



Predicting Trajectories with Directed-Info GAIL

by

Alexander Tsevrenis

Submitted

in partial fulfilment of the requirements for the degree of

Master of Artificial Intelligence

at the


UNIVERSITY OF PIRAEUS

June 2021


Author 

II-MSc “Artificial Intelligence”


June 23, 2021

Certified by..... 

George Vouros
Professor,
University of
Piraeus
Thesis Supervisor

Certified by..... 

George Petasis
Researcher,
NCSR
Demokritos
Member of
Examination
Committee

Certified by..... 

Maria Dagioglou
Researcher,
NCSR
Demokritos
Member of
Examination
Committee

Predicting Trajectories with Directed-Info GAIL

By

Alexander Tsevrenis

Submitted to the II-MSc “Artificial Intelligence” on June 23, 2021, in partial fulfillment of the requirements for the MSc degree

Abstract

As noted in the Directed-Info GAIL paper “the use of imitation learning to learn a single policy for a complex task that has multiple modes or hierarchical structure can be challenging”. This thesis will explore the use of Directed-Info GAIL algorithm, which is based on the generative adversarial imitation learning framework to automatically learn subtask policies from unsegmented demonstrations of robot trajectories and aircraft trajectories, given that flights and robots have indeed different modes of behaviour in different segments of trajectories, depending on tasks they fulfil and many trajectories’ contextual features.

Thesis Supervisor: G. Vouros
Title: Professor

Acknowledgments

I would like to express my special thanks of gratitude to my Professor George Vouros and his PhD student Alevizos Bastas for the continuous support of my MSc study and research, for their patience, motivation and immense knowledge. I am grateful to my Professor who gave me this amazing opportunity to get a glimpse of his knowledge and assigned me this wonderful thesis, which also helped me in doing a lot of research and I came to know about so many new things. I am really thankful.

Besides my advisor, I would like to thank the rest of my MSc Professors and the rest of my thesis committee: Researcher George Petasis and Researcher Maria Dagioglou for their encouragement, insightful comments and hard questions.

I also want to thank my BSc Thesis Professor Evangelos Spyrou that remembered me and strengthened my application with his recommendation letter.

Last but not the least, I would like to thank my family and friends and especially my brother who gifted me this amazing MSc education to improve my skills and be a better programmer and person.

It was really hard to be concentrated in studies while there was the Covid-19 pandemic but I tried my best.

Table of Contents

TABLE OF CONTENTS	1
LIST OF FIGURES	3
LIST OF TABLES	7
1 INTRODUCTION	11
2 PRELIMINARIES	12
2.1 IMITATION LEARNING.....	12
2.2 BASIC IMITATION LEARNING ALGORITHMS	12
2.2.1 <i>Behavioral Cloning</i>	13
2.2.2 <i>Inverse Reinforcement Learning (IRL)</i>	14
2.3 THE GAIL ALGORITHM.....	15
2.3.1 <i>GAIL</i>	15
2.3.2 <i>More detailed information on GAIL</i>	17
2.3.3 <i>GAIL Framework</i>	19
2.3.4 <i>Generalized Advantage Estimation (GAE) and the trade-off between variance and bias</i>	21
2.4 INFO GAIL ALGORITHM.....	22
3 THE DIRECTED-INFO GAIL ALGORITHM	23
3.1 DETAILED DESCRIPTION OF DIRECTED-INFO GAIL	24
3.2 DETAILED DESCRIPTION OF THE IMPLEMENTATION OF THE DIRECTED-INFO GAIL ALGORITHM.....	26
4 ENVIRONMENTS AND DEFINITION OF THE TRAJECTORY IMITATION PROBLEM	30
4.1 AVIATION ENVIRONMENT	30
4.2 AVIATION TRAJECTORY IMITATION PROBLEM	31
4.3 HOPPER ENVIRONMENT	33
4.4 HOPPER TRAJECTORY IMITATION PROBLEM.....	34

5	ALGORITHMS' STRUCTURE	37
5.1	GAIL FRAMEWORK.....	37
5.1.1	<i>Behavioral Cloning – Pre-Training</i>	37
5.1.2	<i>Trusted Region Policy (TRPO) agent with a Discriminator and an Actor critic - GAIL</i>	37
5.2	DIRECTED INFO-GAIL FRAMEWORK.....	38
5.2.1	<i>Variational Auto-Encoder with Gumbel-Softmax Trick – Pre-Training</i>	38
5.3	BASIC PARAMETERS.....	39
6	EXPERIMENTAL EVALUATION.....	41
6.1	EXPERIMENTAL SETTING.....	41
6.2	EVALUATION REWARDS GAINED BY THE ALGORITHMS.....	43
6.2.1	<i>Training</i>	43
6.2.2	<i>Testing</i>	46
6.3	HOPPER TRAJECTORIES	49
6.3.1	<i>Modes</i>	49
6.3.2	<i>Latent variables and training loss</i>	49
6.4	AVIATION TRAJECTORIES	54
6.4.1	<i>Modes and predictions</i>	54
6.4.2	<i>Latent Variables and training loss</i>	63
6.4.3	<i>Deviations between experts and generated trajectories by Directed-Info Gail</i>	72
7	CONCLUSIONS.....	83
	BIBLIOGRAPHY	85

List of Figures

FIGURE 1. ALGORITHM GENERATIVE ADVERSARIAL IMITATION LEARNING (GAIL). [1]	19
FIGURE 2. ALGORITHM GENERATIVE ADVERSARIAL IMITATION LEARNING (GAIL) USING BEHAVIORAL CLONING FOR PRE-TRAINING AND TRPO STEP WITH KL-CONSTRAINED NATURAL GRADIENT STEP.	21
FIGURE 3. C NODES DICTATE THE SUB-ACTIVITY LATENT VARIABLES AND T NODES ARE THE OBSERVED EXPERT DEMONSTRATIONS. THE LEFT GRAPH CORRESPONDS TO THE INFO GAIL ALGORITHM WHILE THE RIGHT ONE TO THE DIRECTED INFO-GAIL. [1]	24
FIGURE 4. DIRECTED-INFO GAIL OVERALL ARCHITECTURE.	27
FIGURE 5. DIRECTED-INFO GAIL' DETAILED APPROACH.	28
FIGURE 6. ALGORITHM 2 DIRECTED-INFO GENERATIVE ADVERSARIAL IMITATION LEARNING (GAIL). ..	29
FIGURE 7. DIRECTED-INFO GAIL TRAINING REWARDS (3 MODES) FIGURE 8. DIRECTED-INFO GAIL TRAINING REWARDS(5 MODES)	43
FIGURE 9. DIRECTED-INFO GAIL TRAINING REWARDS (3 MODES) FIGURE 10. DIRECTED-INFO GAIL TRAINING REWARDS(5 MODES)	44
FIGURE 11. DIRECTED-INFO GAIL TRAINING REWARDS (3 MODES) FIGURE 12. DIRECTED-INFO GAIL TRAINING REWARDS(5 MODES)	44
FIGURE 13. DIRECTED-INFO GAIL TRAINING REWARDS (3 MODES) FIGURE 14. DIRECTED-INFO GAIL TRAINING REWARDS(5 MODES)	45
FIGURE 15. EVALUATING MODES IN HOPPER ENVIRONMENT WITH 3 MODES.....	49
FIGURE 16. EVALUATING MODES IN HOPPER ENVIRONMENT WITH 5 MODES.....	49
FIGURE 17. VAE, LATENT VARIABLES IN FIRST EPOCH FIGURE 18. VAE, LATENT VARIABLES IN 500TH EPOCH.....	50
FIGURE 19. VAE, LATENT VARIABLES IN 1500TH EPOCH FIGURE 20. VAE, LATENT VARIABLES IN 2000TH EPOCH	50
FIGURE 21. VAE, TESTING 3 LATENT VARIABLES... FIGURE 22. TRAINING LOSS OF THE VAE(3MODES)	51
FIGURE 23. VAE, LATENT VARIABLES IN THE FIRST EPOCH FIGURE 24. VAE, LATENT VARIABLES IN THE 500TH EPOCH	52
FIGURE 25. VAE, LATENT VARIABLES IN THE 1500TH EPOCH FIGURE 26. VAE, LATENT VARIABLES IN THE 2000TH EPOCH	52
FIGURE 27. TESTING 5 LATENT VARIABLES .. FIGURE 28. TRAINING LOSS OF THE VAE(5MODES)	53
FIGURE 29. BARCELONA TO MADRID, BEHAVIORAL CLONING RESULTS WITH RAW TRAJECTORIES.....	54

FIGURE 30. BARCELONA TO MADRID GAIL RESULTS WITH BEHAVIORAL CLONING FOR PRE-TRAINING AND RAW EXPERT TRAJECTORIES.	54
FIGURE 31. BARCELONA TO MADRID, BEHAVIORAL CLONING RESULTS WITH EXPERT TRAJECTORIES ENRICHED WITH NOAA.	55
FIGURE 32. BARCELONA TO MADRID GAIL RESULTS WITH BEHAVIORAL CLONING FOR PRE-TRAINING AND EXPERT TRAJECTORIES ENRICHED WITH NOAA.	55
FIGURE 33. BARCELONA TO MADRID, BEHAVIORAL CLONING RESULTS WITH EXPERT TRAJECTORIES ENRICHED WITH NOAA AND METAR.	56
FIGURE 34. BARCELONA TO MADRID GAIL RESULTS WITH BEHAVIORAL CLONING FOR PRE-TRAINING WITH EXPERT TRAJECTORIES ENRICHED WITH NOAA AND METAR.	56
FIGURE 35. BARCELONA TO MADRID VARIATIONAL AUTO-ENCODER RESULTS WITH 3 MODES (FEATURES: 3D POINT WITH TIMESTAMP WITHOUT METEOROLOGICAL FEATURES).	57
FIGURE 36. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS WITH VARIATIONAL AUTO-ENCODER FOR PRE-TRAINING WITH 3 MODES (FEATURES: 3D POINT WITH TIMESTAMP WITHOUT METEOROLOGICAL FEATURES).	57
FIGURE 37. BARCELONA TO MADRID, VARIATIONAL AUTO-ENCODER RESULTS WITH 5 MODES (FEATURES: 3D POINT WITH TIMESTAMP WITHOUT METEOROLOGICAL FEATURES).	58
FIGURE 38. BARCELONA TO MADRID, DIRECTED-INFO GAIL RESULTS WITH VARIATIONAL AUTO-ENCODER FOR PRE-TRAINING WITH 5 MODES (FEATURES: 3D POINT WITH TIMESTAMP WITHOUT METEOROLOGICAL FEATURES).	58
FIGURE 39. BARCELONA TO MADRID VARIATIONAL AUTO-ENCODER RESULTS WITH 3 MODES (FEATURES: 3D POINT WITH TIMESTAMP AND NOAA METEOROLOGICAL FEATURES).	59
FIGURE 40. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS WITH VARIATIONAL AUTO-ENCODER FOR PRE-TRAINING WITH 3 MODES (FEATURES: 3D POINT WITH TIMESTAMP AND NOAA METEOROLOGICAL FEATURES).	59
FIGURE 41. BARCELONA TO MADRID, VARIATIONAL AUTO-ENCODER RESULTS WITH 5 MODES (FEATURES: 3D POINT WITH TIMESTAMP AND NOAA METEOROLOGICAL FEATURES).	60
FIGURE 42. BARCELONA TO MADRID, DIRECTED-INFO GAIL RESULTS WITH VARIATIONAL AUTO-ENCODER FOR PRE-TRAINING WITH 5 MODES (FEATURES: 3D POINT WITH TIMESTAMP AND NOAA METEOROLOGICAL FEATURES).	60
FIGURE 43. BARCELONA TO MADRID VARIATIONAL AUTO-ENCODER RESULTS WITH 3 MODES (FEATURES: 3D POINT WITH TIMESTAMP AND NOAA&METAR METEOROLOGICAL FEATURES)...	61
FIGURE 44. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS WITH VARIATIONAL AUTO-ENCODER FOR PRE-TRAINING WITH 3 MODES (FEATURES: 3D POINT WITH TIMESTAMP AND NOAA&METAR METEOROLOGICAL FEATURES).	61
FIGURE 45. BARCELONA TO MADRID, VARIATIONAL AUTO-ENCODER RESULTS WITH 5 MODES (FEATURES: 3D POINT WITH TIMESTAMP AND NOAA&METAR METEOROLOGICAL FEATURES)...	62

FIGURE 46. BARCELONA TO MADRID, DIRECTED-INFO GAIL RESULTS WITH VARIATIONAL AUTO-ENCODER FOR PRE-TRAINING WITH 5 MODES (FEATURES: 3D POINT WITH TIMESTAMP AND NOAA&METAR METEOROLOGICAL FEATURES).....	63
FIGURE 47. VAE, LATENT VARIABLES IN THE FIRST EPOCH	
FIGURE 48. VAE, LATENT VARIABLES IN THE 500TH EPOCH	64
FIGURE 49. VAE, LATENT VARIABLES IN THE 1500TH EPOCH	
FIGURE 50. VAE, LATENT VARIABLES IN THE 2000TH EPOCH	64
FIGURE 51. VAE, TESTING 3 LATENT VARIABLES (3MODES).....	65
FIGURE 52. TRAINING LOSS OF THE VAE	
FIGURE 53. VAE, LATENT VARIABLES IN THE FIRST EPOCH	
FIGURE 54. VAE, LATENT VARIABLES IN THE 500TH EPOCH	65
FIGURE 55. VAE, LATENT VARIABLES IN THE 1500TH EPOCH	
FIGURE 56. VAE, LATENT VARIABLES IN THE 2000TH EPOCH	66
FIGURE 57. VAE, TESTING 5 LATENT VARIABLES (5MODES).....	66
FIGURE 58. TRAINING LOSS OF THE VAE	
FIGURE 59. VAE, LATENT VARIABLES IN THE FIRST EPOCH	
FIGURE 60. VAE, LATENT VARIABLES IN THE 500TH EPOCH	67
FIGURE 61. VAE, LATENT VARIABLES IN THE 1500TH EPOCH	
FIGURE 62. VAE, LATENT VARIABLES IN THE 2000TH EPOCH	67
FIGURE 63. VAE, TESTING 3 LATENT VARIABLES (3MODES)	68
FIGURE 64. TRAINING LOSS OF THE VAE	
FIGURE 65. VAE, LATENT VARIABLES IN THE FIRST EPOCH	
FIGURE 66. VAE, LATENT VARIABLES IN THE 500TH EPOCH	68
FIGURE 67. VAE, LATENT VARIABLES IN THE 1500TH EPOCH	
FIGURE 68. VAE, LATENT VARIABLES IN THE 2000TH EPOCH	69
FIGURE 69. VAE, TESTING 5 LATENT VARIABLES (5MODES).....	69
FIGURE 70. TRAINING LOSS OF THE VAE	
FIGURE 71. VAE, LATENT VARIABLES IN THE FIRST EPOCH	
FIGURE 72. VAE, LATENT VARIABLES IN THE 500TH EPOCH	70
FIGURE 73. VAE, LATENT VARIABLES IN THE 1500TH EPOCH	
FIGURE 74. VAE, LATENT VARIABLES IN THE 2000TH EPOCH	70
FIGURE 75. VAE, TESTING 3 LATENT VARIABLES (3MODES)	71
FIGURE 76. TRAINING LOSS OF THE VAE	
FIGURE 77. VAE, LATENT VARIABLES IN THE FIRST EPOCH	
FIGURE 78. VAE, LATENT VARIABLES IN THE 500TH EPOCH	71
FIGURE 79. VAE, LATENT VARIABLES IN THE 1500TH EPOCH	
FIGURE 80. VAE, LATENT VARIABLES IN THE 2000TH EPOCH	72
FIGURE 81. VAE, TESTING 5 LATENT VARIABLES (5MODES)	72
FIGURE 82. TRAINING LOSS OF THE VAE	

FIGURE 83. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS WITH 3 MODES. FEATURES: 3D POINT WITH TIMESTAMP WITHOUT METEOROLOGICAL FEATURES. THE BEST CASE SCENARIO WITH 0 MEAN REWARD FOR THIS TESTING EPISODE.	73
FIGURE 84. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS WITH 3 MODES. FEATURES: 3D POINT WITH TIMESTAMP WITHOUT METEOROLOGICAL FEATURES. THE WORST CASE SCENARIO WITH -71.70 MEAN REWARD FOR THIS TESTING EPISODE.	74
FIGURE 85. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS WITH 5 MODES. FEATURES: 3D POINT WITH TIMESTAMP WITHOUT METEOROLOGICAL FEATURES. THE BEST CASE SCENARIO WITH 0 MEAN REWARD FOR THIS TESTING EPISODE.	75
FIGURE 86. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS WITH 5 MODES. FEATURES: 3D POINT WITH TIMESTAMP WITHOUT METEOROLOGICAL FEATURES. THE WORST CASE SCENARIO WITH -82.80 MEAN REWARD FOR THIS TESTING EPISODE.	75
FIGURE 87. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS WITH 3 MODES. FEATURES: 3D POINT WITH TIMESTAMP AND NOAA METEOROLOGICAL FEATURES. THE BEST CASE SCENARIO WITH 0 MEAN REWARD FOR THIS TESTING EPISODE.	76
FIGURE 88. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS WITH 3 MODES. FEATURES: 3D POINT WITH TIMESTAMP AND NOAA METEOROLOGICAL FEATURES. THE WORST CASE SCENARIO WITH -107.70 MEAN REWARD FOR THIS TESTING EPISODE.	76
FIGURE 89. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS WITH 5 MODES. FEATURES: 3D POINT WITH TIMESTAMP AND NOAA METEOROLOGICAL FEATURES. THE BEST CASE SCENARIO WITH 0 MEAN REWARD FOR THIS TESTING EPISODE.	77
FIGURE 90. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS WITH 5 MODES. FEATURES: 3D POINT WITH TIMESTAMP AND NOAA METEOROLOGICAL FEATURES. THE WORST CASE SCENARIO WITH -134.21 MEAN REWARD FOR THIS TESTING EPISODE.	78
FIGURE 91. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS RESULTS WITH 3 MODES. FEATURES: 3D POINT WITH TIMESTAMP AND NOAA&METAR METEOROLOGICAL FEATURES. THE BEST CASE SCENARIO WITH 0 MEAN REWARD FOR THIS TESTING EPISODE.	79
FIGURE 92. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS RESULTS WITH 3 MODES. FEATURES: 3D POINT WITH TIMESTAMP AND NOAA&METAR METEOROLOGICAL FEATURES. THE WORST CASE SCENARIO WITH -352.08 MEAN REWARD FOR THIS TESTING EPISODE.	79
FIGURE 93. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS RESULTS WITH 5 MODES. FEATURES: 3D POINT WITH TIMESTAMP AND NOAA&METAR METEOROLOGICAL FEATURES. THE BEST CASE SCENARIO WITH 0 MEAN REWARD FOR THIS TESTING EPISODE.	81
FIGURE 94. BARCELONA TO MADRID DIRECTED-INFO GAIL RESULTS RESULTS WITH 5 MODES. FEATURES: 3D POINT WITH TIMESTAMP AND NOAA&METAR METEOROLOGICAL FEATURES. THE WORST CASE SCENARIO WITH -253.29 MEAN REWARD FOR THIS TESTING EPISODE.	81

List of Tables

TABLE 1. EVALUATION REWARDS FOR BCLONING/ GAIL, VAE/ DIRECTED-INFO GAIL, IN THE HOPPER ENVIRONMENT	46
TABLE 2. EVALUATION REWARDS FOR BCLONING/ GAIL,VAE/ DIRECTED-INFO GAIL, IN THE AVIATION ENVIRONMENT	47

1 Introduction

Nowadays, imitation learning techniques aim to mimic human behavior in a given task: An agent (a learning machine) is trained to perform a task from demonstrations by learning a mapping between observations and actions. The notion of teaching by imitation has been around for many years, however, the field is gaining attention recently due to advances in computing and sensing as well as rising demand for complex intelligent applications. The paradigm of learning by imitation is gaining popularity because it facilitates teaching complex tasks by demonstrations and with minimal expert knowledge of the tasks. A big difference with other reinforcement learning techniques is that the agent is exploring the environment while is trying to approximate the experts' reward/objective, and thus imitate the expert. In this thesis, we have implemented and evaluated the Directed-Info GAIL algorithm [3] in two continuous environments. The first environment is a terrain with Hopper, a robot-leg from Open-AI gym page¹ and the second environment is based on the aviation domain using real-world flight trajectories that the algorithm imitates. The agent is mimicking the expert trajectories in the first setting as a Hopper robot and in the second setting as an aircraft. In both cases, our aim is to split the trajectories in different modes of behavior that will help the agent identify these modes and learn sub-policies that realize the different modes of behavior towards reaching their ends. The Hopper needs to reach the end of terrain without falling while the aircraft needs to reach the destination airport while departing from an origin airport. The first algorithm that we have implemented and evaluated in this thesis towards identifying different modes (patterns of behavior), while also learning a policy according to the Directed-Info GAIL methodology, is a Variational Auto-Encoder (VAE) [20] with Gumbel-Softmax trick [16], which we explain in the next sections. While studying and experimenting on VAE we tested different hyper-parameters such as: learning

¹ <https://gym.openai.com/envs/Hopper-v2/>

rates, epochs, activation functions between the hidden layers and batch sizes of 1 and 32. Towards learning a better policy while exploiting the mode-identification model learnt, and towards making the agent reach its goal as good as the expert we are using the GAIL [1] algorithm. By combining those two algorithms we get the Directed-Info GAIL but firstly, we need to discuss some background knowledge about basic imitation learning techniques and other algorithms used in similar multi-dimensional imitation problems.

2 Preliminaries

2.1 Imitation Learning

In imitation learning, an agent learns how to behave in an environment with an unknown cost function by mimicking expert demonstrations. Existing imitation learning algorithms typically involve solving a sequential decision making problem, using Reinforcement Learning [5]. Imitation learning is related to Reinforcement Learning, but instead of having the AI agent learn from scratch through its own exploration, the agent learns decision policies from expert demonstrations.

2.2 Basic Imitation Learning Algorithms

There are three ways to recover an expert's reward/cost function: the first one is via behavioral cloning [6, 7], the second using inverse reinforcement learning [8] and the third one with Generative Adversarial Imitation Learning (GAIL) [1].

Behavioral cloning casts the problem as a supervised learning problem over the state-action pairs of the experts' demonstrations. Pros are, that behavioral cloning solves the problem as a regression problem, minimizing the error between the actions demonstrated and the policy actions, over the states of the historical trajectories. The cons are that this method suffers from compounding errors and a regret bound that grows quadratically in the time horizon of the task leading to poor performance. On the other hand, inverse reinforcement

learning aims at deriving a cost function that assigns minimal cost to trajectories demonstrated by experts and maximal cost to trajectories generated by other policies. Other cons of the method include assumptions on the linearity of the cost function, and the high computational cost incurred via the steps of cost approximation and policy computation (using the approximated cost function).

GAIL obtains a model-free imitation learning algorithm that enhances the performance in imitating complex behaviors in high-dimensional environments in a continuous state-action space, without approximating the cost function: However, these happen with a high sample complexity.

2.2.1 Behavioral Cloning

From the time humans are born they learn to imitate actions from the people around them. While they observe different situations in their environment, they learn very quickly the appropriate actions based on these observations. The straightforward and simplest form of imitation learning is called behavioral cloning [6, 7]. It uses the expert's observations and action demonstrations and applies supervised learning. The loss function is being chosen depending on the application, e.g the mean squared error is often used. The learning task exploits state-action pairs, where states are usually observations from the expert demonstrations. The agent learns from experience in a self-supervised way and this experience leads to learning a model that is used by the agent to perform a particular task towards an objective, without explicitly given a reinforcement signal. Behavior cloning succeeds better with a big amount of training data, because of the compounding error caused by covariate shift. Imitation Learning cannot be completely solved with behavior cloning: In traditional supervised learning, the training distribution is separated from the learned model, while in imitation learning, the agent's policy affects what state is queried next. Thus the training and testing distributions are no longer equivalent, and this inconsistency is known as covariate shift. Intuitively, the covariate shift is the difference between the expected error on the agent's distribution of data produced using the policy and the supervisor's distribution.

2.2.2 Inverse Reinforcement Learning (IRL)

Inverse reinforcement learning (IRL) is the problem of inferring the reward function of an agent, given its policy or observed behavior. It does not need such a big training set to work optimally, compared to behavioral cloning.

Inverse Reinforcement Learning, aims at deriving a cost function that assigns minimal cost to trajectories demonstrated by experts and maximal cost to trajectories generated by other policies. Given that many policies may be able to generate the same trajectories, the maximum entropy inverse reinforcement learning approach aims to find the maximum-entropy policy [9,10]. Actually, this process comprises two steps:

The first one outputs a desired cost function according to the following formula,

$$\text{maximize}_{c \in \mathbb{C}} (\min_{\pi \in \Pi} -H(\pi) + E_{\pi}[c(s, a)]) - E_{\pi_E}[c(s, a)] \quad \mathbf{(1)}$$

which can be alternatively [4, 8, 18, 19] be stated as follows using a regularizer:

$$IRL_{\psi}(\pi_E) = \text{argmax}_{c \in R^{S \times A}} -\psi(c) + (\min_{\pi \in \Pi} -\lambda_H H(\pi) + E_{\pi}[c(s, a)] - E_{\pi_E}[c(s, a)]) \quad \mathbf{(2)}$$

where, $\psi(c) : R^{S \times A} \rightarrow \mathbb{R} \cup \{\infty\}$ is a convex cost function regularizer, π_E is the expert policy (provided by the demonstrated trajectories) and Π is the set of all policies, $H(\pi)$ is the entropy function and λ_H is its weighting parameter.

The second step is to find a policy that minimizes the expected cumulative cost and maximizes the entropy by using the cost function approximation, into a standard reinforcement learning problem:

$$RL(c) = \text{argmin}_{\pi \in \Pi} -\lambda_H H(\pi) + E_{\pi}[c(s, a)] \quad \mathbf{(3)}$$

Specific instances of this process result into apprenticeship learning methods, e.g. the one described in paper [18], assuming that the cost function is given by a linear combination of basis functions, which result to feature vectors over states and actions. The linearity assumption can be a constraining one for complex problems, although basis functions can be arbitrarily complex. In addition, the hand-crafted state features are a big engineering burden. Finally, this method is computationally expensive as it runs a Reinforcement Learning algorithm at every cost function update, to find a policy that performs optimally in connection with the approximated cost function.

2.3 The GAIL Algorithm

The main aspect of Generative Adversarial Imitation Learning(GAIL) consists of learning a policy through expert demonstrations without specifically interacting with the expert and without approximating and/or using a reinforcement signal.

