T_i, the model's loss function on this task is $L_{T_i}(\theta)$. The parameter update rule is described as follows:

Step 1—In-task Updates: For each task T_i, initialize the temporary parameters θ' to the global parameters θ. Then, perform K iterations of gradient descent updates, expressed as:

$$\theta' = \theta' - \beta \nabla_{\theta'} L_{T_i}(\theta'),$$

where β is the in-task learning rate.

Step 2—Global Updates: After the K gradient updates for task T_i, the global parameters θ are updated based on the average gradient changes across all tasks:

$$\theta = \theta + \alpha(\theta' - \theta),$$

where α is the global learning rate.

The specific algorithm is as Algorithm 1:

Algorithm 1 Reptile Algorithm [28].

for each iteration **do**
  Sample task T_i
  $\theta' = \theta$
  for $k = 1, 2, \ldots, K$ **do**
    $\theta' = \theta' - \beta \nabla_{\theta'} L_{T_i}(\theta')$
  end for
  $\theta = \theta + \alpha(\theta' - \theta)$
end for

In the Reptile algorithm, the objective of each iteration is to minimize the loss function $L_{T_i}(\theta)$ of task T_i. The algorithm achieves this by performing K steps of gradient descent on task T_i, and then uses these updates to adjust the global parameters θ.

In-task updates aim to quickly adapt to the current task T_i, while the global updates seek to find a good starting point for the parameters so that for new tasks, effective learning can be achieved with only a few steps of gradient updates.

Based on the above, we combine GAIL with the meta-reinforcement learning Reptile algorithm. The core idea is to use GAIL for imitation learning when training subtasks and then use these training results to update the meta-parameters. This process can be divided into the following key steps:

Step 1: Selection and preparation of subtasks—Choose multiple subtasks from the task distribution and prepare expert demonstration data for each task.

Step 2: Training subtasks using GAIL—For each subtask, train using the GAIL framework.

- Generator (i.e., policy network) training: Train the policy network by attempting to imitate expert demonstrations.
- Discriminator training: The discriminator learns to distinguish between the behaviors generated by the policy network and the expert demonstrations.

Step 3: Policy network update—After completing the GAIL training for each subtask, the policy network is improved.

Step 4: Updating meta-parameters—After completing training on all selected subtasks, update the meta-parameters using the logic of the Reptile algorithm. This usually involves averaging or integrating the parameter changes obtained across multiple tasks in some form.

Step 5: Repeat the process—Repeat the above steps until the model performance reaches the expected target or until the predetermined training cycles are completed.

Additionally, during the training process, we implemented normalization of the states with the following steps:

(1) At the start of training, we first store historical training data in the experience replay buffer.

(2) Upon completion of training, we calculate the mean and variance of each dimension of the state in the replay buffer, concluding the entire training process.

(3) Before commencing the next training session, all states inputted into the network are normalized.

Through this methodology, GAIL enhances the policy network's ability to closely replicate expert behaviors across a variety of subtasks. Following this, the Reptile algorithm amalgamates the insights gained to refine the meta-parameters, thus facilitating swift adaptation to new tasks that bear resemblance to those previously encountered. The integration of GAIL with Reptile is designed to capitalize on GAIL's proficiency in imitation learning while harnessing Reptile's meta-learning capabilities to boost the learning process's overall efficiency and adaptability. This combined approach is delineated as Algorithm 2:

Algorithm 2 Combined GAIL and Reptile meta-reinforcement learning algorithm

Require: Task distribution D, expert demonstrations set E, learning rate α, meta-learning rate β, training epochs T
Ensure: Optimized meta-parameters θ
Initialize meta-parameters θ
for $t = 1$ **to** T **do**
  Sample a set of subtasks S from D
  Initialize task-specific parameters $\Phi = \theta$
  for each subtask $s \in S$ **do**
    Initialize policy network π_s with parameters Φ
    Initialize discriminator D_s
    Obtain expert demonstrations E_s from E for task s
    repeat
      Train π_s by imitating E_s
      Train D_s using behaviors generated by π_s and E_s
      Update π_s to maximize the score of D_s for the imitated behavior
    until π_s converges
    Update Φ to optimized parameters of π_s
  end for
  Update meta-parameters $\theta = \theta + \beta \cdot (\Phi - \theta)$
end for
return θ

5. Experiments

Given the complexity of drone control in three-dimensional space, we opted to conduct experimental tests on a two-dimensional plane. We utilized the experimental environment described in the literature [30], which consists of three marked square areas. For detailed information about the specific experimental setup, please visit the following website: https://github.com/marek-robak/Drone-2d-custom-gym-env-for-reinforcement-learning (accessed on 11 December 2023). The smallest square represents the initial space where the drone may appear, while the larger square defines the area where the target point may be located. Here, the agent needs to learn to navigate the drone to the target point. The largest square represents the boundaries of the drone's flyable space, and if the drone flies out of this range, the current round ends.

