ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS

MSc Thesis

Hidden variables' estimation of trajectories' states using Imitation Learning

by

Dimitrios Patiniotis Spyropoulos

Submitted
in partial fulfilment of the requirements for the degree of

Master of Artificial Intelligence

at the

UNIVERSITY OF PIRAEUS

February 2024

Author: Dimitrios Patiniotis Spyropoulos

II-MSc “Artificial Intelligence”

February 29, 2024

Certified by: George Vouros

George Vouros,
Professor,
University of Piraeus
Thesis Supervisor

Certified by: George Giannakopoulos

George Giannakopoulos,
Researcher,
NCSR Demokritos
Member of Examination Committee

Certified by: Maria Dagioglou

Maria Dagioglou,
Researcher,
NCSR Demokritos
Member of Examination Committee

Hidden variables’ estimation of trajectories’ states using Imitation Learning

Dimitrios Patiniotis Spyropoulos

Abstract

Trajectory prediction is a critical problem with profound implications across various domains, from autonomous vehicles and robotics to aerospace and maritime navigation. In this thesis we will be examining trajectories from Paris to Constantinople. Flight trajectories entail navigating through various complexities, including diverse airspace configurations, compliance with international air traffic regulations, and adaptability to dynamic weather conditions. This thesis examines trajectories from Paris to Constantinople, where flight trajectories entail navigating through various complexities, including diverse airspace configurations, compliance with international air traffic regulations, and adaptability to dynamic weather conditions. Our exploration leverages Imitation Learning, focusing on Generative Adversarial Imitation Learning (GAIL), to tackle the trajectory prediction problem in the field of aviation. Specifically, we study the relative performance of two of the most common policy optimization algorithms, Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO), in the context of GAIL. Our findings distinctly highlight TRPO's superior performance over PPO within the GAIL framework, marking the main contribution of our research. This is evaluated through the rate of learning per epoch, the training speed of each optimizer, and the final performance of GAIL configured with each algorithm. Additionally, we examine how using more than one agent to predict the hidden variables of the trajectory affects the accuracy of our setup. An additional goal for this thesis was not to imitate spatio-temporal trajectories per se, but to also learn models for imitating critical KPIs (e.g. fuel consumption) of flight trajectories and examine the impact of learning to imitate spatio-temporal trajectories to predicting these KPIs. Highlighting TRPO's superior performance in the GAIL context underscores the main contribution of our research.

Πρόβλεψη παραμέτρων σε καταστάσεις τροχιών με τη χρήση Μιμητικής Μάθησης

Δημήτριος Πατηνιώτης Σπυρόπουλος

Περίληψη

Η πρόβλεψη της τροχιάς είναι ένα κρίσιμο πρόβλημα με βαθιές επιπτώσεις σε διάφορους τομείς, από τα αυτόνομα οχήματα και τη ρομποτική μέχρι την αεροδιαστημική και τη θαλάσσια ναυσιπλοΐα. Σε αυτή τη διπλωματική εργασία θα εξετάσουμε αεροπορικές τροχιές από το Παρίσι στην Κωνσταντινούπολη. Οι τροχιές πτήσης προϋποθέτουν πλοήγηση μέσα από διάφορους πολύπλοκους παράγοντες, όπως οι ποικίλες διαμορφώσεις του εναέριου χώρου, η συμμόρφωση με τους διεθνείς κανονισμούς εναέριας κυκλοφορίας και η προσαρμοστικότητα στις δυναμικές καιρικές συνθήκες. Η διερεύνησή μας αξιοποιεί τη Μιμητική Μάθηση (Imitation Learning) εστιάζοντας στον αλγόριθμο GAIL (Generative Adversial Imitation Learning), για την αντιμετώπιση του προβλήματος πρόβλεψης τροχιάς αξιοποιώντας παραδείγματα τροχιών. Συγκεκριμένα, μελετάμε τη σχετική απόδοση δύο από τους πιο συνηθισμένους αλγορίθμους βελτιστοποίησης πολιτικής, του TRPO (Trust Region Policy Optimization) και του αλγορίθμου PPO (Proximal Policy Optimization), στο πλαίσιο του GAIL. Τα ευρήματά μας αναδεικνύουν σαφώς την ανώτερη απόδοση του TRPO έναντι του PPO στο πλαίσιο GAIL, σηματοδοτώντας την κύρια συμβολή της έρευνάς μας. Αυτό αξιολογείται μέσω του ρυθμού μάθησης ανά εποχή, της ταχύτητας εκπαίδευσης κάθε παραγωγικού βελτιστοποιητή και της τελικής απόδοσης του GAIL. Επιπλέον, εξετάζουμε πώς η χρήση περισσότερων του ενός πρακτόρων για την πρόβλεψη των κρυφών μεταβλητών της τροχιάς επηρεάζει την ακρίβεια της ρύθμισής μας. Ένας επιπρόσθετος στόχος αυτής της διπλωματικής εργασίας πέρα από τη μίμηση χωροχρονικών τροχιών, είναι η εκμάθηση μοντέλων μίμησης κρίσιμων KPIs (Key Performance Indicators) (π.χ. κατανάλωση καυσίμων) των τροχιών πτήσης και η εξέταση της επίδρασης της μιμητικής μάθησης χωροχρονικών τροχιών στην πρόβλεψη αυτών των KPIs. Η ανάδειξη των ανώτερων επιδόσεων του TRPO στο πλαίσιο GAIL υπογραμμίζει την κύρια συμβολή της έρευνάς μας.

Acknowledgments

The journey to completing this thesis has been an enlightening experience, thanks in large part to the guidance and support of my supervisors. First and foremost, I would like to extend my deepest gratitude to Professor George Vouros. Their expertise and insight have been invaluable throughout this process. Vouros's guidance helped navigate the complexities of my research topic, providing a solid foundation upon which this thesis was built. Their patience and willingness to engage in thoughtful discussions were crucial in refining my ideas and approaches. I am genuinely thankful for the time and effort they dedicated to supporting my work, offering both professional insights and personal encouragement.

I must also express my sincere appreciation to Researcher George Giannakopoulos and Researcher Maria Dagioglou for their contributions to my research. Their practical advice and constructive feedback were essential in streamlining my research methodology, enhancing the overall coherence and impact of my study. Their support played a significant role in overcoming various challenges in my research journey.

Lastly, I would also like to thank Researchers Theocharis Kravaris and Christos Spatharis for giving me guidance in specific areas of my research where their expertise and feedback was incredibly beneficial.

Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the «funding body» or the view of University of Piraeus and Inst. of Informatics and Telecom. of NCSR “Demokritos”

Table of Contents

List of Figures, Diagrams and Tables

1. Introduction

Imitation learning, a technique aimed at replicating behavior in tasks, has gained significant traction due to recent advancements in computing and sensing, alongside a growing demand for complex intelligent applications. This approach involves training an agent, or learning machine, to perform tasks by observing demonstrations and mapping these observations to actions. Unlike traditional reinforcement learning methods where the agent explores the environment and gets rewards, imitation learning focuses on mimicking expert behavior through demonstrations, with no prior knowledge of the reward function.

In this thesis, we compare the performance of two prominent optimization algorithms, Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO), in enhancing the agents' policy learning process within a Generative Adversarial Network (GAN) setup. This combination of TRPO or PPO with a GAN type of setup yields the Generative Adversarial Imitation Learning (GAIL) approach [4]. GAIL leverages the strengths of TRPO or PPO in optimizing policy learning alongside the capabilities of GANs in generating realistic trajectories within the imitation learning framework. A major emphasis of our research is placed on the comparison between TRPO and PPO, aiming to determine the most suitable optimizer for our imitation learning framework, but we also highlight potential strong points and weaknesses of each optimizer in the context of GAIL.

In our research, we mainly focus on imitation learning within the context of aircraft flight plans, specifically examining trajectories from Paris to Constantinople. Flight trajectories entail navigating through various complexities, including diverse airspace configurations, compliance with international air traffic regulations, and adaptability to dynamic weather conditions. This dataset offers a rich tapestry of challenges that allows us to comprehensively evaluate the performance of imitation learning algorithms like GAIL with TRPO and PPO.

When comparing these two optimizers we will be mainly examining three key factors to assess their performance within the Generative Adversarial Imitation Learning (GAIL) framework. Our focus will primarily revolve around evaluating the rate of learning per epoch, the training speed of each optimizer, and final performance of GAIL configured with each of the algorithms. By scrutinizing these critical aspects, we aim to provide a comprehensive understanding of how TRPO and PPO contribute to the overall effectiveness of imitation learning in complex tasks, such as aircraft navigation from Paris to Constantinople. An additional goal for this thesis was not to imitate spatio-temporal trajectories per se, but to also learn models for imitating critical KPIs (e.g. fuel consumption) of flight trajectories and examine the impact of learning to imitate spatio-temporal trajectories to predicting these KPIs. We will

also be looking at how tweaking the learning framework affects the effectiveness of each of those optimizers [19].

The source code from this thesis can be found here.

2. Literature Review

Trajectory prediction is a critical problem with profound implications across various domains, from autonomous vehicles and robotics to aerospace and maritime navigation. Its importance lies in enabling systems to anticipate future positions of moving objects, which is essential for safe and efficient operation. This critical understanding of trajectory prediction's role in ensuring the safety and efficiency of various systems, and predicting their behavior either at the microscopic or at the macroscopic levels, has led to a wealth of research aimed at enhancing trajectory prediction accuracy and reliability.

Trajectory prediction techniques are broadly categorized into traditional models, machine learning approaches, and hybrid methods. Traditional models rely on mathematical formulations and assumptions about motion patterns, offering analytical precision but often lacking adaptability to complex or dynamic environments. Machine learning approaches, in contrast, leverage data to learn patterns and make predictions, showing remarkable adaptability and performance in diverse settings but sometimes requiring extensive data and computational resources. Hybrid methods combine the best of both worlds, integrating the robustness of traditional models with the adaptability of machine learning, aiming to achieve high accuracy while addressing the limitations inherent in each approach [17].

Imitation learning emerges as a powerful approach to tackle trajectory prediction problems by leveraging expert demonstrations to teach models desired behaviors without explicitly programming the complex rules governing those behaviors. This method is particularly effective in environments where defining a comprehensive reward function is challenging, or when the dynamics of the system are too complex for traditional control strategies [3]. Methods that have imitation learning as an integral part have been used in very diverse set of fields [16]. From enhancing the evaluation of a team’s performance in sports [14] to predicting bicycle trajectories in urban environments [15], there are many cases where these methods show their versatility and effectiveness in capturing complex patterns and behaviors.

In this thesis we will be tackling the trajectory prediction problem using imitation learning in the field of aviation. Tackling the trajectory prediction problem in aviation has been critically important for enhancing airspace safety, efficiency, and sustainability. Accurate predictions of aircraft trajectories enable more precise air traffic management operations including, for instance, reducing the risk of mid-air collisions and allowing for tighter spacing between aircraft [21]. This optimization leads to more efficient flight paths, reducing fuel consumption and emissions, and ultimately contributing to

environmental protection. Furthermore, improved trajectory prediction supports the implementation of advanced operational concepts like free flight, where pilots choose their paths dynamically, increasing airspace capacity and flexibility [20]. As air travel continues to grow, solving the trajectory prediction problem becomes essential for maintaining safety standards, minimizing delays, and ensuring the sustainable expansion of the aviation industry.

Aircraft trajectory estimation is a problem almost as old as aircrafts themselves. Historically, aircraft state estimation has relied on a combination of physics-based models and sensor data to predict the future states of aircraft [22]. Older approaches take a more modest approach by using the equation of motion for predicting typically shorter trajectories [18]. Even before the emergence of modern imitation learning algorithms, machine learning techniques have been used to solve this problem with moderate success [19].

As imitation learning methods matured, more and more implementations of such methods have been applied in the aircraft trajectory estimation problem. Both model-free imitation learning [23] and model-free solutions, such as Generative Adversarial Imitation Learning (GAIL) [4] have been used to tackle this problem. It is interesting to note that these methods are sometimes coupled with traditional methods [24].

The implementation we will be using in this thesis will be based on an existing implementation of a Generative Adversarial setup, in the context of imitation learning, originally used in [13]. Some changes have been made when compared to that implementation, mainly the fact that the features used have been reduced and that we will also predict the fuel consumption of the trajectory. The dataset will also differ. In their paper, Alevizos et. al [13] use demonstrated trajectories from Barcelona to Madrid, while we will be using flight plans from Paris to Constantinople, which is a more demanding set of trajectories, given the length of trajectories, and thus the needed prediction horizon from the origin to the destination.

We will also be focusing on modifying the architecture of GAIL. GAIL in its original paper [4] uses Trust Region Policy Optimization (TRPO) [25] as the method of policy optimization. In this thesis we will also incorporate Proximal Policy Optimization (PPO) [11] as a method in our Generative Adversarial framework. PPO was introduced after TRPO by OpenAI to address the complexity and computational inefficiency of earlier policy gradient methods like Trust Region Policy Optimization. According to it’s creators PPO simplifies the optimization process, making it more practical for implementation while retaining strong performance.

Since TRPO (Trust Region Policy Optimization) and PPO (Proximal Policy Optimization) are both widely used policy gradient optimizers in reinforcement learning, understanding their trade-offs

and appropriate application contexts is crucial. TRPO offers precise policy updates at higher computational costs, whereas PPO provides a more practical, scalable alternative with slightly relaxed update rules. Deciding between them hinges on balancing computational resources, desired precision, and specific task requirements.

Indeed, literature is not conclusive on this issue. What is commonly suggested is that superiority of one optimizer over the other is oftentimes dependent on nuanced adjustments in the setup of each model. Seemingly unimportant optimizations fundamentally change algorithm operation in ways unpredicted by the conceptual policy gradient framework [26]. It is important to note that the existing comparisons are done in simpler reinforcement learning structures and not in the Generative Adversarial context we will be using them.

In this thesis, we extend the examination of TRPO and PPO within the unique context of Generative Adversarial Imitation Learning (GAIL), aiming to uncover how each optimizer influences the learning process and performance in this more complex framework. This added complexity should provide an interesting test bed for comparing the capabilities and limitation of the two optimizer. By integrating both optimizers into our GAIL setup, we not only explore their computational efficiency and optimization capabilities but also their impact on the nuanced dynamics of imitation learning in the context of aircraft trajectory prediction, such as the accuracy of trajectory prediction and fuel consumption estimates. This comparative analysis is vital, as it provides insights into the adaptability of TRPO and PPO in handling the intricacies of generative adversarial networks applied to reinforcement learning. Furthermore, our study contributes to the broader discussion on the practicality of implementing advanced optimization methods in real-world applications, where the balance between computational demand and learning performance is paramount. Through this exploration, we aim to offer concrete recommendations on optimizer selection based on specific criteria within the GAIL framework, enhancing the understanding of policy gradient optimizers in complex learning environments.

In the next sections we will present the framework in which we will conduct our experiments. This study aims to contribute to the field by offering a detailed comparison of TRPO and PPO in the context of GAIL for trajectory prediction. We will firstly define and examine the algorithms that we will be using. The preliminaries section provides a foundational understanding of imitation learning, detailing its main algorithms, such as Behavioral Cloning and Inverse Reinforcement Learning (IRL), and introduces GAIL. It discusses the rationale behind GAIL, its benefits, and how optimization is approached within this framework. The thesis further explores policy gradient methods, offering insight into Vanilla Policy Gradient, TRPO, and PPO, culminating in a preliminary comparison between TRPO

and PPO. In addressing the trajectory imitation problem, we outline the data-driven task of trajectory prediction, the approach to the flight trajectory prediction problem, data overview and preprocessing, and the methodology for approaching the imitation problem. The algorithm structure and methodology section describe the use of behavioral cloning for pre-training and the implementation details of TRPO and PPO within the GAIL setup.

Experimental results are presented with an emphasis on evaluation measures such as RMSE, Along-track error (ATE), Cross Track Error (CTE), and Altitude Error, alongside the analysis of SingleModel (modeling spatiotemporal trajectories with the fuel consumption) and DoubleModel results, including a modified fuel consumption prediction model, taking as input the predicted spatiotemporal trajectory. The discussion synthesizes findings, comparing TRPO and PPO's performance in solving the trajectory prediction problem. The conclusion wraps up the thesis, summarizing the significant insights gained and suggesting avenues for future research.

3. Preliminaries

3.1 Imitation Learning

Imitation learning entails an agent acquiring behavioral patterns within an environment with an unspecified reward or cost function by emulating expert demonstrations. Unlike traditional reinforcement learning approaches that involve the agent learning through trial and error, imitation learning focuses on the agent learning decision-making policies from pre-existing expert demonstrations. While imitation learning shares similarities with reinforcement learning, it diverges by leveraging expert knowledge to guide the agents' behavior rather than relying solely on the agents' exploration of the environment.