2.3.1 GAIL

GAIL [1] can imitate multiplex behavior as it does not apply constraining assumptions on the cost function and scales to big, continuous state-action environments. The optimal policy from expert demonstrations is directly learnt from GAIL, quite competently, since it does not need to derive a cost function that will be used by a Reinforcement Learning method to derive a policy. Actually, it aims to bring the distribution of the state-action pairs of the imitator as close as possible to that of the expert. GAIL utilizes an architecture similar to Generative Adversarial Networks to optimize the following objective:

$$\min_{\pi} \max_{D \in (0,1)^{S \times A}} E_{\pi} [\log D(s, a)] + E_{\pi_E} [\log(1 - D(s, a))] - \lambda_h H(\pi) \quad (4)$$

where, π is the imitator policy, π_E is the expert's policy, D is a binary classifier called discriminator that distinguishes state-action pairs generated from π and π_E . $H(\pi) \triangleq E_{\pi} [-\log \pi(a|s)]$, is the γ -discounted causal entropy of the policy π . As shown in [1], Eq. (4) provides a way to solve the two steps in the imitation process described by Eq. (2) and (3).

The GAIL authors [1] to start their research for an imitation learning algorithm that both bypasses an intermediate IRL step and is suitable for large state-action spaces, they study policies found by reinforcement learning on costs learned by IRL on the largest possible set of cost functions C in Eq. (1): all functions $R^{S \times A} = \{c : S \times A \rightarrow R\}$. It is critical when using expressive cost function classes, like Gaussian processes and neural networks, to properly explain complex expert behavior without demanding hand-crafted features. Here, it is shown what IRL can do when finding the best cost function with respect to expressiveness, by examining its capabilities with $C = R^{S \times A}$.

With such a large C , IRL can easily overfit when provided a finite dataset. Therefore, the GAIL authors incorporate a (closed, proper) convex cost function regularizer. Now, equation **(2)** is the definition of an IRL primitive procedure, which finds the best cost function that distinguishes the expert by other lower value policies, with the cost regularized by ψ .

There is a significant importance in a policy given by $RL(\tilde{c})$, where \tilde{c} is the cost function approximation achieved by IRL. The definition of $RL(\tilde{c})$ is critical to transform optimization problems over policies into convex problems. So we have a policy $\pi \in \Pi$ and we define its occupancy measure $\rho_\pi : S \times A \rightarrow \mathbb{R}$ as $\rho_\pi(s, a) = \pi(a|s) \sum_{t=0}^{\infty} \gamma^t P(s_t = s | \pi)$. The occupancy measure can be explained as the distribution of state-action pairs that an agent encounters when navigating through the environment with policy π , and it permits us to write $E_\pi[c(s, a)] = \sum_{s,a} \rho_\pi(s, a) c(s, a)$ for any cost function c . A basic result is that the set of valid occupancy measures $O \triangleq \{\rho_\pi : \pi \in \Pi\}$ can be written as a feasible set of affine constraints: if $\rho_0(s)$ is the distribution of starting states and $P(s'|s, a)$ is the dynamics model, then

$$O = \{\rho : \rho \geq 0 \text{ and } \sum_a \rho(s, a) = \rho_0(s) + \gamma \sum_{s',a} P(s|s', a) \rho(s', a) \forall s \in S\}.$$

Furthermore, there is a one-to-one correspondence between Π and O . [1]

According to the duality principle associated with the optimization theory, it assumes that optimization problems can be perceived as the primal problem or the dual problem. IRL is a dual of an occupancy measure matching problem, and the recovered cost function is the dual optimum. Classic IRL algorithms that solve reinforcement learning repeatedly in an inner loop, such as the maximum entropy inverse reinforcement learning algorithm in reference [19] that runs a variant of value iteration in an inner loop, can be interpreted as a form of dual ascent, in which one repeatedly solves the primal problem (reinforcement learning) with fixed dual values (costs). Dual ascent is effective if solving the unconstrained primal is efficient, which in the case of IRL it amounts to reinforcement learning. The induced optimal policy is the primal optimum. The induced optimal policy is obtained by running RL after IRL, which is exactly the act of recovering the primal optimum from the dual optimum. Strong duality implies that the induced optimal policy is indeed the primal optimum, and therefore matches occupancy measures with the expert.

IRL is traditionally defined as the act of finding a cost function such that the expert policy is uniquely optimal, but now, we can alternatively view IRL as a procedure that tries to induce a policy that matches the expert’s occupancy measure. The next sub-section describes in detail the details behind the GAIL algorithm, the new regularizer, and the algorithm itself.

2.3.2 More detailed information on GAIL

The authors of GAIL proposed a new cost regularizer that gives a solution to the use of other regularizers that lead into two basic problems. The first one is directing to an imitation learning algorithm that exactly matches occupancy measures but is not tractable in large environments. The second one is to have indicator regularizers δ_C for the linear cost function classes leading to algorithms incapable of matching occupancy measures without good tuning. This new regularizer on the other hand, addresses both of these aspects and is defined according to the following formulation:

$$\psi_{GA}(c) \triangleq \begin{cases} E_{\pi_E}[g(c(s, a))] & \text{if } c < 0 \\ +\infty & \text{otherwise} \end{cases} \text{ where } g(x) = \begin{cases} -x - \log(1 - e^x) & \text{if } x < 0 \\ +\infty & \text{otherwise} \end{cases} \quad (5)$$

It places a low penalty on cost functions c that assign an amount of negative cost to the expert state-action pairs; if c , however, assigns large costs (close to zero, which is the upper bound for costs attainable for ψ_{GA}) to the expert, then ψ_{GA} will heavily penalize c . Perhaps the most important difference between ψ_{GA} and δ_C , however, is that δ_C forces costs to lie in a small subspace, whereas ψ_{GA} allows for any cost function, as long as it is negative everywhere. The choice of ψ_{GA} is motivated by the following fact:

$$\psi_{GA}^*(\rho_\pi - \rho_{\pi_E}) = \max_{D \in (0,1)^{S \times A}} E_\pi[\log(D(s, a))] + E_{\pi_E}[\log(1 - D(s, a))] \quad (6)$$

where the maximum ranges over discriminative classifiers $D : S \times A \rightarrow (0, 1)$. Equation (6) is the optimal negative log loss of the binary classification problem of distinguishing between state-action pairs of π and π_E . It turns out that this optimal loss is (up to a constant shift) the Jensen-Shannon divergence

$D_{JS}(\rho_\pi - \rho_{\pi_E}) \triangleq D_{KL}(\rho_\pi || (\rho_\pi + \rho_{\pi_E})/2) + D_{KL}(\rho_{\pi_E} || (\rho_\pi + \rho_{\pi_E})/2)$, which is a squared metric between distributions. Treating the causal entropy H as a policy regularizer, controlled by $\lambda \geq 0$, they obtain a new imitation learning objective:

$$\text{minimize } \psi_{GA}^*(\rho_\pi - \rho_{\pi_E}) - \lambda H(\pi) = D_{JS}(\rho_\pi - \rho_{\pi_E}) - \lambda H(\pi) \quad (7)$$

towards finding a policy whose occupancy measure minimizes Jensen-Shannon divergence to the expert's. Equation (7) minimizes a true metric between occupancy measures, so, unlike linear apprenticeship learning algorithms, it can imitate expert policies exactly.

Ψ -regularized inverse reinforcement learning implicitly seeks a policy whose occupancy measure is close to the expert's, as measured by ψ^* . Here, occupancy measure (ρ), as specified above, is the distribution of state-action pairs encountered while navigating the environment with the policy.

Equation (7) draws a connection between imitation learning and generative adversarial networks, which train a generative model G by having it confuse a discriminative classifier D . The job of D is to distinguish between the distribution of data generated by G and the true data distribution. When D cannot distinguish data generated by G from the true data, then G has successfully matched the true data. In our setting, the learner's occupancy measure ρ_π is analogous to the data distribution generated by G , and the expert's occupancy measure ρ_{π_E} is analogous to the true data distribution. So in reference [1] authors propose a new practical algorithm which is called generative adversarial imitation learning (GAIL) for solving Eq. (7) for model-free imitation, which does not need a specific transition function in large environments. More specifically, the basic outcome is to find a saddle point (π , D) of the expression

$$E_\pi[\log(D(s, a))] + E_{\pi_E}[\log(1 - D(s, a))] - \lambda H(\pi) \quad (8)$$

To do so, there is an introduction of a function approximation for π and D : GAIL fits a parameterized policy π_θ , with weights θ , and a discriminator network $D_w: S \times A \rightarrow (0, 1)$, with weights w . Then, it alternates between an Adam gradient step on w to increase Eq. (8) with respect to D , and a Trust Region Policy Optimization (TRPO) step on θ to decrease Eq. (8) with respect to π . The TRPO prevents the policy from changing too much due to noise in the policy gradient. The discriminator network can be interpreted as a local cost function providing learning signal to the policy, specifically, taking a policy step that decreases expected cost with respect to the cost function $c(s, a) = \log D(s, a)$ will move toward expert-like regions of state-action space, as classified by the discriminator (they derived an estimator for the causal entropy

gradient $\nabla_{\theta}H(\pi_{\theta})$). It is important to clear out that TRPO [17] is an iterative procedure for optimizing policies, while assuring monotonic improvement. The next figure shows how GAIL works in pseudocode.

Algorithm 1 Generative adversarial imitation learning

- 1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
- 2: **for** $i = 0, 1, 2, \dots$ **do**
- 3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
- 4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s, a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s, a))]$$

- 5: Take a policy step from θ_i to θ_{i+1} , using the TRPO rule with cost function $\log(D_{w_{i+1}}(s, a))$. Specifically, take a KL-constrained natural gradient step with

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_{\theta} \log \pi_{\theta}(a|s)Q(s, a)] - \lambda \nabla_{\theta}H(\pi_{\theta}),$$

$$\text{where } Q(\bar{s}, \bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s, a)) \mid s_0 = \bar{s}, a_0 = \bar{a}]$$

- 6: **end for**
-

Figure 1. Algorithm Generative Adversarial Imitation Learning (GAIL). [1]

2.3.3 GAIL Framework

In this work we have used the GAIL implementations described in [4]: GAIL alternates between an Adam gradient step on w to increase equation (4) with respect to D , and a step on θ using the Trust Region Policy Optimization (TRPO) algorithm [17] to decrease equation (4). TRPO optimizes the following objective:

$$\underset{\theta}{\text{minimize}} (\mathbb{E}_{s \sim \rho_{\theta_{\text{old}}}, a \sim q} [\frac{\pi_{\theta}(a|s)}{q(a|s)} Q_{\theta_{\text{old}}}(a|s)])$$

$$\text{Subject to } (\mathbb{E}_{s \sim \rho_{\theta_{\text{old}}}} [D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot | s) \parallel \pi_{\theta}(\cdot | s))]) \leq \delta \quad (9)$$

where $\rho_{\theta_{\text{old}}}$ is the distribution of states generated using the prior-to-update (old) policy $\pi_{\theta_{\text{old}}}$, q is an action sampling distribution that we consider equal to $\pi_{\theta_{\text{old}}}$, π_{θ} is the updated policy with parameters θ , $Q_{\theta_{\text{old}}}$ is the state-action value function of the old policy and δ is a constant that constraints the KL divergence between $\pi_{\theta_{\text{old}}}$ and π_{θ} , preventing the policy from changing too much due to noise in the policy gradient. The TRPO optimization problem is solved as described in [17], using the conjugate gradient method. In reference [4], the authors set $\lambda_H = 0$, so they omit $-\lambda_H H(\pi)$ from the equation (4), following the practice in article [1].

A subtle point in this implementation is that, instead of approximating Q, they used a separate critic algorithm [12] to approximate the state advantage, evaluating the impact of the selected action to the update of policy network, defined as:

$$A_t = A(s_t, a_t | \pi) = Q(s_t, \pi(s_t)) - V^\pi(s_t) \quad ,$$

The aim is to lower the gradient variance. So they use the Generalized Advantage Estimation (GAE) algorithm introduced in [13], which provides a balance between low variance and a small amount of bias introduced. Formally, the advantage from the sampled state-action pairs is estimated as follows:

$$\hat{A}_t^{\text{GAE}(\gamma, \beta)} := (1 - \lambda)(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots)$$

where $\gamma \in [0, 1]$ is the discounting factor, β a hyper-parameter and

$$\hat{A}_t^{(k)} := -V(s_t) + r_t + \gamma r_{t+1} + \dots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}) \quad \mathbf{(10)}$$

Algorithm 1 in Figure 2 shows a part of the GAIL implementation used, in more detail.

Specifically, the authors in [4] pre-train G using Behavioral Cloning. Then, at each GAIL iteration, the algorithm samples from the initial state distribution and G generates roll-out trajectories. It uses the generated state-action samples and the samples of the historical trajectories to update the D parameters w. D is updated with cross entropy loss that pushes the output for the demonstrated state-action samples closer to 0 and π_θ state-action samples closer to 1. Next, the imitation algorithm takes a policy update step using the TRPO [17] update rule and $\log D(s, a)$ as the cost function approximation, to update θ . It must be noted that the t parameter in the denotation of the approximation of the state advantage in Algorithm 1 is left implicit, for simplicity of the presentation.

Algorithm 1 GAIL

- 1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy π_{θ_0} and discriminator parameters w_0
 - 2: **Output:** Policy π_θ
 - 3: Initialize policy using Behavioral Cloning.
 - 4: **for** $i=0,1,2,\dots$ **do**
 - 5: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
 - 6: Update D parameters w with the gradient
 - 7: $\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s, a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s, a))]$
 - 8: Estimate advantages $\hat{A}^{GAE(\gamma, \lambda)}$, according to $\pi_{\theta_{old}}$
 - 9: Take a policy step using the TRPO rule with cost function $\log(D_w(s, a))$:
 Take a KL-constrained natural gradient step with $\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \hat{A}^{GAE(\gamma, \lambda)}]$
 subject to $\mathbb{E}_{s \sim \rho_{\theta_{old}}} [D_{KL}(\pi_{\theta_{old}}(\cdot|s) || \pi_\theta(\cdot|s))] \leq \delta$
 - 10: **end for**
-

Figure 2. Algorithm Generative Adversarial Imitation Learning (GAIL) using Behavioral Cloning for pre-training and TRPO step with KL-constrained natural gradient step.

2.3.4 Generalized Advantage Estimation (GAE) and the trade-off between variance and bias

The main problem formulation in reinforcement learning is to maximize the expected total reward of a policy. A key problem is the long time delay between actions and their positive or negative effect on rewards: This is called the credit assignment problem in the reinforcement learning literature (Minsky, 1961; Sutton & Barto, 1998), and the distal reward problem in the behavioral literature (Hull, 1943). A solution to the credit assignment problem is proposed by using value functions, they make it possible to predict how good is an action before the delayed reward arrives. Value functions are being used by reinforcement learning algorithms in many non-identical ways.

This approach aims at algorithms that optimize a parameterized policy and use value functions to help approximate how the policy could be improved. It is possible to obtain an unbiased estimate of the gradient of the expected total returns, when using a parameterized stochastic policy (Williams, 1992; Sutton et al., 1999; Baxter & Bartlett, 2000). These noisy gradient approximations can be used in a stochastic gradient ascent algorithm. The variance of the gradient estimator scales badly with the time horizon, since the outcome of an action is depended on the effects of past and future actions. Actor-critic methods use a value function rather than the empirical returns, acquiring an estimator with lower variance at the cost of introducing some bias [12]. There is a large possibility, that the introduction of bias can harm the algorithm to fail converge

and give some solution that is not the optimal one. So, while high variance needs more samples, bias can be destructive.

Reinforcement learning can be reduced to stochastic gradient descent, if someone uses policy gradient methods, which provide unbiased gradient estimates. Nevertheless, due to variance and high sample complexity, their success of solving hard control problems is low. GAE's authors delivered an unofficial inspection of the problem of advantage function estimation and explained the generalized advantage estimator, which has two parameters γ , λ which adjust the bias-variance tradeoff. In previous sections, it is described how to combine this idea with trust region policy optimization and a trust region algorithm that optimizes a value function, both defined by neural networks. Combining these techniques, we are able to make the agent learn to solve difficult control tasks that have previously been out of reach for generic reinforcement learning methods. In our experiments as GAE authors suggested, choosing an appropriate intermediate value of λ in the range $[0.9, 0.99]$ usually results in the best performance [13].

2.4 Info GAIL Algorithm

The goal of imitation learning is to mimic expert behavior without access to an explicit reward/cost signal. The expert datasets created by humans, show inconsistencies due to latent variables that are frequently not explicitly modeled. In Info Gail paper [2], authors propose a new algorithm that can infer the latent structure of expert demonstrations in an unsupervised way. Their method, built on top of GAIL, can not only imitate complex behaviors, but also learn interpretable and meaningful representations of complex behavioral data, including visual demonstrations. Compared with various baselines, Info Gail can better capture the latent structure underlying expert demonstrations, often recovering semantically meaningful factors of variation in the data. The makers of Info Gail solved the problem of learning from policies created by several experts. A latent variable c is assigned into the policy function $\pi(a|s, c)$ to identify non-identical type of behaviors appearing in the demonstration. The proposal employs an information-theoretic regularization enforcing that high

mutual information is needed between c and the state-action pairs in the generated trajectory [2]. This concept appeared in a previous proposal on InfoGAN (Chen et al., 2016) [15]. Finally, there was an introduction of a variational lower bound $L_I(\pi, Q)$ of the mutual information $I(c; \tau)$ between latent variable c and the state-action pairs in the generated trajectory to the loss function in GAIL.

$$L_I(\pi, Q) = E_{c \sim p(c), a \sim \pi(\cdot|s, c)} \log Q(c|\tau) + H(c) \leq I(c; \tau) \quad (11)$$

The constructed objective can be derived as,

$$\min_{\pi, q} \max_D E_{\pi}[\log D(s, a)] + E_{\pi E}[1 - \log D(s, a)] - \lambda_1 L_I(\pi, q) - \lambda_2 H(\pi) \quad (12)$$

Info Gail maps the differences between different trajectories to the latent variables, as trajectories may originate from a variety of experts.

While Info Gail identifies inter trajectory variations, we aim to model intra-trajectory variations. Latent codes in our work correspond to sub-tasks (variations) within any single demonstration. As shown in [1], mutual information-based loss is not working in our problem setting. We will explain this in section 3.2.

3 The Directed-Info GAIL Algorithm

As noted in the Directed Info-GAIL paper “the use of imitation learning to learn a single policy for a complex task that has multiple modes or hierarchical structure can be challenging”[3]. This thesis explores the use of Directed-Info GAIL algorithm, which is based on the generative adversarial imitation learning (GAIL) framework [4] to automatically learn subtask policies from unsegmented demonstrations of trajectories in robotic and aviation environments, given that flights and robots’ paths have indeed different modes of behavior in different segments of trajectories, depending on task and trajectories’ contextual features.

3.1 Detailed Description of Directed-Info Gail

Directed Info-Gail [3] serves a great aspect in Deep Imitation Learning while it can break a single policy into smaller sub-policies. A monolithic policy for each task ignores the shared sub-structure between different tasks, so this algorithm can identify sub-tasks to achieve a variety of behavior. The benefits of learning sub-task specific policies are the following:

- a) Each of the sub-tasks is smaller and easier to learn.
- b) There is so much to achieve from shared learning .
- c) The agent needs to learn how to combine these smaller tasks.

The research that has been done from the authors of Directed-Info Gail Algorithm has shown that, if someone has unsegmented expert demonstrations of a complex task then the agent can learn to identify different patterns of behavior and take better decisions than other similar model-free imitation algorithms. The goal of the agent is to learn to segment task demonstrations in sub-tasks, learn smaller policies for each sub-task, and learn a policy which switches between sub-tasks. Hence, someone can comprehend this by imagining the algorithm as a graphical model so as to model sub-tasks given the demonstrations in high-dimensional environments, e.g like the aviation environment. Directed-Info Gail shows that using mutual information maximization as in Info-Gail can be problematic and that using directed/causal information can solve this problem.

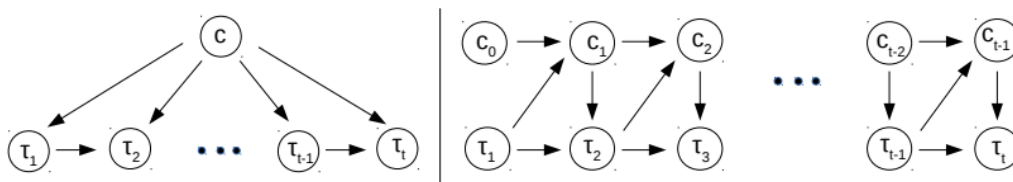


Figure 3. C nodes dictate the sub-activity latent variables and τ nodes are the observed expert demonstrations. The left graph corresponds to the Info Gail algorithm while the right one to the Directed Info-Gail. [1]

In Figure 3(right), we can see the latent code c that is being used by the policy to produce a trajectory by following a specific mode of behavior. The current trajectory up to a specific time-point and the previous latent variable produce the latent code in the next time-point.

The indicated problem [2, 3] can be seen in the following Info Gail objective:

$$L(\pi, q) = \sum_t (E_{c^{1:t} \sim p(c^{1:t}), \alpha^{t-1} \sim \pi(|s^{t-1}, c^{1:t-1})}) [\log q(c^t | c^{1:t-1}, \tau)] + H(c) \leq I(\tau; c) \quad (13)$$

Notice that $L(\pi, q)$ depends on the complete trajectory τ , hence we cannot use q at test time since the future is unknown. To overcome this limitation, the Directed-Info GAIL’s authors suggest to force the policy to generate trajectories that maximize the directed information flow from trajectories to the sequence of latent sub-activity variables, instead. So we will use the directed information instead of mutual information. We can replace the dependence on τ (as indicated in Figure 3 (left)) with a dependence on the trajectory generated up to current time t : $\tau^{1:t} = (s_1, \dots, a_{t-1}, s_t)$ (as indicated in Figure 3 (right)). A new variational lower bound is proposed, based on the directed/causal information, $I(\tau \rightarrow c)$ which uses an approximate posterior $q(c^t | c^{1:t-1}, \tau^{1:t})$ instead of true posterior $p(c^t | c^{1:t-1}, \tau^{1:t})$,

$$L_1(\pi, q) = \sum_t E_{c^{1:t} \sim p(c^{1:t}), \alpha^{t-1} \sim \pi(|s^{t-1}, c^{1:t-1})} [\log q(c^t | c^{1:t-1}, \tau^{1:t})] + H(c) \leq I(\tau \rightarrow c) \quad (14)$$

Notice $L_1(\pi, q)$ now depends on $\tau^{1:t}$, thus we can use q at test time since it only depends on the past. Combining the GAIL objective with this lower bound we have the following objective:

$$\min_{\pi, q} \max_D E_{\pi} [\log D(s, a)] + E_{\pi_E} [1 - \log D(s, a)] - \lambda_1 L_1(\pi, q) - \lambda_2 H(\pi) \quad (15)$$

where $L_1(\pi, q) = \sum_t E_{c^{1:t} \sim p(c^{1:t}), \alpha^{t-1} \sim \pi(|s^{t-1}, c^{1:t-1})} \log q(c^t | c^{1:t-1}, \tau^{1:t})$, H is the policy entropy (that we do not use), and we can learn a posterior distribution over the next latent factor c given the latent factors discovered up to t and the trajectory followed up to t , thereby removing the dependence on the future trajectory. In equation (15) we need to maximize the objective whereas the loss must be minimized. To calculate the loss in equation (14) we need to sample from the prior distribution $p(c^{1:t})$, as we explained. In order to approximate this distribution, together with a first approximation of the expert policy, we first train a variational auto-encoder (VAE) on the expert trajectories. The models learnt by VAE are exploited in conjunction to GAIL towards learning a policy that imitates the expert samples, mimicking the behaviour modes demonstrated.

3.2 Detailed Description of the implementation of the Directed-Info Gail algorithm

This section aims to provide a better understanding of the Directed-Info GAIL algorithm. At the end of the previous sub-section we talked about mapping different sub-policies in the latent features: This is achieved using Variational Auto-Encoder and adding a variational lower bound to GAIL objective.

As proposed in [3] the Variational Auto-Encoder is used in a different way than the vanilla version, also approximating the expert policy, given the demonstrated trajectories. The VAE is made up of one encoder and one decoder. In the vanilla version, the auto-encoder is often used for reconstruction of the input data but in this thesis, we used the encoder to produce latent variables, which divide the trajectory in sub-tasks. The decoder aims to learn predicting the actions of the expert policy. Figure 4(right) gives an overview of the complete method. The VAE pre-training step is used to learn an approximate prior over the latent variables. So the Figure 4(left) also shows the design of the VAE architecture. The VAE consists of two multi-layer perceptrons that play the role of the encoder and the decoder. The encoder uses the current state s_t and the previous latent variable c_{t-1} to produce the current latent variable c_t . We used the Gumbel-softmax trick [16] to obtain samples of latent variables from a categorical distribution. The decoder then takes s_t and c_t as input and outputs the action a_t . We saved only the weights and biases of the encoder and the decoder, which will we ‘feed’ in GAIL which has 2 MLP networks, one predicts the latent variables at any time step (called “encoder clone”) and the other initializes the policy network used by the GAIL generator G .

The predicted latent variables are concatenated with the GAIL policy rollouts, that are created by the generator and pass them through the TRPO policy network (MLP network), that predicts the action and generates rollouts. We present this methodology in Figure 5 in more detail.

To summarize, Directed-Info GAIL contains a TRPO agent, a generator, a discriminator network, a critic network, an encoder clone and a policy to produce the actions (as the decoder does). The generator produces rollouts while the discriminator discriminates the samples it gets from the generator and the expert.

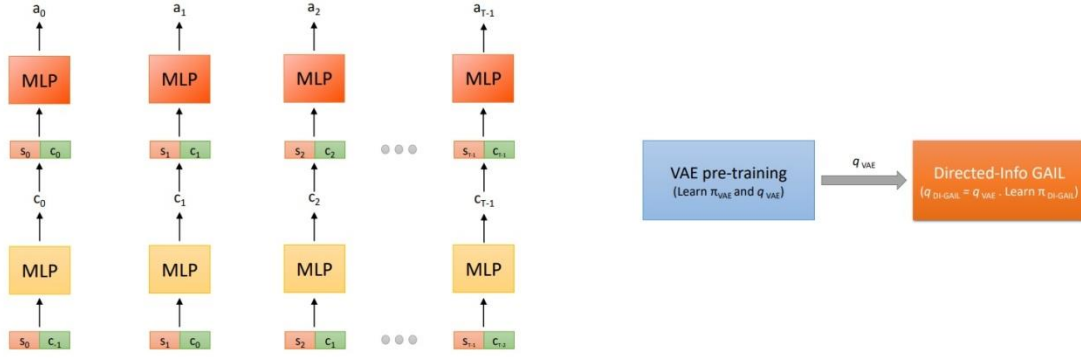


Figure 4. Directed-Info GAIL overall architecture.