The drone model utilized in our study is constructed from a rigid body that comprises three sections, two of which function as motors. These motors are capable of producing lift, as depicted by the elongated red lines, while the grey lines serve as a reference scale. At the initiation of each trial, the drone is propelled in an arbitrary direction, with the motors temporarily disabled for a predetermined sequence of time steps. This arrangement aims to forge a scenario in which the reinforcement learning agent is tasked with regaining control and stabilizing the drone during its descent. The portion of the trajectory delineated in red illustrates the phase of flight where the agent lacks control. Additionally, the simulation of the drone's environment is conducted at a temporal resolution of 60 frames per second, as illustrated in Figure 7.

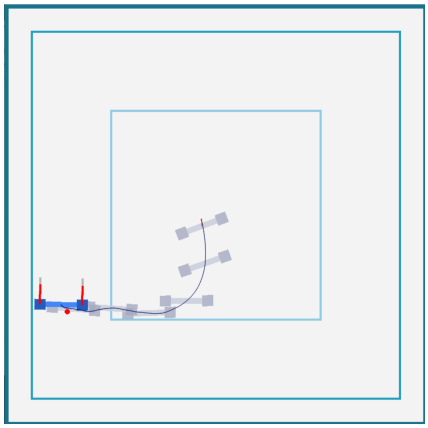

Figure 7. Two-dimensional experimental environment space.

In this experiment, the hardware setup used was as follows:

- CPU: Manufacturer: Mechrevo,Country: China,12th Gen Intel(R) Core(TM) i7-12650H, with a clock speed of 2300 Mhz, with 10 cores and 16 logical processors.
- GPU: Manufacturer: Mechrevo,Country: China,NVIDIA GeForce RTX 4050.

For detailed configurations of the hyperparameters, refer to Table 2.

In our study, we strategically simplified the drone's operational context by confining its movements to the xoz plane, thereby significantly mitigating the complexities associated with navigating a fully three-dimensional space. Within this two-dimensional framework, the drone utilizes its advanced control system to navigate from an initial point to a specified target location. This experimental setup enables a concentrated examination of the drone's control and navigational efficacy on a plane, alongside an assessment of its stability and task execution proficiency. This methodological choice not only facilitated a comprehensive analysis of the drone's control system efficiency but also established a foundational basis for future explorations into three-dimensional spatial navigation, yielding crucial data and insights. Figure 8 simulates the drone's situation in three-dimensional space:

Table 2. Hyperparameter configuration for the experiment.

Hyperparameter	Value	Description
Learning Rate (α)	1×10^{-3}	Learning rate for policy updates.
Meta-Learning Rate (β)	0.001	Learning rate for meta updates in Reptile.
Training Epochs (T)	2500	Total number of training epochs.
Task Sample Number (N)	5, 10, 15, 20	Number of tasks sampled per meta-update.
Gradient Update Steps (K)	5	Number of gradient update steps per task.
GAIL Discriminator Architecture	2 layers, 64 neurons	Two-layer network for the GAIL discriminator, each layer with 64 neurons.
GAIL Discriminator Learning Rate	3×10^{-4}	Learning rate for the GAIL discriminator.
Expert Demonstration Dataset Size	1000 pairs	Size of the expert demonstration dataset (state–action pairs).
Policy Network Architecture	2 layers, 64 neurons	Two-layer network for the policy, each layer with 64 neurons.
Policy Network Learning Rate	3×10^{-4}	Learning rate for the policy network.
Reward Scaling Factor	0.01	Scaling factor for rewards in GAIL.
Exploration Strategy	Epsilon-greedy	Epsilon-greedy strategy with initial epsilon of 1, decaying to 0.1.
Batch Size	64	Batch size for training updates.
Regularization Parameter	1×10^{-3}	L2 regularization coefficient.

