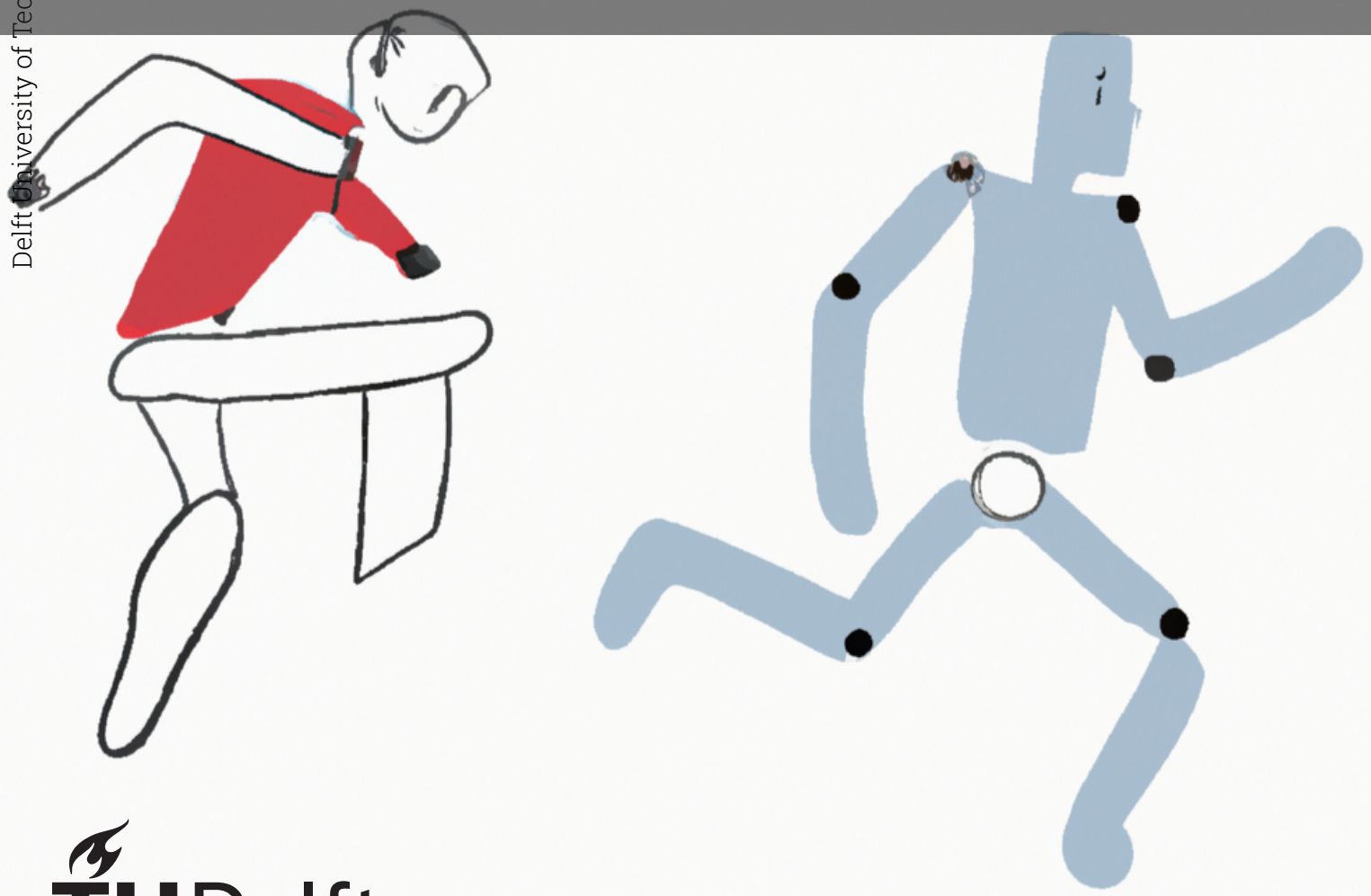
Imitation Learning From Observation Through Generative Modelling

Learning Reward Functions From Demonstrations for Contact-Rich Robotic Manipulation

Literature Research

Anish Abhijit Diwan

Delft University of Technology



TU Delft

Imitation Learning From Observation Through Generative Modelling

Learning Reward Functions From Demonstrations for Contact-Rich Robotic Manipulation

by

Anish Abhijit Diwan

Advisors:	Dr. Jens Kober [1], Dr. Julen Urain De Jesus [2], Dr. Jan Peters [2]
Literature Research Duration:	November 2023 - January 2024
Affiliation:	[1] Department of Cognitive Robotics, TU Delft [2] Intelligent Autonomous Systems Group, TU Darmstadt

Cover: Generated by Dall-E 2. Identifier: "DALL·E 2023-12-12 13.51.06 - An illustration with a robot on the left and a human on the right. The robot is trying to copy the human who is jumping. Light colors, whiteish background"

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Abstract

Right from the early days, humans have envisioned robots as machines that can interact with and manipulate their environment in a “natural" way, just as they do. Imitation learning is a framework for achieving these goals where the robotic agent learns to imitate an expert demonstrator. A large portion of the recent research in this field has strived to work towards building robotic agents that can effortlessly manipulate not only their own complex geometry but also objects in their environment by relying primarily on expert demonstrations. With the rapid rise in the capabilities of generative machine learning methods and their recent success in generating realistic images, videos, and other data modalities, researchers have now started to explore the intersection of these techniques to generate close-to-natural robotic motion.

The objective of this thesis is to explore the use of generative modelling methods such as energy-based models/diffusion within the framework of imitation learning. This literature research specifically focuses on two fundamental methods in the imitation learning literature – **Generative Adversarial Imitation Learning** and **Imitation from Observation** – intending to understand their theoretical background and practical considerations and review their shortcomings. The report discusses the benefits of generative modelling in imitation learning and analyses some issues such as the instability of generative adversarial networks (GANs), non-smoothness and mode-dropping in GANs, imitation under partially observable demonstrations, and some challenges of reinforcement learning as a subproblem in imitation learning. The outcome of this study is a detailed set of open questions in current literature and interesting ways to tackle some of these challenges via algorithmic changes to the current state-of-the-art. This will then guide a rigorous thesis on the use of energy-based generative models (like diffusion) to learn reward distributions within imitation learning.

Contents

1

Introduction

1.1. Motivations, Scope & Research Question

Robots hold immense potential to streamline and simplify human lives across various domains. From dangerous industrial applications like vehicle assembly to mundane household tasks such as cleaning and kitchen preparation to critical medical applications like surgical assistance, robots promise a much safer and more convenient future. However, such a future and the eventual usefulness of these robots predominantly rely on their ability to interact with and manipulate our very complex physical world effectively. The robots of the future must transcend their current dependence on well-defined locomotion spaces and case-specific manipulation strategies – such as "robot safe" zones and predefined motion priors – to be able to walk in unstructured spaces and pick up/manipulate several arbitrary objects. Achieving this requires learning the same motor and dexterous manipulation skills that we humans have perfected over our lifespan. What if robots could learn these skills by simply imitating humans from demonstration?

Imitation learning is a field of machine learning that aims to learn control policies for robotic agents by imitating demonstrations given by an expert (humans in most cases). The imitation learning problem can be boiled down to interpreting these demonstrations to recover scalar reward signals that can then be optimised to learn a control policy. A reward signal provides the robot with the correct incentive to do a certain task. In imitation learning, this reward signal motivates the robot to closely follow the actions of the expert. While a host of methods for recovering reward functions have already been proposed in the imitation learning literature [1], recent advances in the field leverage generative machine learning techniques for this task. One such recently proposed framework is Generative Adversarial Imitation Learning (GAIL) [2]. GAIL uses an adversarial training procedure that learns reward functions by simultaneously training two function approximators in a min-max game. GAIL has been a very influential algorithm and has led to several other methods that attempt to weave

generative adversarial models into imitation learning [3]–[5]. However, a critical analysis of the optimisation procedures of GAIL, its required data modalities, and empirical characteristics highlights shortcomings that, if systematically addressed, could lead to a new, improved algorithm. The primary shortcomings of GAIL-like methods are summarised in the following paragraphs and are analysed critically in the remainder of this report.