We use the following objective, which maximizes the lower bound of the probability of the trajectories $p(\tau)$, to train our VAE,

$$L_{\text{VAE}}(\pi; q; \tau_i) = -\sum_t E_{c^t \sim q} [\log \pi(a^t | s^t, c^{1:t})] + \sum_t D_{\text{KL}}[q(c^t | c^{1:t-1}, \tau^{1:t}) || p(c^t | c^{1:t-1})] \quad (16)$$

In Figure 4, on the left the VAE pre-training step allows us to get approximate samples from the distribution $p(c^{1:t})$ to optimize the Eq. (15). This is done by using q to obtain samples of latent variable sequence c by using its output on the expert demonstrations. In practice, we fix the weights of the network q to those obtained from the VAE pre-training step when optimizing the Directed-Info GAIL loss in equation (15).

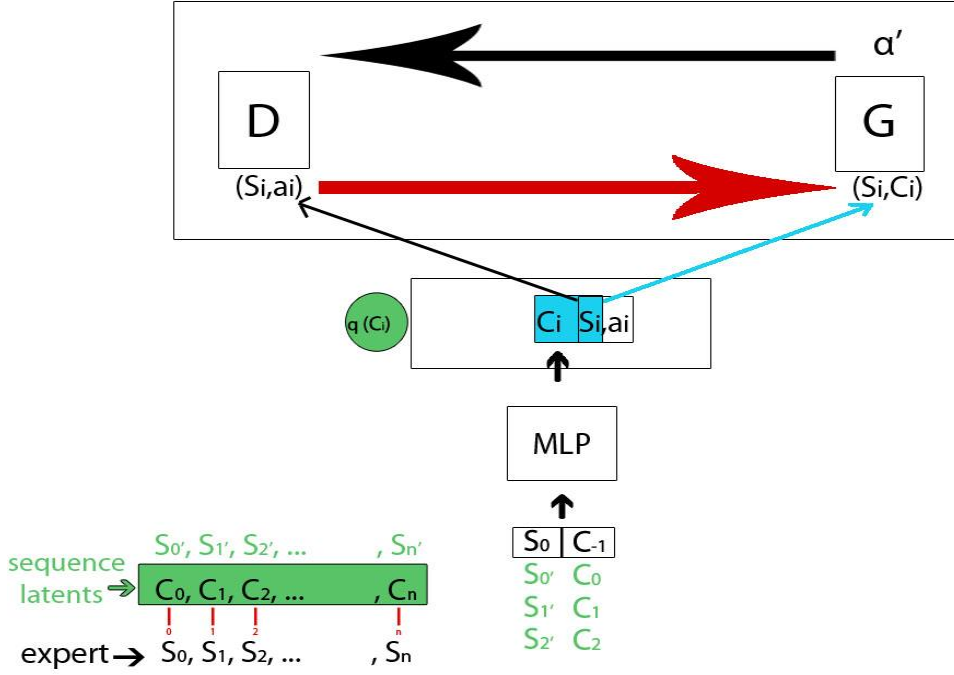


Figure 5. Directed-Info Gail' detailed approach.

In Figure 5: D stands for Discriminator, G for Generator, S for State, C for latent variable and a dictates the action. MLP is the encoder clone. The sequence of latents is predicted by the encoder clone using the pre-trained (exploiting expert samples) Variational Auto-Encoder.

More specifically, in our work we followed the GAIL framework in section 2.3.3, but now the generator now takes as input states and latent variables predicted, and the policy model is pre-trained using the decoder of the Variational Auto-Encoder. In addition to those, the imitation algorithm takes a policy step using the TRPO [17] update rule and $\log D(s, a)$ as the cost function approximation to update θ , together with the penalty of the encoder defined as $\log q(c^t | c^{1:t-1}, \tau^{1:t})$. In our setting, we set $\lambda_1 = 0.01$, which is a regularizer of L1 variational lower bound and we set $\lambda_2 = 0$, so we omit $-\lambda_2 H(\pi)$ from the equation (15), following the practice demonstrated in [1, 3, 4]. GAIL alternates between an Adam gradient step on w to maximize equation (15) with respect to D , and a step on θ using the Trust Region Policy Optimization (TRPO) algorithm [17] to minimize equation (15).

We can explore the Directed-Info GAIL in pseudocode in the next Figure.

Algorithm 2 Directed-Info GAIL

- 1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy π_{θ_0} and discriminator parameters w_0 and **initial latent variable** C -1
 - 2: **Output:** Policy π_θ
 - 3: Initialize policy, encoder using Variational Auto-Encoder
 - 4: **for** $i=0,1,2,\dots$ **do**
 - 5: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
 - 6: Update D parameters w with the gradient
 - 7: $\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s, a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s, a))]$
 - 8: Estimate advantages $\hat{A}^{GAE(\gamma, \lambda)}$, according to $\pi_{\theta_{old}}$
 - 9: Take a policy step using the TRPO rule with cost function $\log(D_w(s, a)) + \log q(c^t | c^{1:t-1}, \tau^{1:t})$
 Take a KL-constrained natural gradient step with $\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \hat{A}^{GAE(\gamma, \lambda)}]$
 subject to $\mathbb{E}_{s \sim \rho_{\theta_{old}}} [D_{KL}(\pi_{\theta_{old}}(\cdot|s) || \pi_\theta(\cdot|s))] \leq \delta$
 - 10: **end for**
-

Figure 6. Algorithm 2 Directed-Info Generative Adversarial Imitation Learning (GAIL).

The GAIL framework appears in Algorithm 2. In line 9, when the policy step is applied using the TRPO rule in conjunction to the cost function we are subtracting $\log q$, which is the penalty enforced by the encoder clone, where q is the approximate posterior distribution.

4 Environments and definition of the trajectory imitation problem

In this thesis, we approach the data-driven trajectory prediction problem as an imitation learning task together with the identification of modes of behavior, in two settings: The Hopper simulated robotic setting, and an aviation setting.

- ❖ In the aviation environment, the flights paths (trajectories) are from Barcelona’s airport to Madrid.
- ❖ In the Hopper trajectories are performed by a two-dimensional one-legged robot. Its goal is to bounce as far as possible without falling. There is more information about the hopper environment on the Open-AI page².

4.1 Aviation Environment

In the aviation domain, using artificial intelligence to predict aircrafts’ trajectories can be challenging. In this thesis, we will study about aircrafts’ trajectories from Barcelona to Madrid. The aircraft trajectory describes the movement of the aircraft both in the air and on the ground. The most relevant observations are: longitude (l), latitude (f) and altitude (h). The raw trajectory can be demonstrated as a sequence of $|T|$ pairs $S_i = (p_i, t_i)$, where p_i is a 3d point (l, f, h) and t_i the corresponding timestamp. An enriched trajectory is a raw trajectory enhanced with a vector v_i , consisting of categorical and numerical variables, $S_{r,i} = (p_i, t_i, v_i)$. The predicted trajectory is the future evolution of the aircraft state given the current flight conditions.

The datasets we used are flights, with meteorological features from METAR and NOAA. In total there are 528 real flight trajectories. Demonstrated trajectories are enriched with eleven (11) numerical variables corresponding to 6 meteorological features at the corresponding 3D state position and time,

² <https://gym.openai.com/envs/Hopper-v2/>

provided by NOAA, and 5 features specifying actual weather conditions at the arrival airport at the time of arrival, provided by METAR. The NOAA features are pressure surface, relative humidity, temperature, wind speed gust surface, u-component of wind, v-component of wind. Features from METAR include wind direction, wind speed in knots, pressure altimeter in inches, visibility in miles, wind gust in knots.

THE EVALUATION REWARD:

As far as the reward is concerned, and as pointed out, the reward signal to the agent is unknown, as only the expert trajectories are known during imitation learning. However, we use the evaluation reward to evaluate the policy of the agent as external observers. This type of reward do not play any role in the learning process and does not impact the agent behavior in the environment. More specifically, the evaluation reward, is as follows:

If the aircraft is within a specific distance from the destination airport the reward for that particular time-step is 0, if it gets out of the boundaries there is a penalty. The reward function is the following:

if $\text{distance}(\text{point}, \text{self.destination}) < \text{conf}[\text{destination distance}]$: reward = 0 ,

The maximum destination distance of the trajectory (denoted above as destination distance) is set to 5000 meters radius from the destination airport. So as long the great-circle distance between the current point and the destination is smaller than the destination distance the reward is 0. The penalty function is the following:

reward = $-\text{distance}(\text{point}, \text{self.destination}) * 0.01$,

The variable $\text{point}[\text{longitude}, \text{latitude}]$ is the current destination of the aircraft in the particular time-step, self.destination are the coordinates of the airport of Madrid and the distance function calculates the great-circle distance between 2 points. The great-circle distance is the shortest distance between two points on the surface of a sphere. The mean rewards' values at test time are between -379.6 – 0.

4.2 Aviation Trajectory Imitation Problem

AI models in Reinforcement and Imitation learning represent among others a main policy, therefore, a strategy that an agent uses in pursuit of goals, but here

our goal is to identify the policy of aircraft trajectory evolution (i.e predict the aircraft trajectory), together with sub-policies. Forming the trajectory prediction to a data-driven problem, and presuming a set $T_E = \{T_{E,i} | i = 1, \dots, N\}$ of historical, demonstrated enriched trajectories, the trajectory prediction problem can be defined as follows: Given T_E and a cost function ‘cost’, the objective is to predict a trajectory T_π such that

$$T_\pi = \underset{\pi}{\operatorname{argmin}} E_\pi[\operatorname{cost}(\langle p, t, v \rangle, a)]$$

where, E dictates the expected cumulative costs for all states s generated along the trajectory by following a policy $\pi(a|s, c)$ describing the probability of applying an action a at an enriched state $s = \langle p, t, v \rangle$ with mode c . Actually, according to the above equation, the ultimate objective is to find the policy π that determines the generation of a minimal expected-cumulative-cost predicted trajectory T_π . The algorithm is going to divide the task in smaller sub-policies that will help the agent distinguish different patterns of behavior in different states along the aircraft’s trajectory on specific time instances depending on aircraft and contextual (e.g meteorological) features. So we can provide the approach to solve the following problem: The data-driven aircraft trajectory prediction problem as an imitation learning task in a hierarchical manner: Learning the behavior modes, and sub-policies corresponding to these modes so as to imitate the whole trajectories.

Let us assume a set $T_E = \{T_{E,i} | i = 1, \dots, N\}$ of historical, raw or enriched aircraft trajectories, depending on the experiment, generated by an expert policy π_E . These trajectories have a number of states $|T_{E,i}|$. The objective is to find sub-policies that ultimately minimize the difference between the expected cumulative cost of the predicted trajectories and of the trajectories in T_E . As noted in paper [1], this objective is equivalent to finding a policy π that brings the distribution of the state-action pairs generated by it, as close as possible to the distribution of the state-action pairs demonstrated by trajectories in T_E . As pointed out in section 4.1 and as done in [4], we aim at determining the progress of the trajectory in space every Δ_t seconds, i.e. at time instances $t_i = t_0 + (\Delta_t * i), i = 1, 2, 3, \dots$, given the position of the aircraft at time instance t_0 .

As done in [4] the set of agent actions A contains all the possible triples $(\Delta l, \Delta f, \Delta h)$ that specify the difference between states' position information in 3D, given the constraint that this difference must be attainable within the constant Δt period considered. Indeed, these actions can be determined by the demonstrated trajectories clear and effectively, although in low-quality surveillance data space-time constraints concerning the evolution of aircraft states may be violated. This action set has three additional important effects. We can tune the resolution of the predicted trajectory by changing the Δt . Given a specific Δt (5 seconds), and the evolution of the trajectory until reaching the destination airport, we can determine the estimated time of arrival, which is simply $(\Delta t * |T_\pi|)$, given the predicted trajectory T_π . The transition between positions is deterministic given an action: Given position (l, f, h) and an action $(\Delta l, \Delta f, \Delta h)$, the position in the next state is $(l + \Delta l, f + \Delta f, h + \Delta h)$.

Given the above, extending the problem specified in [4] the data-driven aircraft trajectory prediction problem as an imitation learning task is specified as follows:

Given a set $T_E = \{T_{E,i}, i = 1, \dots, N\}$ of historical, enriched aircraft trajectories, and a time step Δt , we need to determine a policy $\pi \in \Pi$ and different modes of behavior that the policy follows towards optimizing the objective specified by equation (15). This policy, given the initial state of aircraft $s_0 = \langle (l_0, f_0, h_0), t_0, v_0 \rangle$, decides the evolution of the trajectory at any time instant $t_0 + (\Delta t * i), i = 1, 2, 3, \dots$. Precisely, it regulates $\pi((\Delta l, \Delta f, \Delta h) | s, \Delta t)$, basically the evolution of the aircraft position at state $s_r = \langle (l, f, h), t, v \rangle$, after Δt time steps.

4.3 Hopper Environment

OpenAI research company has created many robots' simulations, one of those is the Hopper, which we used in our experiments. The Hopper, as mentioned above, is a one-legged robot that needs to reach its goal without falling in the environment. If it falls the environment resets and the robot starts again from the beginning. Towards successful execution of tasks, Hopper's creators, added some constraints to check hopper's height and angle bounds and if it reaches its destination they activate a Boolean variable called 'done'. In this thesis we

programmed Hopper to do 10 jumps (1000 time-steps) in each episode. The documentation on Hopper’s observations, which we used to train the algorithms were not clear, but we know that Hopper has three joints and the features acquired are surely position and velocity. We can further explore these in the code of Hopper’s environment. So basically, the dataset consists of eleven observations and three actions at each time point.

THE EVALUATION REWARD:

As specified in the aviation domain, and without explicit knowledge of the reward signal from the environment, we are using an evaluation reward that represent Hopper’s successful behavior. We can understand from the Hopper environment that the evaluation reward in each step equals to:

$$reward = (posafter - posbefore) / self.dt,$$

“posafter” stands for the future position after the step, the variable “posbefore” is referring to the position of hopper before the step and self.dt is the time needed for this action. The hopper gains +1 if it's still alive (up and running) at any time-step. Also, every time the step function is a squared sum of the 3 actions’ values multiplied by 0.001 and all this gets subtracted to the reward variable. Usually, the step function is called maximum 1000 times (i.e. constructing trajectories of 1000 time steps). The mean rewards’ values at test time are in the range of 2719.25-3762.89, depending on how good the agent’s policy has learned to act in every state. So we can expect to have a positive number for total reward.

4.4 Hopper Trajectory Imitation Problem

Similarly to the aviation domain, the algorithm is going to divide the task in smaller sub-policies that correspond to different modes of behavior. So we can provide a formulation of the problem we address here: The data-driven hopper trajectory prediction problem aims to learn a policy model that imitates expert trajectories and the pre-training of the variational auto-encoder aims to solve the problem regarding the identification of sub-policies.

Let us assume a set $T_E = \{T_{E,i}, i = 1, \dots, N\}$ of historical hopper trajectories, depending on the experiment, generated by an expert policy π_E . These trajectories have a number of states $|T_{E,i}|$. The objective is to find sub-policies, that they will ultimately minimize the difference between the expected cumulative cost of the predicted trajectories and of the trajectories in T_E , given an approximation of the cost function (discriminator) together with the penalty of the pre-trained encoder that penalizes any state-action pair generated by any policy in $\Pi - \{\pi_E\}$. As noted in paper [1], this objective is equivalent to finding a policy π that brings the distribution of the state-action pairs generated by it, as close as possible to the distribution of the state-action pairs demonstrated by trajectories in T_E . In this project, we want to predict the hopper's position at specific time steps, given an initial time instant t_0 . Especially, we aim at determining the progress of the trajectory in space every Δ_t seconds, i.e. at time instances $t_i = t_0 + (\Delta_t * i), i = 1, 2, 3, \dots$, given the position of the leg robot at time instance t_0 [4].

5 Algorithms' Structure

Here we provide details on the algorithms implemented and used in the experiments. More analytically: the inputs, the size of the hidden layers, the output layers and the activation functions which had been applied.

5.1 GAIL Framework

5.1.1 Behavioral Cloning – Pre-Training

As we described in section 2.2.1 when using Behavioral Cloning the agent learns in a supervised way. The data is split in training and validation, in 10 folds with K-Fold Cross Validation, the epochs used are 100, with batch size of 64 and with loss function the mean squared error together with Adam Optimizer as the authors in reference [4] suggested.

5.1.2 Trusted Region Policy (TRPO) agent with a Discriminator and an Actor critic - GAIL

For GAIL we used the Trusted Region Policy algorithm [17]. The settings for the GAIL algorithm are as follows:

Gamma = 0.995 for the discounted reward, lamda = 0.97, max_kl= 0.01 and logstd = 0.6. The algorithm makes 1500 updates which correspond to mini batches and in each update almost 50000 state-action pairs (mini batch size) are used. The maximum path size of each trajectory in the Hopper and aviation domains is 1000.

The policy network, is consisted of two hidden layers of 100 units with tanh activation function and outputs three actions in the Hopper and in the aviation environment. The discriminator, uses the same hidden layers as the policy but the output layer has 1 unit. The discriminator epochs are set to 100. The loss function is called 'Categorical Cross Entropy with logits'. The critic algorithm follows a similar architecture. The input comprises only the expert demonstrations, two hidden layers of 100 units and the output layer has 1 unit

to estimate the advantage. The loss we used is mean-squared error with Adam optimizer with an initial learning rate of 0.001 and the epochs are set to 100.

5.2 Directed Info-Gail Framework

5.2.1 Variational Auto-Encoder with Gumbel-Softmax Trick – Pre-Training

As suggested in the original Directed-Info GAIL paper, to compute the mean squared error loss of the VAE, we need to sample from the prior distribution on behavior modes. In order to approximate this distribution we first pre-train a VAE on the expert trajectories. The VAE is being formed of two multi-layer perceptrons that serve as encoder and decoder. The encoder in the first run uses an initialized random one-hot encoded latent variable (1. 0. 0.) or (1. 0. 0. 0. 0.) of shape three or five along with the observations which comprise the first state, in order to produce the next latent variable, sampled from the Gumbel-Softmax distribution. Generally, in the next runs the encoder uses the previous latent variable c_{t-1} along with the current state s_t (the observations) to produce the current latent c_t . The decoder then takes s_t and c_t as input, to decode the action in the proposed environment, according to the expert demonstrations. Each network consists of 2 hidden layers with 100 units in each layer and Tanh as a non-linearity function. We used Adam as optimizer with an initial learning rate of 0.0001. For the VAE pre-training step we set the VAE learning rate to $3e^{-4}$. For the Gumbel-Softmax distribution we set an initial temperature $t = 5.0$ and minimum temperature $t_m = 0.1$. The temperature decays (annealing rate = 0.00003) using an exponential decay with the following schedule $\tau = \max(0.1, \exp(-k*t))$, where $k = 3e - 3$ and t is the current epoch. We used batch size of 32 as suggested, while we encountered problems when trying to train the VAE with batch size of 1. We applied 2000 epochs for training the VAE, as suggested in the Directed-Info Gail paper [3] and we applied 50000 embedded iterations in the Hopper environment, which are the state-action pairs. The same structure applies in the aviation environment but the main difference with the hopper environment is that we use 327000 samples so 327000 embedded iterations which basically are 478 flight trajectories. In the aviation domain, the aircraft doesn't necessarily reach its destination in 1000 time-steps like Hopper, it can

reach it in more or less time-steps. Next, we apply the GAIL algorithm [1, 4] as shown in section 5.1.2. G has a dense output layer with size equal to the number of action variables (i.e. three), while the output layer of D has one node. G outputs for each action variable the mean of a Gaussian distribution with logarithm of standard deviation equal to 0.9, resulting to a stochastic policy. To initialize the policy's parameters, we use Variational Auto-Encoder minimizing the Mean Square Error between demonstrated actions and the policy actions, over the training set, using Adam optimization. This has been trained with 2000 epochs. GAIL is trained for 1500 epochs. At each round the policy generates a batch of 50000 state-action samples. The number of episodes needed to acquire this number of samples is not constant.

5.3 Basic Parameters

Hopper Hyper-parameters used in the experiments:

- Behavioral Cloning, 100 epochs with k-fold cross validation (k=10), 50000 state-action pairs
- GAIL, 1500 epochs, 50000 state-action pairs
- VAE-Batch1, 2000 epochs, 50000 state-action pairs
- VAE-Batch32, 2000 epochs, 50000 state-action pairs
- Directed-Info GAIL, 1500 epochs, 50000 state-action pairs

Aviation Hyper-parameters used in the experiments:

- Behavioral Cloning, 100 epochs with k-fold cross validation (k=10), 327000 state-action pairs
- GAIL, 1500 epochs, 327000 state-action pairs
- VAE-Batch1, 2000 epochs, 327000 state-action pairs
- VAE-Batch32, 2000 epochs, 327000 state-action pairs
- Directed-Info GAIL, 1500 epochs, 327000 state-action pairs

6 Experimental Evaluation

In this section, we will demonstrate the results of the algorithms implemented: Variational Auto-Encoder and Directed-Info Gail (D-INFO GAIL), in both evaluation settings. We will also present some results from the Behavioral Cloning (Bcloning) algorithm and the Generative Adversarial Imitation Learning (GAIL) algorithm, as described and used in reference [4].

6.1 Experimental Setting

Datasets utilized in our experiments include in the aviation environment: radar tracks (surveillance data representing raw trajectories) for flights from Barcelona to Madrid and from the 1st to the 24th of April 2016 , weather data obtained from National Oceanic and Atmospheric Administration (NOAA), and weather reports from airports (METAR). In the Hopper environment we used the position and velocity of the robot from starting point to the goal which is the end of the terrain. The hopper experts' dataset was generated by berkley's code³, which used an MLP policy consisted of 2 hidden layers of 64 units and tanh as the activation function between of the layers. The aim is to predict trajectories with the different modes of behavior along the path in both environments.

To implement the generative model G and the discriminator D in GAIL we have used two neural networks, each consisting of two dense layers of 100 nodes, each layer with tanh activation, as described in the previous section. The input for G in the aviation domain comprises the four 3D position and temporal variables per state, the six meteorological features provided by NOAA and five meteorological features at the destination airport at the time of the arrival provided by METAR. In the other setting the input G comprises the 11 features of Hopper⁴. D takes as additional input the three action variables. At each episode the method selects a starting point regarding a trajectory in the training

³ <https://github.com/berkeleydeeprlcourse/homework/tree/master/hw1>

⁴ There isn't any documentation on hopper's state features, everything is written is based from the environment's code.

set and uses G to generate roll-outs. In aviation domain, the roll-outs terminate either when a trajectory point lies within a 5km radius from the destination airport, or when the trajectory has 1000 points, or when it lies outside the bounding box defined by the geographic (longitude, latitude) coordinates $(-3.7038, 41.4)$, $(2.9504, 39.9864)$.

To evaluate the proposed approach we provide results regarding the prediction of Barcelona-Madrid trajectories and the Hopper trajectories, in the following experimental settings:

- Raw aircraft trajectories with 3 modes
 - 478 training trajectories
 - 50 testing trajectories
- Raw aircraft trajectories with 5 modes
 - 478 training trajectories
 - 50 testing trajectories
- Enriched aircraft trajectories with Raw+Noaa with 3 modes
 - 478 training trajectories
 - 50 testing trajectories
- Enriched aircraft trajectories with Raw+Noaa with 5 modes
 - 478 training trajectories
 - 50 testing trajectories
- Enriched aircraft trajectories with Raw+Noaa+Metar with 3 modes
 - 478 training trajectories
 - 50 testing trajectories
- Enriched aircraft trajectories with Raw+Noaa+Metar with 5 modes
 - 478 training trajectories
 - 50 testing trajectories
- Hopper trajectories with 3 modes
 - 50 training trajectories
 - 50 testing trajectories
- Hopper trajectories with 5 modes
 - 50 training trajectories
 - 50 testing trajectories

The Directed-Info GAIL results reported are generated from 5 independent experiments per setting all the other algorithms are tested in 1 experiment, considering each of the 50 test trajectories. Generally, we report on the testing mean rewards (evaluated in testing) for each case.

6.2 Evaluation rewards gained by the Algorithms

6.2.1 Training

In this subsection, we are going to demonstrate the evaluation rewards achieved by Directed-Info GAIL while the agent is being trained in both environments: Results are indicative, from a single experiment.