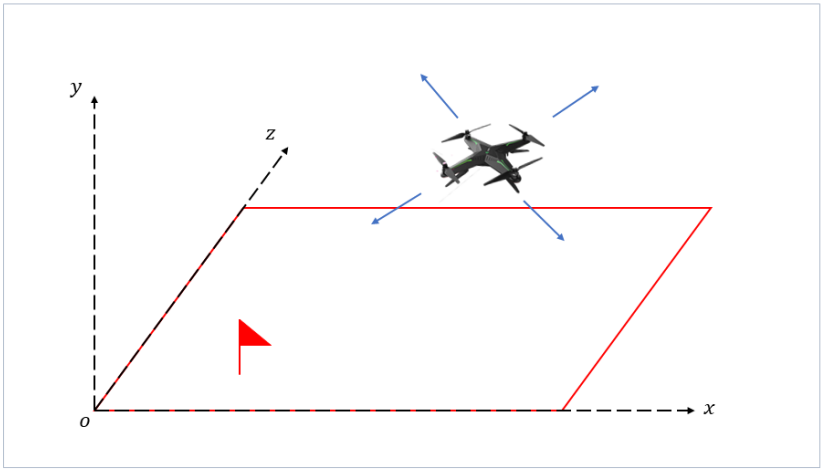

Figure 8. Drone three-dimensional space control task simulation diagram.

Furthermore, we have depicted our strategy through illustrations in Figures 9 and 10. Given the original eight-dimensional state space, we employed Principal Component Analysis (PCA) to compress it into a two-dimensional space for the sake of clarity in visualization. In the derived visual representation, the orientation of the arrows signifies the actions the model is inclined to undertake in specific states. A concentration of arrows pointing in a uniform direction within certain regions implies a degree of consistency in the model's action choices in those states. In contrast, a varied spread of arrow directions may indicate the states' inherent uncertainty or highlight the exploratory behavior of the model during its learning phase. Moreover, the arrow lengths convey the action magnitudes, with longer arrows denoting larger actions and suggesting the model's heightened assurance in its responses to those states. Conversely, shorter arrows could denote a more tentative or measured approach by the model in certain states.

It is particularly interesting to observe that within the core region of the feature space, both the direction and magnitude of the arrows maintain a remarkable consistency. This pattern implies that the model potentially adopts a more stable or decisive strategy when

encountering states within this area. Additionally, in the top-right quadrant of the diagram, we notice a sequence of elongated arrows all pointing in the same direction, suggesting that the model exhibits a pronounced and robust inclination towards certain actions in these specific zones.

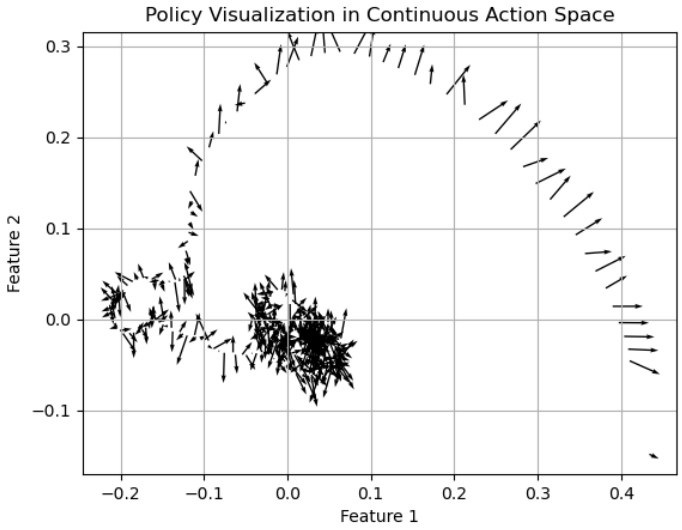

Figure 9. The visualization strategy displayed when the training results are satisfactory.

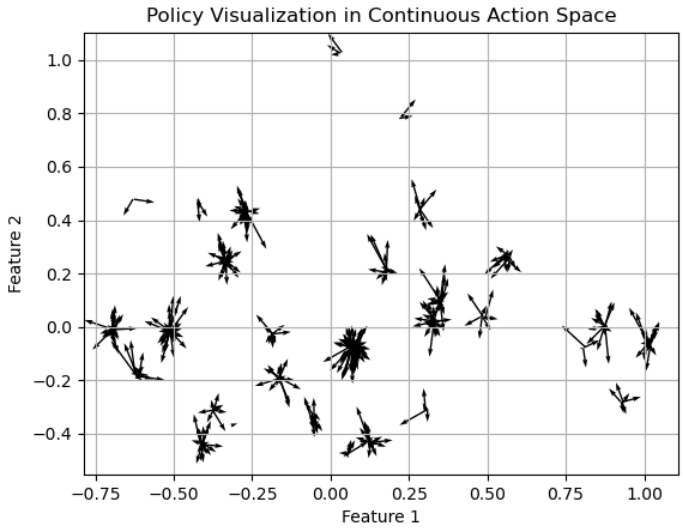

Figure 10. In cases of poor training results, we observe that the distribution of the arrows is more scattered, indicating that a unified and stable strategy has not yet been successfully developed.