Issues in GAIL-like methods arise from the generative adversarial optimisation procedure that they use to learn reward functions. These objectives are famously known to cause unstable training. The functions learnt using these objectives have issues accurately mapping to multi-modal distributions, are not smooth (continuously differentiable) in all parts of the sample space, and are often inadmissible to the process of conditioning. These shortcomings mean that GAIL-like methods often take longer to learn and fluctuate a lot in terms of their performance and the way they solve the task. Further, they often only effectively learn one way of solving tasks. For instance, when learning to pick up an object, such an algorithm might frequently switch between reaching from the left and reaching from the right, while ultimately only converging on one of those behaviours. Additionally, because of the difficulty of conditioning adversarial models, the policies learnt by these algorithms are also theoretically more restricted. From the object-picking analogy, other generative techniques such as energy-based models might naturally learn different policies to pick an object depending on the type of the object, while adversarial learning algorithms like GAIL are difficult to adapt to different conditions. An added challenge in imitation learning – made even more pertinent by the realities of imitation learning in real-robot settings – is to learn to imitate experts in the presence of partially observable demonstrations (only containing trajectories of states and missing other information like the actions executed by the expert). The original formulation of GAIL also does not operate under partially observable demonstrations, rendering it further infeasible for real-world robotics tasks where data availability is often restricted.

Empirically speaking, careful hyperparameter tuning and clever architectural changes have been shown to result in good quality results from generative adversarial objectives – as is also evident from the excellent empirical results of GAIL. From a scientific perspective, although adversarial techniques do seem to work in practice, it is still difficult to attribute their excellent performance solely to algorithmic ingenuity. [6] present a large-scale study of adversarial architectures that showcases that with enough hyperparameter optimisation and random training restarts, most adversarial architectures tend to reach similar performance levels. Their study underscores the inherent unpredictability of GANs and raises questions about reliability and statistical significance in algorithms that use GAN-like objectives.

It seems (to me) that the recent outlook in the machine learning community is to rely more heavily on empirical results backed by training with very large computational budgets than on theoretical argumentation or sample-efficient algorithmic ingenuity ([7] show a graph of the exponential rise in arXiv papers in AI/ML). Relying on large computational budgets to demonstrate performance improvements also has environmental and ethical downsides and might lead to undesirable effects in the machine learning community. First, large machine

learning models that require weeks of training and subsequent hyperparameter optimisation, also require a significant amount of energy [8], [9]. This raises questions about the sustainability of machine learning and points towards the potential downsides of achieving performance gains through excessive training (rather than algorithmic ingenuity). The further deployment of such large models in everyday products [10]–[12] has also been critically analysed [9], [13]–[15] and the environmental impact (including carbon footprint and other energy costs) is something that must be considered as a genuine downside of machine learning. Secondly, the recent preference of machine learning scholarship towards larger models and bigger computational budgets also raises the barrier to entry into the community. If conferences, journals, and industries tend to prefer deeper-architecture research, does that not imply a dismissal of those ideas that are not backed by access to expensive computes? In a broader sense, this might also lead to the stagnation of the field and missing out on influential ideas. Lastly, deep architecture research inevitably comes with the burden of requiring very large datasets. Products like Dall-E and Chat-GPT [10], [12] are trained on several terabytes of data that are often just scraped from text, images, and video material on the internet. Aside from the ethical and legal repercussions of such data scraping, these datasets also require several hours of manual labour for filtering, cleaning, and post-processing. Being closed-source, it is also very hard to determine the biases in these datasets – biases that naturally also transfer over to the machine learning models trained on this data. Works like [16]–[18] analyse the various ethical considerations of such technologies from various perspectives. Their work discusses several facets of the problem like the utility of the technology vs. its human cost, the legal vs. moral considerations of copyright protection, and the idea that the ethics of a technology could ultimately be user-defined. While the conversation around the ethics of artificial intelligence is vast and nuanced, it is my opinion that machine learning research should at least strive to be transparent, openly accessible, and receptive to the idea that such technologies are prone to biases and errors.

Although incapable of making the larger, systemic changes that might be necessary to steer the community away from such pitfalls, this literature study proposes to address issues such as instability, non-smoothness, restricted modality, inoperability in partial observability, and the potential need for large computational budgets that exist in the GAIL-like algorithms mentioned above. This study mainly focuses on (i) exploring better generative model alternatives like energy-based models and diffusion and (ii) focusing on the problem of imitation under partial observability. The goals of this literature study are to

1. Review the existing literature and background work at the intersection of imitation learning and generative modelling. Specifically focus on studying the benefits and drawbacks of **Generative Adversarial Imitation Learning** and **Imitation from Observation**, and other recent methods that are similar to these (AMP, Diffusion Policy etc.).
2. Study GANs and similar generative methods commonly used in the imitation learning literature. Also, review energy-based models and particularly focus on diffusion. Compare their merit via empirical results, theoretical analysis, and suitability for the data modalities that are common in robotic manipulation tasks.

3. Explore potential changes to the identified methods that might promise to improve their performance.

With the scope identified above, the following research questions will be studied

How can generative modelling techniques like energy-based models/diffusion improve imitation learning frameworks like Generative Adversarial Imitation Learning (GAIL) from the perspective of learning contact-rich robotic tasks in a fast, repeatable, and versatile manner?

What changes might be necessary to improve the practical applicability of GAIL-like algorithms, especially for imitation under partial observability? How is the ability to learn in partially observable situations relevant to contact-rich tasks and how can the above-mentioned generative models benefit this cause?

1.2. Literature Review Strategy & Report Outline

As mentioned in the previous section, this literature research is grounded around two fundamental fields in the imitation learning literature – Generative Adversarial Imitation Learning (GAIL) and Imitation from Observation (IfO). The strategy adopted for this literature research mainly focuses on reviewing books, research papers, and technical review articles. The citations from a few fundamental papers, the papers that in turn cite GAIL and IfO, articles recommended by advisors, and online search tools like Google Scholar were the primary sources of information. A broad search of "recent work in generative imitation learning" via sources like Google Scholar, Google Search, academic tools like Scopus/Web-of-Science, and the network of authors that collaborate with researchers in imitation learning, was also conducted to avoid excluding potentially impactful recent work from this literature review. The review process led to insights on the following points that also form the outline of the report.

- Section 1.3 - The fundamental methods/concepts that are used within or act as a theoretical background to imitation learning
- The background work around GAIL and IfO
  - Section 2.2 - The general field of generative machine learning methods, and a specific focus on, Generative Adversarial Networks (GANs)
  - Section 2.2.1 - Studying the challenges of GANs and their pertinence to imitation learning
  - Section 2.3 - Other generative methods like energy-based models, that could potentially be better alternatives
- Section 3.1 - The field of imitation learning as a whole and the main sub-categories of this field of study
  - Section 3.2 - The challenges in imitation learning
- Chapter 4 - Opportunities for the use of generative modelling in imitation learning and works in recent literature surrounding GAIL and the problem of IfO

- Chapter 5 - A broader discussion on the findings of this literature review, and recommendations/new opportunities for the field that will be studied in further depth via a thesis

1.3. Theoretical Background

Section 1.1 briefly introduces imitation learning as a procedure that first recovers a reward function from a set of expert demonstrations and then subsequently optimises that reward function to take the appropriate actions. GAIL [2] and similar adversarial algorithms recover this reward function using generative machine learning techniques. In doing so, GAIL internally minimises the dissimilarity between the actions taken by the expert and the imitator. It then uses the similarity between the imitator's and the expert's actions as an indication of the current performance of the imitator and uses this to incentivise the imitator to iteratively improve its actions via optimisation. Even such a brief description of imitation learning and adversarial methods raises several questions about their specifics and mathematical formalisms. For instance:

1. How is the similarity (or dissimilarity) between sets of trajectories defined?
2. What is a reward function? What are actions? What provides the agent with the reward and how is the interaction between the agent and its surroundings defined mathematically?
3. What does it mean to improve an agent's actions and optimise the reward function?
4. How do these optimisation techniques work and what are some popular examples from recent literature?

A detailed and critical discussion of techniques in imitation learning and the answer to the above-mentioned questions require an understanding of some topics in statistical machine learning. The current section lays down preliminaries and mathematical formalisms that might benefit the reader to follow the rest of this report. Section 1.3.1 discusses the measures of dissimilarity that are often used in generative modelling techniques. Section 1.3.2 lays down the mathematical framework under which imitation learning is defined. Section 1.3.3 discusses a procedure for sequential decision-making and optimising an agent's actions given a reward signal. This procedure is called reinforcement learning and is used within all imitation learning algorithms discussed in this report. Section 1.3.3 then discusses some popular reinforcement learning techniques that are seen in state-of-the-art imitation learning algorithms. Finally, section 1.3.4 goes into some gradient-free optimisation techniques that could be simpler alternatives to reinforcement learning. This report follows the notation adopted in [19] for concepts in reinforcement learning. Variables and other quantities are highlighted in the text as and when necessary.