First, in the Hopper setting:

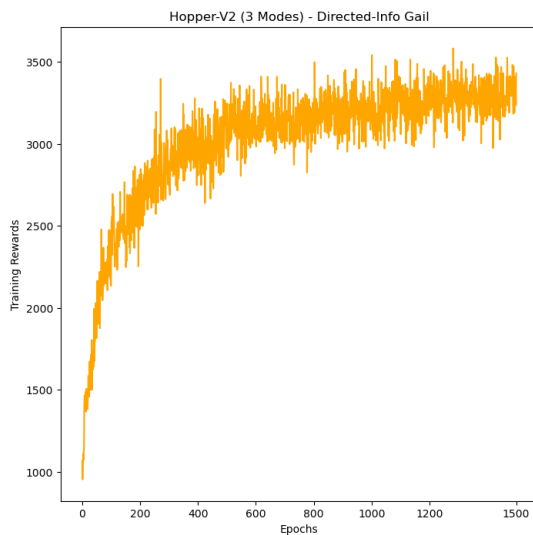


Figure 7. Directed-Info GAIL training rewards (3 Modes)

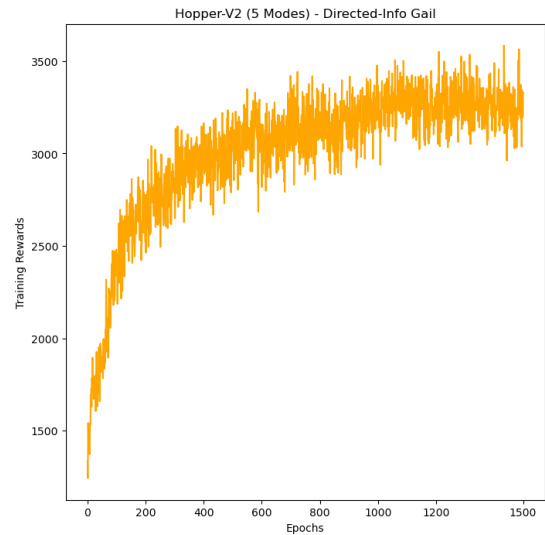


Figure 8. Directed-Info GAIL training rewards(5 Modes)

Second, in the Aviation domain:

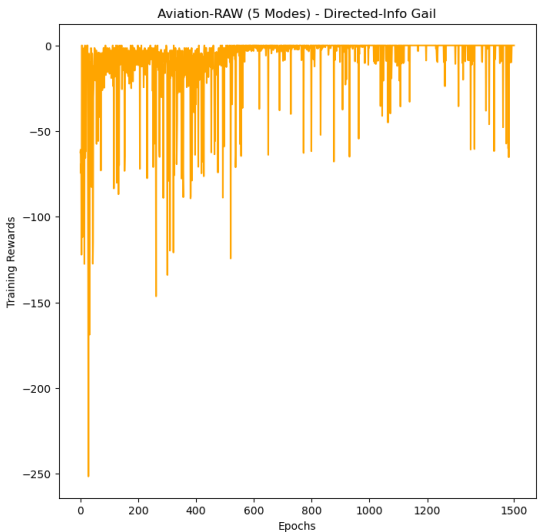
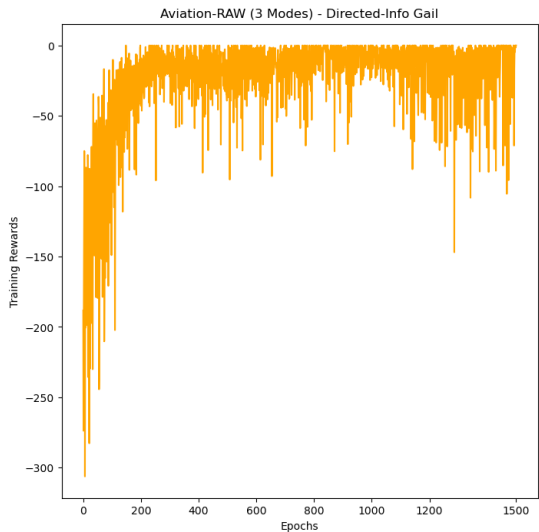


Figure 9. Directed-Info GAIL training rewards (3 Modes) Figure 10. Directed-Info GAIL training rewards(5 Modes)

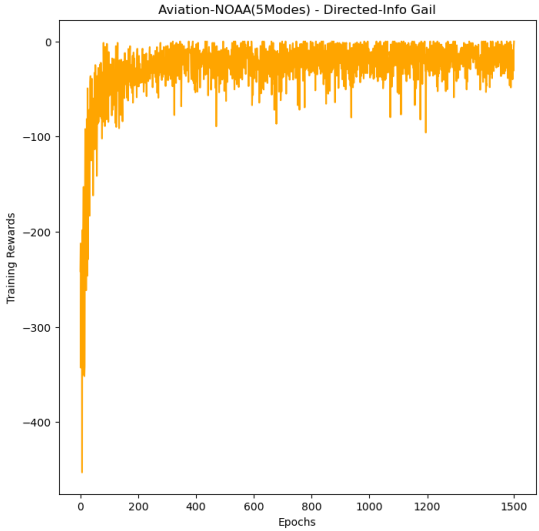
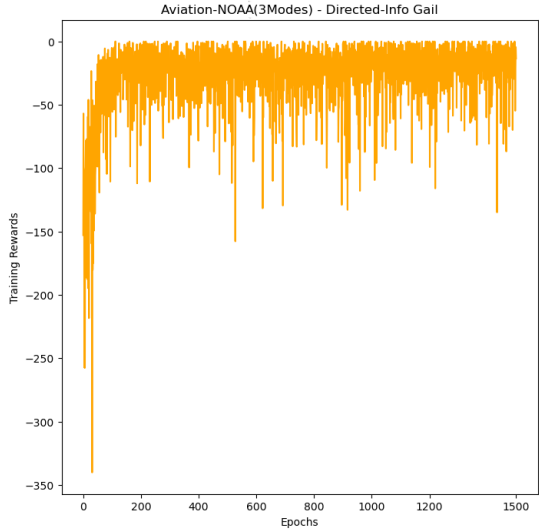


Figure 11. Directed-Info GAIL training rewards (3 Modes) Figure 12. Directed-Info GAIL training rewards(5 Modes)

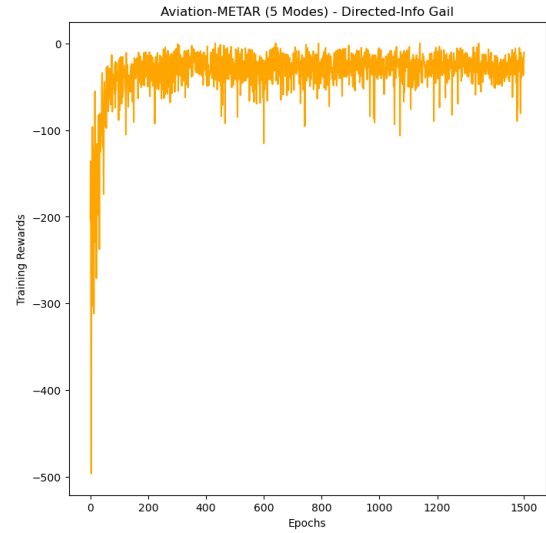
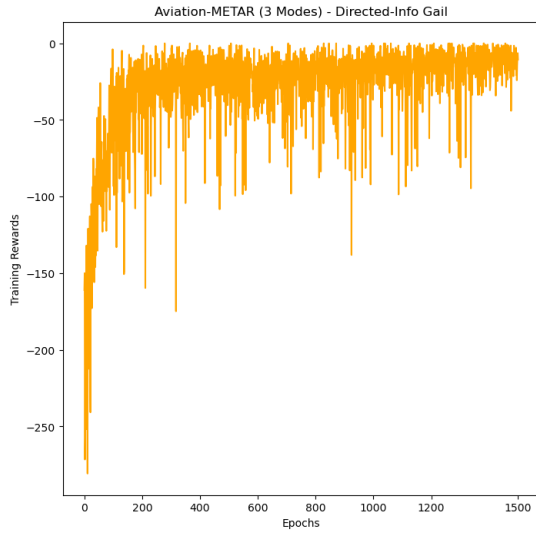


Figure 13. Directed-Info GAIL training rewards (3 Modes) Figure 14. Directed-Info GAIL training rewards(5 Modes)

We can figure out from these figures that in the first epochs the reward in both environments is low but as the epochs advance till 1500, the reward improves. This is because, before we train the agent with GAIL we pre-trained the model using the VAE: We needed to apply some bias in GAIL agent's actions, as it is essential to have some randomness at earlier epochs. That is the reason of the low reward at start in Directed-Info GAIL.

6.2.2 Testing

Tables 1, 2 show the evaluation results while testing the agent in both environments with different state features (11 observations in the Hopper and 4D point [long, lat, alt, timestamp], plus the contextual weather features depending on the experiment in the aviation) and 3 or 5 modes.

- ❖ In the Hopper environment we tested 2 different environment versions and the evaluation reward ranges of [0-3800].
- ❖ In the Aviation environment the evaluation reward ranges of [-400 – 0].

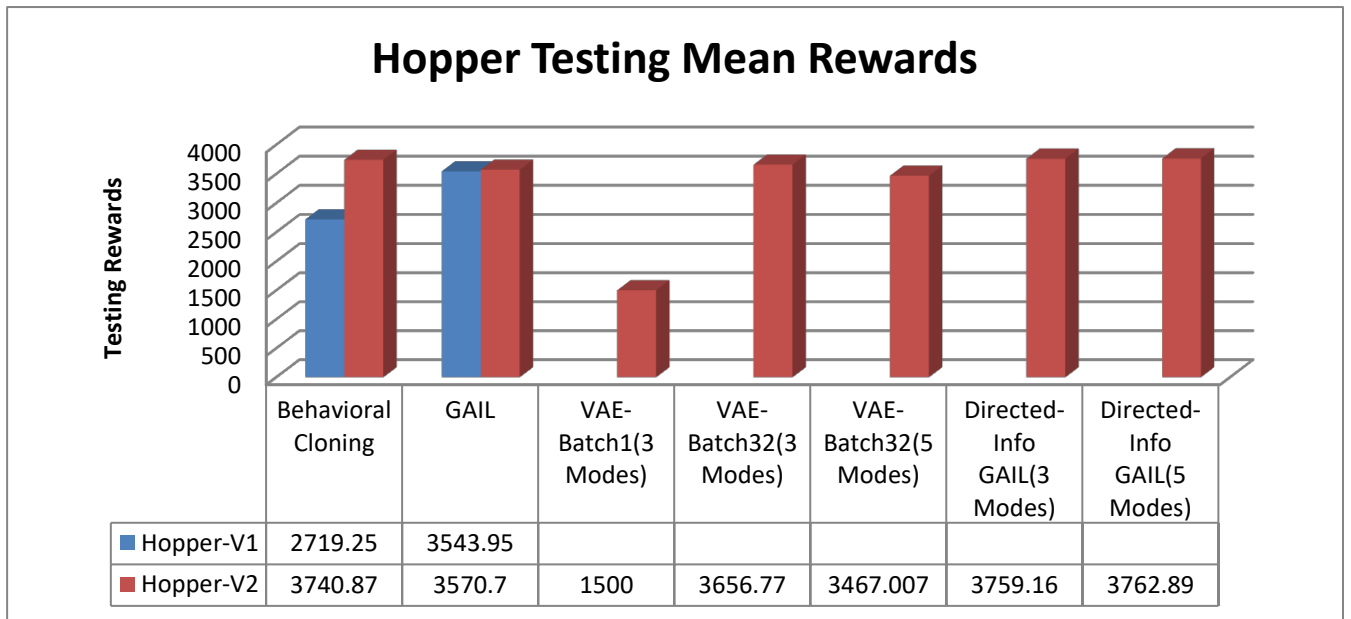


Table 1. Evaluation Rewards for Bcloning/ Gail, VAE/ Directed-Info GAIL, in the Hopper environment

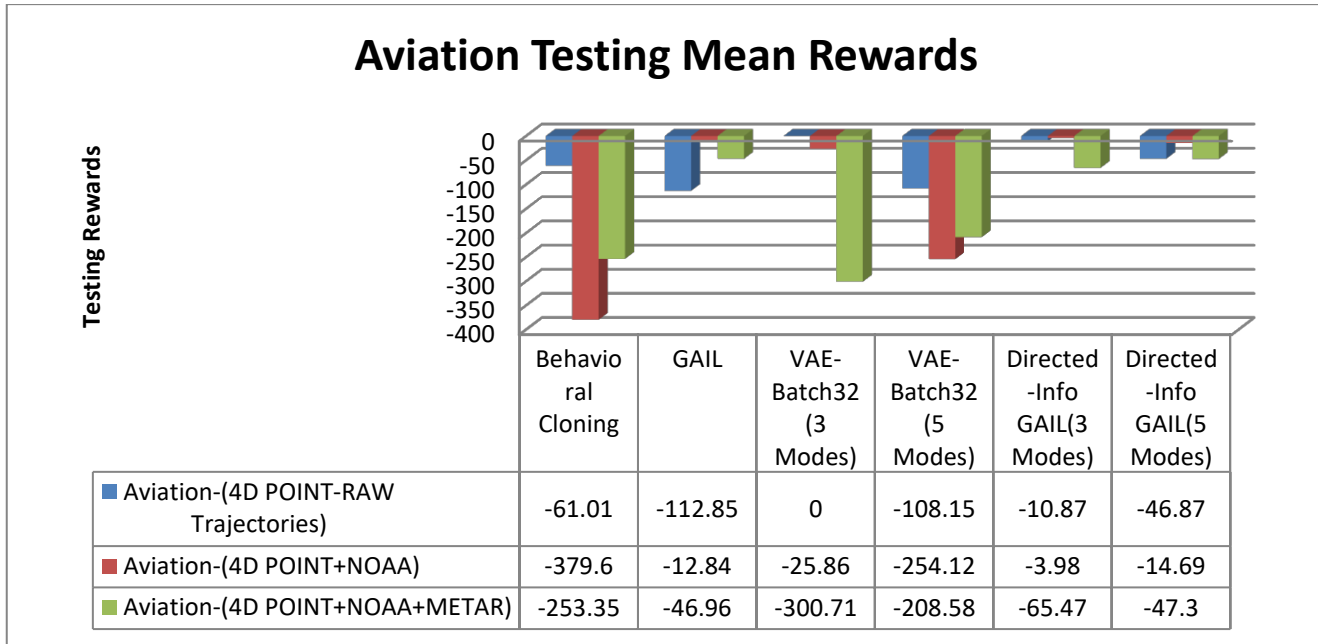


Table 2. Evaluation Rewards for Behavioral Cloning/ GAIL, VAE/ Directed-Info GAIL, in the Aviation environment

Regarding the results in table 1, it must be noted that when we used VAE with batch size 1 for pre-training the hopper, which regularly is used in online learning, the results were poor (eg. 1500 reward). So we use batch size of 32 for pre-training the VAE in the rest of our experiments. The results were really better, the agent when was trained using VAE with batch size of 32 scored more than 2000 extra reward points than with batch size of 1. In the Hopper environment version 1(Hopper-V1), we evaluated the agent using behavioral cloning (for pre-training) and GAIL. Results were a little better on version 2 (Hopper-V2) which we decided to use in the rest of our experiments. These versions correspond to the changes that the Hopper's makers did in the environment. We can understand from table 1 that VAE did a better job with 3 modes than in the scenario with VAE with 5 modes, and Directed-Info GAIL reached almost the same mean reward after 50 episodes in both mode settings. It seems that Directed-Info GAIL managed to score the highest cumulative mean reward in Hopper.

It must be noted, in all of our experiments with Behavioral Cloning for pre-training we used batch size of 64.

Second, in the Aviation environment, we can understand from table 2 that the experiments made with Variational Auto-Encoder and Directed-Info GAIL are improving the policy of our agent and let it gain the highest cumulative reward than previous algorithms such as Behavioral Cloning and GAIL. We can understand as a matter of the policy that with 3 modes in cases with raw trajectories and enriched trajectories with NOAA, the method scored better rewards than in the 5 modes. The approach with enriched trajectories with NOAA and METAR for meteorological features did good but not the best. However, these state features are the more dynamic and necessary in this setting. Thus, for the setting with METAR features we can say that the aircraft scored better rewards in 5 modes than 3.

The hopper has imitated the expert perfectly when the reward is closer to 3800 while the aircraft when the reward is close to 0.

In the next sub-sections we will discuss more about how the modes/latent variables of hopper and aircraft indicate interesting patterns of behavior depending on the setting.

6.3 Hopper Trajectories

6.3.1 Modes

In the next images, we have captured some of the different hopper states in rendering while evaluating the agent (hopper). The Latent Variables in Figure 15 are three (0, 1, 2) and they are getting started enumerated by 0. After five experiments, we can presume that mode 2 most commonly appears in states that the robot leg is reaching the peak, 0 when is ascending and 1 when is landing. In the other experiment we made, which we used five latent variables (0, 1, 2, 3, 4), the path is getting split differently, as we can see it in Figure 16. In this case, after five experiments we can conclude that mode 0 is reaching the peak, 4 is for ascending and 1 is for landing, but there are two extra modes that correspond to when hopper is in mid-air with value of 2 and mode 3 is just before landing.

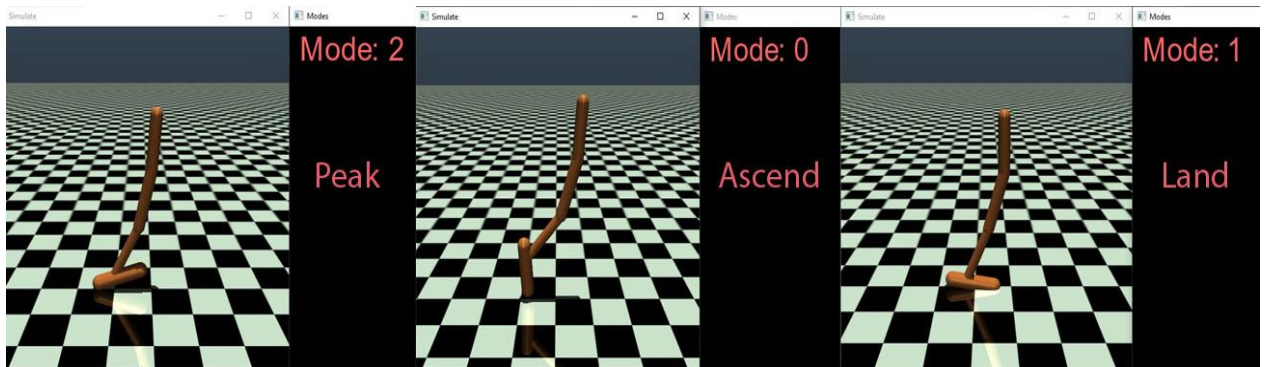


Figure 15. Evaluating Modes in Hopper Environment with 3 Modes



Figure 16. Evaluating Modes in Hopper Environment with 5 Modes

6.3.2 Latent variables and training loss

In the next figures we are exploring, how the encoder of VAE divides the sub-tasks in Hopper's trajectories. We present some epochs while the network is trained, and second, while testing the agent. For simplicity of the presentation,

only the first trajectory of the dataset will be demonstrated. The y axis indicates the Latent Variables which in the first experiment are integers 0, 1 and 2, and in the second experiment the latents are 0, 1, 2, 3 and 4. The x axis indicates Hopper's time-steps with a maximum limit of 1000.

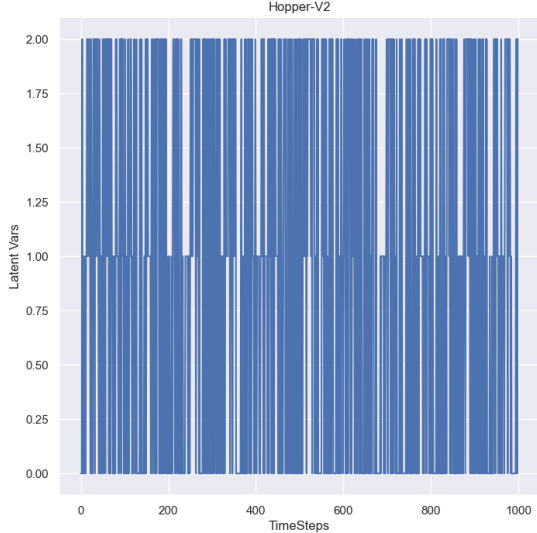


Figure 17. VAE, Latent Variables in first epoch

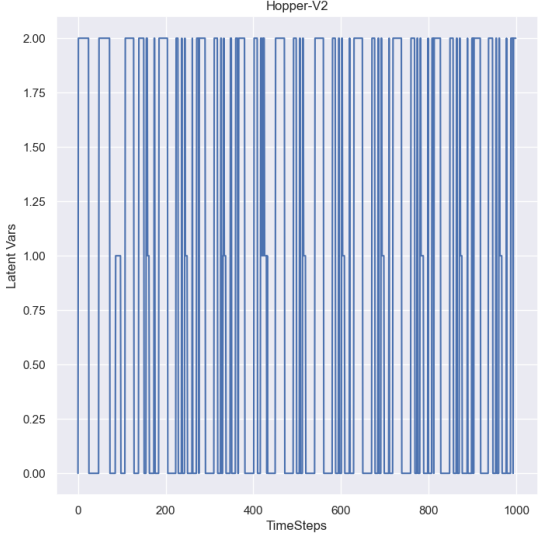


Figure 18. VAE, Latent Variables in 500th epoch

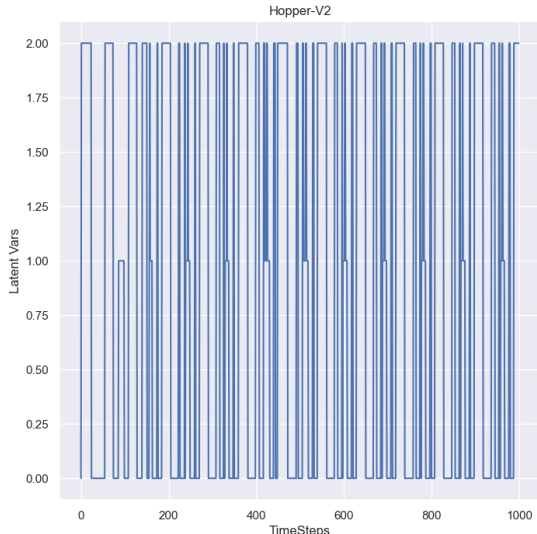


Figure 19. VAE, Latent Variables in 1500th epoch

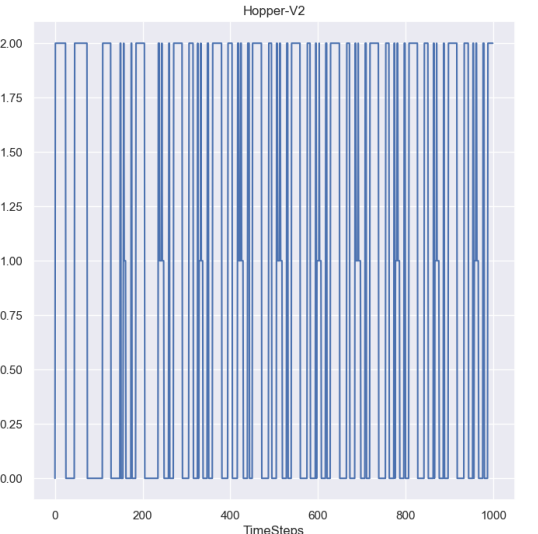


Figure 20. VAE, Latent Variables in 2000th epoch

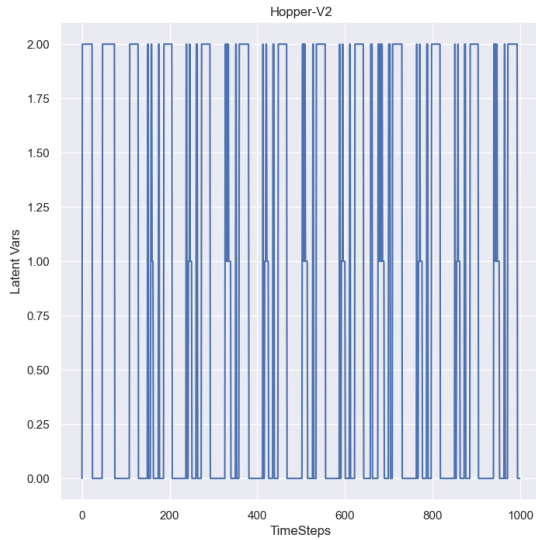


Figure 21. VAE, Testing 3 Latent Variables

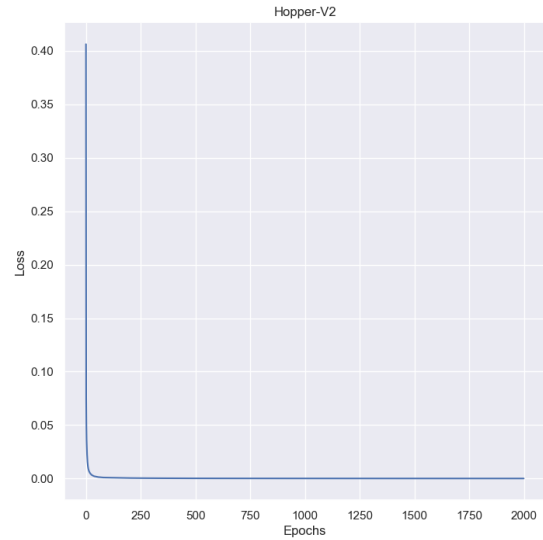


Figure 22. Training Loss of the VAE(3modes)

We can clearly see in figures 17-20 that while the encoder is trained with the use of expert trajectories, the first mode discovered is 1, which corresponds to landing. The same as the above can be seen in testing in Figure 21. When hopper environment is reset, the robot is already touching the terrain so after the first jump almost in 180th time-step, hopper is landing. Reaching the peak corresponds to mode 2 and ascending to 0. Sub-section 6.3.1 further refers to detecting these modes, as also presented in figures 15-16. We can see in Figure 22, the training loss of the Variational Auto-Encoder (VAE)'s pre-training. The Mean-Squared Error loss function is being used to train the network. The y axis indicates the training mean loss across all the dataset (50000 state-action pairs). And the x-axis indicates the training epochs which are 2000. It's important here to say that given that the hopper environment is a simulated setting, the expert's observations from the dataset are really precise, the loss is a float in 0.4-0.0001, and the loss is never equal to 0.

In the second experiment, with the use of 5 modes, figures 23-27 show the modes along the trajectory.