In this experimental environment, we conducted comparative tests of our algorithm using the PPO algorithm as a baseline. As shown in Figures 11 and 12, the use of either GAIL (Generative Adversarial Imitation Learning) alone or Reptile alone positively impacted the results, with GAIL showing particularly noticeable effects. Our combined algorithm, Reptile+GAIL, demonstrated even more superior performance. Especially in the early stages of training, our algorithm was able to achieve good results quickly. During the convergence phase, compared to using GAIL alone, our algorithm exhibited better stability (Table 3). This success is attributed to the rapid adaptability to new tasks of meta-learning and the guiding role of expert data in imitation learning.

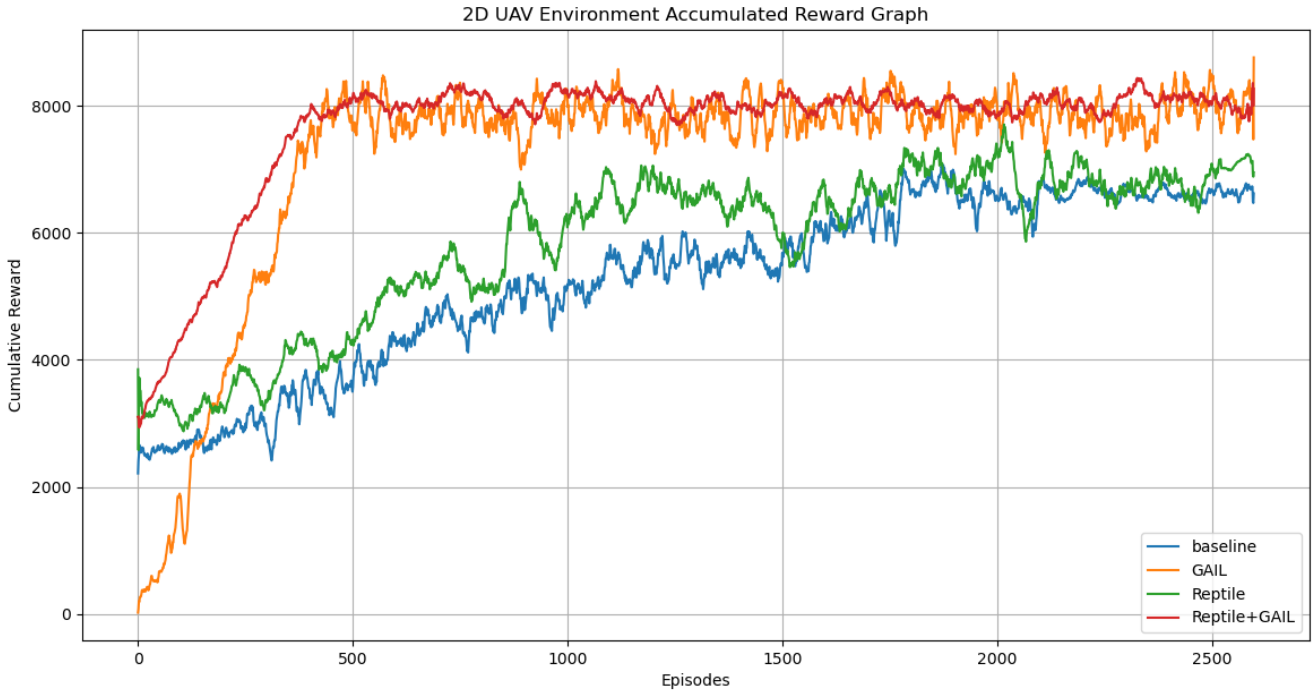

Figure 11. Comparative analysis of algorithm performance.