1.3.1. Cross-Entropy, KL Divergence & Jensen-Shannon Divergence

Optimisation in the context of function approximators usually involves matching the output of the function approximator with a known ground truth value. Most machine learning techniques use function approximators that return probability distributions. A natural way to optimise such functions is to then match the returned probability distribution with a known one. Measures of divergence are functions that in turn specify the similarity between two probability distributions.

The cross-entropy $H(p||q)$ between two distributions $p(x)$ and $q(x)$ is a measure that defines the distance as the average number of data samples needed to correctly identify the source of the data given that the source is ambiguous prior to the sampling process. The log-likelihood of a distribution is a similar measure that also defines the likelihood of the source of the data given samples drawn from it. However, the log-likelihood is usually used in isolation and not in the context of two distributions. In practice, these two differ minutely in their formulation and are used almost interchangeably depending on the problem definition.

The Kullback-Leibler (KL) Divergence [20] is another distance measure that is a widely used metric in machine learning research and is often a part of optimisation objectives. It can be shown that optimising w.r.t the KL divergence is effectively the same as optimising the cross-entropy. It is important to note that the KL divergence is not symmetric, ie $D_{KL}(p||q) \neq D_{KL}(q||p)$. The Jensen-Shannon (JS) Divergence [21] is a symmetric version of the KL divergence that measures the divergence of both distributions from a new averaged version of both.

$$
\begin{aligned}
H(p||q) &= \mathbb{E}_{x\sim p(x)}[\log q(x)] \\
Log - Likelihood(p) &= \mathbb{E}_{x\sim p(x)}[\log p(x)] \\
D_{KL}(p||q) &= \mathbb{E}_{x\sim p(x)}[\log \frac{p(x)}{q(x)}] \\
D_{JS}(p||q) &= \frac{1}{2}KL(p||A) + \frac{1}{2}KL(q||A) \text{ where } A = \frac{p(x)+q(x)}{2}
\end{aligned}
$$

1.3.2. Markov Decision Process

Many of the machine learning methods discussed in this report – including imitation learning and by extension, reinforcement learning – involve a robotic agent sequentially making decisions to change its current state and interact with its surroundings. The agent's decision-making and the interaction between the agent and its surroundings are defined by the mathematical framework of a Markov Decision Process (MDP). Loosely speaking, an MDP is a framework in which a *rational agent* [22] takes *actions* to interact with its *environment* and subsequently obtains a numeric *reward* signal while also influencing the state of the environment. The agent's goal is to maximise its long-term reward. Some problems choose to define a *cost* instead of the reward.

The cost provides the same motivations as a reward but in a negative direction. It is hence minimised whereas rewards as maximised.

Formally, a Markov Decision Process is a framework for formulating problems involving sequential decision-making [19], [22]. An MDP (finite) is a [22] discrete time model (the subscript t is used to indicate the time step), formally defined as a 5-tuple $< S, A, T, R, \gamma >$ where

1. S is a set of states
2. A is a set of actions that the agent can take
3. $T(s_{t+1}|s_t, a_t)$ is a function of the agent's current state ($s_t \in S$) and executed action ($a_t \in A$), defining the transition dynamics of the environment that take the agent from the current to the next state ($s_{t+1} \in S$)
4. $R(s_t, a_t, s_{t+1})$ is a reward function that maps the agent's current state, executed action, and next state to a scalar reward signal
5. $\gamma \in (0, 1]$, called the discount factor, determines the agent's outlook towards the reward signal (more on this below)
6. In some definitions an additional initial-state distribution $\rho(s_0)$ that defines the distribution of the initial state $s_0 \in S$ is also included. However, this report omits it and mentions it whenever necessary
7. In problems where the state space is not fully observable, an additional observation set O and an observation model $O(o_{t+1}|o_t, a_t)$ is also included in the formalism (in this case, the framework is formally called a Partially Observable Markov Decision Process (POMDP)).

When defining a Markov Decision Process, we make the *Markov* assumption. In problems involving sequential decision-making, the state of the environment is assumed to retain all relevant information from all past actions and state transitions. The Markov assumption (or property) captures the same. It states that an agent's future state and the reward signal it receives are independent of its history of state transitions and actions, given its current state, action taken (and the next state in the case of the reward). The discount factor γ determines the agent's preference for immediate or delayed rewards. Unlike other hyperparameters such as the learning rate, is defined as an integral part of the MDP because it influences the optimal solution to the problem. i.e. varying the value of γ can change the optimal solution to the MDP.

In general "solving" an MDP often boils down to computing a *policy*, $\pi(a|s)$. A policy is a mapping from the set of states S to the set of actions A. This essentially tells the agent which action to take in its current state for any state $s \in S$. The agent's goal is to maximise its expected discounted long-term reward (also called the *return*, G_t). In an infinite-horizon setting,

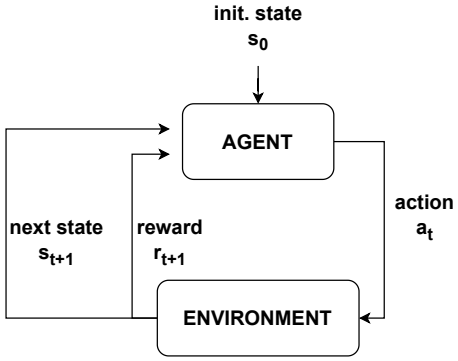


Figure 1.1: The agent-environment interaction loop

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s_0,a_0}[G_0]$$

where $s_0 \sim \rho(s_0); a_0 \sim \pi$ (ρ is the initial state distribution)

1.3.3. Reinforcement Learning - Background & Popular Methods

Given that the agent's current situation is defined by its state, and that it can take actions to interact with its surroundings to both change its state and simultaneously obtain a reward, how then does the agent learn to take the right actions so as to maximise the obtained reward? Reinforcement learning is one such procedure to achieve this.

Reinforcement learning is a paradigm of learning algorithms that attempt to learn an optimal policy through trial and error by *repeated interaction* with the *environment*. The agent's environment can be thought of as the "thing" that tells the agent its current state and the effects of its actions. It is "everything **other** than the agent" in a learning setting. For example, in a video game such as space-invaders, the environment is the actual video game that accepts the agent's actions, updates the positions of non-player characters (NPCs) and gives back the score (which could be thought of as the reward). In multi-agent games such as chess, the environment is a combination of both the chess board (along with the game logic) and the opponent player. [19] also provide examples from nature where they view animals like bees as agents while their cognitive chemistry can be considered the environment that gives them a reward signal when taking desirable actions like collecting nectar. Figure 1.1 shows a schematic of the agent-environment interaction. As described in section 1.3.2, the goal of the learning algorithm is to maximise the expected discounted return.

Model Based & Model Free Methods

A common theme across most sequential decision-making methods discussed in this report is their dependence on the "model" of the dynamics of the system. A learning algorithm

is considered *model-free* if it does not assume knowledge of or try to recover the transition dynamics $T(s_{t+1}|s_t, a_t)$ of the system [19]. Conversely, it is considered *model-based* if the problem-designer either provides this information or the algorithm attempts to learn it via interaction with its environment. Model-free methods have the benefit of being easier to apply on systems where actuation and low-level control are pre-solved problems for any motion plan returned by the algorithm. This is usually the case for fully actuated robots or systems where low-level actuation is not a hurdle. On the other hand, model-based methods are more applicable in cases where the output of the algorithm must comply with the complex dynamics and actuation constraints of the system. As a result of this, model-based algorithms are generally more data-efficient than their model-free counterparts. However, they also suffer from poor performance when the underlying dynamics are far too complex to learn from data. On the other hand, model-free methods are simpler as they do not rely on any prior knowledge about the system. They are also more general-purpose and less computationally expensive [1].

Value Based & Policy Search Methods

Reinforcement learning algorithms come in a variety of flavours. They can be roughly categorised as either those algorithms that assign some sort of *value* to states and actions (and subsequently change their policy to encounter high-value states/actions more frequently) or as those that directly search in the space of policies for a policy that gives a high expected discounted return.