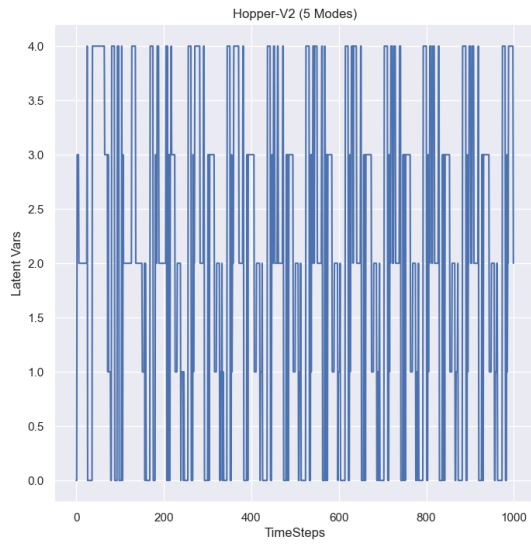
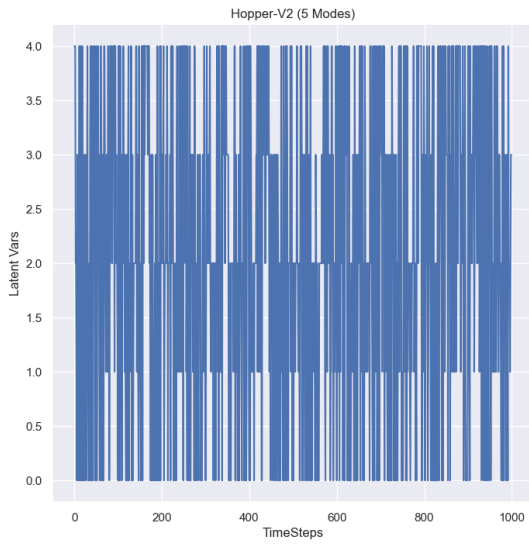


Figure 23. VAE, Latent Variables in the first epoch Figure 24. VAE, Latent Variables in the 500th epoch

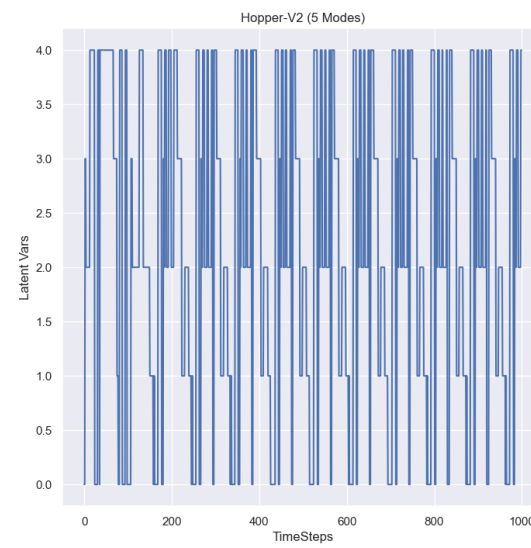
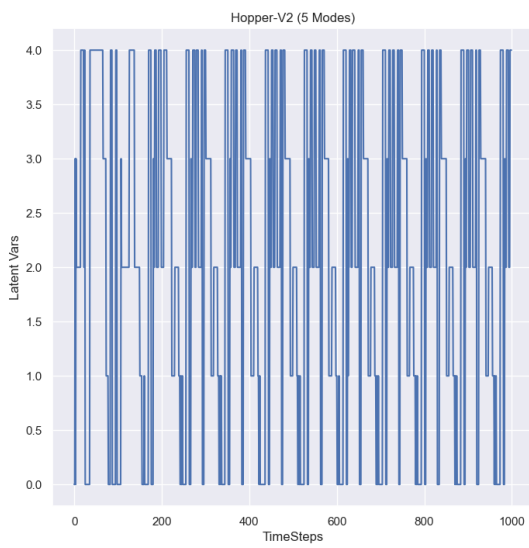


Figure 25. VAE, Latent Variables in the 1500th epoch Figure 26. VAE, Latent Variables in the 2000th epoch

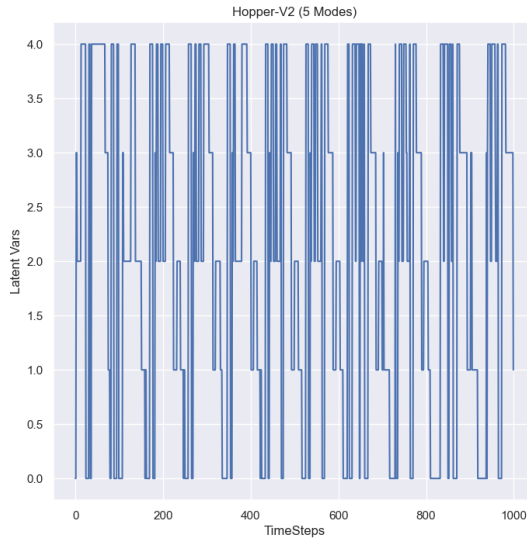


Figure 27. Testing 5 Latent Variables

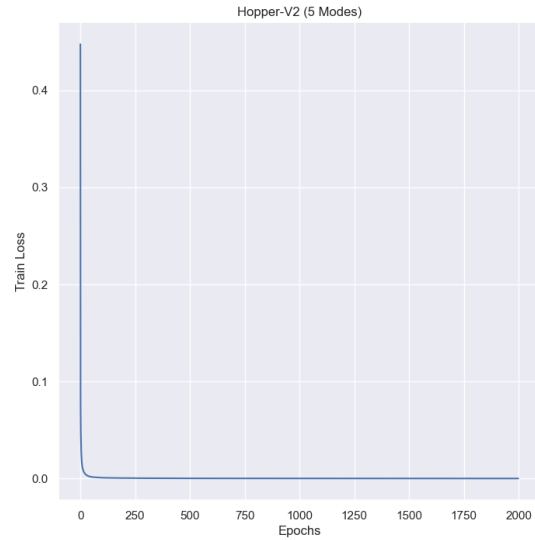


Figure 28. Training Loss of the VAE(5modes)

So in this case, mode 1 indicates the landing phase, as in using the 3 modes showed in the previous experiment. In most cases the neural network has done well but there are always flaws in the modes between 1000 time-steps. The latent variable 0 corresponds to reaching the peak, while 4 is ascending, 2 appears more frequently when the robot leg is in mid-air. Mode 3 is before landing as we can see in Figure 27 and mode 1 appears just after. This pattern continues appearing in the next time-steps, in a periodic manner. The training loss is a float number very close to 0 but never 0.

6.4 Aviation Trajectories

6.4.1 Modes and predictions

In figures 29-34, the 50 red trajectories in each case are generated using behavioral cloning while the 50 blue trajectories are generated by GAIL using behavioral cloning for pre-training.

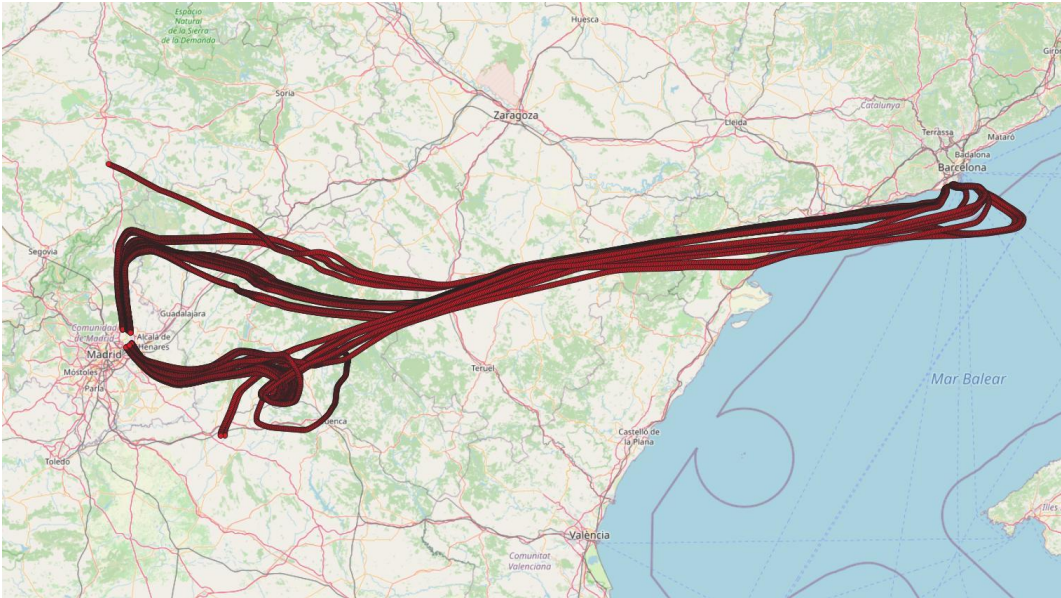


Figure 29. Barcelona to Madrid, Behavioral Cloning Results with raw trajectories.

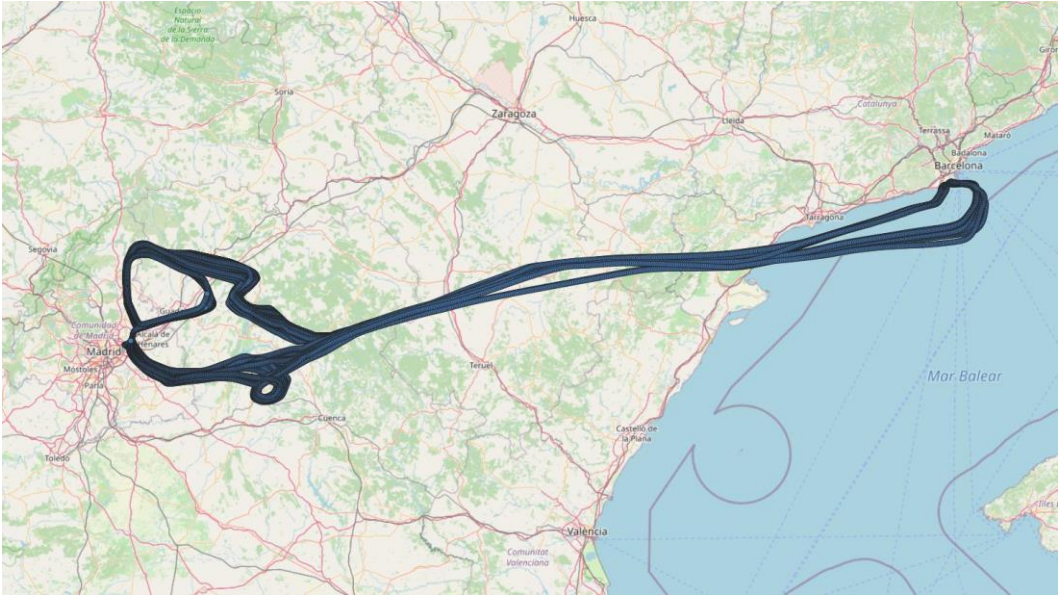


Figure 30. Barcelona to Madrid GAIL Results with behavioral cloning for pre-training and raw expert trajectories.

In figures 29-30, we can see the trajectories generated by behavioral cloning and GAIL in a visual way: The policy learned by GAIL imitates expert trajectories successfully, and the aircraft reaches the destination airport, scoring a high reward.



Figure 31. Barcelona to Madrid, Behavioral Cloning Results with expert trajectories enriched with NOAA.



Figure 32. Barcelona to Madrid Gail Results with behavioral cloning for pre-training and expert trajectories enriched with NOAA.

Figures 31-32 show the results of behavioral cloning and GAIL for trajectories enriched with meteorological features from NOAA. As noted above, the policy learned by GAIL imitates expert trajectories successfully, and the aircraft

reaches the destination airport, scoring a high reward. In Figure 31, it is clear that behavioral cloning has difficulties in imitating experts' trajectories because of compounding error caused by covariate shift.

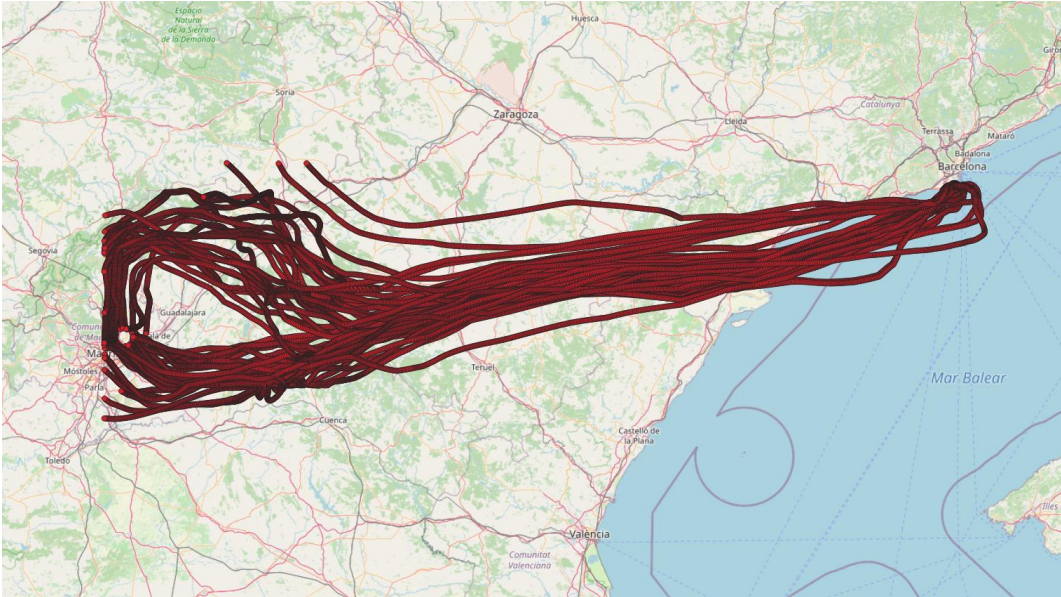


Figure 33. Barcelona to Madrid, Behavioral Cloning Results with expert trajectories enriched with NOAA and METAR.



Figure 34. Barcelona to Madrid Gail Results with behavioral cloning for pre-training with expert trajectories enriched with NOAA and METAR.

Figures 33, 34 show results from trajectories enriched with NOAA and METAR, resembling better real flight scenarios considering the weather along the flight and the weather in the destination airport. Trajectories generated from the behavioral cloning policy shown in Figure 33, do not reach the destination, in contrast to those generated by GAIL and shown with Figure 34.

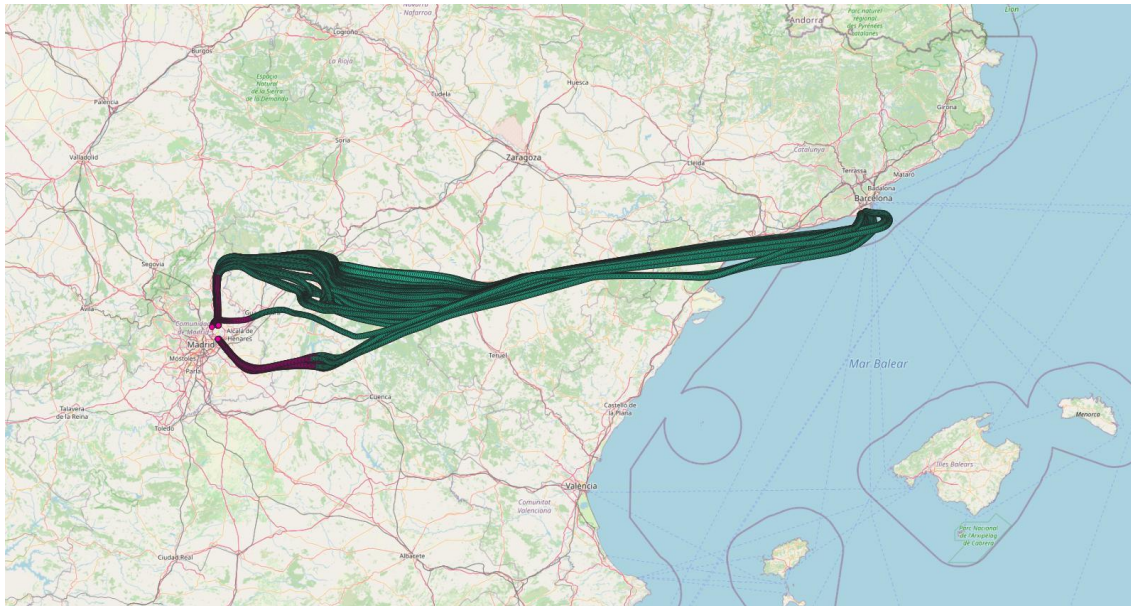


Figure 35. Barcelona to Madrid Variational Auto-Encoder Results with 3 Modes (features: 3D Point with timestamp without meteorological features).

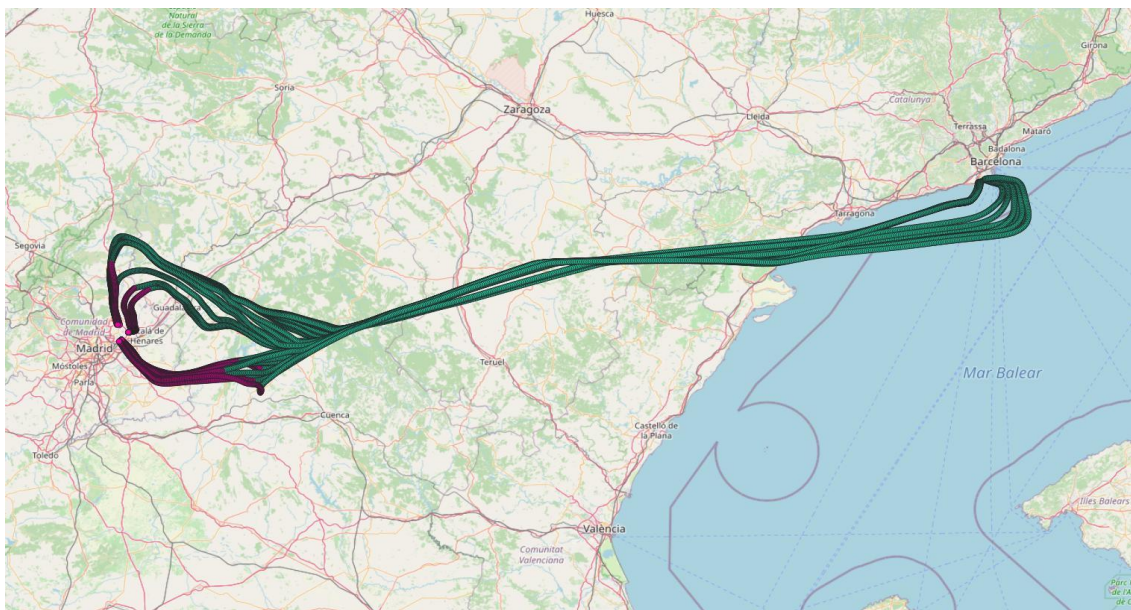


Figure 36. Barcelona to Madrid Directed-Info GAIL Results with Variational Auto-Encoder for pre-training with 3 Modes (features: 3D Point with timestamp without meteorological features).

In figures 35, 36 we can see the results with 3 modes in raw trajectories using Variational Auto-Encoder and Directed-Info GAIL. We can see that the agent using the VAE was able to identify 2 of the 3 modes. Green color (mode: 2) is the take-off along with the flight and dark pink color (mode: 0) indicates the landing phase in the destination airport.



Figure 37. Barcelona to Madrid, Variational Auto-Encoder Results with 5 Modes (features: 3D Point with timestamp without meteorological features).

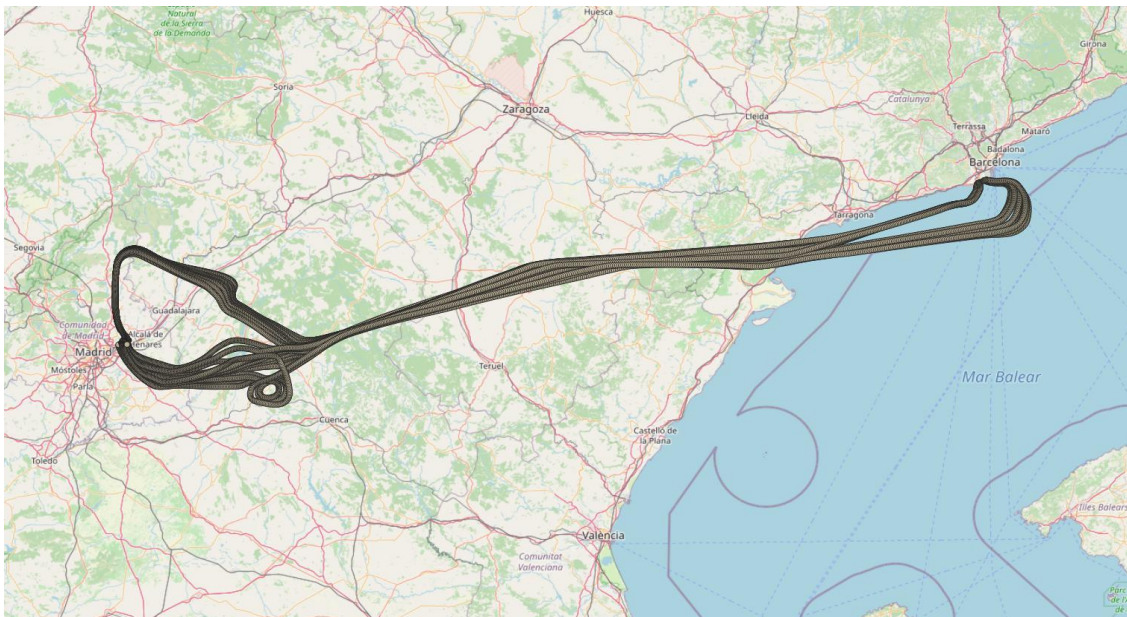


Figure 38. Barcelona to Madrid, Directed-Info Gail Results with Variational Auto-Encoder for pre-training with 5 Modes (features: 3D Point with timestamp without meteorological features).

In figures 37, 38 we can see the results with 5 modes in the raw trajectories. We can see that the agent using the VAE was able to identify only 1 of the 5 modes. Most probably this happened because the raw state features are not enough to indicate splitting the trajectory in 5 modes. So the brown color (mode: 1) is the whole mode identified by VAE and we can see in Figure 38 that GAIL algorithm slightly improved the policy.

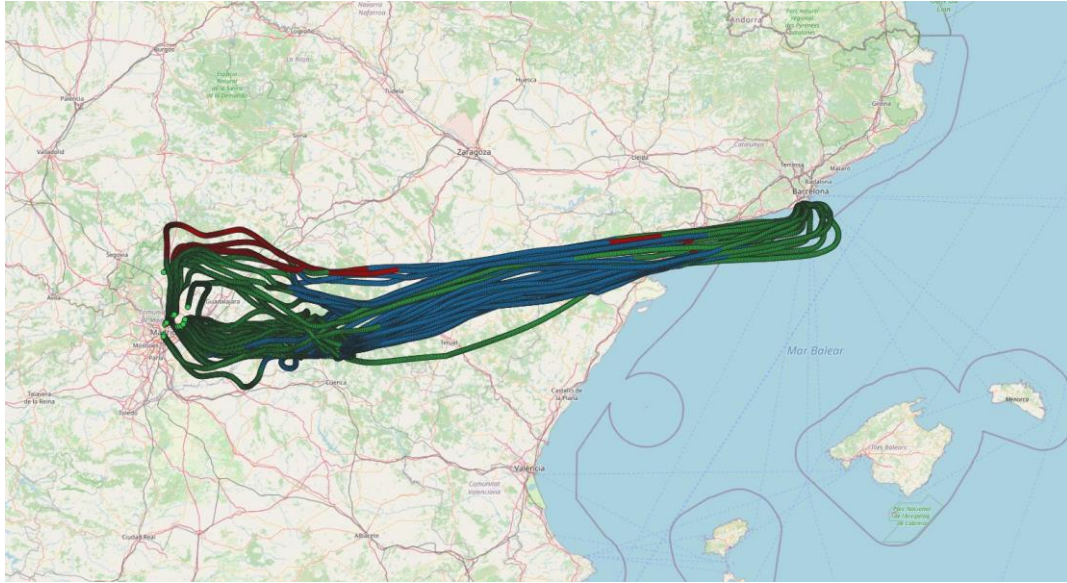


Figure 39. Barcelona to Madrid Variational Auto-Encoder Results with 3 Modes (features: 3D Point with timestamp and NOAA meteorological features).

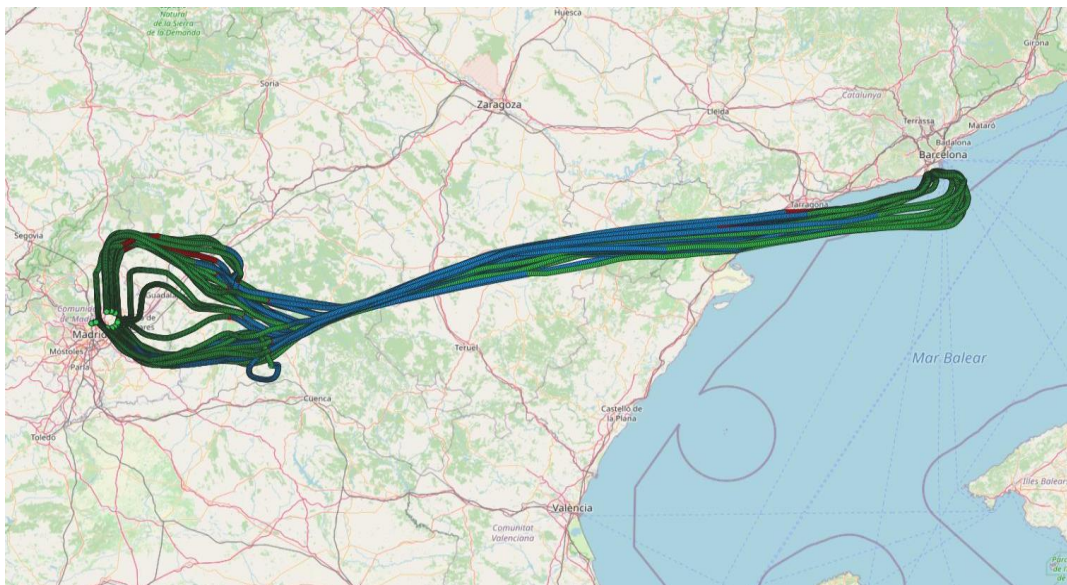


Figure 40. Barcelona to Madrid Directed-Info Gail Results with Variational Auto-Encoder for pre-training with 3 Modes (features: 3D Point with timestamp and NOAA meteorological features).

In figures 39, 40 we can see the results with 3 modes in the corresponding enriched trajectories with NOAA. We can see that the agent using the VAE was able to identify all 3 modes. Green color (mode: 1) indicates the take-off and the landing, where the aircraft gains or loses altitude, the blue color (mode: 2) shows the whole flight where the aircraft has high altitude. The red color (mode: 0) appears in some trajectories, indicating changes in longitude, latitude and altitude and significant changes in the NOAA contextual features which

were: relative humidity and temperature, u-component of wind and v-component of wind. Directed-Info GAIL in Figure 40, appears to have improved the policy, so the aircraft in each trajectory reaches the destination airport.



Figure 41. Barcelona to Madrid, Variational Auto-Encoder Results with 5 Modes (features: 3D Point with timestamp and NOAA meteorological features).



Figure 42. Barcelona to Madrid, Directed-Info Gail Results with Variational Auto-Encoder for pre-training with 5 Modes (features: 3D Point with timestamp and NOAA meteorological features).

In figures 41, 42 we can see the results with 5 modes from both algorithms with enriched trajectories with NOAA. The purple color (mode: 0) indicates big increase or decrease in altitude when the aircrafts takes-off and landing. Green

(mode: 3) and orange color (mode: 4) show changes in altitude depending on the meteorological NOAA features, while the orange mode in some trajectories in Figure 41 indicate turns. Red color (mode: 2) appears to represent sudden changes in all NOAA features. Blue color (mode: 1) in the trajectories appears before landing.

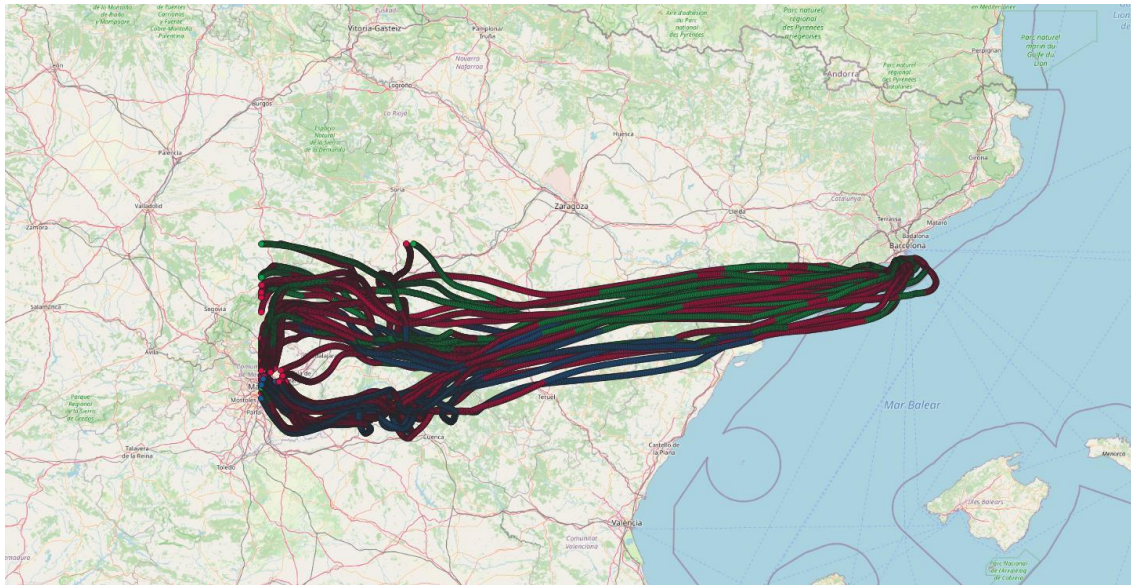


Figure 43. Barcelona to Madrid Variational Auto-Encoder Results with 3 Modes (features: 3D Point with timestamp and NOAA&METAR meteorological features).