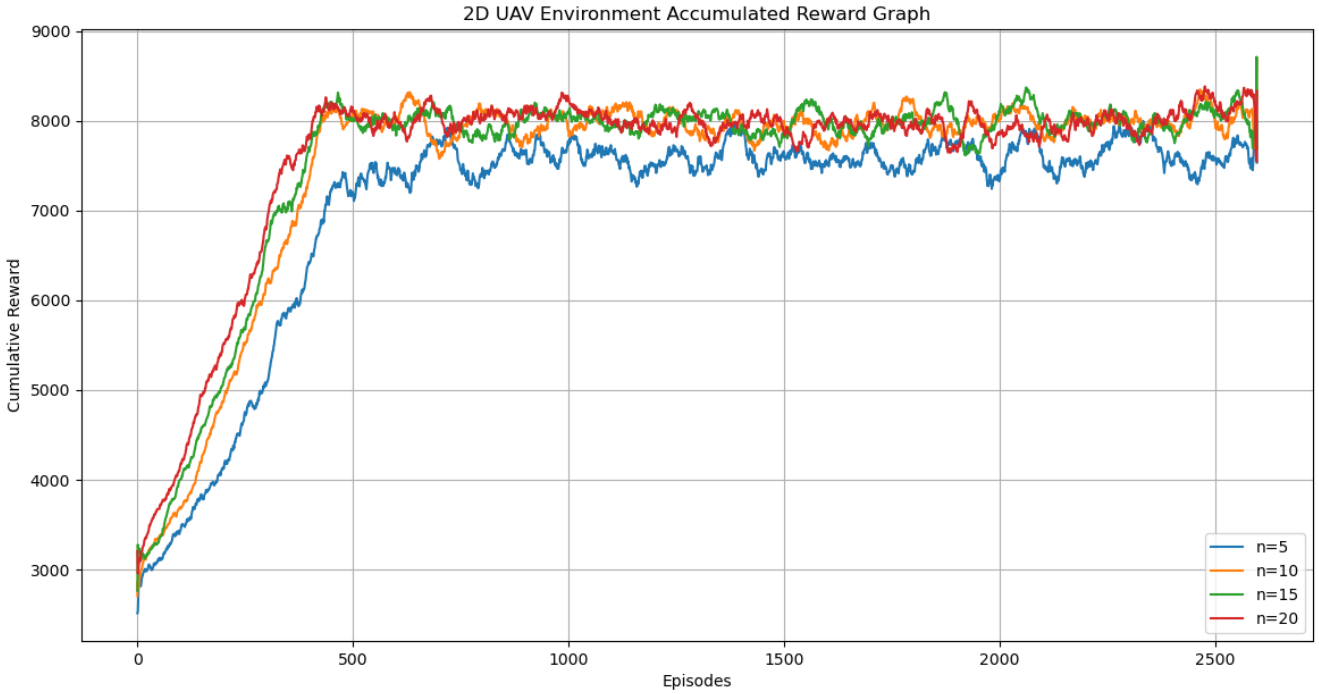

Figure 12. This graph illustrates the effect of varying the number of task samples. It can be observed that with the increase in the number of tasks, the performance of the algorithm correspondingly improves.

Table 3. The analysis table of return conditions under different n and num. It is observed that an increase in the number of expert trajectories and subtasks accelerates the model's learning pace during the initial phase of training until convergence is achieved. The returns in the table are moving averages.

Episodes	0	250	2500
n = 05, num = 100	2650.582136	4022.159734	7603.132461
n = 10, num = 100	2998.124612	4953.316497	7988.314579
n = 20, num = 100	3158.919738	5603.789134	7999.314725
n = 05, num = 250	2836.316472	4183.216475	8020.461346
n = 10, num = 250	3104.164917	5287.642197	8035.914238
n = 20, num = 250	3223.845127	5584.264975	8027.313671

Additionally, we conducted a thorough analysis of the process for generating expert trajectories and observed some key findings. Notably, when the number of available expert trajectories is limited, or the quality of these trajectories is insufficient, we found that the experimental results were significantly below expectations (see Figure 13). This discovery underscores the importance of high-quality and sufficient quantities of expert trajectories for the performance of our algorithm.