3.2 The Main Imitation Learning Algorithms

There are three methods of approaching the imitation problem. The initial approach is through behavioral cloning [1, 2], which focuses on replicating the observed actions of the expert in given states, without explicitly attempting to infer the expert's underlying reward function or cost function. The other two methods are inverse reinforcement learning [3] and Generative Adversarial Imitation Learning (GAIL) [4], which try to reconstruct an expert's reward/cost function.

Behavioral cloning tackles the problem by framing it as a supervised learning task, focusing on the state-action pairs observed in expert demonstrations. It boasts advantages such as addressing the problem as a regression task, aiming to minimize the discrepancy between demonstrated actions and policy actions across historical trajectories. However, drawbacks include compounding errors and regret bounds that escalate quadratically with the task's time horizon, resulting in suboptimal performance.

Inverse reinforcement learning [3] seeks to derive a cost function that assigns minimal cost to expert-demonstrated trajectories and maximal cost to trajectories generated by alternative policies. Challenges associated with this method include assumptions regarding the linearity of the cost function and the computational burden incurred during cost approximation and policy computation stages.

Generative Adversarial Imitation Learning (GAIL) [4] presents a model-free imitation learning algorithm designed to improve performance in replicating complex behaviors within high-dimensional environments featuring continuous state-action spaces. However, this approach comes with a trade-off in high sample complexity.

3.2.1 Behavioral Cloning

Behavioral cloning emerges as the simplest form of imitation learning, leveraging expert observations and action demonstrations through supervised learning. This learning approach hinges on state-action pairs, with states usually derived from expert demonstrations whilst the choice of loss function is typically application-dependent. Through self-supervised learning from experience, the agent develops a model utilized to perform specific tasks towards achieving an objective, without the need for explicit reinforcement signals. Behavioral cloning thrives with ample training data but faces challenges due to compounding errors caused by covariate shift. This is due to a discrepancy introduced by imitation learning: the agent's policy influences the next queried state. This discrepancy implies that the training and testing distributions no longer align (covariate shift), and since behavioral cloning heavily relies on supervised learning from expert demonstrations, it faces challenges when training and testing under most circumstances.

3.2.2 Inverse Reinforcement Learning (IRL)

Inverse reinforcement learning (IRL) tackles the challenge of inferring an agent's reward or cost function based on its observed behavior or policy. Unlike behavioral cloning, IRL does not require a large training dataset to achieve optimal performance. The primary objective of IRL is to derive a cost function that assigns minimal cost to trajectories demonstrated by experts and maximal cost to trajectories generated by alternative policies.

Inverse Reinforcement Learning (IRL) encompasses a diverse array of approaches aimed at inferring an agent's underlying reward function or cost function from observed behavior or expert demonstrations. There are many approaches within the realm of IRL such as Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL)[5], Bayesian Inverse Reinforcement Learning[6] and Adversarial Inverse Reinforcement Learning[7].

MaxEnt IRL, as the name suggests, incorporates the principle of maximum entropy into the learning process. Its intuition is that there may exist multiple optimal policies that can achieve the same task or behavior. The objective of MaxEnt IRL is to infer a reward function that not only explains the observed behavior of an expert demonstrator but also encourages diversity and flexibility in the agent's behavior. By maximizing entropy, MaxEnt IRL seeks to avoid overly deterministic or rigid policies,

allowing the agent to exhibit a wider range of behaviors while still remaining consistent with the expert demonstrations.

On the other hand, Bayesian Inverse Reinforcement Learning introduces Bayesian methods to estimate the uncertainty associated with the learned reward function. Bayesian approaches have been widely used in various areas of machine learning and statistics for decades. In this approach, the goal is to infer a posterior distribution over possible reward functions given the observed behavior of the agent. By incorporating uncertainty into the learning process, Bayesian IRL provides a probabilistic framework for decision-making under uncertainty, allowing the agent to make more informed decisions in ambiguous or uncertain environments.

Adversarial Inverse Reinforcement Learning stands out among other IRL approaches by introducing an adversarial training paradigm. Unlike traditional IRL methods that directly infer a reward function, Adversarial IRL leverages a generator-discriminator framework inspired by generative adversarial networks (GANs). This setup allows the generator to learn from expert demonstrations by generating trajectories, while the discriminator learns to distinguish between expert and generated trajectories. Through adversarial training, the generator improves its performance by producing trajectories that are indistinguishable from expert demonstrations, offering a unique approach to inferring reward functions in imitation learning tasks.

3.3 Generative Adversarial Imitation Learning (GAIL)

3.3.1 The idea behind GAIL

Generative Adversarial Imitation Learning (GAIL) [4] is a prominent example of Adversarial Inverse Reinforcement Learning, known for its ability to imitate multiplex behavior without imposing constraints on the cost function and for its scalability to large, continuous state-action environments. It directly learns the optimal policy from expert demonstrations, bypassing the need to derive a separate cost function for Reinforcement Learning methods to derive a policy. Instead, GAIL focuses on aligning the distribution of state-action pairs of the imitator with that of the expert. It operates by employing two neural network models: a generative model (the imitating agent) and a discriminative model (the discriminator). The generative model aims to replicate the expert's policy by generating actions given the states, while the discriminative model evaluates the pairs of states and actions to determine whether they are from the expert or the imitating agent.

At its core, GAIL seeks to minimize the divergence between the state-action distribution of the imitator's policy and the expert's policy. This is analogous to the way Generative Adversarial Networks (GANs) work, where the generator tries to produce data that is indistinguishable from real data, as judged by the discriminator. However, in GAIL, the 'real data' is the trajectory of the expert, and the 'generated data' is the trajectory produced by the agent's policy. The discriminator in GAIL, denoted by D, plays a critical role. It is trained to distinguish between the state-action pairs generated by the expert and those by the agent. The better the discriminator becomes at this task, the better the signal it provides to the agent about how to improve its policy. The agent's policy π is adjusted to make the discriminator's job as hard as possible, thereby making the agent's behavior more closely resemble that of the expert.

GAIL optimizes the objective:

$$\min_{\pi} \max_{D \in (0,1)^{S \times A}} \mathbb{E}_{\pi} \left[\log(D(s,a)) \right] + \mathbb{E}_{\pi_E} \left[\log(1 - D(s,a)) \right] - \lambda_h H(\pi)$$

where π_E is the expert's policy, π is the agent's policy and D is the discriminator, a binary classifier that classifies state-action pairs, where the actions are generated by the policies π_E and π. H(π) represents the causal entropy of the agent's policy, which encourages exploration and ensures that the policy does not collapse into a deterministic one. λ is a hyperparameter that balances the importance of the entropy term against the other objectives.

3.3.2 Benefits of GAIL

Generative Adversarial Imitation Learning (GAIL) represents a significant advancement in the domain of machine learning, particularly in the context of IRL. Unlike traditional IRL methods that rely heavily on the derivation of a cost function to understand and replicate expert behavior, GAIL adopts a novel approach by leveraging the framework of Generative Adversarial Networks (GANs). This strategic shift enables GAIL to directly learn from expert demonstrations without the intermediary step of cost function inference, thereby streamlining the imitation learning process. This direct learning capability is instrumental in GAIL's ability to address complex behaviors across vast, continuous state-action spaces, which has historically been challenging for conventional IRL approaches.

At the core of GAIL's innovation is its ability to learn complex behaviors without the need for explicit cost functions. This aspect is crucial in environments where defining a comprehensive cost function is impractical due to the complexity or the nuanced nature of optimal behaviors. This direct policy

learning from expert demonstrations not only enhances the efficiency of the imitation process (avoiding policy optimization for every single cost function approximation throughout the process) but also ensures higher fidelity in the replication of expert behavior. The interconnectedness of direct learning and the ability to handle complex behaviors underscores GAIL's robustness in modeling and mimicking intricate decision-making processes without the constraints typical of traditional IRL methods.

Another standout feature of GAIL is its scalability to large, continuous state-action spaces. The generative adversarial framework at the heart of GAIL empowers it to efficiently navigate and learn within high-dimensional environments. This scalability is a testament to GAIL's adaptability, allowing it to be applied to a wide array of tasks ranging from robotic manipulation in physical spaces to complex strategy games in virtual environments. The underlying efficiency of GAIL's learning process, fueled by its focus on the most informative aspects of the expert's behavior, further facilitates its ability to generalize well to unseen scenarios. By learning policies that closely mimic the expert's across diverse situations, GAIL ensures that the imitated behaviors are not just applicable within the confines of the training data but extend effectively to new, unencountered states. This generalization capability is crucial for real-world applications where conditions and requirements can vary significantly from the training environment.

The last major advantage of GAIL is its incorporation of causal entropy in the learning objective, which promotes exploration. Exploration is vital in reinforcement learning and imitation learning contexts, as it prevents premature convergence on suboptimal, deterministic policies. By encouraging the agent to explore a broad range of actions, GAIL ensures the development of robust and flexible policies that are capable of adapting to various situations. This emphasis on exploration not only aids in discovering effective strategies but also in enhancing the policy's adaptability and resilience to changes in the environment. The balance GAIL strikes between exploration and exploitation exemplifies its advanced learning capabilities, further solidifying its position as a superior method for imitation learning.

3.3.3 Policy Optimization in GAIL

GAIL incorporates two distinct optimizers: one for the actor and one for the discriminator. This is a logical and effective strategy, directly aligned with the different roles and optimization challenges each component presents.

The discriminator, tasked with distinguishing between the actor's generated actions and the genuine expert actions, operates in a more straightforward classification context. Here, the Adam optimizer [8] stands out for its efficiency and adaptability. Adam's ability to adjust learning rates

individually for each parameter based on the gradients' first and second moments makes it highly effective for the discriminator. This optimizer quickly adapts to the shifting landscape as the actor evolves, maintaining the discriminator's accuracy and responsiveness.

Optimization when it comes to the actor stands as the pivotal yet challenging endeavor of GAIL. The actor's role is to emulate the expert's behavior as closely as possible by making decisions based on the observed states. This process, however, is not straightforward, particularly because of the adversarial nature of GAIL, where the actor is in a constant tug-of-war with the discriminator. This dynamic interplay creates a non-stationary optimization landscape, fraught with deceptive gradients that can mislead the optimization process, suggesting short-term improvements that do not genuinely contribute to achieving expert-like imitation over time. So, whilst in GANs both the generator and the discriminator are typically trained using stochastic gradient descent-based optimizers such as Adam, in the context of GAIL they seem to be insufficient.

3.4 Policy Optimization: Policy Gradient & Trusted-Region Methods

3.4.1 Introduction to Policy Gradient Methods

Policy Gradient methods occupy a pivotal role in the optimization landscape of reinforcement learning (RL), presenting a powerful class of algorithms designed to directly optimize the policy—a function that maps states to probability distribution over actions—towards maximizing cumulative rewards. Unlike value-based methods, which indirectly learn a policy through the value function, Policy Gradient methods adjust the policy parameters via gradients of the expected reward. This direct approach to policy optimization enables the learning of stochastic policies, offering significant advantages in environments where the optimal policy may involve randomness or the action space is continuous and high-dimensional.

The theoretical foundation of Policy Gradient methods is rooted in the policy gradient theorem [9], which provides a formal mechanism to compute the gradient of the expected return with respect to the policy parameters. This theorem ensures that by following the gradient of the expected return, it is possible to iteratively improve the policy.

3.4.2 Vanilla Policy Gradient

The key idea underlying policy gradients is to push up the probabilities of actions that lead to higher return, and push down the probabilities of actions that lead to lower return, until you arrive at the optimal policy. Vanilla Policy Gradient (VPG) is an on-policy algorithm that can be used for environments with either discrete or continuous action spaces. The gradient of the expected finite-horizon undiscounted return of the policy $J(\pi_\theta)$ is

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t) \right],$$

where π_θ is a policy with parameters θ, τ is a trajectory and A^{π_θ} is the advantage function for the current policy π_θ. The policy gradient algorithm operates by updating the policy parameters through stochastic gradient ascent on policy performance:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_{\theta_k})$$

In policy gradient implementations, it is common to calculate advantage function approximations based on the infinite-horizon discounted return, even when applying the finite-horizon undiscounted policy gradient formula in other aspects. The Vanilla Policy Gradient (VPG) algorithm begins with predefined policy parameters, θ_0 and value function parameters ϕ_0. It improves the policy by iteratively collecting trajectories under the current policy $\pi(\theta_k)$, from the environment. These trajectories are used to calculate rewards-to-go, $\hat{R}_t$ and to estimate the advantage, $\hat{A}_t$, which measures the relative quality of actions taken. The algorithm computes the policy gradient using these advantage estimates to adjust the policy parameters in the direction that increases expected returns. This process is repeated, with the policy and value function parameters being refined incrementally with the goal of maximizing the expected return of the agent's actions in the environment. The policy is updated either by standard gradient ascent or an optimizer like Adam, while the value function is tuned via regression on the mean-squared error between predicted and actual returns. This dual process of policy and value function updates continues until the policy converges to an optimal set of parameters that dictate the most rewarding actions.

Algorithm 1 Vanilla Policy Gradient Algorithm

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, ...$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

  Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

  or via another gradient ascent algorithm like Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

In Vanilla Policy Gradient (VPG), the method of addressing the exploration versus exploitation dilemma is dynamic. The algorithm starts by exploring the action space stochastically, guided by the most current policy iteration. As training advances, the policy's stochastic nature is gradually refined, favoring actions that have historically led to better rewards. However, this shift from exploration to exploitation can sometimes lead to premature convergence to suboptimal policies if the algorithm becomes too focused on exploiting known rewards, potentially neglecting better options yet to be discovered and causing the policy to get trapped in local optima.

VPG's sample inefficiency and susceptibility to high variance in gradient estimates are major pitfalls. It requires fresh samples from the environment for each policy update, leading to computational and time demands. The on-policy nature means it cannot leverage previous experiences, exacerbating sample inefficiency. Additionally, VPG's simple update mechanism does not include trust region methods, which can help in making safer, more reliable policy updates. These issues contribute

to potential instability in training and reduce the risk of converging to suboptimal policies due to inadequate exploration strategies and difficulty in escaping local optima.

3.4.3 Trust Region Policy Optimization

Trust Region Policy Optimization (TRPO) [25] represents a breakthrough in reinforcement learning, specifically in the optimization of policy gradient methods for better stability and efficiency. TRPO addresses the shortcomings of previous approaches by introducing a mechanism that ensures updates to the policy that do not deviate significantly from the current policy, thereby maintaining the learning process's stability and reliability.

The primary motivation behind TRPO is to overcome the challenges associated with policy gradient methods, particularly their tendency to take overly large steps in the policy space, which can lead to performance collapse. TRPO introduces a 'trust region' around the current policy, a concept borrowed from trust region optimization methods in numerical optimization. This region defines a boundary within which the policy can be updated, ensuring the new policy remains within a safe distance of the old policy, as measured by the Kullback-Leibler (KL) divergence.

The objective function in TRPO is designed to maximize the expected return while ensuring the new policy π_θ does not stray too far from the old policy $\pi_{\theta_{\text{old}}}$. This is achieved through the following objective function:

$$L(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right]$$

Here, $\pi_\theta(a_t|s_t)$ represents the probability of taking action a_t in state s_t under the policy parameterized by θ, and $\hat{A}_t$is an estimator of the advantage at time t, which indicates how much better it is to take a specific action compared to the average.

The trust region constraint is defined by the KL divergence, which measures how one probability distribution diverges from a second, expected probability distribution. TRPO ensures that the average KL divergence between the new policy and the old policy across all states does not exceed a predefined threshold δ:

$$\mathbb{E}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s) \,\|\, \pi_\theta(\cdot|s)\right)\right] \leq \delta$$

This constraint is crucial for controlling the size of the policy update and ensuring that the optimization process does not lead to policies that are too different from what has been proven to work, thus maintaining stability.

TRPO starts with an initial policy and value function. It progresses through iterations, collecting trajectories under the current policy, computing rewards-to-go, and estimating advantages using any chosen method. The policy gradient, $\hat{g}_k$, is then calculated as an average over the sampled trajectories.

A crucial step in TRPO is using the conjugate gradient algorithm to approximate the natural gradient, $\tilde{x}_k \approx H_k^{-1}\hat{g}_k$, where H_k is the Hessian of the sample average KL-divergence. This step is essential to TRPO's stability, as it ensures policy updates are within a theoretically justified trust region, preventing policy updates that are too large and might destabilize learning.

Policy parameters are updated using a backtracking line search, which is a methodical approach to finding the step size that satisfies both the improvement in the sample loss and the KL-divergence constraint. This helps maintain the balance between exploring new strategies and refining the current policy within a safe range. Lastly, the value function is updated by minimizing the mean-squared error between the value function's predictions and the computed rewards-to-go, thus improving the policy's value estimates. The algorithm iterates these steps, leading to a policy that not only imitates the expert's behavior but does so by incrementally learning in a stable and efficient manner.