The state value function $v_\pi(s)$ is a function of the agent's state $s \in S$ that tells it how "good" it is to be in this state. It is defined as the expected discounted return starting from state s and following policy π. The action value function $q_\pi(s, a)$ tells the agent how "good" it is to execute a certain action in a certain state and then following policy π from there onwards [19]. It can be shown mathematically that the state value function is just the expected action value given a policy.

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t|s_t = s] \\
q_\pi(s, a) &= \mathbb{E}_\pi[G_t|s_t = s, a_t = a] \\
v_\pi(s) &= \sum_a \pi(a|s) * q_\pi(s, a)
\end{aligned}
$$

Value function based methods such as value iteration, Q-learning, SARSA, DQN [19], [23] etc. all work with the general procedure of approximating the value function of a policy through repeated interaction with the environment and then updating the policy to pick high value states/actions more frequently.

The second class of reinforcement learning methods, called policy search methods, uses a parameterised representation of the policy π_θ and searches in the parameter space instead of iteratively approximating optimal value functions and then updating the policy. In these

algorithms, the policy is evaluated using a performance measure $J(\theta)$. Policy search methods are generally more robust in cases of partial observability (section 3.2) while also easily allowing for continuous state and action spaces. The general procedure used in such methods is to first start with some random initial policy with parameters θ, sample trajectories with this policy in the environment to obtain the performance measure, and then update the parameters to maximise $J(\theta)$.

Popular RL Algorithms - Actor-Critic, TRPO, PPO

This section explains some popular policy search RL algorithms from recent literature (called policy gradient algorithms) that are often used to learn control policies within the imitation learning (section 3.1) framework. Policy gradient algorithms operate under the conforms defined by the *policy gradient theorem* [24]. The basic idea of this theorem is that

1. Assuming that all episodes start from a single start state s_0 from which every action leads to a state sampled from a uniform distribution $\rho(s_1)$
2. Under the specific definition of the performance measure $J(\theta)$ as the value of this start state $v_{\pi_\theta}(s_0)$
3. It can be shown that the **gradient of the performance measure is proportional to the gradient of the policy**

$$\nabla_\theta J(\theta) \propto \sum_s \rho(s) \sum_a q_\pi(s,a) \nabla_\theta \pi_\theta(a|s)$$

While sounding obscure at first, the policy gradient theorem simply implies that – under assumptions that are perfectly reasonable – one can compute the gradient of the performance measure by computing the gradient of the policy. The gradient of the performance measure tells us how to change the policy parameters θ to get the steepest improvement in performance. Under this theorem, this gradient can now be defined in a form that is suitable for **computation by experience** (given an approximate action value $q_\pi(s,a)$). The very first policy gradient algorithms [25] compute this gradient as an expectation over action values sampled during the agent-environment interaction.

The stability of policy gradient methods can be improved by modifying this expectation to compare the action values to a *baseline* b. The idea of the baseline is to reduce variance in learning by providing an indication of the relative goodness of the action value. The baseline could be something like the average action value and $q(s,a) - b$ indicates just how much "better" the current action is compared to the average. *Actor-critic* (AC) methods [19] use an evolving baseline called a *critic* that is formulated as an approximation of the state value. $q(s,a) - v(s)$ then tells us the *advantage* of picking action a in state s when compared to the baseline value of the state. AC methods use learnt functions for both the policy (the *actor*) and the critic and update these at every iteration of training.

Trust Region Policy Optimisation (TRPO) [26] and Proximal Policy Optimisation (PPO) [27]

are two more very popular policy gradient algorithms that are a standard choice for learning control policies in continuous action spaces. A significant flaw in the policy gradient procedure explained above is that the policy parameters are updated by evaluating the performance of the policy by sampling trajectories generated by the very same policy. This means that the optimisation target ($J(\theta)$) is itself a result of the optimisation parameters (θ). If the policy parameters are updated recklessly, this can change the policy enough to lead to arbitrarily inconsistent changes in performance – i.e. changes in the parameter space are not consistent with changes in policy space. TRPO uses a procedure that limits the change in the policy between consecutive updates and ensures that updates strive towards a forward (monotonic) progress in performance. This is achieved by updating parameters within the confines of a "trust region", defining a "surrogate" advantage that ensures that the updated policy performs better, and by using the KL divergence between the policies before and after the update to ensure that there are no drastic policy changes. PPO is a first-order simplification of the TRPO objective – using adaptive KL penalties and clipping – that drops some of the guarantees of TRPO for an improvement in computational simplicity and sample efficiency. Under some considerations, PPO can be shown to be analogous to actor-critic methods in continuous spaces.

Goal-Conditioned RL

Goal-conditioned reinforcement learning is a slightly modified version of the RL problem where the agent interacts with the environment to fulfil a goal from a set of, goals $g \in G$ and $g \sim p(g)$ where $p(g)$ is a distribution of goals. The policy that the agent learns $\pi(a|s, g)$, the rewards it receives $r(s, a, s', g)$, and subsequently the optimisation target $\mathbb{E}_{p(g)}\mathbb{E}_{\pi(.|s,g)}[return]$, are all in the context of this goal and are distributed conditionally to g. This minor change in formulation enables the agent to take its decisions based on the goal. However, the utility of goal-conditioned RL is quite reliant on the larger setting of its use as it can easily be shown that the standard RL problem is also a goal-conditioned problem with just one goal.

1.3.4. Gradient Free Methods - CEM & CMA

Having discussed a rather complex procedure to optimise the rewards given by the environment (reinforcement learning), one might wonder whether there are easier ways to take actions to maximise rewards that do not involve neural networks or learning policies. Cross Entropy Method (CEM) [28], [29] and Covariance Matrix Adaptation (CMA) [30] are two such rather simple, but surprisingly effective optimisation algorithms. These are both evolutionary strategies that treat the optimisation objective as a black-box function and evolve a population of solutions to incrementally maximise the cumulative return from this function. CEM works by assuming that the black-box function is Gaussian (or a mixture of Gaussians). It hence learns the mean and covariance of a Gaussian distribution from which solutions (actions) are sampled. These parameters as then iteratively updated towards solutions that maximise the objective function. CMA employs a similar strategy of sampling, performance evaluation, and section/recombination, however uses a clever adaptation procedure (to balance learning speed and sample diversity) to update the covariance matrix of the Gaussian distribution.

These algorithms perform quite well in some reinforcement learning tasks and sometimes also outperform gradient-based methods [31]–[33]. They are hence appropriate baselines for comparison against any newly proposed algorithm.

2

Generative Machine Learning

A generative model can be loosely defined as a learnt function that generates *new* (previously unseen) data samples. Typically, such a function is learnt in an unsupervised manner by using machine learning techniques that take in unlabelled data samples as input. For example, [10], [34]–[36] present generative models capable of generating images of human faces, or handwritten digits, [37] present models that generate audio samples and [38], [39] present video-generation models.

The underlying idea in almost all generative models is to think of the data samples in the dataset, $\{x_1, x_2, .., x_N\}$ to have come from some unknown probability distribution $p_{data}(x)$. The goal of generative modelling is to learn a probability distribution $p_\theta(x)$ parameterised by parameters θ that very closely resembles the original data distribution. Some generative modelling techniques explicitly learn the distribution while others learn it implicitly by learning a model of the procedure to sample from this distribution. Section 2.1 provides an overview of these categories and introduces some popular generative modelling methods. The remainder of this chapter then focuses on two such methods – generative adversarial networks (section 2.2) and energy-based models (section 2.3) – that are the most relevant to the discussions that follow in upcoming chapters.

2.1. Likelihood-Based & Implicit Generative Models

A likelihood-based model is one that attempts to explicitly learn the data distribution by updating its parameters to maximise the log-likelihood of the sample having come from the learnt distribution. Popular generative models in this category are variational autoencoders (VAEs) [36], normalising flow models [40], energy-based models [41], [42], and autoregressors [43].