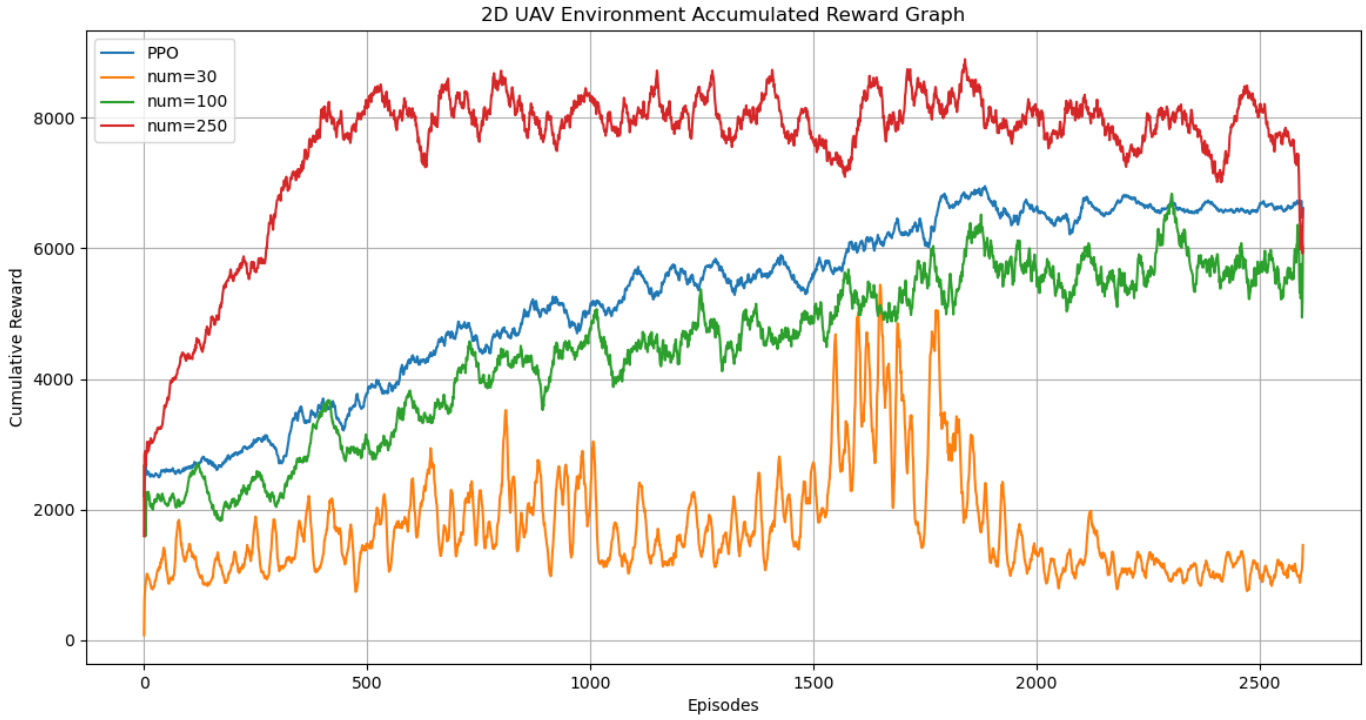

Figure 13. Effects of experiments with different numbers of expert trajectories.

We further investigated the reasons behind this phenomenon. As shown in Figure 14, with a limited number of expert trajectories, the learning algorithm faces a problem of insufficient data, which restricts the model's learning capabilities and generalization performance. On the other hand, when the quality of trajectories is low, the algorithm may learn incorrect or suboptimal behavioral patterns, leading to poor experimental outcomes.

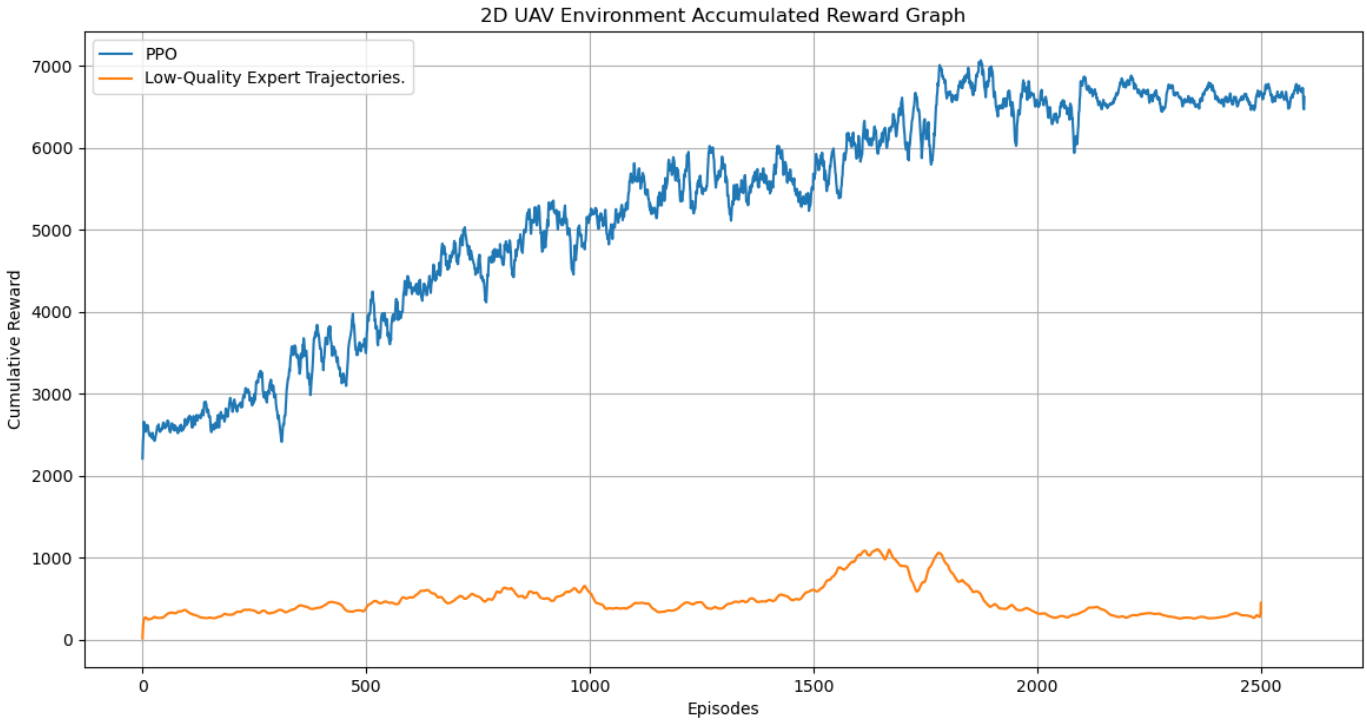

Figure 14. The impact of low-quality expert trajectories.