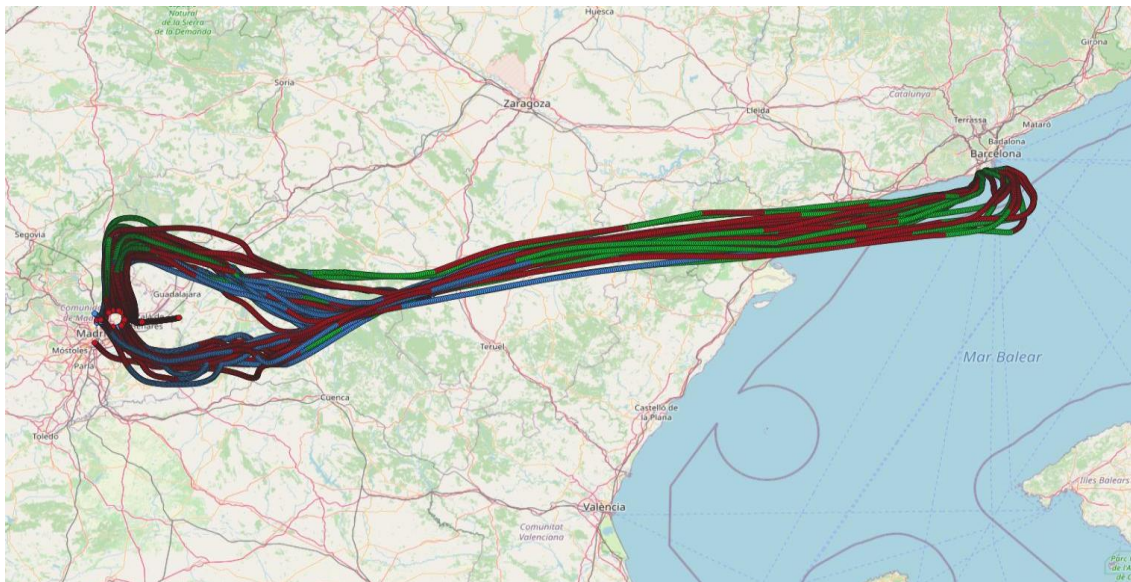


Figure 44. Barcelona to Madrid Directed-Info Gail Results with Variational Auto-Encoder for pre-training with 3 Modes (features: 3D Point with timestamp and NOAA&METAR meteorological features).

In figures 43, 44 we can see the results with 3 modes from both algorithms with enriched trajectories with NOAA and METAR. In this case, the algorithm doesn't appear to divide the trajectory like the previous experiments. The modes seem to be switching too often, so we can hypothesize that the algorithm has identified different patterns of behavior than the other settings. The green color (mode: 2) indicates big increase or decrease in altitude when the aircraft takes-off or landing. Red color (mode: 1) indicate changes in altitude depending also on the meteorological NOAA features. Blue color (mode: 0) appears at the end of the flight.



Figure 45. Barcelona to Madrid, Variational Auto-Encoder Results with 5 Modes (features: 3D Point with timestamp and NOAA&METAR meteorological features).

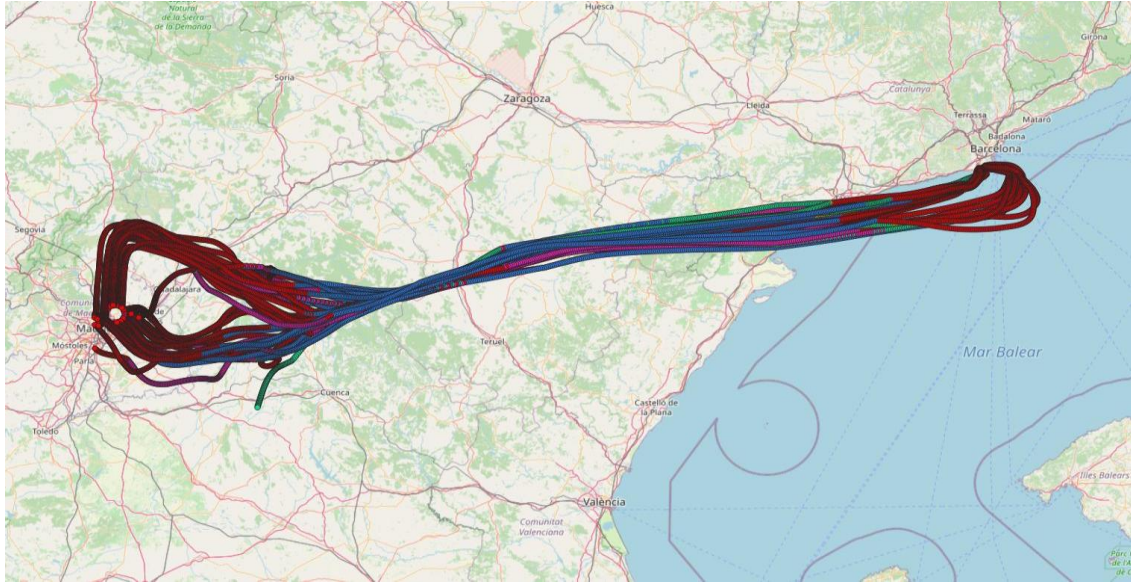


Figure 46. Barcelona to Madrid, Directed-Info Gail Results with Variational Auto-Encoder for pre-training with 5 Modes (features: 3D Point with timestamp and NOAA&METAR meteorological features).

In figures 45, 46 we can see the results with 5 modes from both algorithms with enriched trajectories with NOAA and METAR. Red color (mode: 1) indicates the take-off and the landing of the aircraft. The blue color (mode: 0) indicates the en-route phase. Inspecting meteorological features, dark pink color (mode: 4) appears mostly when there is a change in wind speed gust surface. Green color (mode: 2) appears mostly when there was a shift in wind features. Yellow color (mode: 3) appears mostly in cases when relative humidity and temperature was changing.

6.4.2 Latent Variables and training loss

In the aviation environment, in the following we are exploring how the encoder of the VAE indicates the trajectories modes. First, in epochs while the models are trained, and then while testing. For simplicity of the presentation, only the first trajectory of the dataset is demonstrated. The y axis is for latent variables, which in the first experiment are integers 0, 1 and 2 and in the second experiment the latents are 0, 1, 2, 3 and 4. x axis indicates the aircraft' time-steps with a maximum limit of 1000. There is a difference between the time-steps of Hopper and those in the aviation domain. In aviation domain as noted before the maximum limit is 1000 but one trajectory isn't necessarily 1000 steps-long as in Hopper. For that reason in the figures that follow a part of the

second trajectory may appear. For simplicity of the presentation, as we describe the figures, we examining only the first trajectory (591 time-steps). A part of the second trajectory will also appear in the figures after the 591th time-step because in the aviation domain the whole flight isn't necessary 1000 time-steps. These trajectories are exploited in all different experiments with various meteorological features.

Experiment with 3 modes and Raw observations:

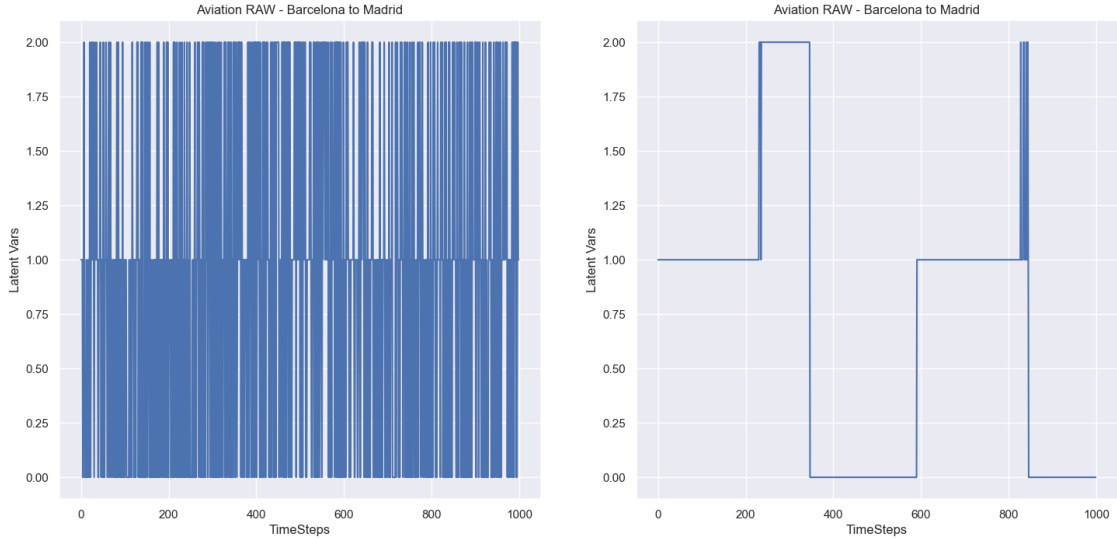


Figure 47. VAE, Latent Variables in the first epoch Figure 48. VAE, Latent Variables in the 500th epoch

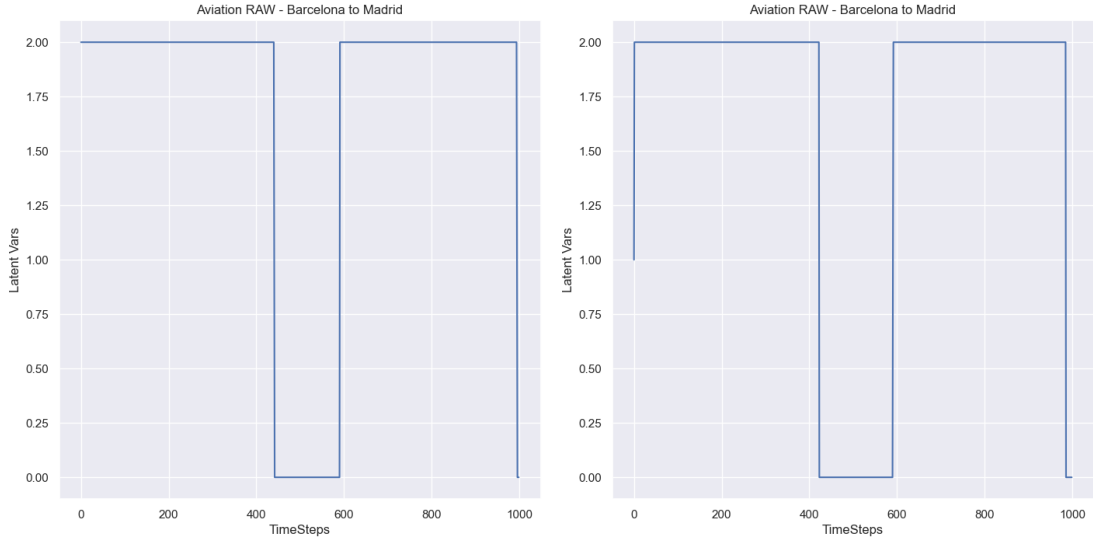


Figure 49. VAE, Latent Variables in the 1500th epoch Figure 50. VAE, Latent Variables in the 2000th epoch

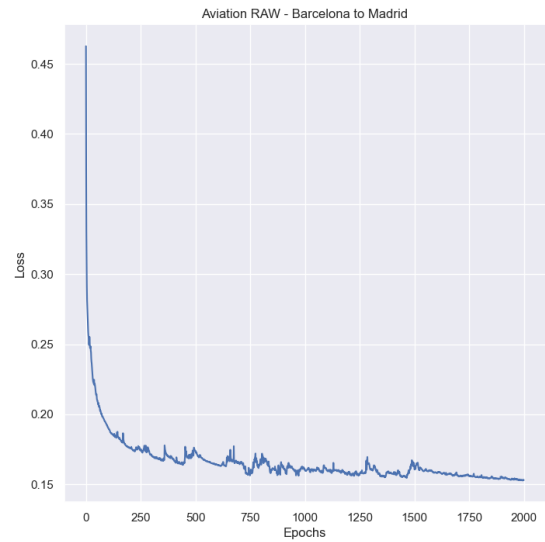
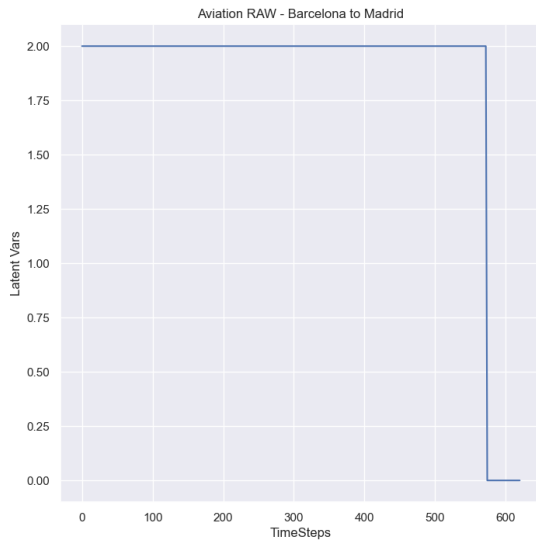


Figure 51. VAE, Testing 3 Latent Variables Figure 52. Training Loss of the VAE (3modes)

As can be seen in figures 47-50, while the VAE is being trained, at 2000th epoch mode 1 appears at the start of the trajectory but resulted in testing in Figure 51 to not be identified while the take-off and the journey were identified together (in the same mode: 2). Only the landing is appearing separately.

Experiment with 5 modes and Raw observations:

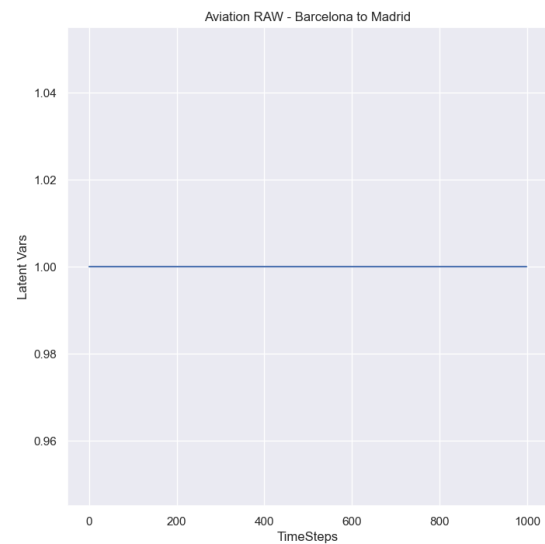
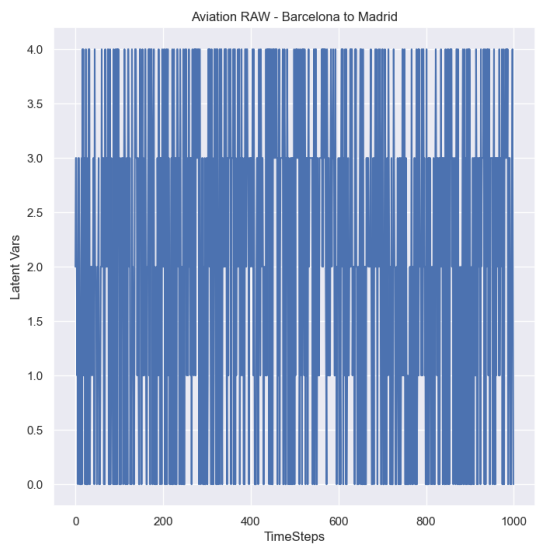


Figure 53. VAE, Latent Variables in the first epoch Figure 54. VAE, Latent Variables in the 500th epoch

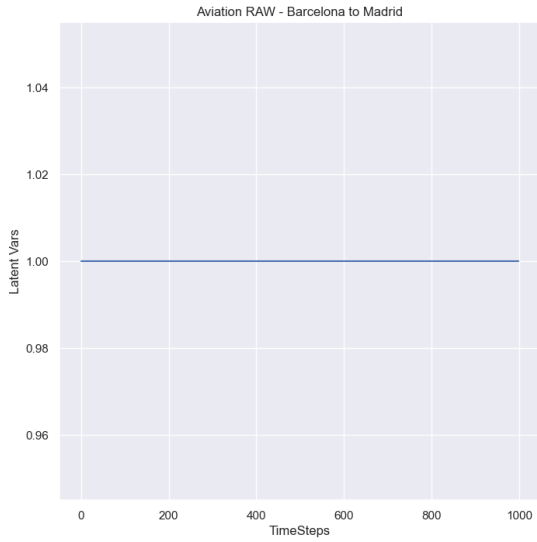


Figure 55. VAE, Latent Variables in the 1500th epoch

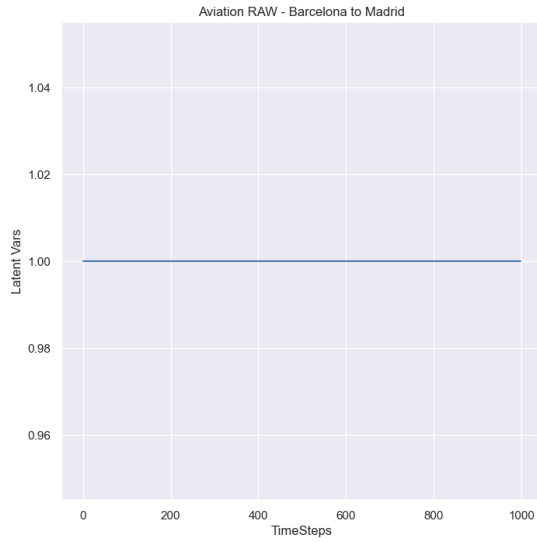


Figure 56. VAE, Latent Variables in the 2000th epoch

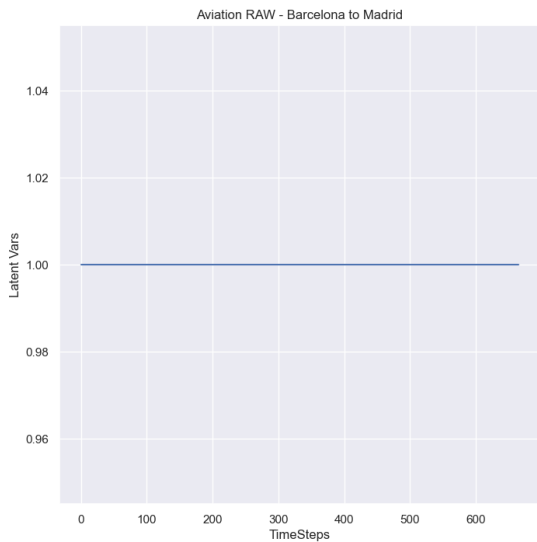


Figure 57. VAE, Testing 5 Latent Variables

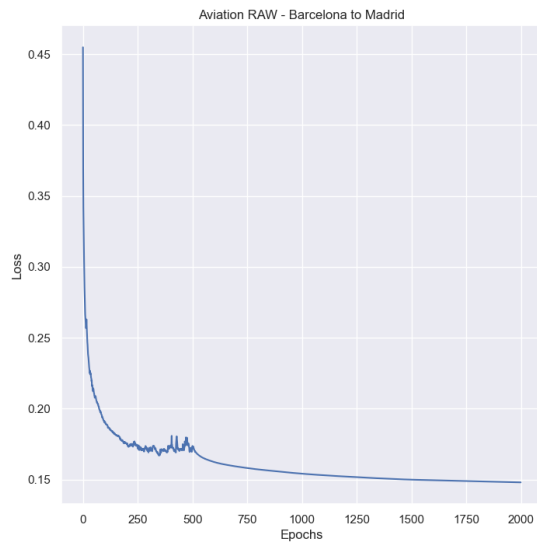


Figure 58. Training Loss of the VAE (5modes)

In this setting, the VAE didn't identify any patterns of aircraft's behavior and we can say this was the worst experiment. This can be seen in Figure 58, in the training loss because it isn't minimized as we would expect, as the epochs advance. We have a speculation this happened because of the random initialization of the Xavier weights in the pre-training of the VAE (in this particular experiment) or the raw trajectories individually cannot be split into five different patterns of behavior.

Experiment with 3 modes and Raw+NOAA observations:

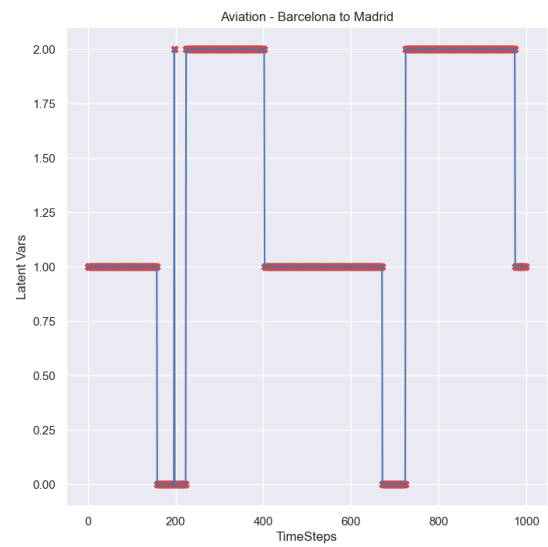
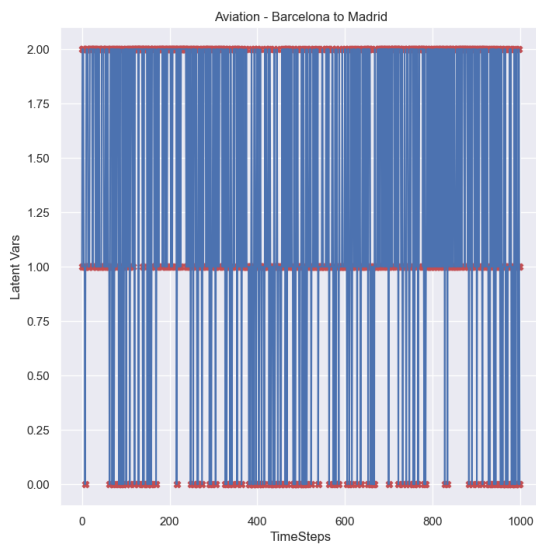


Figure 59. VAE, Latent Variables in the first epoch Figure 60. VAE, Latent Variables in the 500th epoch

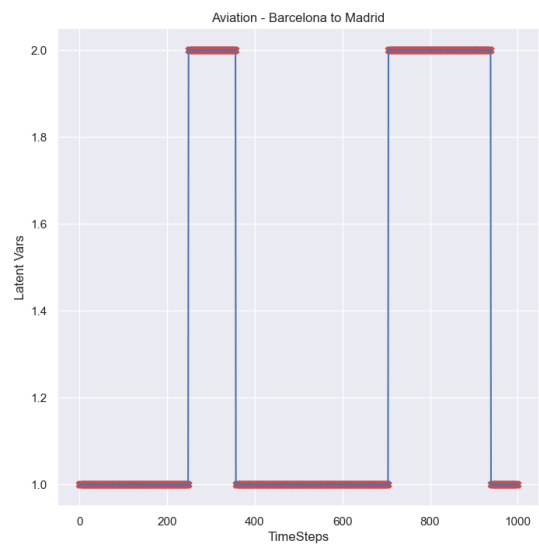
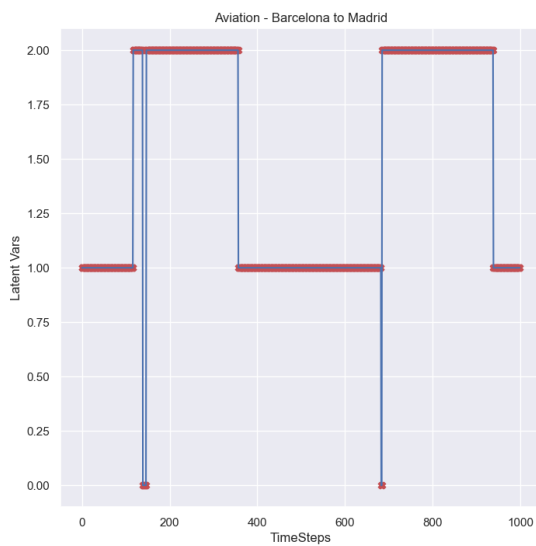


Figure 61. VAE, Latent Variables in the 1500th epoch Figure 62. VAE, Latent Variables in the 2000th epoch

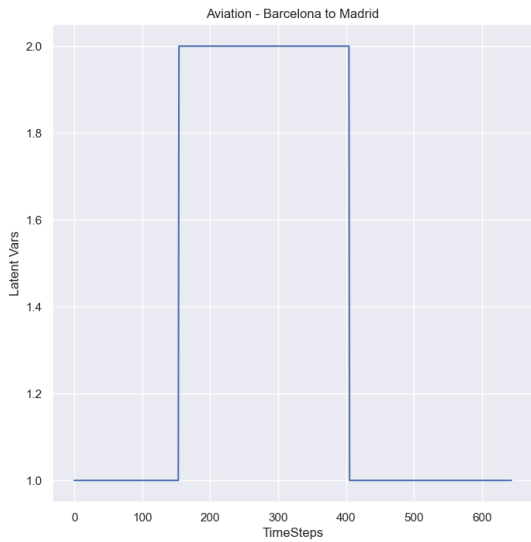


Figure 63. VAE, Testing 3 Latent Variables

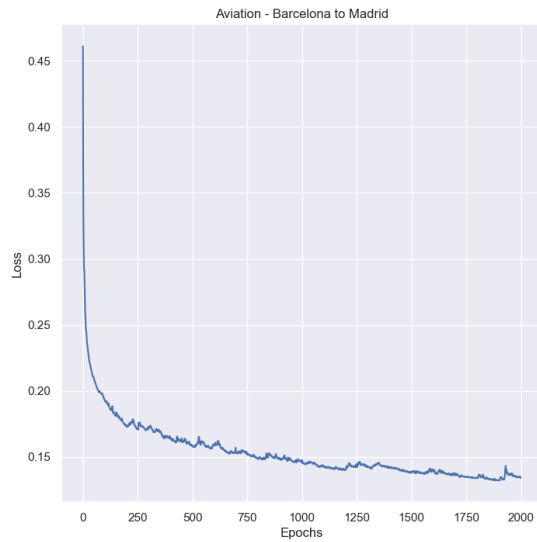


Figure 64. Training Loss of the VAE (3modes)

In this setting, mode 1 indicates the take-off and the landing phase, while mode 2 appears in the middle of the flight. NOAA meteorological features can only be seen in sections 6.4.1 and 6.4.3 while these figures aren't sufficient to show these patterns of behavior.