Algorithm 1 Trust Region Policy Optimization

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

Hyperparameters: KL-divergence limit δ, backtracking coefficient α, maximum number of backtracking steps K

for $k = 0, 1, 2, ...$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

Use the conjugate gradient algorithm to compute

$$\hat{x}_k \approx \hat{H}_k^{-1} \hat{g}_k,$$

where $\hat{H}_k$ is the Hessian of the sample average KL-divergence.

Update the policy by backtracking line search with

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{\hat{x}_k^T \hat{H}_k \hat{x}_k}} \hat{x}_k,$$

where $j \in \{0, 1, 2, ...K\}$ is the smallest value which improves the sample loss and satisfies the sample KL-divergence constraint.

Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

end for

TRPO trains a stochastic policy in an on-policy way, which means that it explores by sampling actions according to the latest version of its stochastic policy. The amount of randomness in action selection depends on the training procedure and the initial conditions. Over the course of training, the policy usually becomes progressively less random, as the update rule encourages it to exploit rewards that it has already found. This means that TRPO, like VPG, may cause the policy to get trapped in local optima, but this tendency is not equal. This is due to the fact that VPG uses simpler policy updates which can overfit to the current policy's behavior, especially in environments with deceptive gradients.

TRPO, on the other hand, employs a trust region to constrain the policy updates, reducing the likelihood of making large, detrimental updates that could result in poor local optima. This mechanism helps TRPO in taking more informed steps during the optimization process, ideally leading to better global policy performance.

The incorporation of a trust region introduces additional benefits in TRPO when compared to VPG. Firstly, it limits the size of policy updates to prevent drastic changes that could harm learning. This results in more stable training and typically leads to better performance, especially in challenging environments with complex dynamics. TRPO also systematically ensures that the updated policy remains within a certain statistical distance from the old policy, thus providing a more controlled exploration of the policy space and reducing the likelihood of performance collapse.

Lastly, TRPO also tackles the major issue of policy improvement guarantees, which VPG lacks. TRPO's objective function is designed to ensure that the policy improves with high probability, leading to monotonic performance improvements. This is achieved through the use of a surrogate objective function that is maximized while keeping the policy within a trust region, defined by the KL-divergence. This theoretically grounded approach provides stronger assurances against performance drops after policy updates, which is a significant improvement over the more heuristic update strategy used in VPG.

Despite TRPO's advancements in policy optimization, it presents practical challenges. The computation of the Hessian matrix for large policy networks is resource-intensive, making it less feasible for real-time or low-resource settings. Furthermore, TRPO's complex implementation requires precise tuning of its trust region, which can be difficult to calibrate correctly. The algorithm also struggles with scalability to very high-dimensional action spaces or extremely large policy networks due to its computational intensity. These factors limit the algorithm's widespread adoption in environments where computational efficiency is paramount.

3.4.4 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [11] emerges as an evolution in reinforcement learning, refining policy gradient optimization to enhance both stability and accessibility. Designed to mitigate the computational complexities and implementation hurdles of its predecessor, TRPO, PPO simplifies the approach to policy updates. It retains the core idea of limiting updates to ensure stability but implements this through a clipped objective function, making it more straightforward to apply and

scale. PPO's innovation lies in balancing performance improvement with practical applicability, facilitating easier adoption for a broad range of reinforcement learning tasks.

PPO introduces two variants for policy updates: PPO-Penalty [11] and PPO-Clip [11]. PPO-Penalty adapts the TRPO framework by incorporating a penalty based on the KL divergence to maintain updates within a specified trust region, mirroring TRPO's approach to managing policy updates. PPO-Clip, alternatively, simplifies the process by clipping the policy update ratio, ensuring moderation without direct reliance on KL divergence calculations, distinguishing it from the more computationally intense TRPO methodology. PPO-Clip simplifies implementation even further than PPO-Penalty by employing a clipping mechanism in its objective function. This approach negates the need for calculating the exact KL divergence and adjusting penalty coefficients, making it more straightforward and computationally efficient to implement, especially in large-scale applications or when rapid development cycles are preferred.

In our research we will be using PPO-Clip. In selecting PPO-Clip for our study, we prioritize both its implementation efficiency and its broad acceptance in current research. This variant, by forgoing the need for complex KL divergence computations inherent in PPO-Penalty, presents a leaner approach conducive to rapid experimentation and adaptation in varied domains. Its widespread use across the scientific community not only speaks to its effectiveness but also ensures a rich ecosystem of shared insights and optimizations, facilitating smoother application and potentially greater reproducibility in our findings. These considerations, aligned with our research's objectives for streamlined and efficient development, underscore our choice of PPO-Clip. For simplicity, "PPO" will refer specifically to the PPO-Clip variant throughout this discussion.

PPO updates policies via the following function:

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{s,a\sim\pi_{\theta_k}} \left[L(s, a, \theta_k, \theta) \right],$$

The equation represents a key step in Proximal Policy Optimization (PPO) known as PPO-Clip, where the new policy parameters, denoted by θ_{k+1}, are determined by maximizing the expectation of the clipped surrogate objective function. E stands for the expectation operator, which calculates the expected value of the clipped surrogate objective function $L(s, a, \theta_k, \theta)$ over the distribution of states s and actions a sampled according to the policy π_{θ_k}. This function, L, is designed to temper the policy updates to ensure they do not stray too far from the old policy, thus fostering gradual improvement and maintaining training stability. The clipping mechanism specifically mitigates the risk of large policy updates that could degrade performance, steering the optimization in a controlled fashion. Typically

PPO updates typically takes multiple steps of SGD, often utilizing minibatches, to maximize the objective function mentioned above. The L function is the following:

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \quad \text{clip} \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right),$$

which can be simplified into:

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \quad g(\epsilon, A^{\pi_{\theta_k}}(s, a)) \right),$$

where

$$\mathrm{g}(\epsilon, A) = \begin{cases} (1 + \epsilon)A & A \geq 0 \\ (1 - \epsilon)A & A < 0. \end{cases}$$

The function $g(\epsilon, A)$ defines how the advantage A is scaled within the context of the PPO-Clip algorithm. If the advantage is non-negative, suggesting the action is better than the current policy's average, it's scaled by $1 + \epsilon$ if the advantage is negative, indicating a less beneficial action, it's scaled by $1 - \epsilon$. This adjustment is part of the clipping mechanism that keeps the policy updates conservative, ensuring that the policy does not move too far from the old policy during an update step. The clipping mechanism acts as a safeguard, limiting abrupt changes in the policy by setting boundaries on how much it can deviate from its previous version. The ϵ hyperparameter defines the extent to which the new policy can differ from the old and still benefit the overall goal.

The Proximal Policy Optimization (PPO) algorithm, operates on a cyclical process to refine the policy parameters for decision-making in an environment. It commences with a given set of initial parameters for the policy and value function. Through interaction with the environment, the algorithm collects trajectories, which are sequences of states, actions, and rewards experienced by the agent.

From these trajectories, the algorithm computes "rewards-to-go," which are estimations of future rewards that an agent can expect to receive from a particular state. These calculations are essential for understanding the potential benefit of actions taken in specific states. Concurrently, the algorithm estimates the "advantage" of actions, which reflects the relative value of the chosen action compared to the average action at a given state, based on the current policy.

The core of PPO is the policy update rule, where the parameters are adjusted to maximize an objective function. This function incorporates the probability ratio between the new and the old policy's likelihoods of having selected a particular action in a given state, modulated by the calculated advantage. To ensure stability and avoid drastic policy changes, PPO introduces a clipping mechanism that bounds the probability ratio, keeping it within a narrow interval around 1, denoted by $1\pm\epsilon$.

This objective is optimized using stochastic gradient ascent, often implemented with algorithms like Adam, which is a popular choice due to its adaptive learning rate capabilities. The value function parameters are also refined by regressing on the mean-squared error between the value function's predictions and the observed rewards. This regression helps the value function better predict the expected return from each state, which is crucial for accurate advantage estimation in subsequent iterations. By repeatedly iterating through these steps—sampling trajectories, computing advantages, updating policies within a controlled range, and refining value functions—PPO efficiently improves the policy. This results in an agent that is better equipped to make decisions that will maximize cumulative rewards over time, all while maintaining a careful balance between exploration of new strategies and exploitation of known, successful behaviors.

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

end for

PPO employs an on-policy approach to optimize a stochastic policy, meaning it iteratively refines its action-selection strategy based on the most recent policy iteration. As training progresses, the policy's stochastic nature diminishes, favoring actions with known rewards, which could potentially lead to convergence on local optima due to a decrease in exploratory behavior. When compared to VPG, PPO is generally considered more resilient when it comes to getting trapped in local optima. PPO is more resilient due to its objective function, which uses a clipping mechanism to limit the extent of policy updates, fostering more gradual learning and reducing the risk of overfitting to suboptimal strategies. This clipping ensures that the policy doesn't change too drastically, allowing it to maintain beneficial exploratory behavior while still improving the exploitation of known rewards.

3.4.5 TRPO vs PPO – A Preliminary Comparison

Trust Region Policy Optimization and Proximal Policy Optimization are both advanced policy gradient methods in reinforcement learning, designed to train policies for sequential decision-making tasks. Their relationship stems from the shared objective of effectively balancing the exploration-exploitation trade-off while ensuring stable policy updates.

Both TRPO and PPO emerge from the same lineage, aiming to address the shortcomings of earlier policy gradient methods, like VPG, which could suffer from high variance in policy updates and potentially get trapped in local optima. TRPO introduces a trust region approach, enforcing a constraint on the KL divergence between successive policies to maintain a controlled exploration of the policy space. This constraint is derived from the natural policy gradient method and ensures that the policy does not update too aggressively, which might lead to suboptimal performance.

PPO, developed after TRPO, shares the motivation to prevent drastic policy updates but aims for greater practicality. Instead of the computationally demanding second-order methods required by TRPO, PPO simplifies the optimization process by using first-order methods that are easier to implement. PPO’s primary contribution to practicality is its two variants: PPO-Clip and PPO-Penalty. PPO-Clip, in particular, uses a clipping function in the objective to prevent the ratio of new to old policy probabilities from exceeding a predetermined threshold. This allows for a more straightforward and computationally efficient implementation, which is particularly advantageous in large-scale applications or when resources are limited.

Efficiency in reinforcement learning not only pertains to computational resources but also to sample efficiency – the ability to learn effective policies from fewer interactions with the environment.

Here, PPO sometimes outperforms TRPO due to its simpler optimization procedure, which allows for more frequent policy updates with less computational overhead. PPO's ability to reuse data more effectively through multiple epochs of minibatch updates further contributes to its sample efficiency.

The differences in mechanisms for avoiding local optima between TRPO and PPO are subtle yet significant. TRPO's strict trust region defined by the KL divergence provides strong theoretical guarantees for monotonic improvement, which is a powerful deterrent against poor local optima. PPO, while still maintaining some of these guarantees, relies on its clipping mechanism to offer a more forgiving and less rigid update process. This can sometimes result in more significant policy changes, provided they stay within the clipping bounds, allowing PPO to potentially escape suboptimal policies that TRPO might adhere to more conservatively, but also runs the risk of stepping into local optima.

In summary, while TRPO and PPO are closely related in their goals and methods, they differ in practicality, efficiency, and their approach to maintaining stable learning progress. PPO's design as a more practical and efficient algorithm, coupled with its clipping mechanism to prevent drastic policy updates, allows it to be widely adopted in various applications, from robotics to gaming. On the other hand, TRPO's more conservative updates, grounded in a strong theoretical foundation, make it a robust choice when computational resources allow for its more intricate implementation. Each algorithm has its strengths and ideal use cases, making them both valuable tools in the arsenal of modern reinforcement learning.

In our upcoming study, we plan to test these widely accepted notions about the performance of Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO). This examination is crucial for understanding the practical implications of these algorithms under specific conditions. One thing that is important to remember is that we will be using these optimizers within the framework of Generative Adversarial Imitation Learning, so we have to be careful when generalizing our findings. A significant focus will be on the dataset employed for this study, as its characteristics are fundamental to the validity and relevance of our findings. The choice of dataset not only influences the complexity of the learning environment but also highlights the adaptability and efficiency of TRPO and PPO in real-world scenarios. By situating our comparison within the GAIL structure, we aim to provide a comprehensive analysis of how these algorithms support learning to imitate expert behavior, manage exploration and exploitation, and navigate the challenges posed by the learning environment.

4. The Trajectory Imitation Problem

4.1 Trajectory prediction as a data-driven task

In the aviation domain, a trajectory is defined as the description of movement of an aircraft both in the air and on the ground [12]. Aircraft trajectories can be outlined as a time-ordered series of states, characterized by variables such as airspeed, three-dimensional position (latitude, longitude, and geodetic altitude), direction (bearing or heading), and current mass. Trajectories that only include spatial and temporal data (i.e. 3D positions and timestamps), which are called raw trajectories, can be identified using surveillance data.

A raw trajectory (T) is comprised of a series of 4-dimentional trajectory states. A trajectory state (s_i) is an array that contains the longitude (l), latitude(f), geodetic altitude (h) as well as a timestamp t_i. We can represent a trajectory state as $s_i = \langle p_i, t_i \rangle$, where p_i represents the geospatial point of the trajectory state. The length of a raw trajectory T is equal to the number of trajectory states it contains [13]. By adding more categorical and/or numerical variables in a trajectory state we create an enriched trajectory that contains enriched trajectory points. An enriched trajectory point is a triplet $s_{r,i} = \langle p_i, t_i, v_i \rangle$ where v_i is a vector containing all the additional variables. For the purpose of this thesis we will be using enriched trajectories that also contain the fuel consumption (kg/h).

A predicted trajectory is essentially the anticipated progression of an aircraft's status, determined by (a) the present flying conditions (such as the starting state under current weather conditions), (b) predictions of relevant contextual factors (like expected weather conditions at certain locations), and (c) a strategy for how the aircraft will move through successive states from its starting point forward, in other words, a "policy" guiding the development of the trajectory [12].

Reframing the trajectory prediction issue as a data-driven task and presuming a set $T_E = \{T_{E_i} \mid i = 1, 2, 3, ...\}$ of enhanced trajectory data, the problem of predicting trajectories can be articulated as follows: With T_E and a "cost" function c, the objective is to forecast a trajectory T_π that satisfies the following equation:

$$T_\pi \sim \underset{\pi}{\operatorname{argmin}} \ \mathbb{E} \sum_{s} [C(s, a) | \pi]$$

Here, $\mathbb{E}$ represents the anticipated aggregate costs for all states s created while following a policy $\pi(a \mid s)$ that indicates the likelihood of choosing an action a given an augmented state $s = (p, t, v)$, where the discussion about states and actions is to be elaborated later (as mentioned in the next paragraph). In essence, the primary goal is to identify the policy π that leads to a trajectory T_π with the least expected cumulative cost.

4.2 Approaching the flight trajectory prediction problem

When analyzing a collection of flight trajectories between any two airports, one may reveal varying behavioral patterns. These patterns are likely influenced by distinct contextual elements that shape how stakeholders decide on the development of a flight's trajectory. For example, the selection of a landing runway at the destination airport might be influenced by factors such as local weather conditions, air traffic, or the preferences of the airline, leading to notably diverse flight paths. To advance the automation of data-driven trajectory prediction, it is crucial to identify and understand these unique trajectory patterns and the distinguishing characteristics of each. Subsequently, we could formulate specific policies for each identified trajectory category, corresponding to a certain behavior pattern. This tailored approach could greatly enhance the efficiency and effectiveness of the learning process compared to the creation of a single, all-encompassing model. Nevertheless, to accurately predict an individual trajectory, it's necessary to ascertain the applicable policy, hence determining the most probable behavioral pattern it will exhibit. A potential method to achieve this is to anticipate the contextual factors likely to influence the trajectory's progression and to categorize the forthcoming trajectory based on these factors. This categorization process should focus solely on those features that are predictive of different behavioral patterns and that can be anticipated or known at any stage of Air Traffic Management (ATM) operations.

For the purpose of our study we will be skipping the classification step, since it introduces more complexity and noise to our main goal which is comparing TRPO and PPO in the GAIL context. We will be making clusters of trajectories with the same main features (origin, destination and load capacity). Then we will be applying the trajectory imitation step. But let’s first talk a little bit about our dataset.

4.3 Data overview & pre-processing

The data we will be using will be flight plans from Paris, France to Constantinople, Turkey. These flight plans will be for a specific payload capacity of the aircraft. This is done to skip the classification mentioned in the previous section. We will be using a cluster of 43 enriched trajectories which contain the longitude, latitude, geodetic altitude and the fuel consumption at any given time. Since the time steps of these flight plans are not linear we will be interpolating the data to create a more uniform time series for analysis. Interpolating between the recorded data points to estimate values at regular intervals will help in smoothing out the trajectory data and ensuring that changes in direction, altitude, or fuel consumption are captured in a consistent time step. Each step will be 10 seconds, so that we have a little over 1150 data points for every flight.