Since the data distribution can be arbitrarily complex, it is not always guaranteed that the

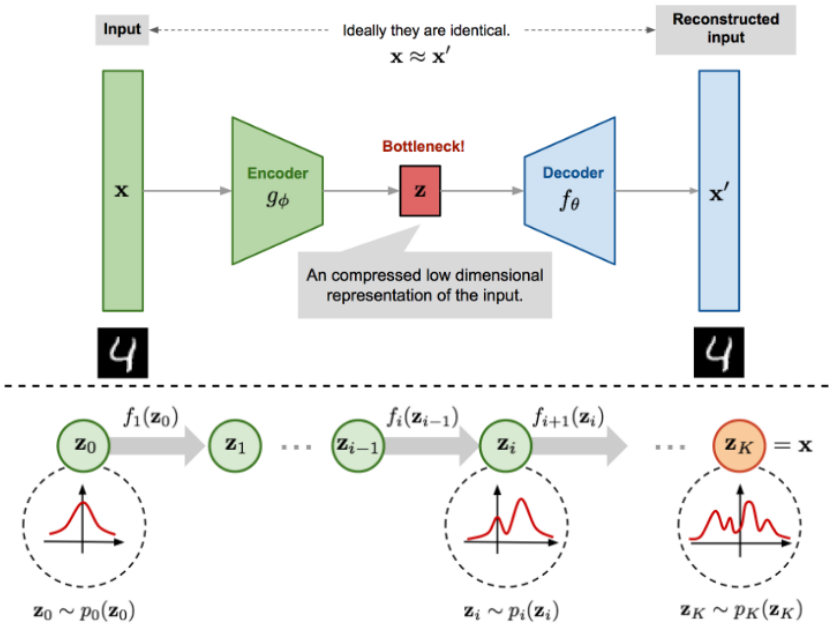


Figure 2.1: Top: an illustration of the training process of a variational autoencoder. Bottom: an illustration of the transformation of the learnt distribution in normalising flow models [44], [45]

process of sampling from it would be analytically tractable. Models that attempt to learn explicit representations of the data distribution often have to either make architectural simplifications that allow tractable sampling or optimise surrogate objectives that deviate from the maximum log-likelihood. VAEs work by learning an encoder-decoder couple. The encoder maps the input data points to a latent space from which another learnt layer then generates estimates for mean and variance, making the assumption that samples in the data distribution can be mapped to a set of Gaussian distributions with arbitrary parameters. The decoder then maps this mean and variance vector to a higher dimensional space and returns the generated sample. Training is facilitated by matching the similarity of the generated sample with the ground truth. Once trained, the encoder is discarded and the decoder is used to directly generate samples by passing in arbitrary values of mean and variance. Flow-based models use the concept of normalising flow. Here, the idea is to directly learn an explicit representation of the data distribution by first starting from a simpler distribution and then applying a series of invertible transformation functions that gradually convert it into a complex distribution. The resulting transformed function is analytically computable and allows for direct sampling. Figure 2.1 [44], [45] shows an illustration of the VAE and normalising flow model training procedures.

An implicit generative model on the other hand directly learns the procedure to sample from the learnt distribution, thereby not necessitating analytical tractability. Examples of popular implicit models are the generative adversarial network (GAN) [35] and the denoising diffusion probabilistic model (DDPM) [34]. GANs learn to generate samples by learning a mapping from a latent space to the sample space. While quite similar to VAEs in an architectural sense, GANs differ as they do not assume that the latent space represents parameters for a Gaussian distribution. Like normalising flow models, diffusion models also learn to convert

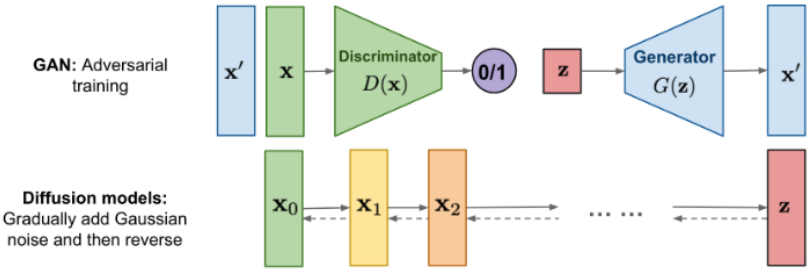


Figure 2.2: An illustration of the training procedure of GANs (top) and diffusion models (bottom) [46]

simple, tractable distributions into more complex ones. However, instead of computing explicit functions for the complex distribution, they learn an iterative denoising process that directly converts samples from one distribution to another. Figure 2.2 [46] illustrates the training process of both these models. Section 2.2 and section 2.3 discuss these two generative models in further detail and compare their characteristics from a robotics viewpoint.

2.2. A Deeper Look Into GANs

A generative adversarial network (GAN) [35] is an implicit generative model that uses a learnt *generator* function $g_{\theta^G}(z)$ to transform a vector in some latent space $z \in Z$ to a data sample x [47]. The generator function here is described by a neural network parameterised by parameters θ^G. GANs use an interesting training procedure inspired by game theoretic approaches, where the generator network "competes" with an adversary who attempts to spot whether the sample is artificially generated or truly from the data distribution. The adversary, called the *discriminator*, $d_{\theta^D}(x)$ takes in the generated sample x and returns a probability value describing the likelihood of that sample having come from the learnt distribution.

Learning in GANs is facilitated by formulating a zero-sum game where each network aims to maximise its own payoff that is defined by a function $v(\theta^G, \theta^D)$. The payoff is positive for one network while it is negative for another (typically the generator). The default choice for the payoff is

$$v(\theta^G, \theta^D) = \mathbb{E}_{x \sim p_{data}}[\log(d(x))] + \mathbb{E}_{x \sim p_{model}}[\log(1 - d(x))]$$

where x is the true data sample or a sample generated by the generator. In the ideal scenario, at convergence

$$g* = \arg\min_{g} \max_{d} v(g, d)$$

where $g*$ is the optimal generator function. The optimal generator function ideally generates

samples that are indistinguishable from the data distribution and the discriminator returns an output that is approximately 0.5. GANs can be theoretically shown to have convergence guarantees assuming that the payoff is convex in the generator's parameters. Unfortunately, in practice, this is rarely the case. GANs are famously known for being difficult to train, requiring a very careful selection of hyperparameters. Section 2.2.1 discusses some of these issues in detail and presents challenges that could arise from the use of GAN-like optimisation objectives in imitation learning.

2.2.1. Instability In GANs

[35], [48]–[50] present a detailed analysis of some of the issues related to GANs. As seen in section 2.2, training is formulated as the maximisation of the log-likelihood (or equivalently, minimisation of the Kullback-Leibler divergence) between the original data distribution p_{data} and the learnt distribution p_θ. The KL divergence has the desirable property of not requiring knowledge of the distributions (only samples are sufficient) and being at its minimum when $p_{data} = p_\theta$. However, $KL(P||Q) \neq KL(Q||P))$, i.e. the KL divergence is not symmetric. This leads to two forms of discrepancy.

If $p_{data}(x) > p_\theta(x)$ – the probability of the sample having come from the data distribution is higher than the probability of it having come from the learnt distribution – it leads to a phenomenon known as *mode dropping* (or *mode collapse*) where the learnt distribution fails to completely capture the data distribution. If $p_{data}(x) < p_\theta(x)$, it simply leads to undesirable results where the generator returns samples that do not resemble the desired distribution. Both these phenomena could reverse depending on the choice of $KL(p_{data}||p_\theta)$ or $KL(p_\theta||p_{data})$. This leads to additional ambiguity. In the work of [35], the model can be shown to minimise the Jensen-Shannon divergence (section 1.3.1), which can be seen as a "symmetric middle ground" to either formulation of the KL divergence. Minimising this distance measure has shown to greatly improve performance in terms of stability when compared to the GAN formulation that minimises just the KL divergence. However, the distance measure being minimised is not the only source of instability.

Mode dropping also happens as a result of the zero-sum nature of the GAN training procedure. The generator attempts to not explicitly learn the data distribution, but instead learn it under the assumption that the discriminator points it towards this data distribution with the right motivation. In reality, the generator is simply trying to minimise the discriminator's outputs. This often leads to the generator simply learning a distribution that the discriminator can consistently predict as being quite similar to the data distribution. Since the discriminator does not bother about diversity in the samples passed to it, the generator can get away with simply learning one of the many modes in the data distribution. Figure 2.3 shows experiments from [51] that aim to solve the issue of mode collapse by modifying the generator's objective. This however could come with the cost of other changes to the performance of GANs. Generating samples with high diversity is a continuing challenge in the GAN community and is an active research topic.