Experiment with 5 modes and Raw+NOAA observations:

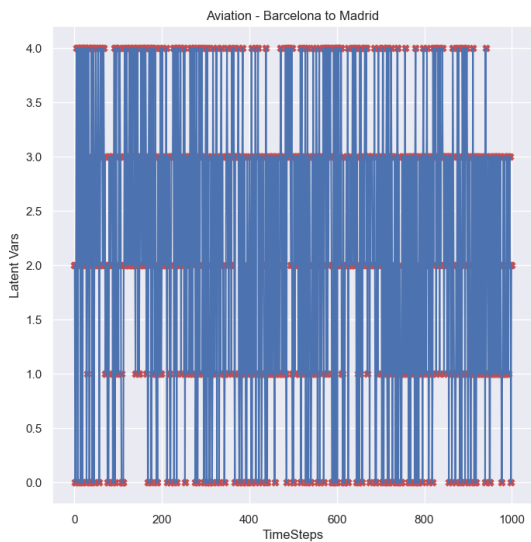


Figure 65. VAE, Latent Variables in the first epoch

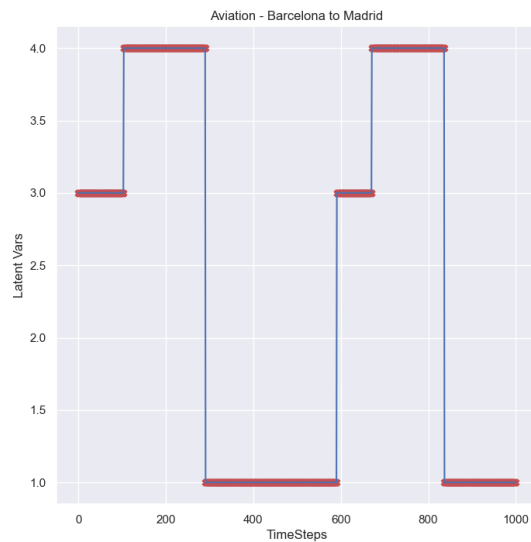


Figure 66. VAE, Latent Variables in the 500th epoch

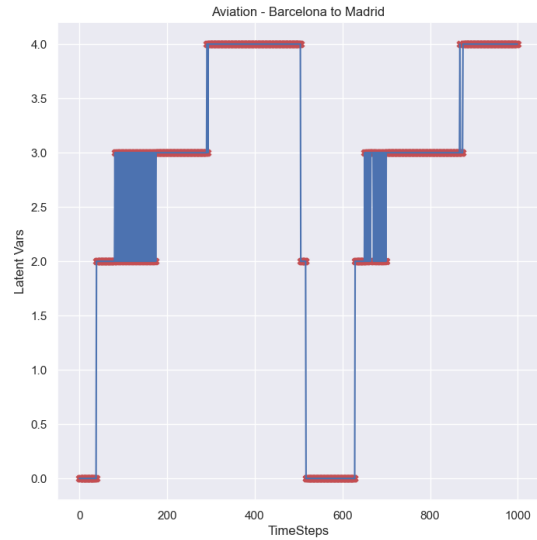
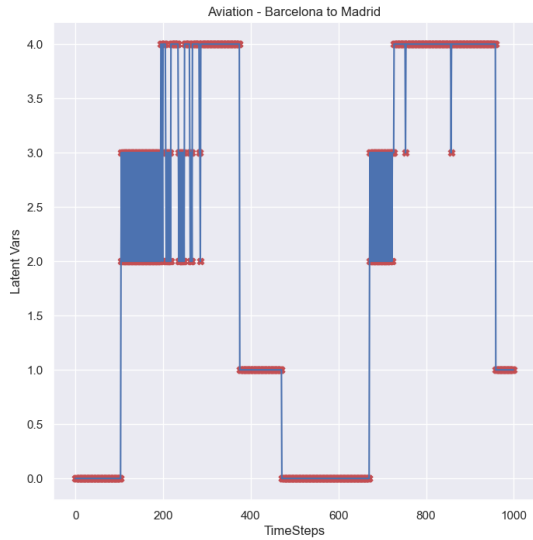


Figure 67. VAE, Latent Variables in the 1500th epoch

Figure 68. VAE, Latent Variables in the 2000th epoch

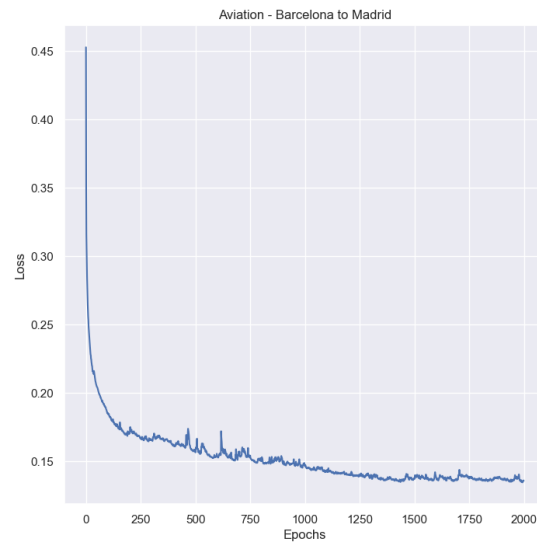
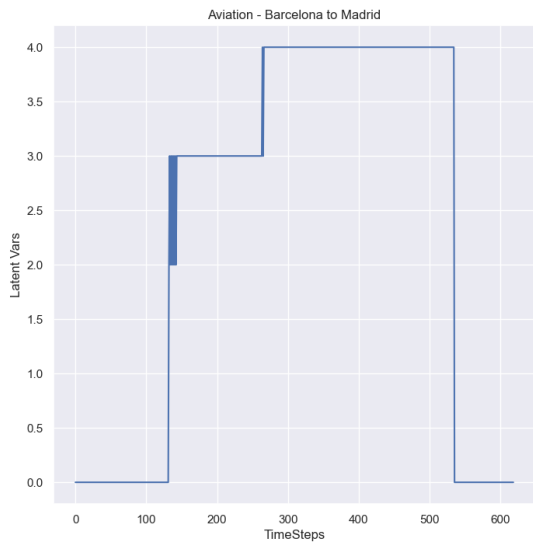


Figure 69. VAE, Testing 5 Latent Variables

Figure 70. Training Loss of the VAE (5 modes)

In this setting, with 5 modes and enriched trajectories with NOAA, the algorithm identified the following modes: 0 Mode appears in take-off and landing. We can't say for sure what modes 2-3-4 represent. They might be changes in the weather and we can't show it in these figures. Figure 70 shows the training loss in epochs.

Experiment with 3 modes and Raw+NOAA+Metar observations:

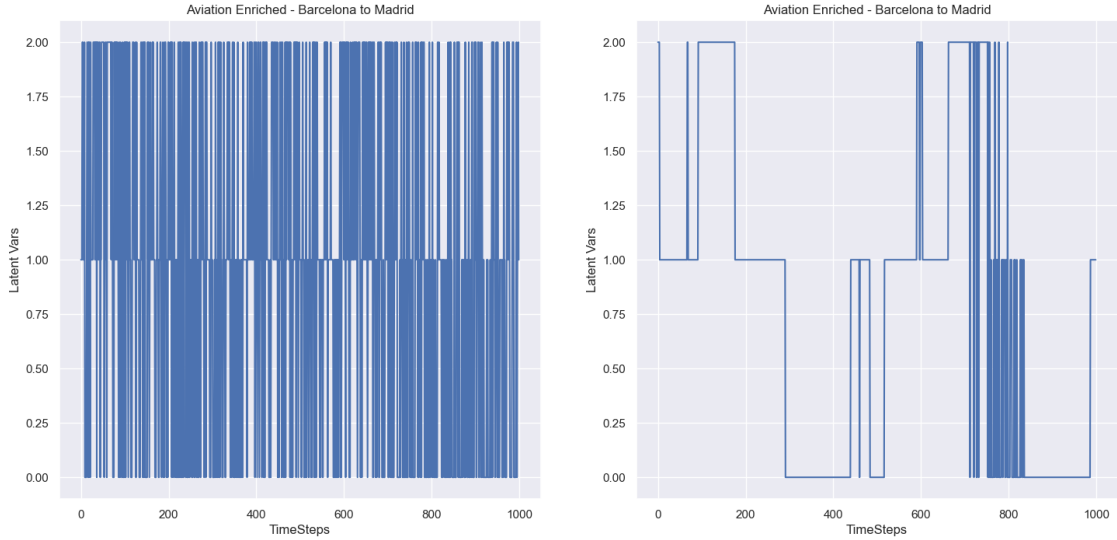


Figure 71. VAE, Latent Variables in the first epoch Figure 72. VAE, Latent Variables in the 500th epoch

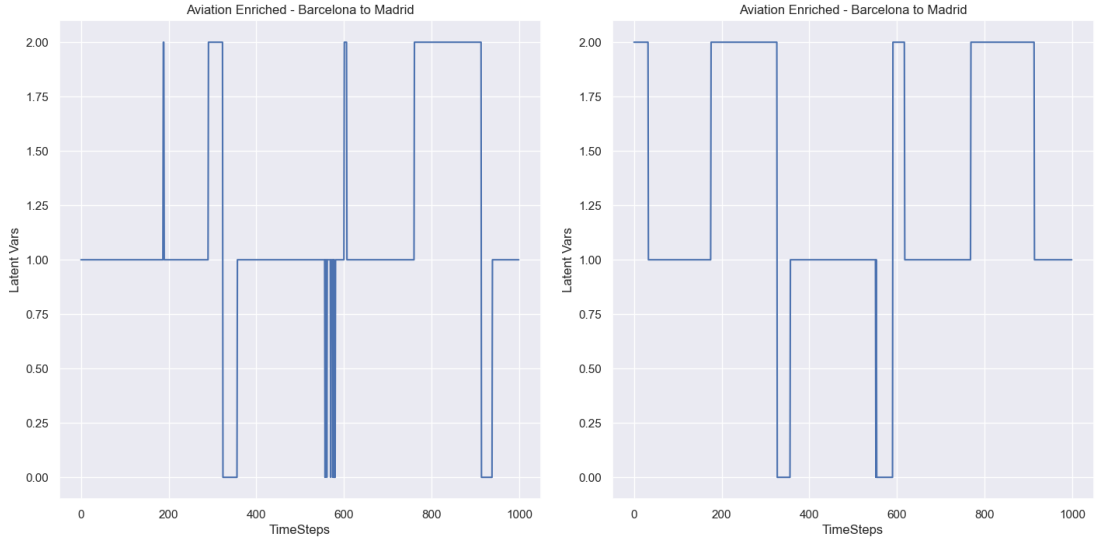


Figure 73. VAE, Latent Variables in the 1500th epoch Figure 74. VAE, Latent Variables in the 2000th epoch

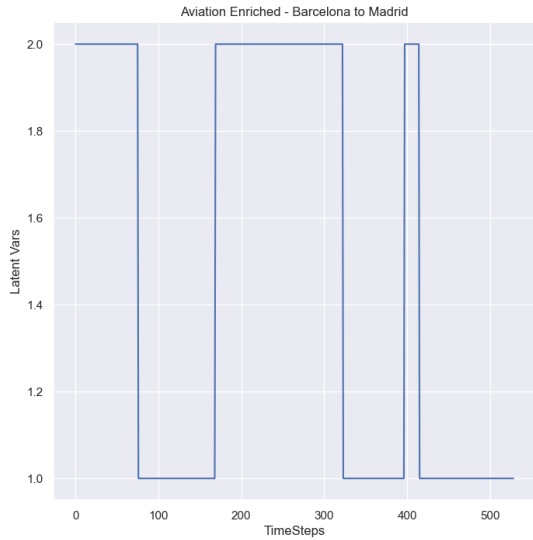


Figure 75. VAE, Testing 3 Latent Variables

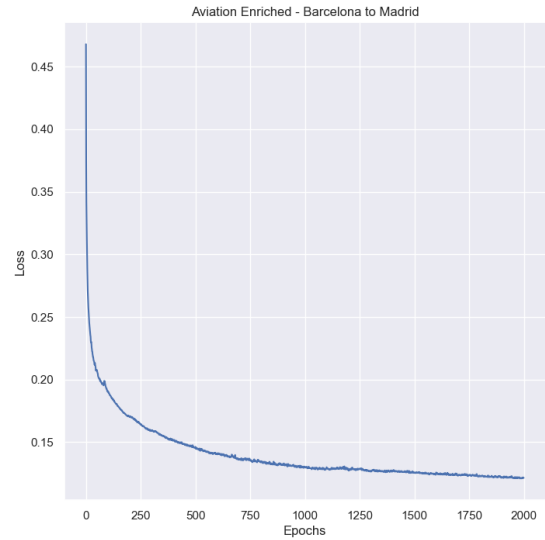


Figure 76. Training Loss of the VAE (3modes)

In this setting, the algorithm appears to have identified 2 modes in Figure 75. However, there are 49 trajectories where the algorithm has identified 3 modes. The patterns identified mostly depend on the NOAA and METAR features. As Figure 76 shows, the training loss appears to be decreasing while the epochs advance.

Experiment with 5 modes and Raw+NOAA+Metar observations:

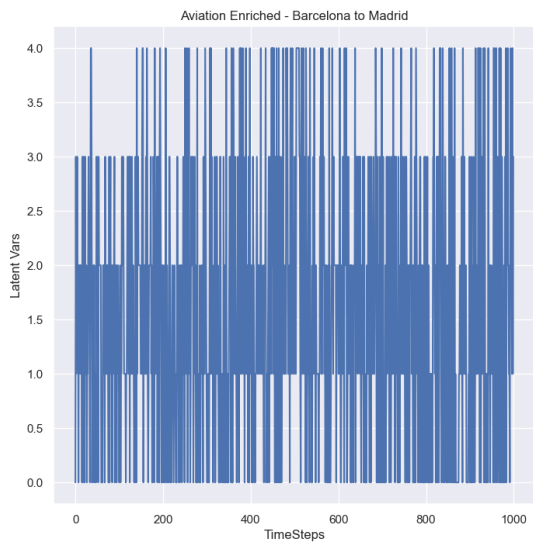


Figure 77. VAE, Latent Variables in the first epoch

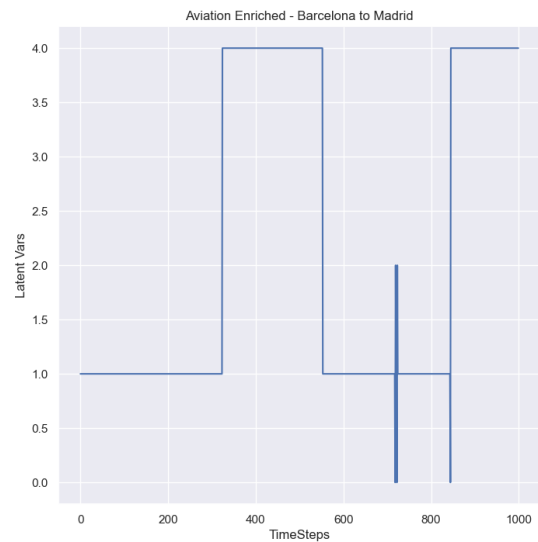


Figure 78. VAE, Latent Variables in the 500th epoch

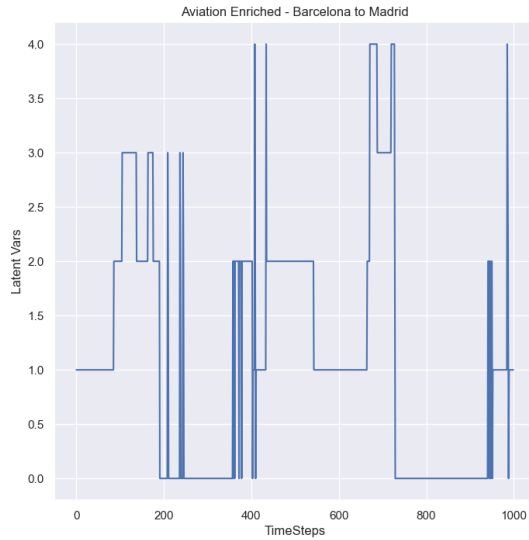
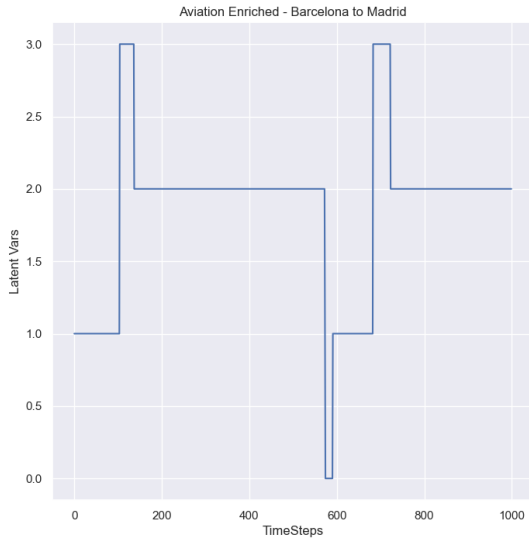


Figure 79. VAE, Latent Variables in the 1500th epoch Figure 80. VAE, Latent Variables in the 2000th epoch

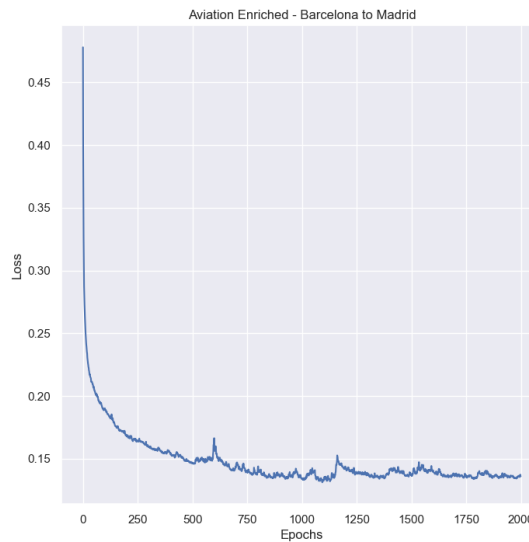
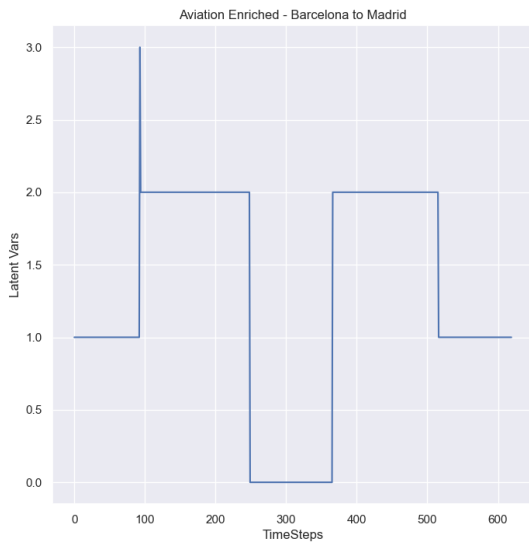


Figure 81. VAE, Testing 5 Latent Variables Figure 82 Training Loss of the VAE (5modes)

In the last setting with trajectories enriched with NOAA and METAR features, figures 77-81 show the modes identified by the VAE. The aircraft seems to be at take-off and landing phases when mode 1 appears. Changes regarding the meteorological features are indicated by modes switching between 2, 3 and 0. Figure 82 shows the training loss in subsequent epochs.

6.4.3 Deviations between experts and generated trajectories by Directed-Info Gail.

In all of the following figures, there is one expert trajectory with color red. The other colors indicate the identified sub-tasks of the generated predicted trajectory. The colors

of the generated trajectory are the same as in section 6.4.1. By depicting pairs of expert and predicted trajectories we can visually inspect the deviations of the policy trajectories from expert trajectories, and how the modes are split in the best and worst case scenarios, given the rewards scored.

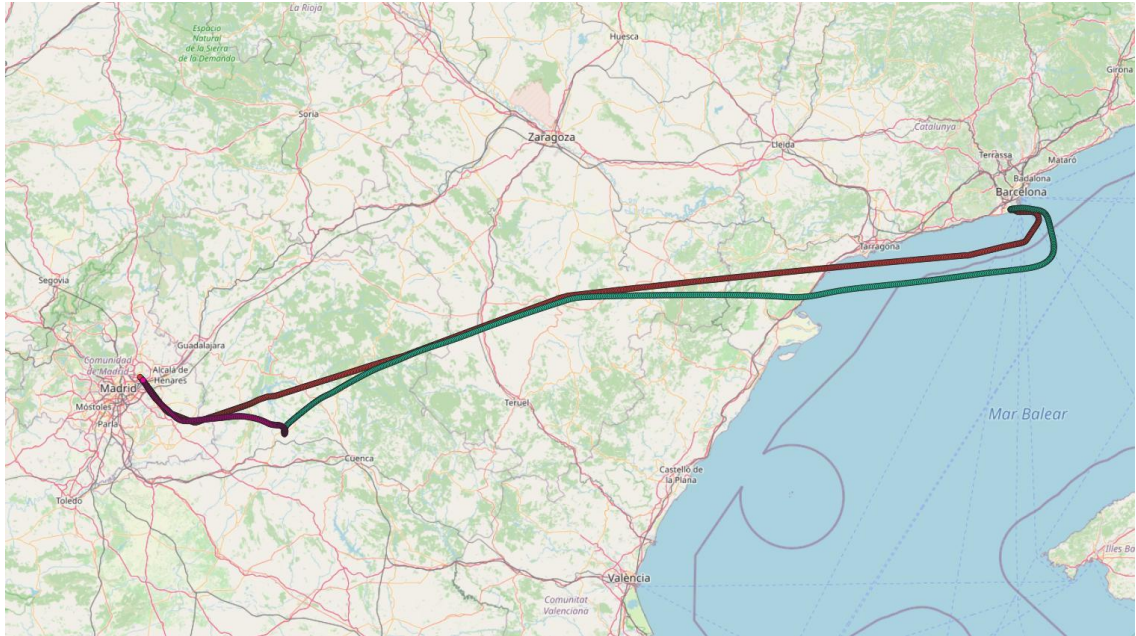


Figure 83. Barcelona to Madrid Directed-Info GAIL Results with 3 Modes. Features: 3D Point with timestamp without meteorological features. The best case scenario with 0 mean reward for this testing episode.

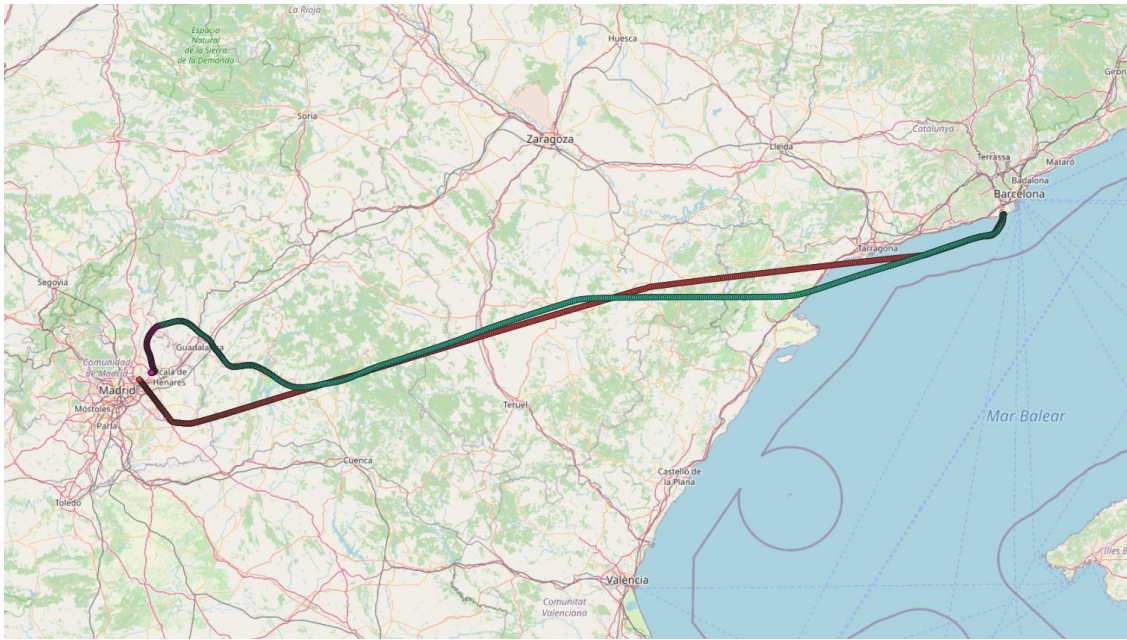


Figure 84. Barcelona to Madrid Directed-Info GAIL Results with 3 Modes. Features: 3D Point with timestamp without meteorological features. The worst case scenario with -71.70 mean reward for this testing episode.

The results in figures 83, 84 were tested using 3 modes. We can see that Directed-Info GAIL has learned to identify the take-off and en-route phases (indicated with green color) and the landing phase (indicated with dark pink), both, in the best and worst case scenario. So the decrease of altitude is the basic factor that switches the mode from green to dark pink. In this specific case while the policy of the aircraft imitates the expert trajectory with high fidelity, the encoder did not indicate one of the three modes needed.

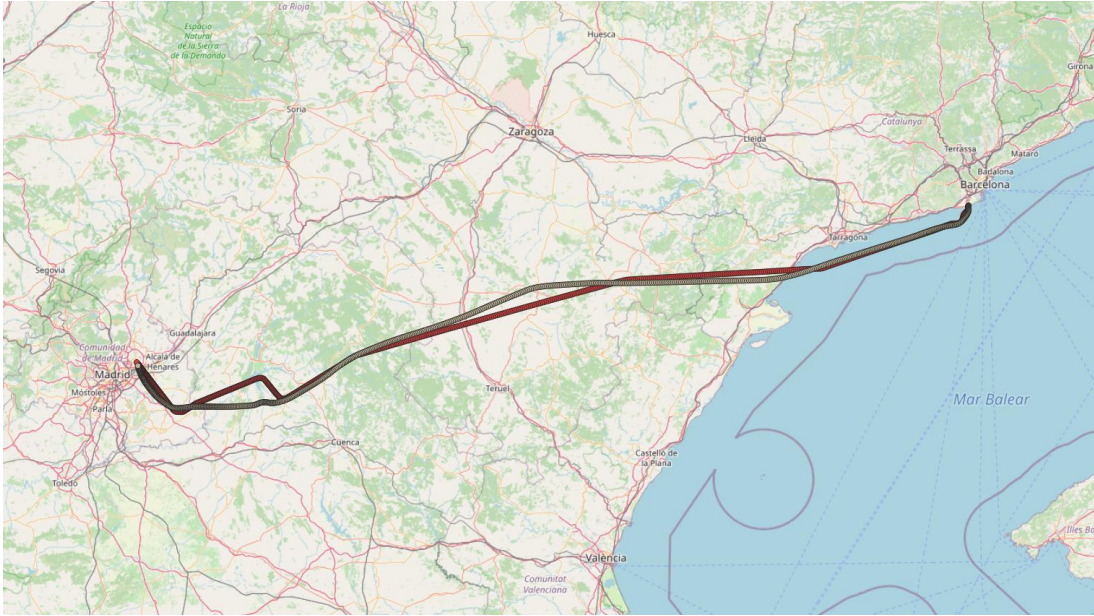


Figure 85. Barcelona to Madrid Directed-Info GAIL Results with 5 Modes. Features: 3D Point with timestamp without meteorological features. The best case scenario with 0 mean reward for this testing episode.