For every step in our analysis, corresponding to a 10-second interval, we will calculate the difference in each measure—longitude, latitude, geodetic altitude, and fuel consumption—compared to the previous step. This method allows us to quantify the incremental changes occurring within each interval, providing a granular view of the trajectory's evolution. By focusing on the differences for every measure, we can more accurately capture and analyze the dynamics of the flight trajectory, including subtle changes in direction, altitude adjustments, and variations in fuel consumption.

Incorporating this method of calculating first-order differences at every step also proves beneficial when the changes at each interval are minor compared to the overall magnitude of a variable. This approach enhances the analysis by highlighting even the smallest shifts in trajectory, altitude, or fuel consumption that might otherwise be overlooked when considering the total values alone. This in turn helps our model more easily and accurately predict the change of the value of each respective variable.

4.4 Approaching the imitation problem

Let us assume $T_E = \{T_{E,i} \mid i = 1, \ldots, N\}$ a set of historical, raw or enriched aircraft trajectories, depending on the experiment, generated by an expert policy π_E, in this case the flight plans. The goal is to identify a policy that effectively reduces the disparity between the anticipated total cost of the predicted trajectories and that of the actual trajectories. This goal is equivalent to discovering a policy π that aligns the distribution of state-action pairs it generates as closely as possible with the distribution of state-action pairs observed in the expert trajectories [4]. Our aim is to track the trajectory's spatial progression as well as the fuel consumption at Δ_t intervals (in our case t=10s).

The agent action A is the quartet $\Delta_l, \Delta_f, \Delta_h, \Delta_k$ where l is longitude, f is latitude, h is geodetic altitude and k is fuel consumption. This quartet specifies the 4 dimensions of the aircraft's state, given the requirement that this difference must be achievable within the consistent time interval Δ_t under consideration. The input of the agent is calculated by applying the changes provided by the agent action to the s_0. In that way we deterministically arrive to s_1 which can be then used as input for the agent, and so on and so forth. s_1 can be represented as the quartet $(l_0 + \Delta_l, f_0 + \Delta_f, h_0 + \Delta_h, k_0 + \Delta_k)$.

We will also be experimenting with breaking down our main objective into 2 different objectives. We will train one agent to predict the raw objective (using only spatial variables) and one to predict the fuel consumption. When training, each state will be the same for both models, and will be equal to the spatial part of the state mentioned above. However, when testing, each subsequent state will be calculated by using the actions suggested by both agents.

5. Algorithm Structure & Methodology

5.1 Behavioral Cloning as Pre-training

Behavioral Cloning (BC) serves as an effective strategy to accelerate the training of agents in reinforcement learning tasks. By directly imitating expert demonstrations, BC provides a quick way to establish a competent baseline policy, bypassing the initial trial-and-error phase that often characterizes the early stages of learning in complex environments. This rapid acquisition of a reasonably good policy can significantly reduce the time and computational resources required for an agent to reach proficiency. Moreover, starting with a BC pre-trained model can help in navigating the exploration-exploitation trade-off more effectively, as the agent already has a foundational understanding of the task at hand. This approach is especially beneficial in scenarios where exploration costs are high or can lead to undesirable outcomes. Therefore, integrating BC as a preliminary step in training regimes can lead to more sample-efficient learning processes, setting a solid groundwork upon which more sophisticated learning algorithms, like GAIL, can further refine and optimize the agent's performance. For those reasons, before training each GAIL models, in all instances, we will be first pre-train the actor (policy) network in a BC environment. This will be done for 100 epochs with k-fold cross validation where k=10. In this phase almost 76.000 state-action pairs will be used.

5.2 Using TRPO - The original GAIL algorithm

In their original paper [4] Ho and Ermon introduced in 2016 the GAIL algorithm that used TRPO as its' agents' optimizer. At the time it was arguably the best option. Trust Region Policy Optimization had made significant progress in addressing the primary challenges associated with Vanilla Gradient Policy methods. This combination of a GAN type of structure, paired with the latest at the time gradient policy optimizer allowed for the development of more robust and efficient learning algorithms, capable of learning complex behaviors from expert demonstrations without explicit reward functions.

In our experiments we trained our GAIL with TRPO model for 3000 epochs, using almost 76.000 state-action pairs for each epoch with a 20.000 bach-size. This, as mentioned above, will be done after the pre-training part, and will serve as a standard baseline for our comparison for GAIL with PPO, a less documented combination. The TRPO hyper-parameters are as follows: The damping coefficient (which ensures the conjugate gradient method can find a stable direction for policy updates) is set to 0.1. The maximum Kullback-Leibler (KL) divergence δ between the new and old policies

during a policy update, which ensures that the policy update doesn't deviate too much, is set to 0.01. Lambda, which refers to the scaling factor applied to the gradient of the discounted causal entropy, is set to 1e-4. The gamma hyper-parameter of the Generalized Advantage Estimation (GAE), which influences how much the agent prioritizes immediate rewards over future ones, is set to 0.995. The lambda hyper-parameter of the GAE, which controls the trade-off between bias and variance in the advantage estimates, is also set to 0.995. Lastly, the epsilon parameter, which is used to calculate the step size (alpha) for updating the policy parameters, is set to 0.01.

5.3 Using PPO - The alternative GAIL setup

When GAIL was introduced in 2016 PPO hadn't been developed yet. PPO, introduced shortly after GAIL, simplifies many of TRPO's complexities while retaining its benefits. It uses a clipping mechanism to prevent overly large policy updates, making it computationally more efficient and easier to implement than TRPO. This efficiency and simplicity could potentially make PPO a better optimizer for GAIL by speeding up the learning process, reducing computational demands, and making the implementation more accessible. Additionally, PPO's flexibility in balancing exploration and exploitation might lead to improved learning outcomes when combined with GAIL's framework.

In our experiments we trained our GAIL with PPO model for 3000 epochs, using, again, almost 76.000 state-action pairs for each epoch with a 20.000 bach-size. This, again, will be done after the pre-training part, and will serve as a standard baseline for our comparison for GAIL with PPO, a less documented combination. The TRPO hyper-parameters are as follows: The clipping parameter used to limit the ratio of policy updates is set to 0.1. The gamma hyper-parameter of the GAE, which influences how much the agent prioritizes immediate rewards over future ones, is set to 0.995. The lambda hyper-parameter of the GAE, which controls the trade-off between bias and variance in the advantage estimates, is also set to 0.995.

5.4 Setup Description

In our experiments we have used two types of setups: one where we use one agent that predicts the following state on its own, and one where we use two agents. We will refer to the first setup as the SingleModel setup and to the second one as the DoubleModel setup.

5.4.1 SingleModel

The single model setup is shown in the Diagram 1. It's input is the normalized spacial and fuel consumption data points of the initial state. A single agent predicts the difference in between the initial values of s_0 and the ones of the next state s_1. By adding that difference to the original values we get the next state s_1 that can then be used as input. By doing for every step we create the full predicted trajectory.

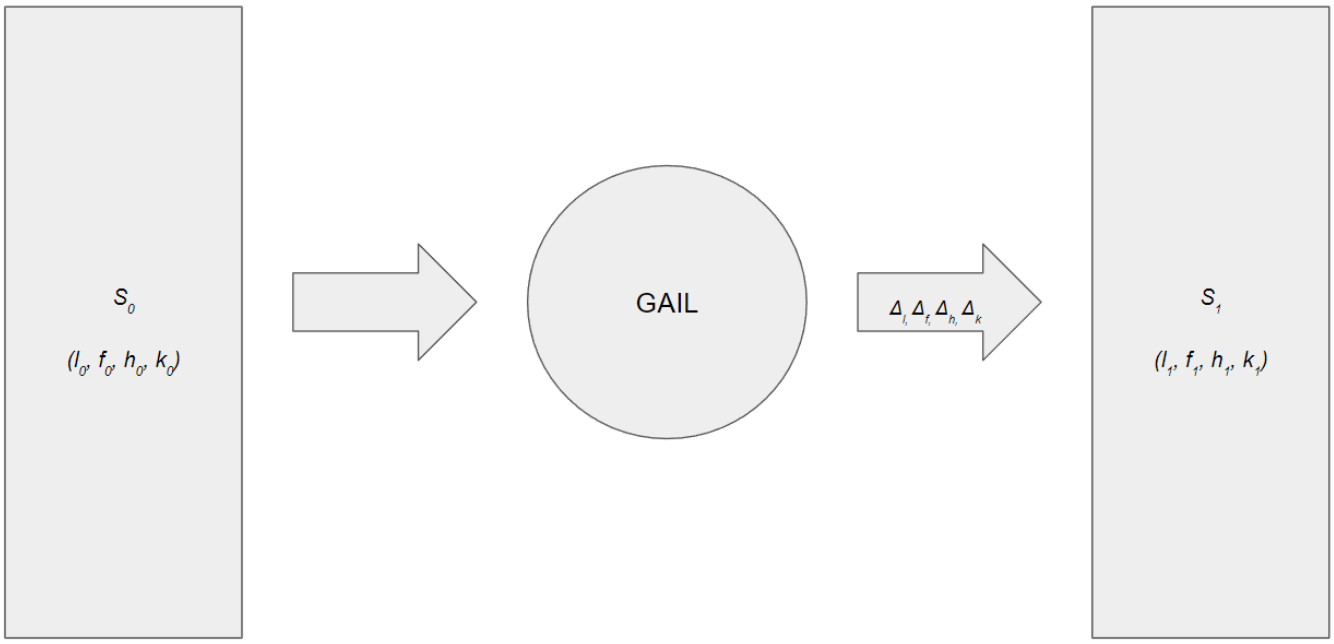

Diagram 1. The SingleModel setup

5.4.2 DoubleModel

The double model setup is shown in the Diagram 2. The input of this setup is the normalized spacial and fuel consumption data points of the initial state. This input is used in two agents. The first one is responsible for predicting the difference in between the initial spatial values of s_0 and the ones of the next state s_1 and the second one for predicting the difference in between their fuel consumption values. By adding the difference to the values of s_0 we get s_1. By doing for every step we create the full predicted trajectory.

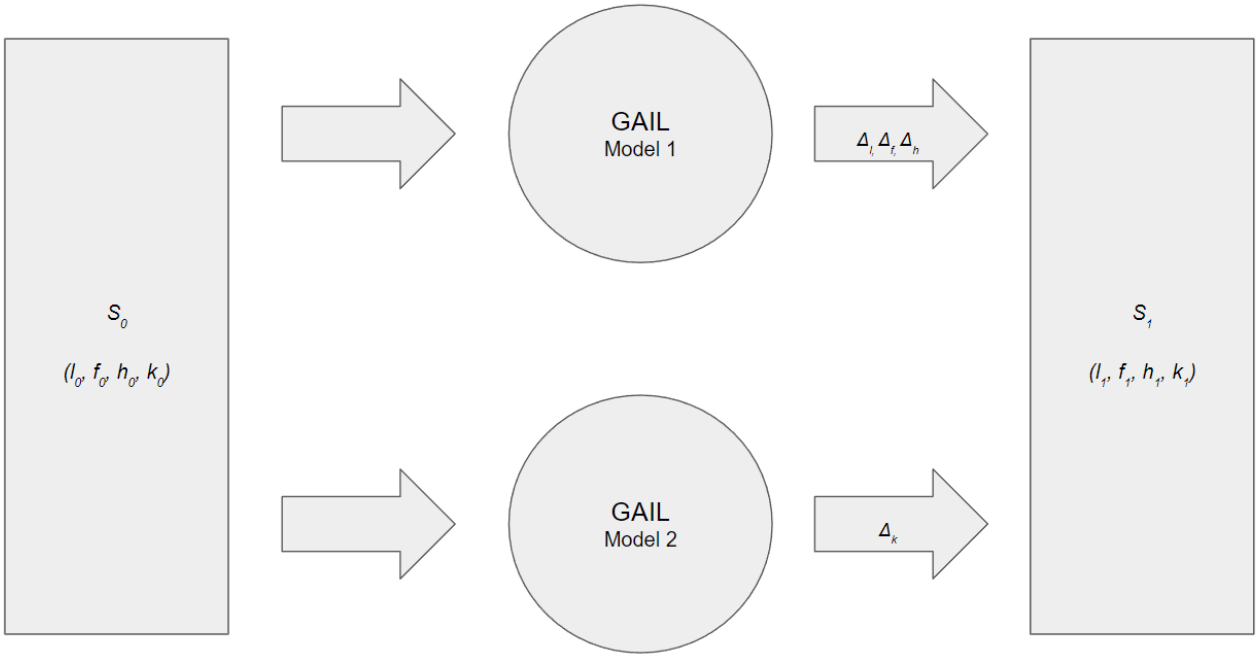

Diagram 2. The DoubleModel setup

6. Experimental Results

6.1 Evaluation Measures

Before delving into our results let's take a look at the evaluation measures we will be using. By incorporating a broad spectrum of evaluation criteria—including Root Mean Square Error for overall trajectory prediction accuracy, along-track and cross-track errors for horizontal spatial precision, and altitude difference for vertical accuracy—we aim to provide a comprehensive analysis of model performance. This section explains the definition, importance, and application of each measure, ensuring clarity and facilitating the understanding of how these measures collectively contribute to a nuanced evaluation of predictive capabilities. Our chosen measures are instrumental in identifying strengths and pinpointing areas of improvement, thereby driving advancements in trajectory prediction research.

6.1.1 Root Mean Square Error (RMSE) and Mean Absolute Error (MAE)

RMSE is a widely used metric for quantifying the accuracy of predictions, representing the square root of the average squared differences between predicted and actual values. It is crucial because it provides a clear measure of model performance by highlighting the magnitude of prediction errors. RMSE is particularly valued for its sensitivity to large errors, ensuring that models are penalized more significantly for big mistakes. This characteristic makes RMSE indispensable for comparing different models or algorithms, ensuring the selection of the most accurate and reliable one for predicting outcomes across various fields, from finance to weather forecasting.

We will use RMSE mainly for evaluating the fuel consumption prediction by our models. While RMSE is valuable for evaluating the overall accuracy of trajectory predictions, it falls short in offering insights into the nature of an agent's failures. RMSE aggregates errors into a single measure, obscuring specific error types such as along-track, cross-track, and altitude discrepancies. In trajectory prediction, understanding these specific errors is crucial for diagnosing model weaknesses and improving navigational accuracy. For instance, a model might accurately predict trajectory distance but consistently fail in lateral positioning, a nuance RMSE alone cannot reveal. Therefore, while RMSE is useful for a general accuracy assessment, it is somewhat inadequate for fully understanding and addressing the complexities of trajectory prediction errors. For this reason we will be introducing the measures mentioned to get more meaningful insights to our agent's performance.

Mean Absolute Error (MAE) is a fundamental metric employed to evaluate the accuracy of predictions by computing the average magnitude of the absolute differences between predicted and actual values. This metric is pivotal for its simplicity and direct interpretation, offering a straightforward measure of prediction accuracy without complicating the understanding with squared terms, as seen in RMSE. MAE is particularly appreciated for its robustness to outliers, as it does not disproportionately penalize large errors like RMSE does.

6.1.2 Along-track error (ATE)

Along-track error is a critical measure in trajectory prediction problems, quantifying the longitudinal deviation of a predicted path from the actual trajectory. This measure assesses how well a predictive model can replicate the timing and pacing of an object's movement along its path. In essence, it reflects the distance by which the predicted position lags behind or exceeds the true position along the direction of travel at any given point in time. This measure is especially important in applications where precise timing and speed control are crucial, such as autonomous vehicle navigation, aircraft flight path prediction, and maritime routing.

The significance of along-track error lies in its ability to evaluate the predictive accuracy of temporal aspects of a trajectory. For autonomous systems, minimizing along-track error is imperative to ensure synchrony with real-world dynamics, enabling vehicles to arrive at destinations on time, maintain optimal spacing with other objects, and efficiently adjust to changes in velocity. In aviation, accurate along-track predictions are vital for flight scheduling, airspace management, and avoiding potential conflicts with other aircraft by ensuring adherence to assigned flight paths and times.

Moreover, along-track error provides insights into the model's understanding of the underlying physics and dynamics governing the movement of objects. A low along-track error indicates a model that can accurately forecast the speed and acceleration patterns over time, reflecting a deep comprehension of the factors influencing the trajectory. Consequently, it is a crucial measure for evaluating the performance of trajectory prediction algorithms, contributing to the development of safer, more reliable, and efficient navigation systems across various domains.