6. Conclusions

This paper elaborated on an innovative framework that integrated the meta-reinforcement learning algorithm Reptile with GAIL to achieve UAV control on a 2D plane. In experiments, this framework demonstrated significant advantages and improvements over traditional single-reinforcement learning algorithms. However, the framework still faces challenges that need to be addressed. Firstly, the effective operation of GAIL depends on high-quality expert trajectories, and efficiently acquiring these expert trajectories remains a key issue. Secondly, the real-time nature of online learning and its interaction with the environment poses certain risks, necessitating further research and optimization. Based on these analyses, our future research will focus on two main aspects: firstly, developing methods to generate high-quality expert trajectories, and secondly, exploring how to effectively translate online meta-reinforcement learning algorithms for use in offline scenarios, thereby enhancing the safety and practicality of the algorithm. Through these efforts, we aimed to further improve the performance and application scope of this framework.

7. Statement of Use of Artificial Intelligence Tools

In this paper, a small portion of the introduction to meta-learning was generated by ChatGPT, specifically the "To address these issues...explicit reward signals" section, where ChatGPT was tasked to articulate the roles of meta-learning and imitation learning separately. We then synthesized and fine-tuned these contributions; in the "Related Work" section, we provided relevant literature and directed ChatGPT to summarize the content. For example: Pham et al. [16] implemented autonomous navigation of UAVs using RL and the TEXPLORE algorithm, enhancing sophisticated UAV control; in the "Preliminary" section on Meta-RL, "During the meta-training phase...seemingly unrelated tasks", ChatGPT was employed to explain the process of meta-reinforcement learning and the significance of meta-knowledge, which we subsequently compiled and organized; in the "Methods" section, "In contrast…across a set of tasks", we collected information on meta-reinforcement learning algorithms Reptile and MAML from the internet and used ChatGPT to compare their strengths and weaknesses, after which we integrated and corrected the

content. The rest of the article was also moderately polished with the help of ChatGPT. All parts concerning the methods and experiments of this paper are original content.

Author Contributions: Conceptualization, Y.G. and S.J.; methodology, Y.G.; software, Y.G. and S.J.; experiment and validation, X.Y.; writing—original draft preparation, Y.G. and S.J.; writing—review and editing, X.Y. and W.Y.; supervision, H.C. All authors have read and agreed to the published version of the manuscript.