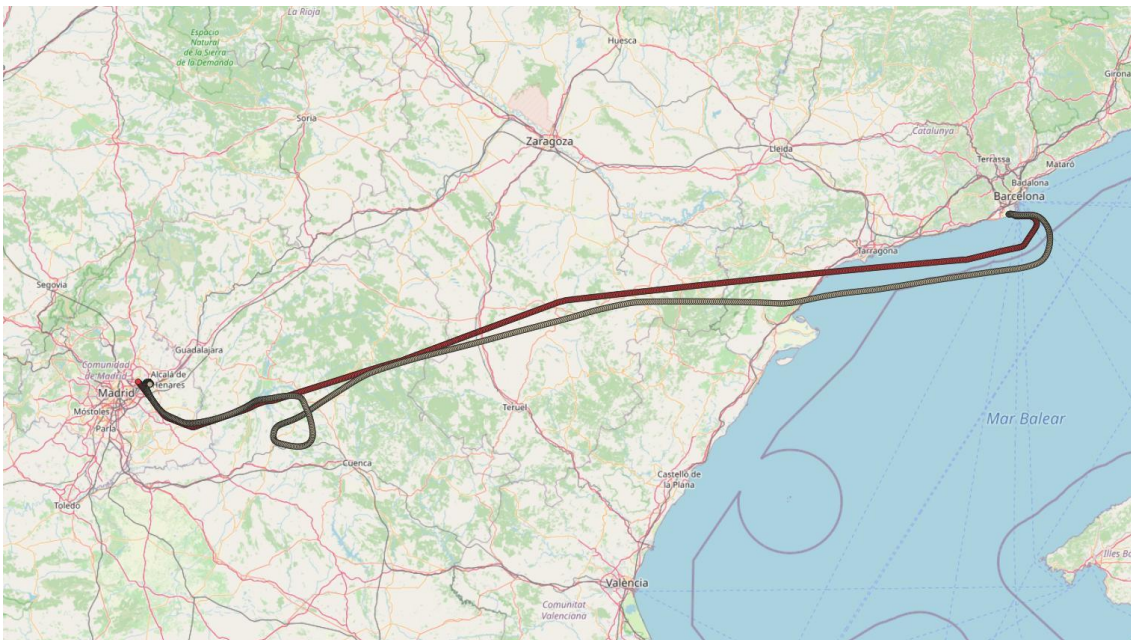


Figure 86. Barcelona to Madrid Directed-Info GAIL Results with 5 Modes. Features: 3D Point with timestamp without meteorological features. The worst case scenario with -82.80 mean reward for this testing episode.

We can see in figures 85-86 that Directed-Info GAIL has learned to split the modes in brown color in the generated trajectory in the best and worst case scenario. The setting with raw trajectories with 5 modes, was the worst experiment because we can assume that the observations' features (longitude, latitude, altitude and timestamp) weren't enough to split the trajectory with 5

modes. In this specific case the policy shows that can predict the expert policy with high accuracy but the encoder cannot identify the modes.

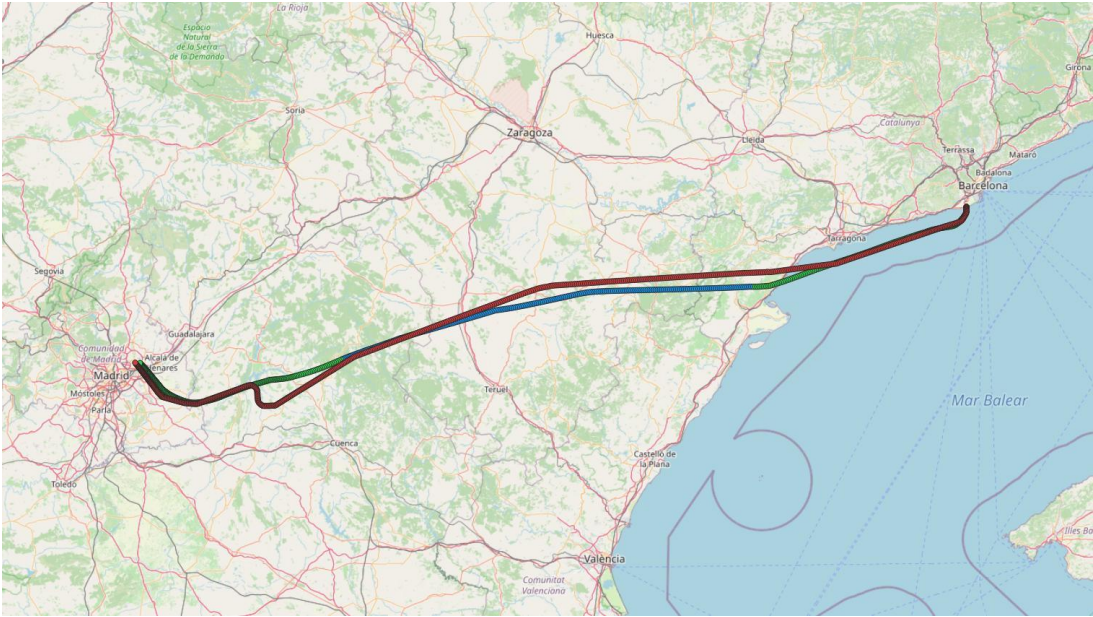


Figure 87. Barcelona to Madrid Directed-Info GAIL Results with 3 Modes. Features: 3D Point with timestamp and NOAA meteorological features. The best case scenario with 0 mean reward for this testing episode.

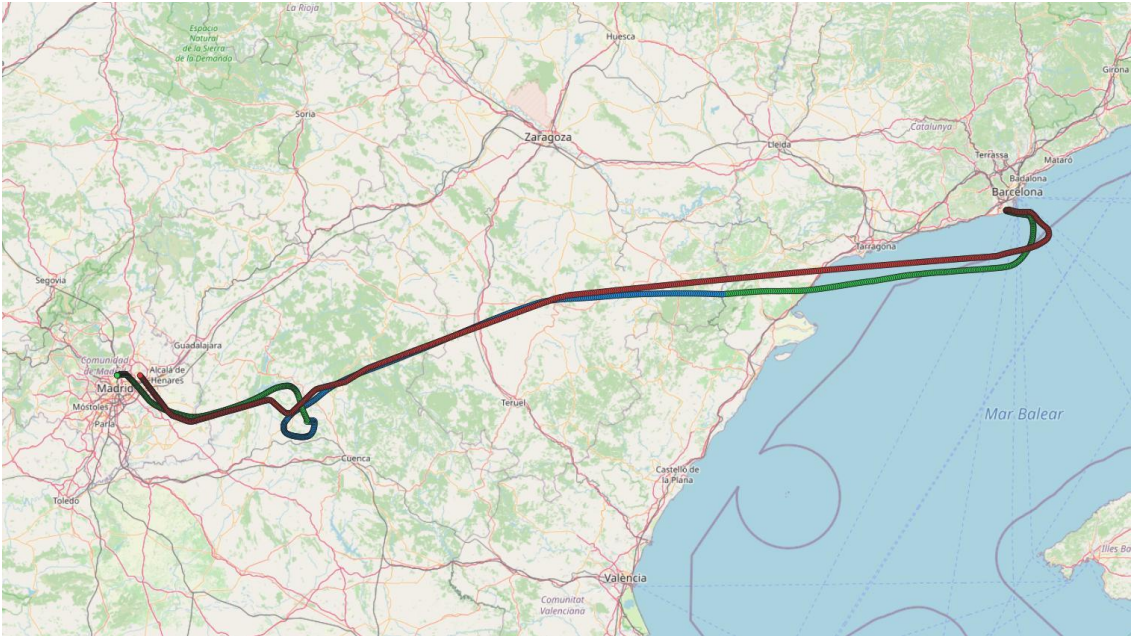


Figure 88. Barcelona to Madrid Directed-Info GAIL Results with 3 Modes. Features: 3D Point with timestamp and NOAA meteorological features. The worst case scenario with -107.70 mean reward for this testing episode.

In this experiment, in figures 87-88 we used 3 modes, with raw trajectories enriched with NOAA features. Directed-Info GAIL has learned the modes indicated as follows: Green color indicates the take-off and landing phases of the aircraft and blue color the en-route phase of the generated trajectory, in the best and worst case scenario. In both scenarios the aircraft is switching modes from blue to green when there is a decrease in altitude along with a change in NOAA meteorological features. Specifically, there was an increase of relative humidity and temperature and a decrease in u-component of wind and v-component of wind. In this experiment the method managed to indicate the modes in aircrafts' behaviors. So we can summarize, that the policy of the aircraft in this case is good enough and the encoder identified the modes.

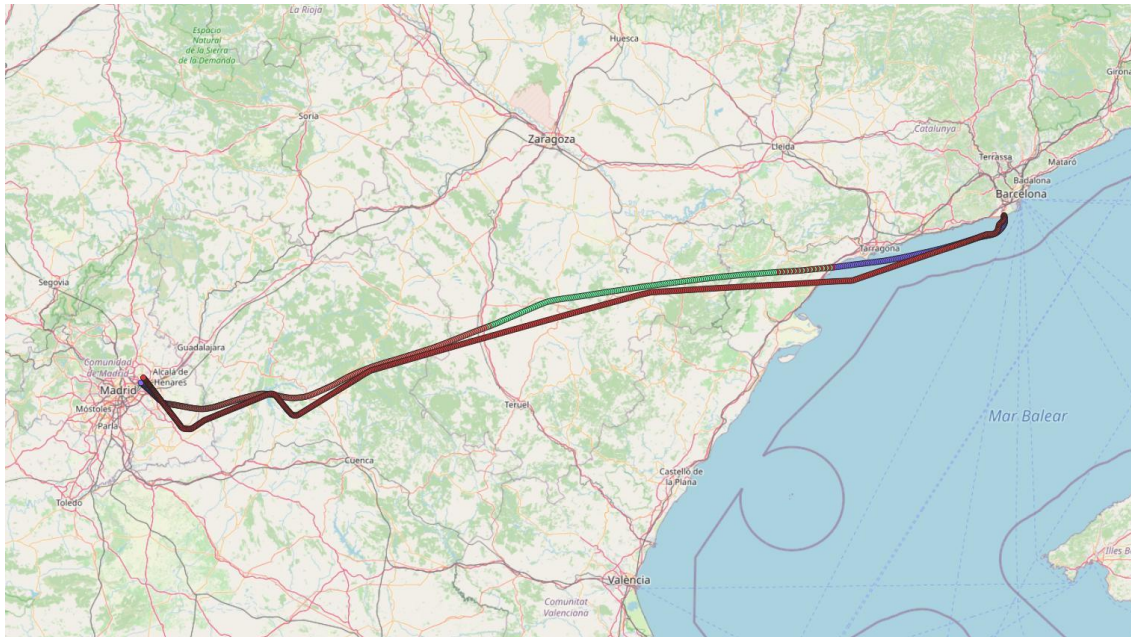


Figure 89. Barcelona to Madrid Directed-Info GAIL Results with 5 Modes. Features: 3D Point with timestamp and NOAA meteorological features. The best case scenario with 0 mean reward for this testing episode.

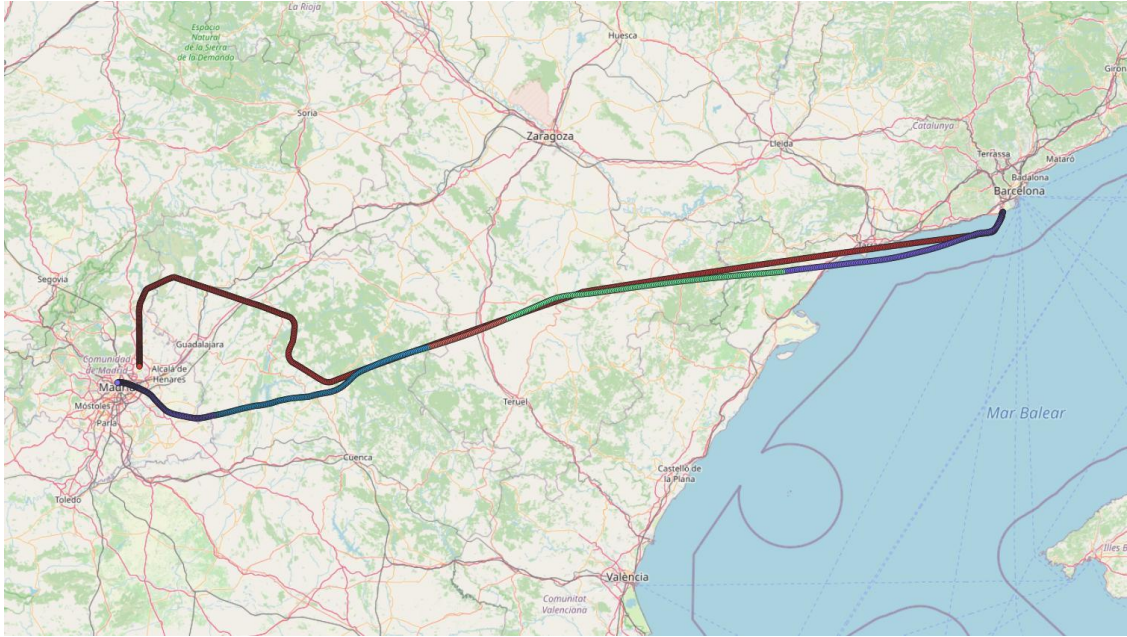


Figure 90. Barcelona to Madrid Directed-Info GAIL Results with 5 Modes. Features: 3D Point with timestamp and NOAA meteorological features. The worst case scenario with -134.21 mean reward for this testing episode.

In this experiment, with enriched NOAA features and 5 modes, Directed-Info GAIL in the best case scenario has learned to split the trajectory in the following sub-tasks: the modes with purple color is the take-off and landing of the aircraft and modes indicated in red, green and orange color depend on the meteorological NOAA features and the trajectory altitude. We found out that the orange color was appearing when the aircraft was losing altitude while reaching its destination. The red was switching with green mode every time-step until the green mode was stabilized. There is an assumption that the red and green mode are slightly similar with escalations in longitude, latitude, altitude and almost the same NOAA features, but they are different from purple and orange mode. If we take a closer look in Figure 89, there was a sudden decrease to 2.7 from 9.7 in relative humidity and a little change of temperature, an increase of u-component of wind and a decrease to v-component of wind, while switching to red and green from purple mode after the take-off. In the worst case scenario another mode appears before landing with blue color but the red mode didn't appear. In Figure 90, when blue mode appeared there was a sudden change in relative humidity from 68.9 decreased to 38.2, an increase in temperature and u-component of wind and a light decrease in v-component of wind . That was

one good experiment because we were able to see different patterns of aircrafts' behaviors. So we can presume, that the generated trajectory is really close to the expert trajectory and the encoder did indicate meaningful modes.



Figure 91. Barcelona to Madrid Directed-Info GAIL Results Results with 3 Modes. Features: 3D Point with timestamp and NOAA&METAR meteorological features. The best case scenario with 0 mean reward for this testing episode.

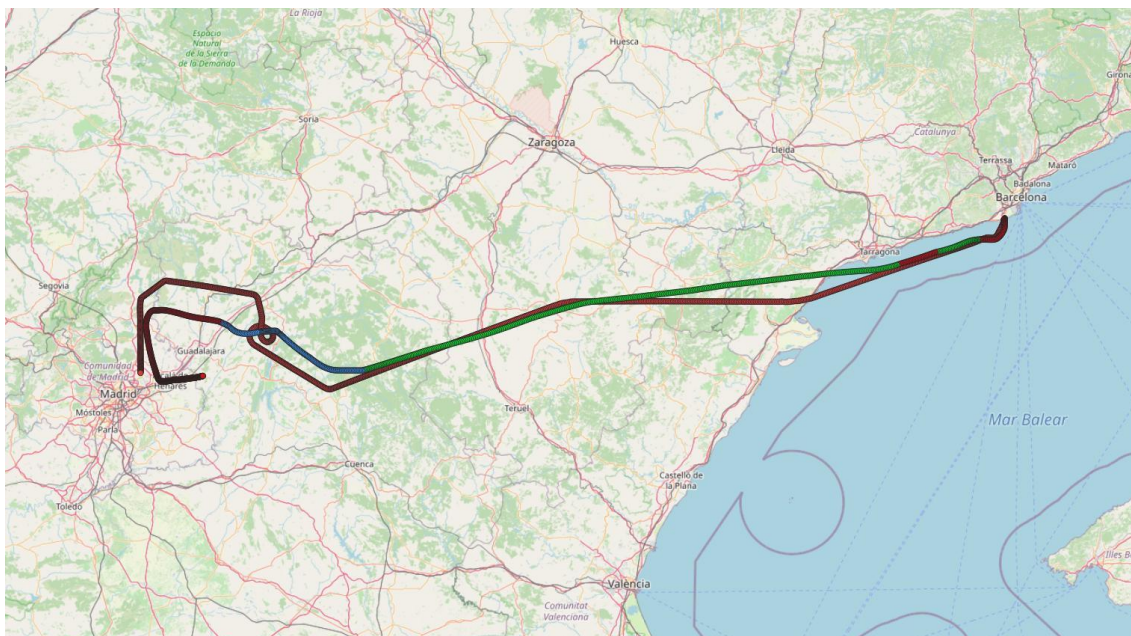


Figure 92. Barcelona to Madrid Directed-Info GAIL Results Results with 3 Modes. Features: 3D Point with timestamp and NOAA&METAR meteorological features. The worst case scenario with -352.08 mean reward for this testing episode.

In this experiment we used 3 modes, with raw trajectories along enriched with NOAA and METAR features. In the best case Directed-Info GAIL has learned to identify the following modes: With green color the take-off, with blue color the landing of the aircraft and with red color the en-route phase of the generated trajectory. In this case, the algorithm learnt to split the trajectory at take-off and landing phases, but in QGIS we checked the trajectory's modes more precisely and we found out that there was another part of the journey with blue mode. So there is high probability the weather changed because there was an increase in pressure surface and relative humidity. In the worst case the aircraft didn't manage to reach near Madrid's airport. We predicted this would happen because METAR indicated that there will be high gust with value of 30 and wind direction 190 degrees and wind speed in knots over 10 at Madrid airport. The colors of the modes are different as we can see in Figure 92 in contrast with Figure 91. That was not the best experiment because we weren't able to see similarities we wanted from best and worst case experiment. In these specific scenarios, the policy shows that can predict the expert policy with high precision. The encoder indicated the modes better in the best case scenario in comparison with the worst scenario in in Figure 92.



Figure 93. Barcelona to Madrid Directed-Info GAIL Results Results with 5 Modes. Features: 3D Point with timestamp and NOAA&METAR meteorological features. The best case scenario with 0 mean reward for this testing episode.

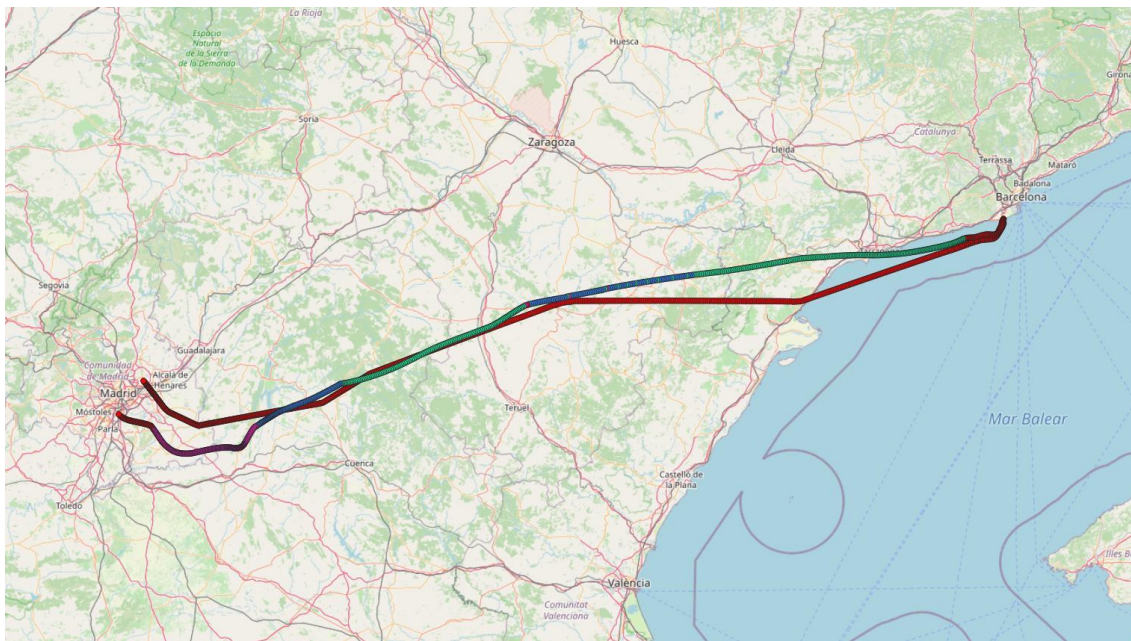


Figure 94. Barcelona to Madrid Directed-Info GAIL Results Results with 5 Modes. Features: 3D Point with timestamp and NOAA&METAR meteorological features. The worst case scenario with -253.29 mean reward for this testing episode.

In this experiment we used 5 modes, with raw trajectories along with timestamp and enriched with NOAA & METAR features respectively. In the best case in Figure 93, Directed-Info GAIL has learned to identify the take-off and landing phases with red color and with blue color the en-route phase of the generated

trajectory. While the blue mode appears, the aircraft was gaining altitude over 20000 and the longitude was decreasing drastically in contrast with the red mode where the altitude was increasing but in a lower range (600-19000). In the worst case in Figure 94, the algorithm identified 4 modes: with red the take-off and landing, and with green, blue and dark pink the en-route phase. The green mode could be a change in NOAA features while relative humidity and temperature were decreasing lower than 93.4 and 269.2 respectively. Also there was an increase in u-component of wind and v-component of wind. There is a big chance the dark pink mode could appear while the aircraft was making turns before landing. There is another color yellow but you can see it in Figure 45 which has all the trajectories. We checked the data of the yellow mode and we decided that there was an increase in altitude and a decrease in relative humidity, temperature, u-component of wind and increase in v-component of wind. That was one good experiment because we were able to see the differences in aircraft behaviors via the identification of modes. According to our understanding, the generated trajectory is predicted very similar to the expert trajectory and the encoder split the trajectory in the necessary modes.

7 Conclusions

In this thesis, we explored a way of model-free imitation learning, using the Directed-Info GAIL algorithm. The agent followed a trajectory while was switching between subtasks to imitate expert's behavior, executing the appropriate actions under a specific policy learnt. The VAE model according to the Directed-Info GAIL methodology is necessary to split the trajectory and indicate the different modes of behavior to be followed, depending on the experimental setting: For instance, reaching the peak, ascending or landing in the hopper environment. But this model is not enough to make the policy of our agent able to decode the action in the best way and reproduce (imitate) the expert policy towards evolving the trajectories. So, we have to use the GAIL framework to monotonically improve agents' behavior given the expert state-action pairs. The first algorithm has low bias and high variance as it is happening in behavioral cloning approaches. The second algorithm GAIL [1] appears to improve the outcome and lower the variance, balancing the bias-variance trade-off so as to train the agent to imitate the trajectories in the best way.

In the future, we hope that there will be better algorithms that will be able to divide a trajectory of a vehicle or of a robot in a more advanced way. More specific neural networks can control more accurately the landing at an airport, in the aviation domain. Also, some information in the data of the aircrafts' trajectory is missing while turns are made. This method can be tested in the future for different origin-destination airports. There is a lot of space for improvement and in the future we hope to make a more consistent algorithm that can divide every expert's trajectory in such way that can be easily be comprehended by agents in dynamic environments, in more than 4 dimensions. In such a way, after these algorithms are tested in simulations, one can start impacting the self-driving domain in real world environments. In future works, the VAE pre-training algorithm can be modified in such a way that the latent variables are used as a sequence of all previous latent variables while

pre-training the agent, instead of using only the previous latent variable at each iteration. Last but not least, the game industry can use these algorithms in their engines with their environments and map the movement and the actions of characters, NPCs or vehicles behaviors in the simulating game world.

Bibliography

1. Ho, J., Ermon, S.: Generative adversarial imitation learning. In: NIPS, pp.4565–4573 (2016).
2. Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In Advances in Neural Information Processing Systems, pp. 3815–3825, 2017.
3. Arjun Sharma, Mohit Sharma, Nicholas Rhinehart, Kris M. Kitani. DIRECTED-INFO GAIL: LEARNING HIERARCHICAL POLICIES FROM UNSEGMENTED DEMONSTRATIONS USING DIRECTED INFORMATION(2019).
4. Alevizos Bastas, Theocharis Kravaris, and George A. Vouros. Data Driven Aircraft Trajectory Prediction with Deep Imitation Learning, CoRR abs/2005.07960 (2020).
5. Jonathan Ho H, Jayesh K. Gupta , Stefano Ermon. Model-Free Imitation Learning with Policy Optimization (2016).
6. BD. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. NeuralComputation, 3(1):88–97, 1991.
7. Michael Bain, Claude Sammut. A Framework for Behavioural Cloning (2001).
8. Saurabh Arora, Prashant Doshi. A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress(2019).
9. Brian D. Ziebart J. Andrew Bagnell Anind K. Dey. Modeling Interaction via the Principle of Maximum Causal Entropy (2010).
10. Seyed Kamyar Seyed Ghasemipour, Shixiang Gu, Richard Zemel.UNDERSTANDING THE RELATION BETWEEN MAXIMUM-ENTROPY INVERSE REINFORCEMENT LEARNING AND BEHAVIOUR CLONING.
11. T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel and J. Peters. An Algorithmic Perspective on Imitation Learning (2018).
12. Konda, Vijay R and Tsitsiklis, John N. On actor-critic algorithms. SIAM journal on Control and Optimization,42(4):1143–1166, 2003.
13. Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, a. P.: High-dimensional continuous control using generalized advantage estimation (2016-2018).
14. Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav Sukhatme, and Joseph J Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In Advances in Neural Information Processing Systems, pp. 1235–1245, 2017.
15. Xi Chen, Yan Duan, Rein Houthoofd, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan:Interpretable representation learning by information maximizing generative adversarial nets. In Advances in Neural Information Processing Systems, pp. 2172–2180, 2016.
16. Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144, 2016.
17. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: ICML, pp. 1889–1897 (2015).
18. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning.In: ICML, p. 1 (2004).

19. Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: Aaai, vol. 8, pp. 1433–1438. Chicago, IL, USA (2008).
20. D. P. Kingma and M. Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