6.1.3 Cross Track Error (CTE)

CTE is a pivotal measure in evaluating trajectory prediction algorithms, measuring the lateral deviation of a predicted trajectory from the actual path. This measure assesses the accuracy of a model's ability to predict the precise lateral position of an object as it moves, essentially quantifying how far to the side

(either left or right) the predicted path strays from the intended course. Cross-track error is crucial in contexts where maintaining a specific route is vital for safety and efficiency, such as in autonomous driving, aircraft navigation, and maritime operations.

The importance of cross-track error stems from its direct impact on navigational precision and the avoidance of obstacles or hazardous zones. In autonomous driving, for example, minimizing cross-track error is essential for ensuring that vehicles stay within their lanes, navigate turns accurately, and avoid veering off course, which could lead to accidents or inefficient routes. Similarly, in aviation, small deviations can significantly affect flight efficiency and safety, making the accurate prediction of lateral movements a necessity for adherence to flight corridors and the avoidance of airspace conflicts.

Furthermore, cross-track error serves as an indicator of a model's ability to understand and predict the dynamics influencing directional changes. A low cross-track error signifies a model that can accurately forecast shifts in direction and handle complex maneuvers, reflecting a sophisticated grasp of environmental and operational factors affecting the trajectory. As such, cross-track error is an indispensable measure for the development and evaluation of trajectory prediction models, playing a critical role in enhancing the reliability and safety of navigational systems across various transportation sectors.

6.1.4 Altitude Error

Altitude difference is an essential measure in trajectory prediction, especially in applications where vertical positioning is critical, such as our case - aviation. It measures the vertical discrepancy between the predicted and actual trajectories, offering a precise assessment of a model's ability to predict altitude changes over time. This measure is crucial for ensuring compliance with flight safety regulations, optimizing airspace usage, and preventing collisions by maintaining prescribed flight levels. Additionally, accurate altitude predictions contribute to fuel efficiency and environmental protection by enabling more efficient flight paths. The ability to minimize altitude difference enhances the reliability and safety of navigational systems, making it a valuable measure for evaluating the performance of trajectory prediction algorithms in three-dimensional space.

6.2 SingleModel Results

6.2.1 Training graphs

Figures 1 and 2 show the training graphs for GAIL with TRPO and PPO after the pre-training was completed. Here the RMSE is referring to all the output variables. These visual representations allow for direct comparison between optimization strategies, highlighting how quickly and reliably each model learns to imitate desired behaviors.

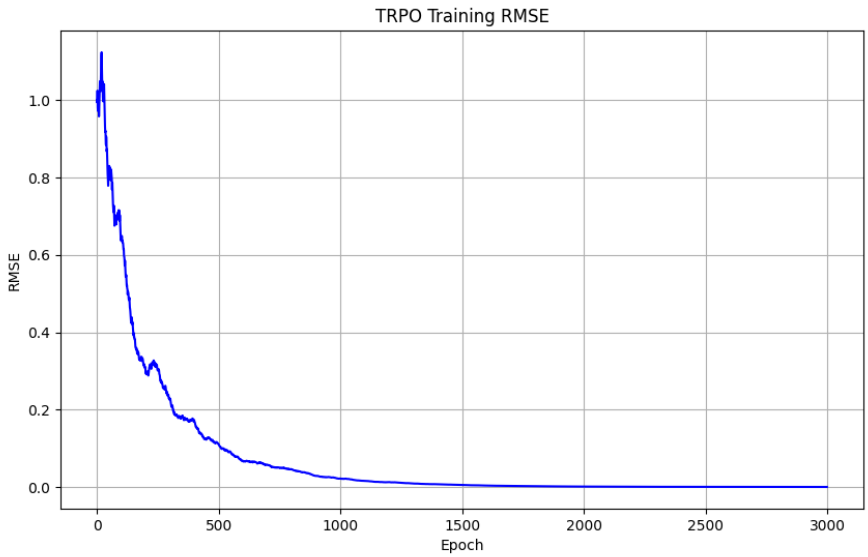

Figure 1. SingleModel GAIL with TRPO RMSE graph while training

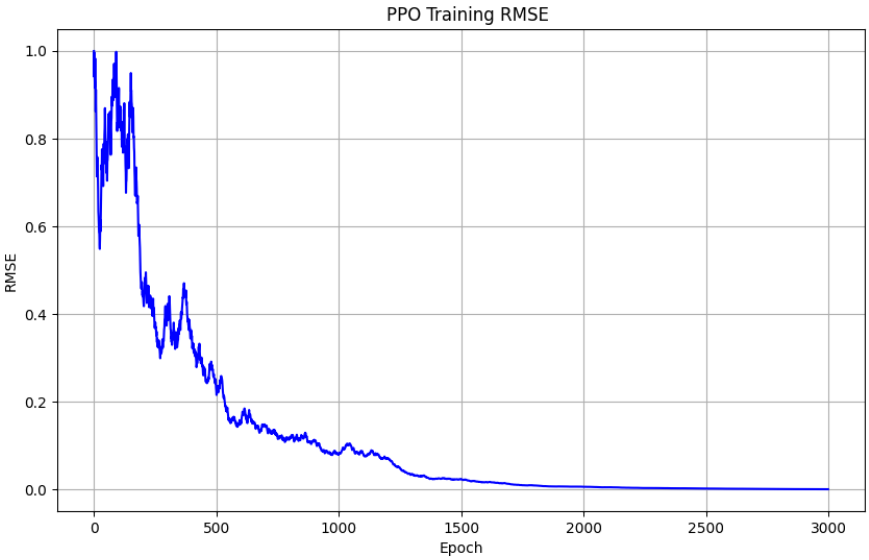

Figure 2. SingleModel GAIL with PPO RMSE graph while training

The PPO training RMSE graph features a steep initial decline followed by notable fluctuations and gradual convergence, indicating variability in learning efficiency. Conversely, the TRPO graph presents a more consistent and smooth decline towards convergence, reflecting a more stable learning trajectory. Both methods ultimately reach a plateau, suggesting at least some learning stabilization, but TRPO appears to achieve this with less volatility and faster compared to PPO.

6.2.2 Trajectory Graphs

In Figures 3-4, we can see the best case and worst case scenarios of the GAIL models using behavioral cloning for pre-training, with TRPO and PPO respectively. All the ‘best’ and ‘worst’ trajectories, including the following ones, are chosen based on 100 trajectories created by each agent respectively. Those trajectories are used to calculate the measures shown in tables 1 and 2.

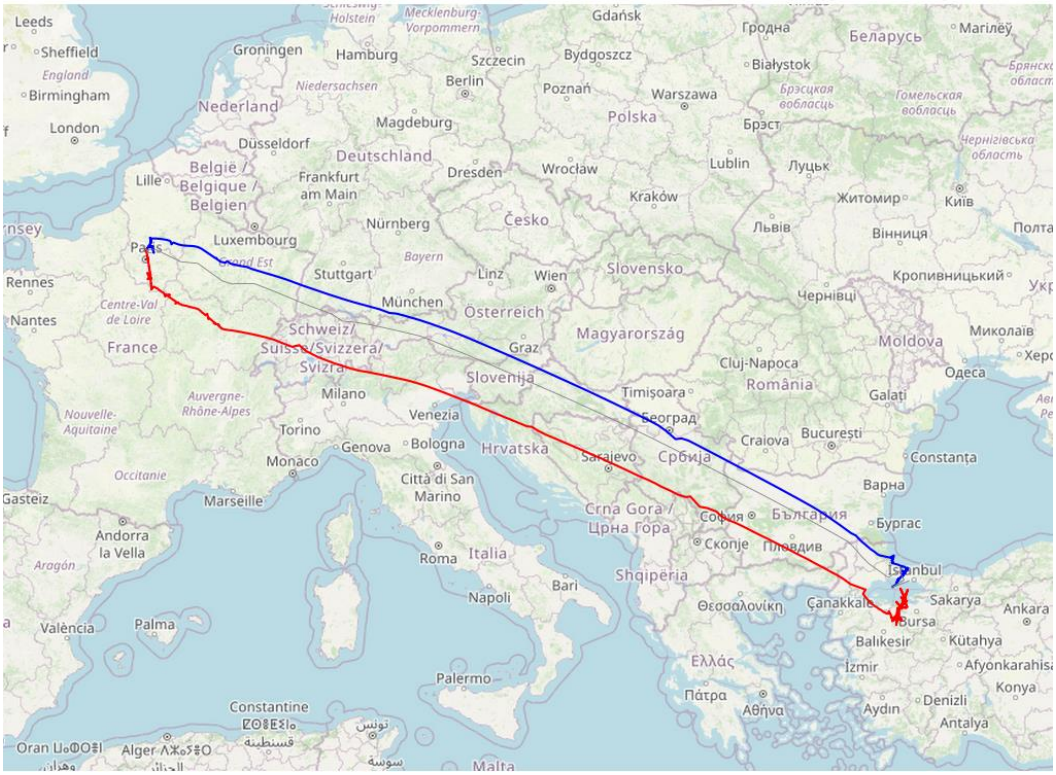

Figure 3. The best (blue) and worst (red) trajectories of the SingleModel GAIL with TRPO agent when training a single model. The gray line is an one of the expert trajectories included in the training cluster.

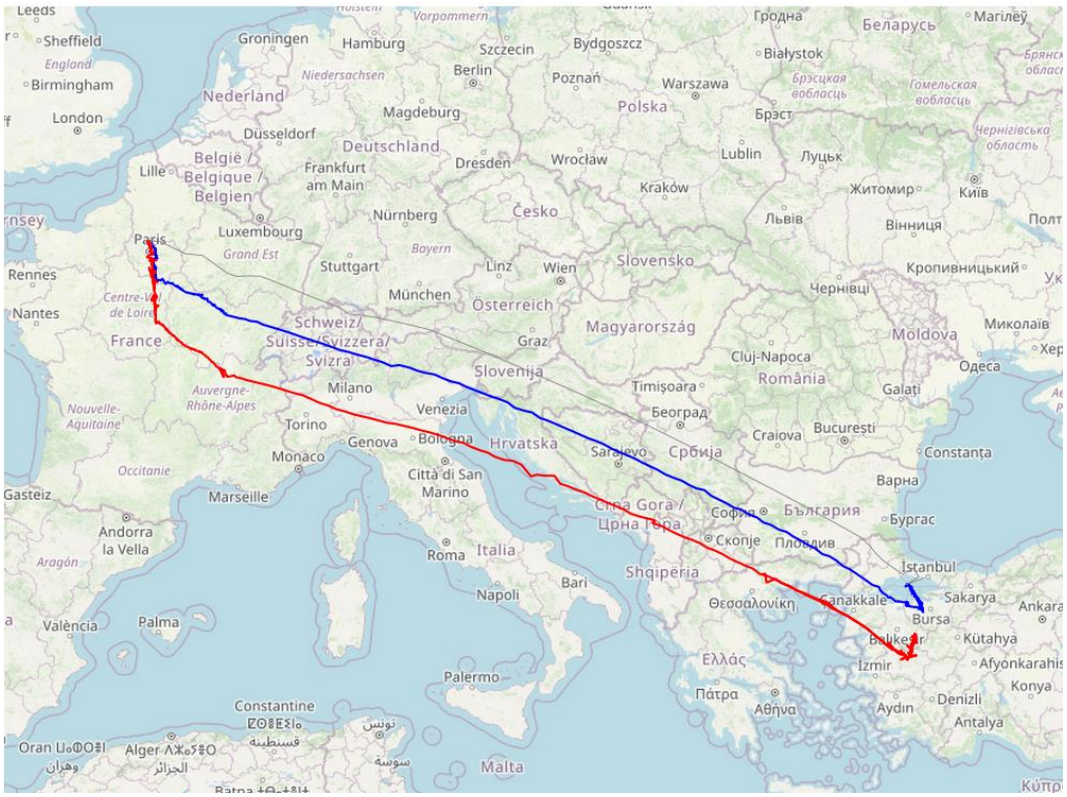

Figure 4. The best (blue) and worst (red) trajectories of the SingleModel GAIL with PPO agent when training a single model. The gray line is an one of the expert trajectories included in the training cluster.

The TRPO trajectory more closely follows the expert's path, showing tighter adherence and fewer deviations. In contrast, the PPO trajectory exhibits greater divergence from the expert's route as well as more (unrealistic) sharp turns. TRPO also shows that it can better approach the target, in this case Constantinople. Both the TRPO and PPO trajectories demonstrate challenges during takeoff and landing phases, probably due to these segments likely represent more complex maneuvers compared to the rest of the trajectory.

We can see the predicted fuel consumption of the TRPO and PPO models in figures 3 and 4.

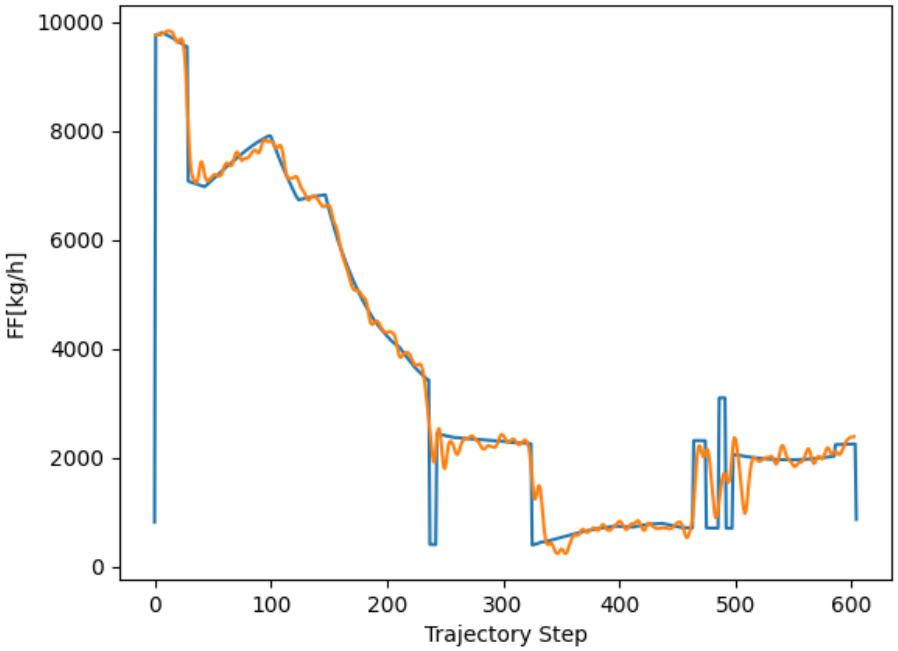

Figure 5. Fuel consumption predicted by SingleModel GAIL with TRPO agent when training a single model. The orange line is the agent's prediction, while the blue is the expert's actions.

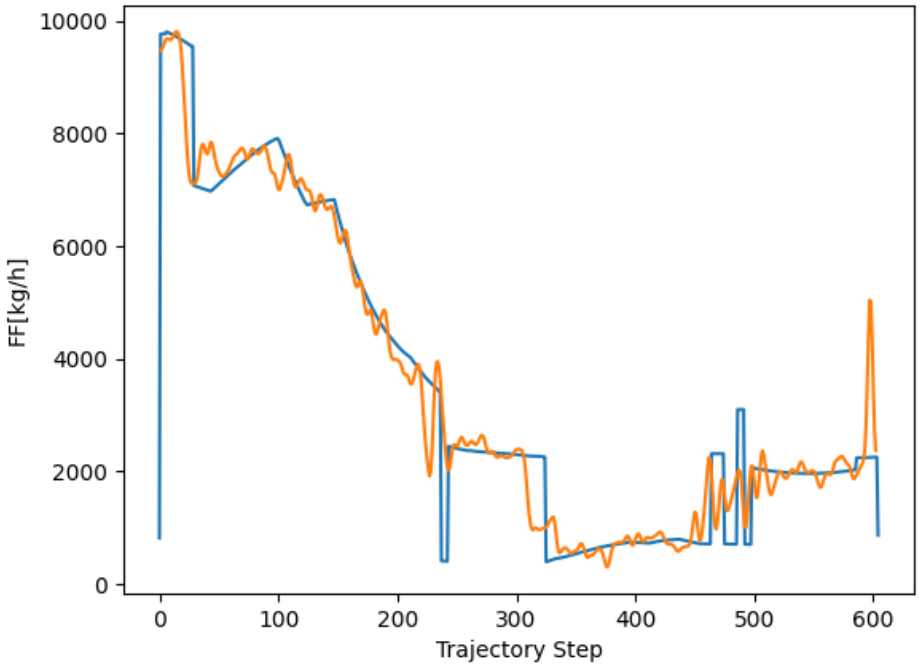

Figure 6. Fuel consumption predicted by SingleModel GAIL with PPO. The orange line is the agent's prediction, while the blue is the expert's actions.