Data Availability Statement: The data associated with this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Minsky, M. Steps toward artificial intelligence. *Proc. IRE* **1961**, *49*, 8–30. [CrossRef]
2. Kendall, A.; Hawke, J.; Janz, D.; Mazur, P.; Reda, D.; Allen, J.M.; Lam, V.D.; Bewley, A.; Shah, A. Learning to drive in a day. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; IEEE: Piscateville, NJ, USA, 2019; pp. 8248–8254.
3. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef] [PubMed]
4. Yang, D.; Qin, X.; Xu, X.; Li, C.; Wei, G. Sample Efficient Reinforcement Learning Method via High Efficient Episodic Memory. *IEEE Access* **2020**, *8*, 129274–129284. [CrossRef]
5. Eschmann, J. Reward function design in reinforcement learning. In *Reinforcement Learning Algorithms: Analysis and Applications*; 2021 ; pp. 25–33.
6. Zhou, W.; Li, W. Programmatic reward design by example. In Proceedings of the AAAI Conference on Artificial Intelligence 2022, Virtual, 22 February–1 March 2022; Volume 36, pp. 9233–9241.
7. Belkhale, S.; Li, R.; Kahn, G.; McAllister, R.; Calandra, R.; Levine, S. Model-based meta-reinforcement learning for flight with suspended payloads. *IEEE Robot. Autom. Lett.* **2021**, *6*, 1471–1478. [CrossRef]
8. Zhou, S.; Cheng, Y.; Lei, X.; Duan, H. Multi-agent few-shot meta reinforcement learning for trajectory design and channel selection in UAV-assisted networks. *China Commun.* **2022**, *19*, 166–176. [CrossRef]
9. Park, B.; Oh, H. Vision-based obstacle avoidance for UAVs via imitation learning with sequential neural networks. *Int. J. Aeronaut. Space Sci.* **2020**, *21*, 768–779. [CrossRef]
10. Beck, J.; Vuorio, R.; Liu, E.Z.; Xiong, Z.; Zintgraf, L.; Finn, C.; Whiteson, S. A survey of meta-reinforcement learning. *arXiv* **2023**, arXiv:2301.08028.
11. Azar, A.T.; Koubaa, A.; Ali Mohamed, N.; Ibrahim, H.A.; Ibrahim, Z.F.; Kazim, M.; Ammar, A.; Benjdira, B.; Khamis, A.M.; Hameed, I.A.; et al. Drone deep reinforcement learning: A review. *Electronics* **2021**, *10*, 999. [CrossRef]
12. Hussein, A.; Gaber, M.M.; Elyan, E.; Jayne, C. Imitation learning: A survey of learning methods. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–35. [CrossRef]
13. Cui, J.; Liu, Y.; Nallanathan, A. Multi-agent reinforcement learning-based resource allocation for UAV networks. *IEEE Trans. Wirel. Commun.* **2019**, *19*, 729–743. [CrossRef]
14. Osa, T.; Pajarinen, J.; Neumann, G.; Bagnell, J.A.; Abbeel, P.; Peters, J. An Algorithmic Perspective on Imitation Learning; Foundations and Trends®in Robotics. 2018, pp. 1–179. Available online: https://www.nowpublishers.com/article/Details/ROB-053 (accessed on 1 March 2024).
15. Lin, K.; Li, C.; Li, Y.; Savaglio, C.; Fortino, G. Distributed learning for vehicle routing decision in software defined Internet of vehicles. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 3730–3741. [CrossRef]
16. Ho, J.; Ermon, S. Generative adversarial imitation learning. *Adv. Neural Inf. Process. Syst.* **2016**, *29*. [CrossRef]
17. Koch, W.; Mancuso, R.; West, R.; Bestavros, A. Reinforcement learning for UAV attitude control. *ACM Trans. Cyber-Phys. Syst.* **2019**, *3*, 1–21. [CrossRef]
18. Yijing, Z.; Zheng, Z.; Xiaoyi, Z.; Yang, L. Q learning algorithm based UAV path learning and obstacle avoidence approach. In Proceedings of the 2017 36th Chinese control conference (CCC), Dalian, China, 26–28 July 2017; IEEE: Piscateville, NJ, USA, 2017; pp. 3397–3402.
19. Pham, H.; La, H.; Feil-Seifer, D.; Nguyen, L. Autonomous uav navigation using reinforcement learning. *arXiv* **2018**, arXiv:1801.05086.
20. He, L.; Aouf, N.; Whidborne, J.F.; Song, B. Deep reinforcement learning based local planner for UAV obstacle avoidance using demonstration data. *arXiv* **2020**, arXiv:2008.02521.
21. Yang, B.; Ma, C.; Xia, X. Drone formation control via belief-correlated imitation learning. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, Online, 3–7 May 2021; pp. 1407–1415.
22. Wang, T.; Chang, D.E. Robust navigation for racing drones based on imitation learning and modularization. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; IEEE: Piscateville, NJ, USA, 2021; pp. 13724–13730.

23. Hu, Y.; Chen, M.; Saad, W.; Poor, H.V.; Cui, S. Meta-reinforcement learning for trajectory design in wireless UAV networks. In Proceedings of the GLOBECOM 2020—2020 IEEE Global Communications Conference, Taipei, Taiwan, 7–11 December 2020; IEEE: Piscateville, NJ, USA, 2020; pp. 1–6.
24. Prat, A.; Johns, E. PERIL: Probabilistic embeddings for hybrid meta-reinforcement and imitation learning. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 4 May 2021.
25. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
26. Yu, T.; Quillen, D.; He, Z.; Julian, R.; Hausman, K.; Finn, C.; Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In Proceedings of the Conference on Robot Learning PMLR, Virtual, 16–18 November 2020; pp. 1094–1100.
27. Nagabandi, A.; Clavera, I.; Liu, S.; Fearing, R.S.; Abbeel, P.; Levine, S.; Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv* **2018**, arXiv:1803.11347.
28. Nichol, A.; Achiam, J.; Schulman, J. On first-order meta-learning algorithms. *arXiv* **2018**, arXiv:1803.02999.
29. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the International Conference on Machine Learning PMLR, Sydney, Australia, 6–11 August 2017; pp. 1126–1135.
30. Robak, M. Zastosowanie Uczenia ze Wzmocnieniem (Reinforcement Learning) do Stabilizacji Ruchu. 2021. Available online: https://ruj.uj.edu.pl/xmlui/handle/item/287455 (accessed on 1 March 2024).