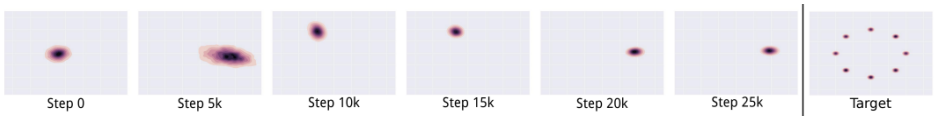


Figure 2.3: A demonstration of mode dropping [51]. The model can be seen to rotate through the multiple modes in the target distribution. However, without architectural and training parameter-related changes, the model inherently only learns a single mode

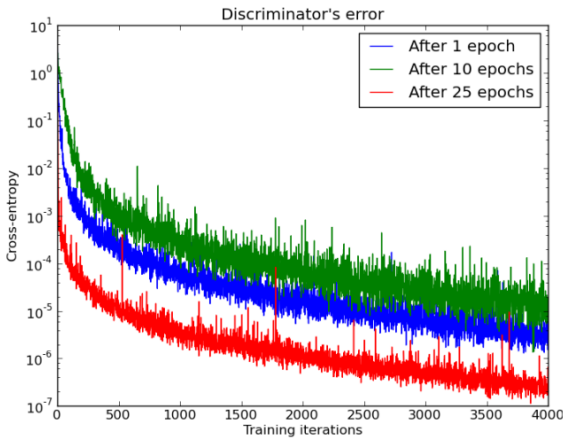


Figure 2.4: Discriminator error w.r.t training timesteps with a fixed generator recovered from a separate model trained up to a certain number of epochs. It can be seen that the discriminator error quickly drops to zero leading to its gradient dropping to zero, leading to worse updates to the generator [50]

Furthermore, without very fine-tuned hyperparameters, as the discriminator gradually receives better updates, the updates to the generator tend to get consistently worse. This is also an inherent feature of zero-sum games where, if one player learns a strategy that consistently overpowers the other, learning tends to come to a halt. Imagine a scenario at the early stage of training where the two networks have only begun to start learning. In such cases, the generator is often quite poor and generates samples that are not very similar to the data distribution. The discriminator then quickly learns to differentiate the real samples from the fake ones, causing $d(g(z)) \approx 0$. When this happens, $\log(1 - d(g(z))) \approx 0$ and $\nabla_{\theta^G} \log(1 - d(g(z))) \approx 0$. This means that the gradient of the objective function – the signal that tells the generator how to improve – gets weaker as the discriminator does better. This cascades as training continues and leads to the generator not being able to learn the data distribution. Figure 2.4 shows experiments from [50] that demonstrate the same.

Limitations Of Adversarial Objectives In Imitation Learning

The imitation learning algorithms discussed in chapter 4 use GAN-like adversarial objectives to learn a discriminator that is in turn used as a cost function for subsequent optimisation with reinforcement learning. The idea here is to learn a discriminator that can differentiate between the motions produced by the agent and the motions in the expert demonstration dataset and then iteratively update the agent's policy to minimise this discriminator-based cost. While this methodology seems technically sound and has shown good empirical results, the shortcomings of GAN-like optimisation objectives also tend to cause problems with such adversarial imitation learning techniques.

From the point of view of suitability as a cost function, the discriminator suffers from the issue of non-smoothness in the sample space. Smoothness refers to the ability to be continuously differentiable, thereby allowing the computation of gradients with respect to the parameters of a function approximator (trying to minimise costs). It is quite challenging to optimise non-smooth objectives with gradient-based optimisation techniques since the gradient – the signal that indicates the direction of the steepest ascent/descent – is often very close to zero in non-smooth regions of the sample space. Since the discriminator is simply optimised to maximise the divergence between the learnt and data distributions, it is not guaranteed to be smooth in the sample space. In contrast, generative models discussed in section 2.3, model the learnt probability distributions via smooth functions that are indeed continuously differentiable and capable of providing informative gradients. Figure 2.5 shows an illustration of this issue. Issues like mode-dropping also arise when using adversarial objectives in imitation learning. Instead of limiting the diversity in the generated samples, mode-dropping in adversarial imitation learning algorithms limits the expressiveness of the cost function (the discriminator). In such cases, the cost function might only reward some of the desirable actions seen in the expert demonstrations. This could cause the agent to only learn partially correct motions or perhaps not learn to imitate the expert in all possible ways.

GANs also lack some important features that are desirable qualities for generative modelling techniques in robotics tasks. The original definition of the adversarial objective – the definition that is used in most adversarial imitation learning algorithms – lacks the ability to condition the output of the generator on features that could define output modality or qualitative characteristics. This highly desirable characteristic would allow the user to define exactly what kind of output is needed. From the point of view of the learning policies, conditioning can enable the user to change the cost function (discriminator) depending on the desired motion characteristics. For example, it might be beneficial to be able to generate robot motions conditioned on the objects that the end effector is manipulating. Conditioning could also let the user slowly update the robotic agent's reward function as training progresses. For example, the cost function could be set to reward the agent for vaguely similar actions in the beginning and then progressively change to reward the agent only for actions that are exactly the same as the expert. This might greatly improve the stability of the learning algorithm.

[52] propose a conditional GAN architecture that can encode conditioning features in the latent space vector z, thereby allowing the generator to also generate conditioned outputs. However, this simple addition does not alleviate the problem of mode dropping and the generated outputs are still restricted in modality. Further, methods for probabilistic conditioning (such as [53], seen in diffusion) that can drastically improve sample diversity have not been fully explored in the GAN setting. In comparison, generative models discussed in section 2.3 are more open to conditioning in general. The task of conditioning a generative model can also be thought of as changing the learnt distribution by multiplication with another "modifying" distribution. The inability of GANs to adapt the learnt distribution means that GANs are less effective at tasks that involve denoising or computing posteriors from the generated samples.

In contrast, methods like diffusion can easily be adapted for tasks like in-painting or image colourisation. In the context of robotics, it is often necessary to iteratively update samples based on the evolving dynamics of the environment. For example, consider updating the end-effector's target pose depending on a noisy current state (as a result of actuator errors). Such iterative changes are quite difficult when using adversarial objectives.

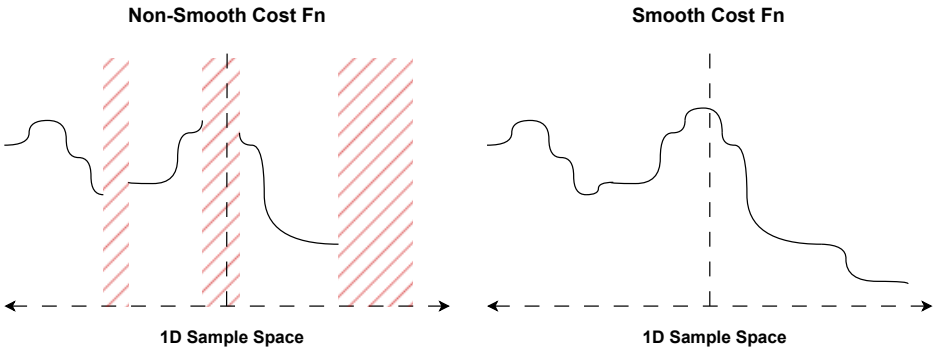


Figure 2.5: A 1D example of the non-smooth nature of GAN discriminators and the subsequent optimisation issues. The GAN discriminator function is shown on the left. The hatched regions indicate parts of the sample space that the discriminator identifies as not being from the data distribution. For these regions, $d(x) = 0$ and consequently $\nabla_\theta d(x) = 0$. This means that a function approximator parameterised by θ can not obtain informative gradients in the hatched regions, thereby failing to learn properly.

These limitations in GAN-based learning algorithms could be addressed by a different choice of generative model. The desired generative model must be capable of providing smooth reward functions that accurately incentivise the agent to learn policies that imitate all styles of motions in the expert dataset. The model must also be stable to train and capable of conditioning. The following chapter discusses energy-based models that seem to check all these requirements.

2.3. Energy Based Models

Energy-based models (EBMs) are based on a probability density modelling framework that defines the probability distribution of a random variable x through the Boltzmann distribution [41], [47]. Under such a framework

$$P(x) = \frac{\exp(-E(x))}{Z}$$

where $E(x)$ is the energy function and Z is known as the partition function (or normalising constant) that ensures that $\int P(x)dx = 1$. This requirement on Z is a source of challenge for most likelihood-based generative models. Most techniques balance a trade-off between model simplicity and expressiveness by varying the formulation of $E(x)$ and Z. For instance, generative models such as autoregressive or flow-based models attempt to learn the underlying data distribution through restricted but tractable (analytically computable) versions of the normalising constant, but in doing so also restrict their expressiveness and are only able

to capture simpler data distributions. Energy-based models attempt to maintain model expressiveness while also sidestepping the constraints imposed on Z. Instead of specifying a normalized probability, they only specify the unnormalized negative log probability (energy function). Since the energy function does not need to integrate to one, it can be parameterized with any nonlinear regression function [41].