Both TRPO and PPO algorithms generally follow the expert's fuel consumption trends but exhibit challenges in capturing sharp changes in fuel usage. The TRPO algorithm, appears to align more closely with the expert's consumption pattern and has less variability, suggesting it performed better than PPO in replicating the expert's efficiency.

6.3 DoubleModel Results

6.3.1 Training graphs

Figures 7, 8, 9, 10 show the training graphs for GAIL with TRPO and PPO after the pre-training was completed. In the case of the DoubleModel setup, we showcase the learning curve of the two agents (one predicting the 3D features and the other predicting the fuel consumption) for each approach. So in total we have 4 graphs, 2 graphs for each optimizer, one for the geo-location prediction one for the fuel consumption prediction.

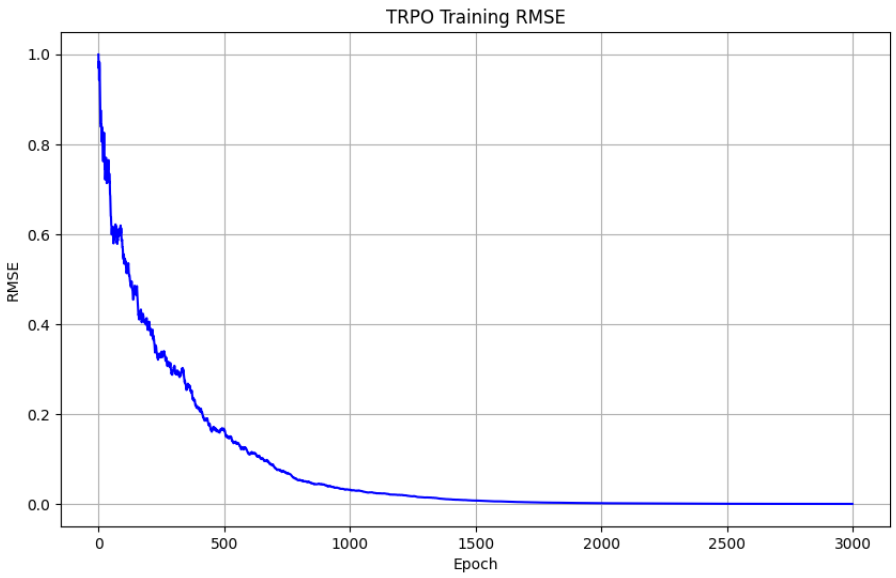

Figure 7. DoubleModel GAIL with TRPO RMSE graph while training – 3D Location Model

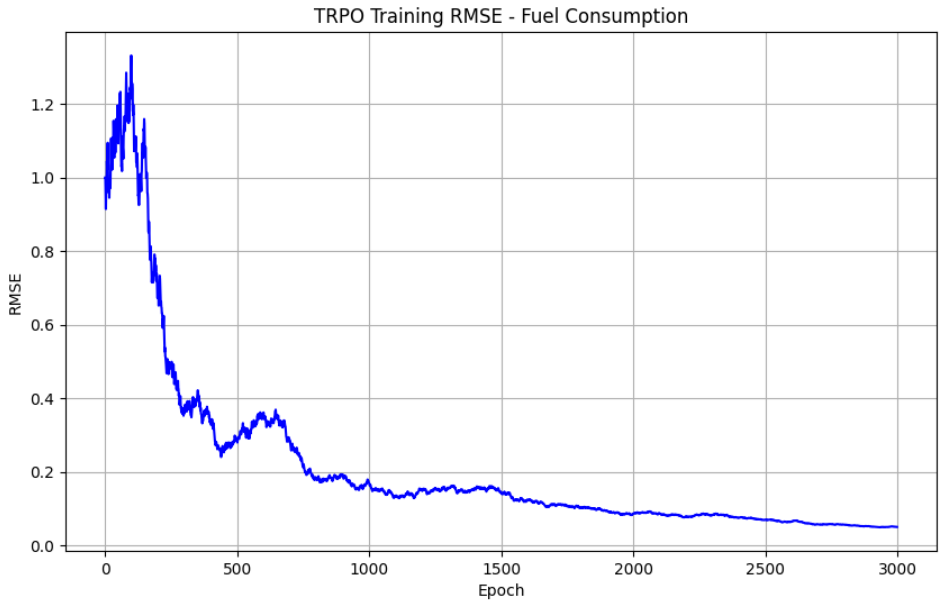

Figure 8. DoubleModel GAIL with TRPO - RMSE training graph – Fuel Consumption Model

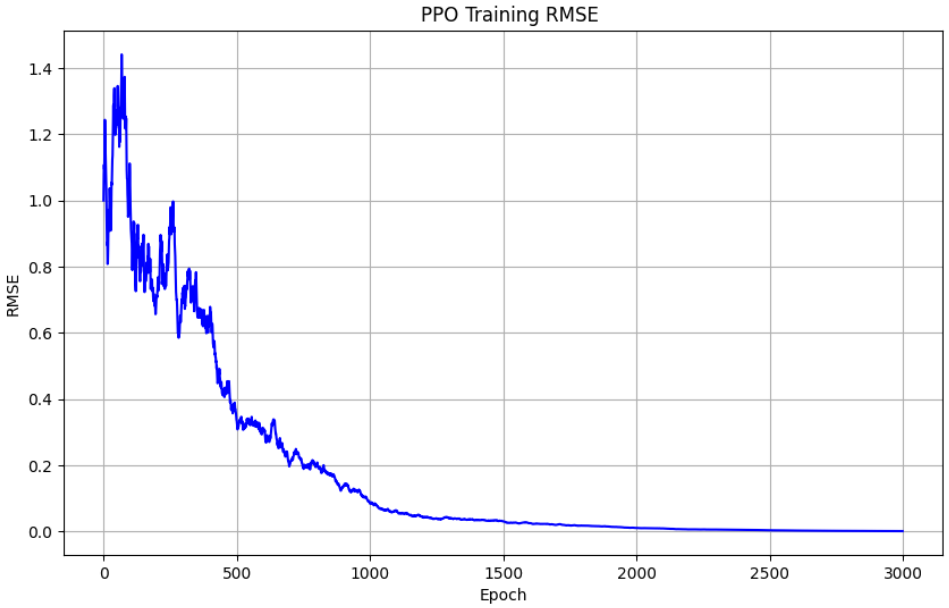

Figure 9. DoubleModel GAIL with PPO - RMSE training graph – 3D Location Model

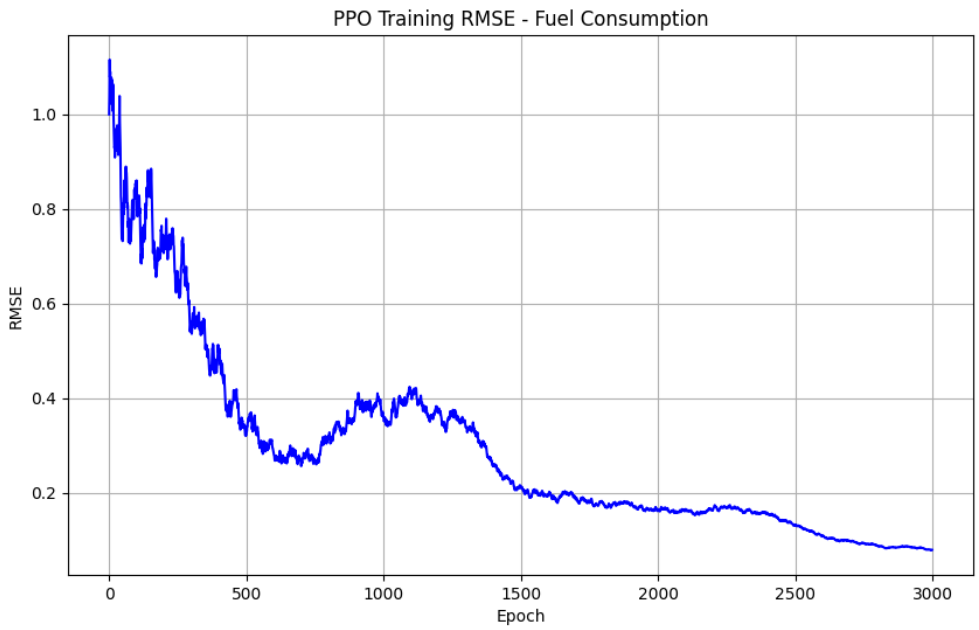

Figure 10. DoubleModel GAIL with TRPO - RMSE training graph – Fuel Consumption Model

The RMSE graphs of the DoubleModel setup have clear similarities to the SingleModel ones. PPO again displays notable fluctuations and gradual convergence, indicating variability in learning efficiency. The TRPO graph presents a smoother and faster decline towards convergence, reflecting more stable learning. The training of the 3D Location Model for each optimizer seems to have a better convergence, although that could be largely explained by the nature of the normalized data, where, in the case of fuel consumption, has a larger range and variability in the flight plans.

6.3.2 Trajectory Graphs

In Figures 11-12, we can see the best case and worst case scenarios of the GAIL models using the 3D Location Model, after behavioral cloning for pre-training, with TRPO and PPO respectively. All the ‘best’ and ‘worst’ trajectories, including the following ones, are chosen based on 100 trajectories created by each agent respectively. Those same trajectories are used to calculate the measures shown in tables 1 and 2.

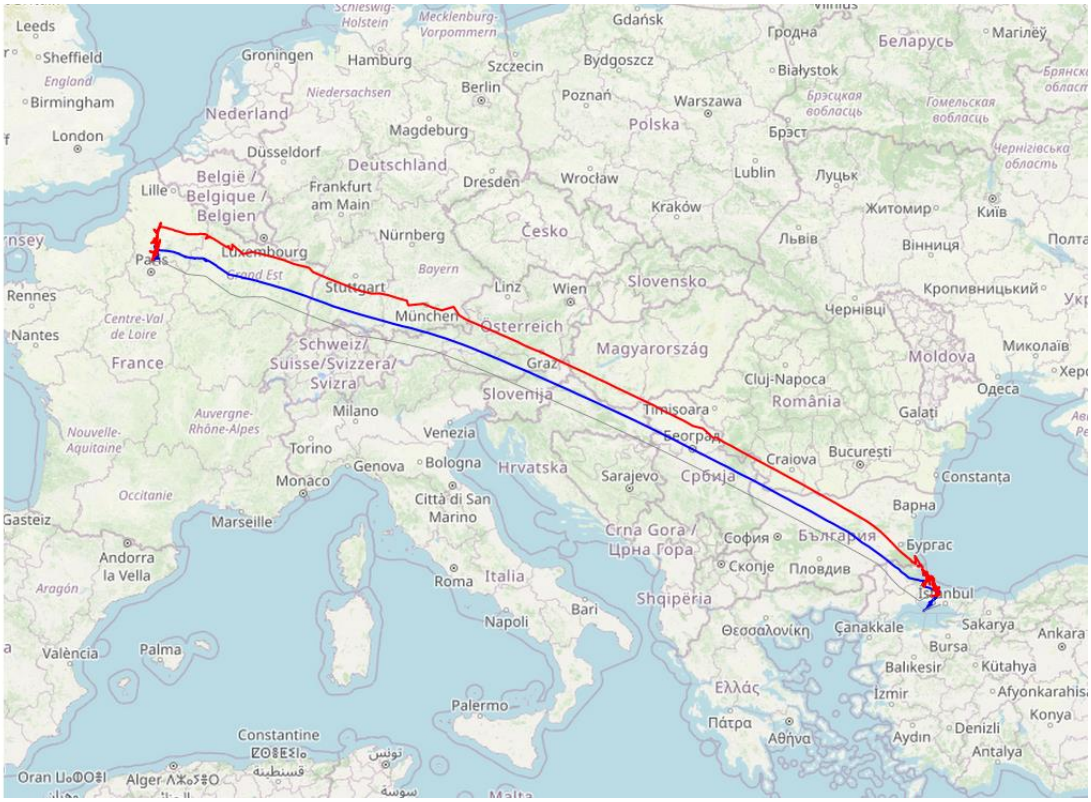

Figure 11. The best (blue) and worst (red) trajectories of the DoubleModel GAIL with TRPO 3D Location Model when training 2 models. The gray line is an one of the expert trajectories included in the training cluster

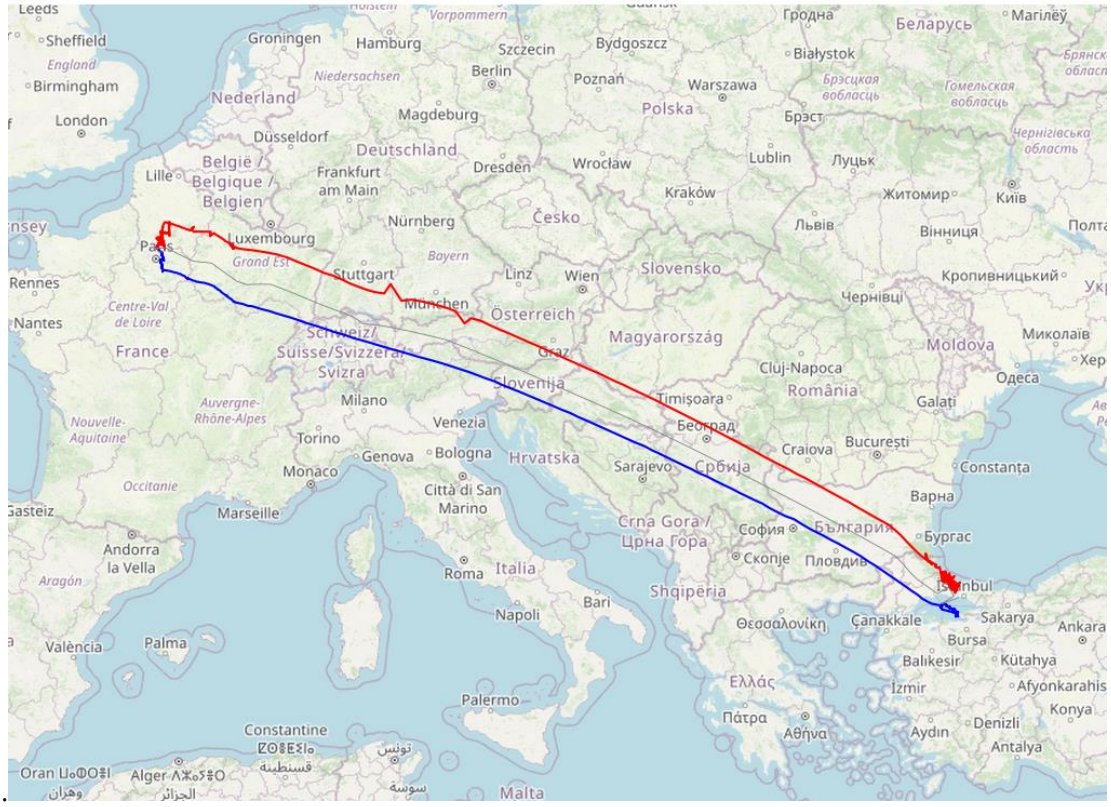

Figure 12. The best (blue) and worst (red) trajectories of the DoubleModel GAIL with PPO 3D Location Model when training 2 models. The gray line is an one of the expert trajectories included in the training cluster.

The TRPO trajectory again more closely follows the expert's path, although this time the difference appears to be significantly smaller. Both algorithms have improved performance over their SingleModel counterpart but still struggle on takeoff. Sharp turns can still be seen, especially around the takeoff and landing areas, indicating that the agents couldn’t adapt to the complexities of those phases.

Figures 13 and 14 show the fuel consumption predicted by the Fuel Consumption Model.

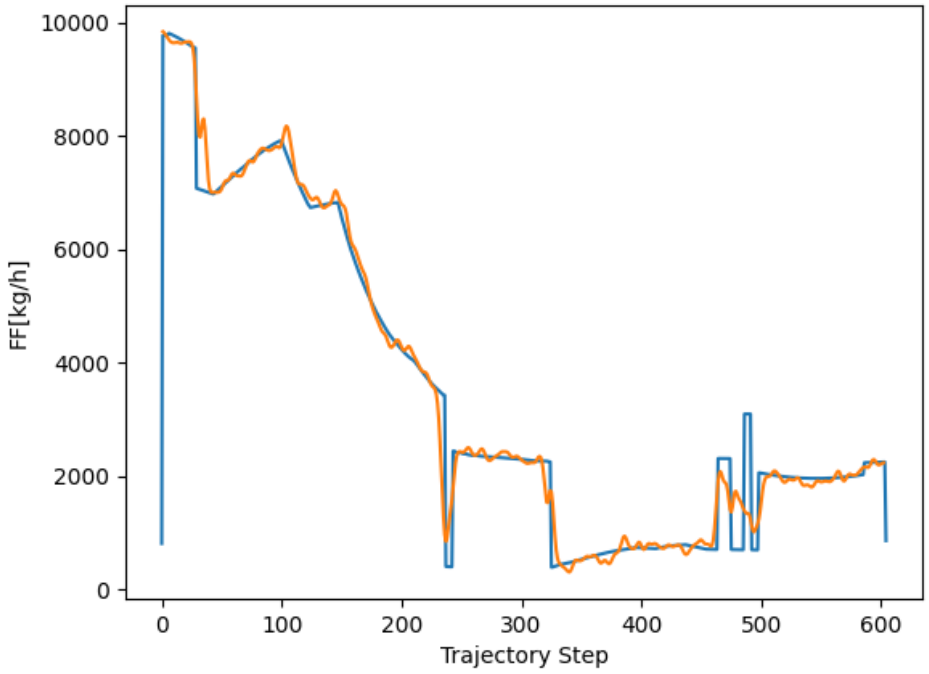

Figure 13. Fuel consumption of the DoubleModel GAIL with TRPO Fuel Consumption Model. The orange line is the agent’s prediction, while the blue is the expert’s actions.

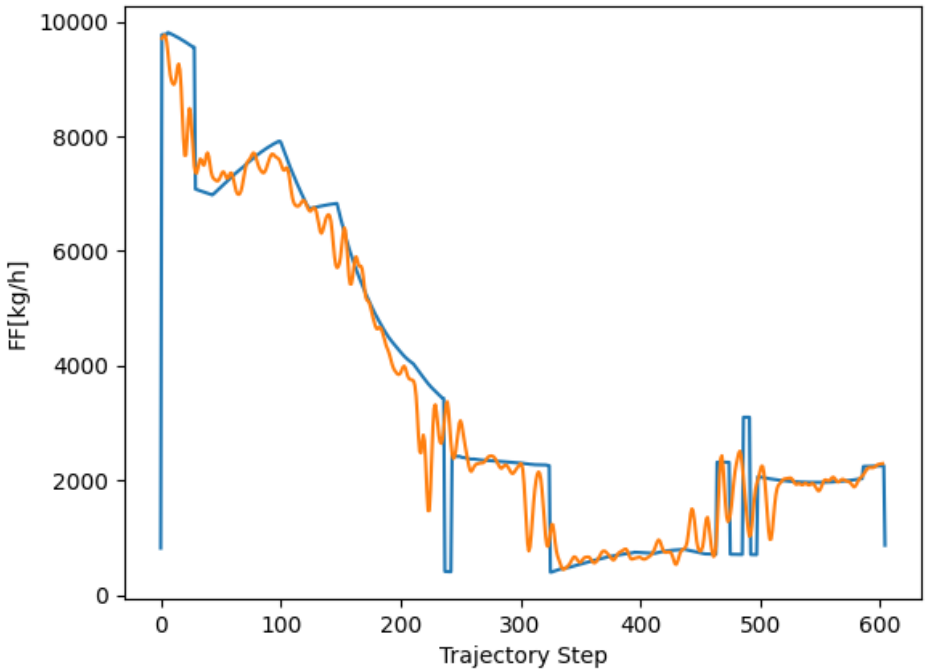

Figure 14. Fuel consumption of the DoubleModel GAIL with PPO Fuel Consumption Model. The orange line is the agent's prediction, while the blue is the expert's actions.

The TRPO's graph shows a slightly closer adherence to the expert's fuel consumption pattern with fewer spikes and less deviation from the blue line, indicating a more precise match to the expert's performance. Meanwhile, the PPO's graph, while still closely aligned, has a bit more noise and occasional spikes that deviate from the expert's pattern, suggesting a less smooth approximation. Overall both optimizer seem to perform better when dealing with the fuel consumption problem when compared to the SingleModel counterparts.

6.4 Two Model Results with Modified Consumption Model

In addition to the simple DoubleModel framework we will also test a modified version of it. The 3D Location Model will remain the same but the Fuel Consumption Model will take in as input the first order differences of the 3D trajectory state features. This means that instead of the network taking as input the current location and the previous fuel consumption as input, it will instead take the 3D vector of the last transition of the aircraft along with the fuel consumption that occurred in the previous step.

In Figures 15 and 16 we can see Fuel Consumption of the agent that takes in the 3D vector of the previous step as input:

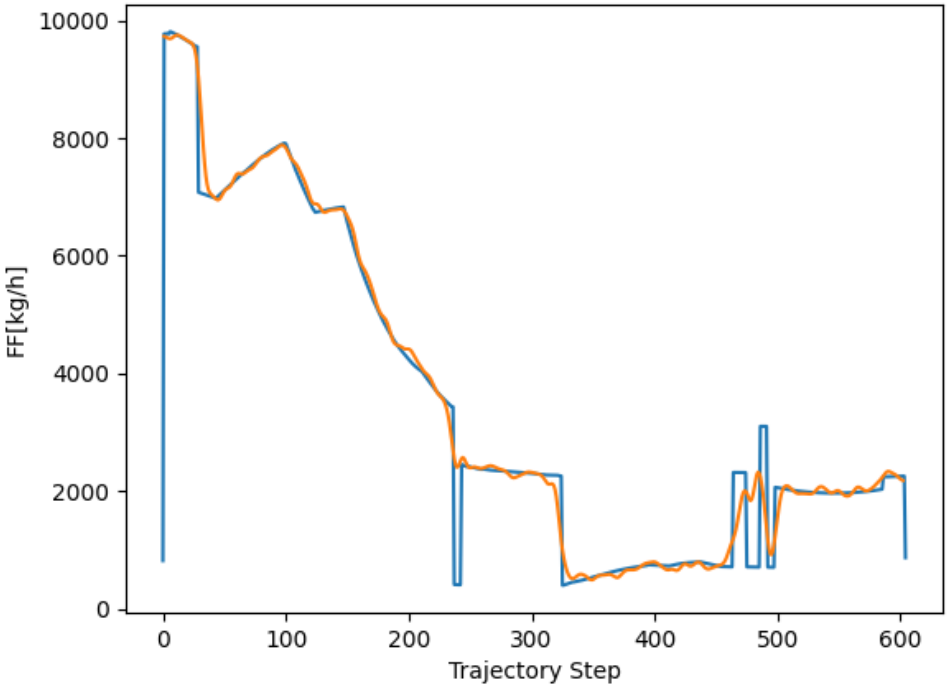

Figure 15. Fuel consumption of the Modified DoubleModel GAIL with TRPO Fuel Consumption Model. The orange line is the agent's prediction, while the blue is the expert's actions.

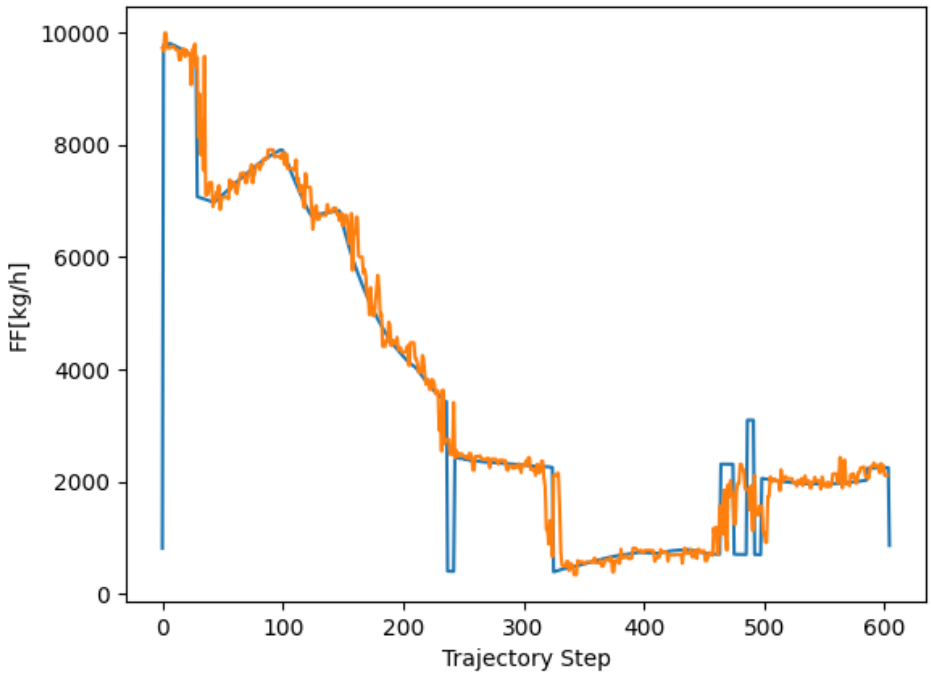

Figure 16. Fuel consumption of the modified DoubleModel GAIL with PPO Fuel Consumption Model. The orange line is the agent's prediction, while the blue is the expert's actions.

The modified DoubleModel seems to result in better performance that the original one, while also carrying over the same weak points. It is also interesting to note the highher variance in the PPO graph. Overall the modification seems to be beneficial to the fuel consumption agent, something that is confirmed by Table 2.

6.5 Measures

The following Figures contain the measurements from our experiments. For the 3D Location these measures are the ATE, CTE and Altitude Error and our shown in their respective box plots. For Fuel Consumption we will use the RMSE in Table 1 as well as the mean absolute error of each of our test runs in a box plot.

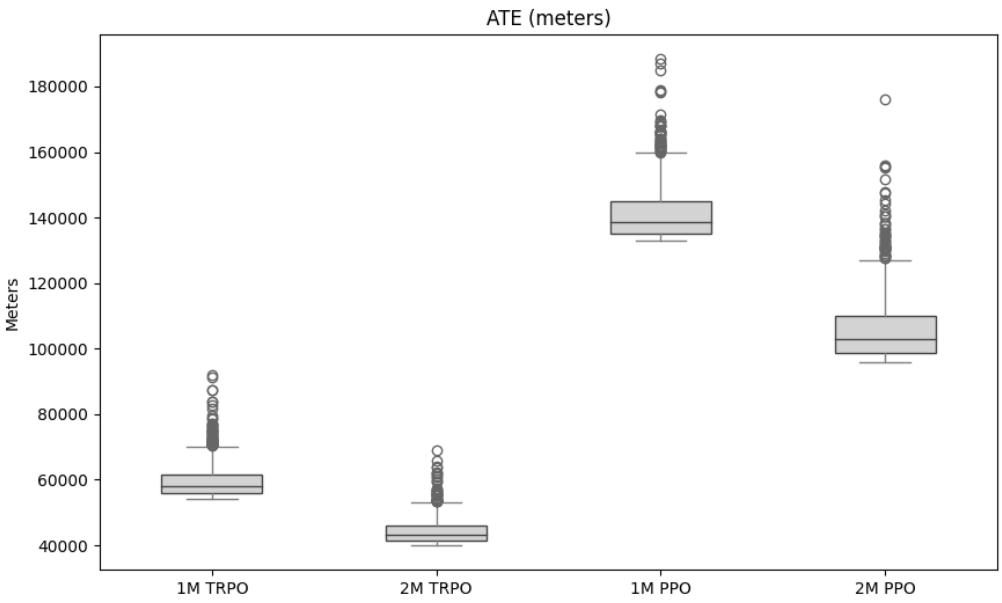

Figure 17. ATE measures (in meters) for each of the optimizer-model setup combination

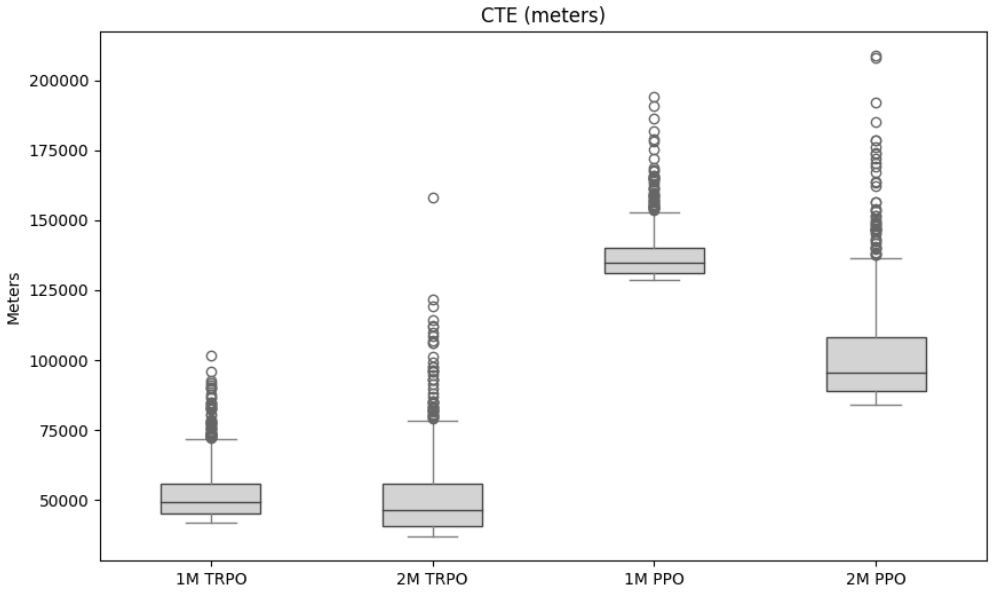

Figure 18. CTE measures (in meters) for each of the optimizer-model setup combination

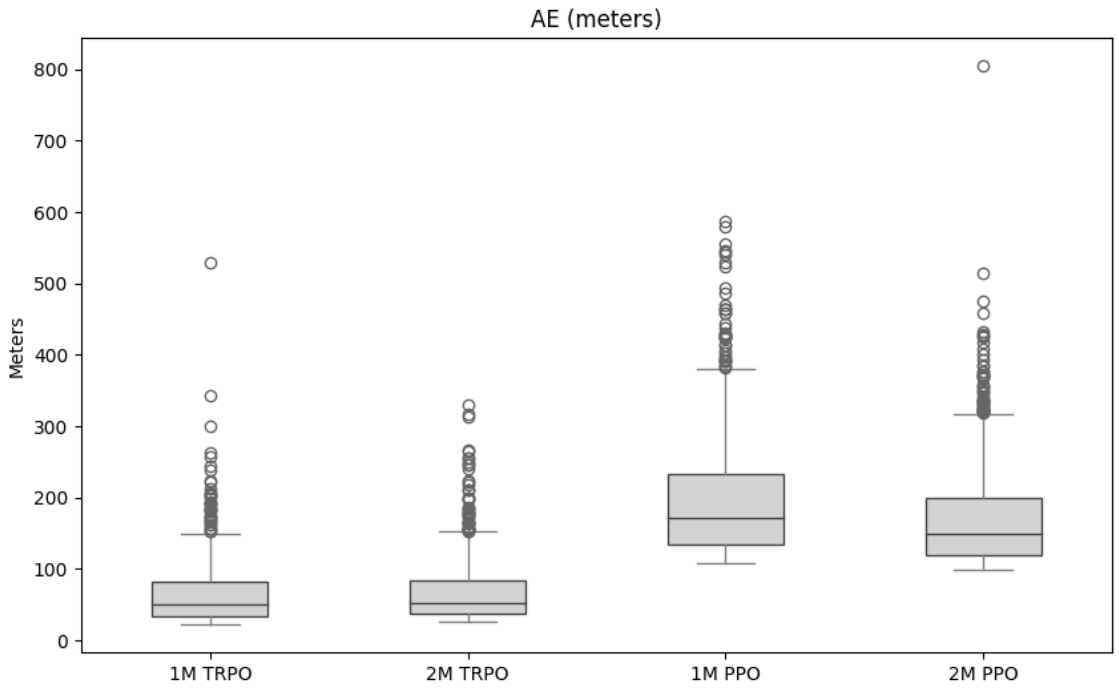

Figure 19. AE measures (in meters) for each of the optimizer-model setup combinations

In all 3 measures in Figures 17, 18 and 19 the TRPO models exhibit better performance than their PPO counterparts as well as less variance. When comparing the SingleModel to the DoubleModel solution, it seems that the Double model aids to enhance the performance of our setup. Having said that it doesn't seem to be any significant improvement in variance.