While there are several procedures for training EBMs, this report mainly focuses on score-matching. The *score* is defined as the gradient of the log probability w.r.t the sample. Given

$$p_\theta(x) = \frac{\exp(-E_\theta(x))}{Z_\theta}$$

Where the learnt distribution p_θ is computed as a Boltzmann distribution over a learnt energy function E_θ and some normalising constant Z_θ that is not a function of x but only θ,

$$score = \nabla_x \log p_\theta(x) = \nabla_x \log \frac{\exp(-E_\theta(x))}{Z_\theta} = -\nabla_x E_\theta(x) - \cancelto{0}{\nabla_x \log Z_\theta} = -\nabla_x E_\theta(x)$$

As it turns out, defining the score as a gradient negates the need to learn an approximation for the normalising constant. Techniques like Langevin MCMC [54]–[56] use this fact to sample from the learnt distribution by first sampling from a simple tractable distribution (like a Gaussian distribution) and then following a schedule of iterative “denoising" by traversing along the gradient of the log probability (or as shown above, the negative gradient of the energy). Algorithm 1 shows a simple procedure to use Langevin MCMC to sample from a distribution determined by a learnt energy function $E_\theta()$

Algorithm 1 Sampling from $p_\theta()$ with Langevin MCMC [41]

Require: $score = \nabla_x \log p_\theta(x) = -\nabla_x E_\theta(x)$
Require: Noisy sample $x_0 \sim \mathcal{N}(0, I)$, number of timesteps K, step size ϵ
1: $x_k = x_0$
2: $k = 0$
3: **repeat**
4: $\quad x_{k+1} = x_k - \frac{\epsilon}{2}(-\nabla_{x_k} E_\theta(x_k))$
5: $\quad x_k = x_{k+1}$
6: $\quad k = k + 1$
7: **until** $k = K$
return x_k

While Langevin MCMC is a way of sampling from a complex, intractable distribution (represented by a learnt energy function in this case), score-matching is the procedure to train a network to learn that energy function. The theoretical basis of score-matching is as follows [41]. Given two real-valued functions $f(x)$ and $g(x)$, $f(x) = g(x) + const.$ if their first derivatives are the same for all values x (i.e. $\frac{d}{dx} f(x) = \frac{d}{dx} g(x)$). Think of one of these functions – say $f(x)$ –

as the learnt probability density function and the other as the data probability density function. This implies that $g(x)$ can be recovered by learning a function $f(x)$ whose first derivative is the same as $g(x)$. Score-matching is the procedure of learning the data distribution by matching the derivative of the log probability of a learnt function with that of the data. Given the definition of score from above, this simply amounts to learning to "match" the scores of the learnt and the data distribution. Given an unknown data probability density function p_{data}, we can draw i.i.d samples from this and learn a parameterised probability density function p_θ to minimise the objective

$$D_{Fisher}(p_{data}||p_\theta) = \mathbb{E}_{p_{data}}[\frac{1}{2}\left\|\nabla_x \log p_{data}(x) - \nabla_x \log p_\theta(x)\right\|_2^2]$$
$$D_{Fisher}(p_{data}||p_\theta) = \mathbb{E}_{p_{data}}[\frac{1}{2}\left\|\nabla_x \log p_{data}(x) - \nabla_x E_\theta(x)\right\|_2^2]$$

Even though $\nabla_x \log p_{data}(x)$ is unknown, under the assumption that $p_{data}(x)$ is continuous, differentiable, and tends to 0 as x tends to infinity, it can be shown that the Fisher divergence can be rewritten independent of the data distribution [41].

2.3.1. Denoising Score Matching & Diffusion

While the Fisher divergence can theoretically be shown to be independent of the data distribution, this requires the assumption that it is continuous and differentiable everywhere. This is surely not the case for a host of datasets and modalities that might be discrete and bounded in value or simply derived from discontinuous data distributions (given that datasets are sets of samples and the continuity of the unknown data density is not provable). This would render $\log p_{data}(x)$ discontinuous and its derivatives undefined. Further, the data density independent objective derived from the Fisher divergence has the downside of requiring the computation of second derivatives (a process that is quadratic in data dimensionality).

Denoising score matching (DSM) [57] is a modification of the default score-matching procedure that allows for the use of discontinuous data distributions that might not always be defined everywhere, without requiring the computation of second derivatives. The issues with discontinuity are resolved by the addition of a small amount of noise vectors ϵ sampled from a continuous noise distribution (normally Gaussian) to the data samples. The resulting distribution is $q(\tilde{x})$

$$\tilde{x} = x + \epsilon$$
$$q(\tilde{x}) = \int q(\tilde{x}|x)\, p_{data}(x)\, dx$$

The Fisher divergence objective is then modified to be

$$D(q(\tilde{x})||p_\theta(\tilde{x})) = \mathbb{E}_{q(\tilde{x})}[\frac{1}{2}\left\|\nabla_x \log q(\tilde{x}) - \nabla_x \log p_\theta(\tilde{x})\right\|_2^2]$$
$$D(q(\tilde{x})||p_\theta(\tilde{x})) = \mathbb{E}_{q(\tilde{x})}[\frac{1}{2}\left\|\nabla_x \log q(\tilde{x}) - \nabla_x E_\theta(\tilde{x})\right\|_2^2]$$

As a result of this, the p_{data} term is completely removed from the objective and the computation of second derivatives is negated.

Diffusion models [34], [58] are generative models that interweave Langevin MCMC and score-matching. Diffusion models can be thought of as denoising functions that convert simple, tractable distributions (like Gaussians) into complex data distributions by directly learning to predict the score (gradient of log probability) for samples from the tractable distribution. The samples from the tractable distribution are often viewed as “noisy" representations of samples from the data distribution and the score is viewed as a “denoising vector" that nudges the noisy sample towards the data distribution.

Diffusion models are trained in a two-step procedure, where samples from the data distribution are first distorted by the addition of noise (as per a pre-defined Markov Chain) to convert them to samples resembling the known, tractable distribution. In the second step, the noise added to the samples is matched with a prediction of the added noise (returned by the diffusion model) via denoising score matching. Thus, by applying score matching on a per-sample basis, a diffusion model learns to convert samples resembling random noise into samples from some unknown data distribution. In doing so, diffusion models also implicitly learn the energy function (since the score is just the negative gradient of the energy of a sample) but save the processing time of explicitly computing these gradients.

The artificially added noise is added as per a *schedule* following a forward Markov Chain using an iterative kernel $T(x_{k+1}|x_k, \beta_k)$ where β_k is the diffusion rate that defines the type of schedule used. Most recent works use a linear schedule as [58] also highlight that they did not notice stark differences compared to other choices of the schedule. Most diffusion models use a Markov kernel that converts the data distribution into an identity-covariance Gaussian distribution. In this case, it is also possible to directly obtain $x_k = f(x_0)$ without having to follow all steps in the Markov Chain by defining

$$\alpha_k = 1 - \beta_k$$
$$\bar{\alpha_k} = \Pi_{i=1}^{k} \alpha_k$$
$$x_k \sim \mathcal{N}(\sqrt{\bar{\alpha_k}} x_0, (1 - \alpha_k) I)$$

This greatly speeds up training time and most modern architectures also vary the timestep across samples in the training minibatch for stable optimisation via gradient descent. The reverse

process is to learn the transformation $T(x_k|x_{k+1}, \beta_k)$ by predicting the noise $\epsilon_k = \epsilon_\theta(x_0, k)$ given timestep k and a noise-free sample from the data distribution x_0. [36] also show that a "reparameterisation" trick can be used to learn the variances (dictated by β_k). However, it can also be assumed to be constant as a hyperparameter (given that it is kept "small" in value) [34]. Algorithm 2 shows the diffusion procedures for training and sampling from [34]. Figure 2.6 shows both the forward and reverse diffusion processes on the toy Swiss-roll dataset.