	1M TRPO	1M PPO	2M TRPO	2M PPO	2M TRPO MOD	2M PPO MOD
RMSE	416.3	621.3	457.7	538.8	370.7	514.2

Table 1. Fuel Consumption RMSE measures (in kg/h) for our SingleModel and DoubleModel setups (1M and 2M respectively) as well as the DoubleModel with modifications (2M MOD)

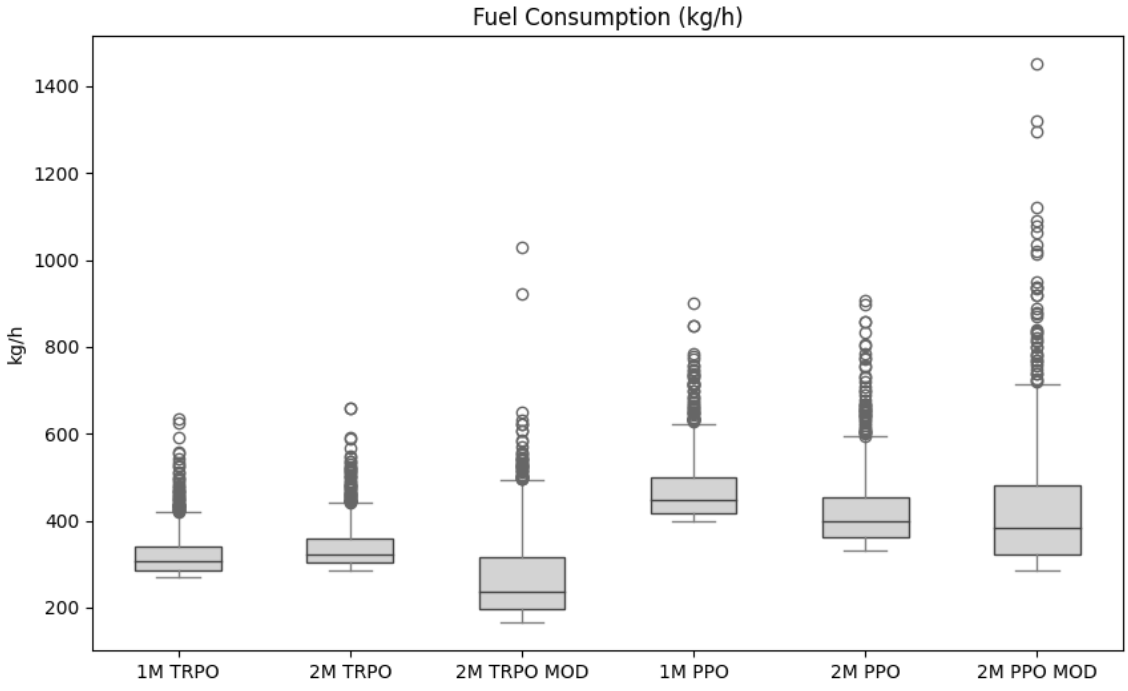

Figure 20. Fuel Consumption measures (in kg/h) for each of the optimizer-model setup combinations

In Table 1, it is evident that the DoubleModel setup outperforms the SingleModel across both of the chosen optimizers in terms of fuel consumption. Figure 20 offers an interesting insight into our findings. Although the modified versions of the DoubleModel setups exhibit a better mean estimation of the fuel consumption, it's noteworthy that their variance is significantly larger compared to the non-modified version for both TRPO and PPO optimizers.

7. Discussion

7.1 Solving the Trajectory prediction problem

The analysis reveals that both the single-model and the dual-model setups fell short in effectively solving the trajectory prediction challenge. A common difficulty encountered by agents in all configurations was achieving successful takeoffs and landings, despite maintaining relatively accurate bearings during flight. This challenge is most probably due to the complexities of the trajectories while approaching the destination airport, such as holding patterns and maneuvers [13]. Particularly noteworthy is the TRPO implementations' performance in fuel consumption, where agents nearly replicated expert trajectory patterns but struggled with abrupt changes. Among the strategies evaluated, the dual-model approach emerged as the most effective, showcasing enhancements in various aspects of trajectory prediction as detailed in Table 1. This approach's revised version exhibited even slight improvements in fuel consumption measures, without the need to change the 3D location model.

When comparing our results with the existing literature, particularly the findings from A. Bastas et al. [13], it becomes evident that our results are challenging to compare directly. This discrepancy arises not only because our smaller dataset that contains flight plans with nonlinear time frames, but also because their training incorporated more features than ours, notably meteorological data. None the less, one common observation was that using a multi-policy approach enhanced the results of the resulting pipeline. We can also observe in our findings, as mentioned in A. Bastas et al. [13], that using SingleModel type of setups is more prone to output trajectories that follow a different pattern of behavior than the actual one. Overall, although our results may not offer an immediate solution to the trajectory problem, they reinforce the optimism surrounding the implementation of imitation learning as a viable solution to trajectory prediction problems, by giving us a glimpse of the capabilities of such architecture.

Before we compare the performance of the policy optimizers, it is important to note the performance boost given by inputting the 3D Location vector of the last state to the fuel consumption agent. This, along with other datapoints, could be a beneficial addition to future models that not only aids performance but could also help with creating more generalized models. Experimenting with such nuances, as mentioned by [26], can benefit context specific problem solving.

To enhance the model's performance further, a multifaceted approach could be beneficial. Expanding the dataset would provide a richer basis for training, allowing the model to capture a wider

array of flight conditions and trajectory nuances. Introducing additional parameters, particularly those capturing environmental factors such as weather, could enrich the model's understanding and prediction accuracy. Moreover, incorporating GAIL in more complex pipelines could also enhance the performance and flexibility of the final policies. Such advancements could lead to models that not only better replicate expert patterns but also adapt more dynamically to unforeseen flight conditions, enhancing overall prediction reliability and applicability.

7.2 TRPO vs PPO

With a quick glance at our results we can see that TRPO is more suitable when tackling the imitation problem addressed. Starting from the training graphs of each optimizer, TRPO tends to have a smoother and faster convergence, hesitating to take bigger and sometimes reckless steps. PPO, even though simpler to implement, was more likely to make bigger steps and struggled to converge smoothly. Both TRPO and PPO models exhibited similar levels of fluctuation when adjustments were made to the training setup of the agent. The DoubleModel solution greatly aided the performance of both implementations, and the addition of the modified input of the fuel consumption network further assisted that performance enhancement. The trajectories generated by the TRPO models significantly outperformed those from their PPO counterparts across all measures. There could be many reasons for this significant disparity in performance.

Firstly, TRPO enforces a strict constraint on policy updates, ensuring minimal deviation from the previous policy, thereby promoting stability and consistency in learning outcomes. This method calculates an optimal step size within a defined trust region, preventing excessively large updates that could destabilize learning. Conversely, PPO seeks to streamline this process by implementing a clipped objective function. This clipping mechanism limits the policy update by penalizing changes that deviate too far from a certain range, aiming to retain the advantages of trust region methods while simplifying their implementation [11]. However, PPO's approach may not always guarantee the same level of adherence to the desired policy update constraints, particularly in environments characterized by complex dynamics, such as non-linearity and complex temporal dependencies. This could potentially result in less precise control over policy evolution, making it challenging to achieve the same level of learning stability and effectiveness as TRPO, especially in scenarios where precise adjustments to the policy are critical for navigating the environment successfully.

Another possible reason could be that TRPO exhibits higher sample efficiency compared to PPO in certain contexts, notably within complex environments where precise estimation of the

advantage function is essential. TRPO's methodology, characterized by its conservative update strategy, aims to make the most out of each data sample. By cautiously adjusting the policy within a well-defined trust region, TRPO minimizes the risk of making detrimental updates, thereby enhancing the likelihood of performance improvement with each iteration. This approach is particularly beneficial in our scenario where we encounter a limited dataset. Although this conservative strategy may lead to increased computational demands due to the intricate calculations required to ensure each update stays within the trust region, the trade-off typically results in more efficient learning [26]. In contrast, PPO, despite being designed for ease of use and computational efficiency [11], might not utilize data as effectively in environments where the accuracy of advantage estimation significantly influences learning success, leading to potentially slower progress or less optimal policy development with the same amount of data.

Additionally, the differential performance between TRPO and PPO can be partly attributed to the sensitivity of PPO to its hyperparameters, such as the clipping range. This aspect of PPO demands precise tuning to achieve optimal results, making its performance highly contingent on finding the right settings for each specific problem. The challenge here is that the optimal hyperparameter configuration can vary significantly across different environments, necessitating a more meticulous and sometimes trial-and-error approach to tuning. This sensitivity contrasts with TRPO's more forgiving nature regarding hyperparameter adjustments, as its core mechanism of enforcing strict policy update constraints inherently provides a buffer against suboptimal parameter choices [11]. In our context, where the dynamics of the problem are complex, the ability of TRPO to maintain learning stability and effectiveness without requiring the fine-tuning necessary for PPO becomes a decisive advantage. And, although in the context of this theses, different hyperparameters have been tested in the PPO optimizers, there is no guarantee that we chose the best option. This requirement for precise hyperparameter tuning in PPO could lead to its underperformance compared to TRPO. This factor further underscores the suitability of TRPO over PPO when addressing the nuanced challenges presented by our trajectory prediction problem.

Most importantly, the utilization of TRPO and Proximal Policy Optimization PPO within GAIL frameworks introduces nuanced dynamics, especially considering the nature of rewards generated by the discriminator. In GAIL, the discriminator's role is to differentiate between the agent's actions and those from a demonstration, providing a reward signal based on this differentiation. This process can introduce variability and potential inaccuracies in the reward signal, as the discriminator's assessments may not always perfectly align with optimal policy behaviors. Given this context, TRPO's inherent characteristics may offer a systemic advantage.

TRPO's strict policy update constraints and robustness to hyperparameter sensitivity ensure a more stable learning trajectory, which is crucial in environments where the reward signal is inherently noisy or imprecise. This stability is particularly beneficial in GAIL, where the exactitude of rewards is contingent on the evolving accuracy of the discriminator, making the reward landscape more unpredictable than in standard reinforcement learning setups.

Moreover, TRPO's sample efficiency, as mentioned above, and conservative update strategy mean that it can achieve significant learning progress from fewer interactions, an advantage when tuning the algorithm in the face of uncertain rewards. The need for precise hyperparameter tuning in PPO, on the other hand, becomes an even more pronounced challenge in GAIL. The unpredictable nature of discriminator-based rewards compounds the difficulty of finding optimal settings, potentially exacerbating PPO's sensitivity to hyperparameter adjustments.

These factors combined suggest that TRPO could be inherently better suited to the unique challenges presented by GAIL. Its ability to navigate uncertain reward landscapes with greater stability and less sensitivity to tuning nuances may lead to more effective learning outcomes, particularly in complex environments where the discriminator's reward signal does not always reflect the true value of policy actions accurately. This perspective does not discount the utility of PPO in GAIL applications but suggests that TRPO's specific attributes could offer systematic benefits under certain conditions. The choice between TRPO and PPO should be informed by the specific dynamics and requirements of the task at hand.

8. Conclusion

In this thesis we approached the trajectory prediction problem in the context of aviation using the GAIL framework. Our investigation revealed that both single-model and dual-model approaches encountered difficulties in achieving successful takeoffs and landings, despite maintaining accurate flight bearings, attributed to the complexities of flight trajectory near destination airports. Notably, TRPO implementations demonstrated superior performance in fuel consumption, closely mirroring expert trajectory patterns, albeit with challenges in adapting to abrupt changes.

Comparative analysis with existing literature, especially with works incorporating meteorological data, indicated that our results, while promising, are not directly comparable due to differences in datasets and features used. However, the utilization of a multi-policy approach emerged as a significant enhancer of performance, aligning with findings from A. Bastas et al [13]. This approach, particularly the dual-model strategy, demonstrated marked improvements across various measures, suggesting its superiority in enhancing the accuracy of trajectory predictions.

A focal point of our study was the discernible performance disparity between TRPO and PPO. TRPO's conservative approach to policy updates, prioritizing stability and sample efficiency, proved to be particularly advantageous. This approach is crucial in the context of GAIL, where the discriminator generates inherently uncertain reward signals. TRPO's design, which is less sensitive to hyperparameter adjustments, facilitates a more stable learning trajectory, a critical asset when navigating the unpredictable reward landscape of GAIL. Conversely, PPO's performance was notably hampered by its hyperparameter sensitivity and its difficulties in coping with the variable nature of discriminator-based rewards, making it a less optimal choice for our specific trajectory prediction challenges.

The main contribution of this thesis lies in demonstrating the superiority of TRPO over PPO in the GAIL framework for aviation trajectory prediction, highlighting its stable learning progress and efficiency under the complex, uncertain conditions of discriminator-based rewards. Through rigorous testing and comparative analysis, this research has uncovered valuable insights into how TRPO's approach significantly outperforms PPO, especially in scenarios requiring high adaptability and precision in response to dynamic environmental factors and reward signals. This study not only reinforces the viability of imitation learning for addressing trajectory prediction issues but also highlights the critical need for algorithmic precision and adaptability in such complex tasks.

Looking forward, this thesis aspires to lay the groundwork for future exploration into more sophisticated models and the integration of a wider array of features, including environmental

variables, to further enhance the accuracy and generalizability of trajectory prediction models. The insights gained here pave the way for refining imitation learning approaches, offering promising directions for research in enhancing the fidelity and reliability of trajectory prediction methodologies.

Bibliography

1. BD. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. Neural Computation, 3(1):88–97 (1991).

2. Michael Bain, Claude Sammut. A Framework for Behavioural Cloning, 2001.

3. Abbeel P., Ng A.Y.: Apprenticeship learning via inverse reinforcement learning. Proceedings of the twenty-first international conference on Machine learning; 1 (2004).

4. Ho, J., Ermon, S.: Generative Adversarial Imitation Learning. Advances in neural information processing systems 29 : 4565– 4573 (2016).

5. Ziebart, Brian D., et al., Maximum entropy inverse reinforcement learning., Aaai. Vol. 8. 2008.

6. Ramachandran, Deepak, and Eyal Amir., Bayesian Inverse Reinforcement Learning., IJCAI. Vol. 7. 2007.

7. Pinto, Lerrel, et al. "Robust adversarial reinforcement learning." International Conference on Machine Learning. PMLR, (2017).

8. Kingma, D. P., Ba, J.: Adam: A Method for Stochastic Optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, Conference Track Proceedings (2015).

9. Sutton, R. S., McAllester, D. A., Singh, S. P., Mansour, Y.: Policy Gradient Methods for Reinforcement Learning with Function Approximation. Advances in Neural Information Processing Systems 12 : 1057–1063 (2000).

10. Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In Machine Learning (Vol. 8, No. 3-4, pp. 229-256) (1992).

11. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal Policy Optimization Algorithms., CoRR (2017).

12. https://ext.eurocontrol.int/lexicon/index.php/Trajectory

13. A. Bastas, T.Kravaris, G.A.Vouros, Data Driven Aircraft Trajectory Prediction with Deep Imitation Learning, CoRR abs/2005.07960 (2020).

14. M. Teranishi, K. Fujii, K. Takeda, Trajectory Prediction with Imitation Learning Reflecting Defensive Evaluation in Team Sports, 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE), Kobe, Japan, pp. 124-125 (2020).

15. A. Weißmann, D. Görges, An Empirical Study on Ego Vehicle Trajectory Prediction for Bicycles in Urban Environments Based on Conditional Imitation Learning, 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, pp. 1482-1489 (2021).

16. A. Hussein, M.M. Gaber, E. Elyan, C. Jayne, Imitation Learning: A Survey of Learning Methods, ACM Comput. Surv. 50, 2, Article 21 (March 2018).
17. D. Kamal M, Tahir A, Babar Kamal M, Moeen F, Naeem MA, A Survey for the Ranking of Trajectory Prediction Algorithms on Ubiquitous Wireless Sensors, Sensors (Basel), 2020.
18. Gano B. Chatterji, Short-term Trajectory Prediction Methods, In: Guidance, Navigation, and Control Conference and Exhibit, p. 4233 (1999).
19. I. Lymperopoulos, J. Lygeros, Sequential Monte Carlo Methods for Multi-Aircraft Trajectory Prediction in Air Traffic Management, Int. J. Adapt. Control Signal Process., 24, pp. 830-849 (2010).
20. J.V. Benavides, J. Kaneshige, S. Sharma, R. Panda, M. Steglinski, Implementation of a Trajectory Prediction Function for Trajectory Based Operations, In AIAA Atmospheric Flight Mechanics Conference, p. 2198 (2014).
21. W. Liu, I. Hwang, Probabilistic Trajectory Prediction and Conflict Detection for Air Traffic Control, Journal of Guidance, Control, and Dynamics, 34(6), pp. 1779-1789 (2011).
22. J. Sturdy, J. Andrews, J. Welch, Aircraft Trajectory Prediction for Terminal Automation, In Guidance, Navigation and Control Conference, p. 3634 (August 1989).
23. P. Englert, et al., Probabilistic Model-Based Imitation Learning, Adaptive Behavior, 21(5), pp. 388-403 (2013).
24. H.-C. Choi, C. Deng, I. Hwang, Hybrid Machine Learning and Estimation-Based Flight Trajectory Prediction in Terminal Airspace, IEEE Access, 9 (2021).
25. J. Schulman, et al., Trust Region Policy Optimization, International Conference on Machine Learning, PMLR (2015).
26. L. Engstrom, et al., Implementation Matters in Deep RL: A Case Study on PPO and TRPO, International Conference on Learning Representations (2019).
27. Boeing Research and Technology Europe (patent filed in the European Patent Office): Method and system for autonomously operating an aircraft (2017).