Algorithm 2 Diffusion: Training & Sampling [34]

Require: Data distribution p_{data} from which samples can be drawn, max steps in schedule K
Require: Score prediction network ϵ_θ parameterised by θ
procedure TRAINING
  repeat
    $x_0 \sim p_{data}()$
    $k \sim$ `Uniform(0,1,2... K)`
    $\epsilon \sim \mathcal{N}(0, I)$
    Predict noise vector (score) = $\epsilon_\theta(\sqrt{\bar{\alpha}_k}x_0, (1-\alpha_k)\epsilon, k)$
    Step using $\nabla_\theta \|\epsilon - score\|_2^2$
  until convergence
end procedure

procedure SAMPLING
  Sample $x_K \sim \mathcal{N}(0, I)$
  for k = K, K-1, .., 0 **do**
    $z \sim \mathcal{N}(0, I)$ if $k > 1$ else $z = 0$
    $x_{k-1} = \frac{1}{\sqrt{\alpha_k}}(x_k - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta(x_k, k)) + \sigma_k z$
  end for
return x_0
end procedure

2.4. Discussion: Comparing GANs & EBMs

This section evaluates the implications of energy-based generative modelling and contrasts the properties of diffusion models with the drawbacks of GANs seen in section 2.2.1. An inherent benefit of learning energy functions instead of direct mappings from the latent space to the sample space is the ability to run arbitrarily long inference loops. Given an energy function $E_\theta()$ the procedure from Algorithm 1 can be run with an arbitrarily large maximum timestep K to generate denoised samples with varying quality. This means that once trained, an EBM is quite flexible in its application and can be implemented on systems with varying compute characteristics. Under the limit of infinite time, Langevin MCMC is also known to generate samples truly from the data distribution [59]. Energy-based models also sidestep issues of constrained expressiveness that come with having to rely on direct mappings from the latent space to the sample space. Architectures like GANs and VAEs often fail to correctly map the whole latent space to multi-modal distributions with large disconnected regions in between modes (mode dropping). A reason for this could simply be the inherent complexity of learning a multimodal piecewise mapping in a large sample space. In contrast, EBMs map samples to

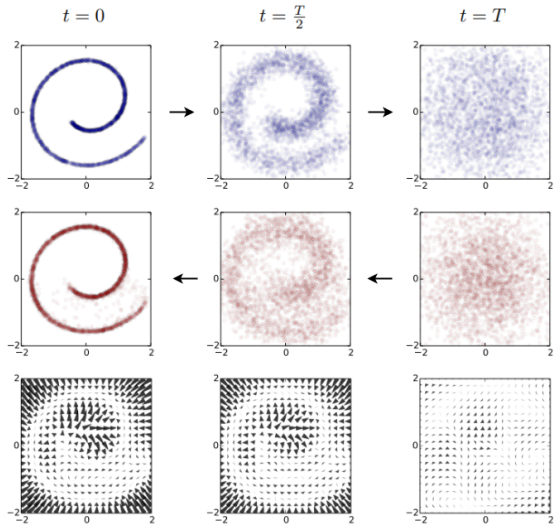


Figure 2.6: The application of a diffusion model on the Swiss roll dataset [58]. The top row shows the distribution at different points in the forward Markov Chain. The middle row shows the denoised distribution obtained by iteratively removing some amount of predicted noise from a Gaussian distribution. The third row shows a gradient field of the score (denoising vectors) predicted by the diffusion model

a scalar value and can easily represent disjoint distributions by assigning low energies [59]. Since EBMs do not require the simultaneous optimisation of multiple independent function approximators with different objectives, they also tend to be far more stable during training [48]–[50], [60]. [6] also perform large-scale comparative studies of multiple state-of-the-art GAN architectures to argue that many of the GAN performance improvements proposed in recent literature might be a result of the inherent variance in GAN training, hyperparameter sensitivity, and access to larger computational budgets than meaningful algorithmic changes.

EBMs (or score-based models like diffusion) also allow natural conditioning on arbitrary features that could define the characteristics of the generated sample. Diffusion for example – while already using the denoising timestep as conditioning – can condition its denoising prediction on features that define output characteristics. For example, [10], [61] present image generation diffusion models that use text prompt embeddings to condition the output of the model towards the desired mode in the data distribution. [53] also present a technique to further improve sample quality by probabilistically dropping the conditioning during training. From a robotics viewpoint, the ability to condition the generation process proves to be a rather important advantage and allows agents to dynamically vary the generated samples based on the current context. For example, a robot could generate trajectories of motions conditioned on the items that are held by the end-effector.

From the perspective of reinforcement learning, energy could also be interpreted as the "desirability" of that sample. Viewing energy functions as cost functions for optimisation could lead to interesting techniques for learning optimal control. EBMs also naturally offer composability [59] – viewing energies as costs, the summation of two energies evaluates to the summation of the goals implied by them. Further, being smooth in the sample space,

energy functions are also easier to optimise and promise better gradient-based learning. These characteristics make energy-based models quite enticing as a substitute for adversarial architectures in robotics tasks that require data generation.

3

Imitating Experts

3.1. The Need For Imitation Learning

As briefly introduced in the motivation section of this report, imitation learning is a field of machine learning that aims to teach robotic agents to mimic human experts. This kind of mimicry seems like a very natural way to teach robots crucial physical skills like locomotion, picking and manoeuvring everyday objects, interacting with tools etc. Much like reinforcement learning (section 1.3.3), the goal in imitation learning is to learn a policy that maps the agent's current state to a suitable action. In fact, quite like reinforcement learning, many popular imitation learning algorithms also learn such a policy by maximising the rewards given to the agent, with the added step of first formulating a suitable reward function that facilitates solving the required task. With so many similarities between the two frameworks, it is natural to wonder whether imitation learning is even worth pursuing. Can these physical skills not be learnt simply through reinforcement learning with a carefully designed reward function? The following few paragraphs differentiate imitation learning from reinforcement learning and present arguments in favour of reward learning.

As described in section 1.3.3, reinforcement learning is the problem of learning a policy for some "intended goal behaviour" by repeated interaction with an environment that provides a reward signal. [19] formalise this intended goal behaviour through the *reward hypothesis*, stating that the goals and purposes of a rational agent can be fully defined as the maximization of the expected value of the cumulative sum of rewards. While in theory, the reward hypothesis can be thought of as the foundational pillar upon which much of the formal analysis of reinforcement learning is developed, in practice, it is often quite challenging to learn the desired behaviour **only** through a well-designed reward function. This is the primary challenge in reinforcement learning for complex physical tasks.

First, the notion that reinforcement learning can converge to the desired behaviour rests on

the implicit assumption that the reward signal developed by the problem designer correctly captures the required motivation for an agent to learn the intended behaviour [1]. While designing a reward signal that provides the correct motivation for the intended behaviour might be trivial in simple problems, it is often a major challenge when trying to learn complex skills such as those required in the field of robotics. A major drawback of reinforcement learning is when the agent learns to maximise rewards through behaviour that was not intended by the problem designer [62], [63]. Imagine trying to motivate a robot to solve a maze by only providing a positive reward for forward progress. The agent can easily learn to hack this reward function by repeatedly making a small amount of forward progress, retracing its steps, and making the same forward progress again. This problem can be sidestepped if the "true" reward signal is instead something that the agent could learn through demonstrations of the intended behaviour – the fundamental idea of imitation learning.

Further, as the complexity of the problem increases, so do the possibilities for trial and error. Converging on sequences of actions that provide the highest expected return is a lot more challenging in problems involving large state and action spaces with tons of stochasticity – the salient features of most robotics tasks. While it is theoretically possible to asymptotically converge on an optimal policy purely through intermittent random exploration, it is far easier to do so given some kind of guidance from an expert [1]. This is especially enticing when **humans indeed can provide such guidance**. In most cases of robotic manipulation, it is likely that a human already knows how to solve the given task. This knowledge might not be as an explicit policy that maps states to actions, but it could be provided implicitly as a set of demonstrations. Leveraging this information is likely to simplify the learning process for robotic agents. In fact, [64] demonstrate both empirically and theoretically that methods that leverage such expert guidance can achieve faster and better solutions with less training data than their less-informed reinforcement learning counterparts.

In short, learning a policy from scratch (without any prior knowledge) is a rather challenging task. Even as humans, learning complex tasks such as tool operation, dance, acrobatics etc. often involves instructions or demonstrations from other expert humans. Imitation Learning can be thought of as a set of problems where the agent uses at least some amount of input from an expert or a demonstrator when attempting to learn a task [1], [65], [66]. The type and modality of input taken from the expert varies across methods but it is generally a set of demonstrations from the expert showcasing the intended way to complete a task.

Formally, imitation learning is defined as the set of problems where the learner's goal is to minimise some distance measure (`dist()`) between the probability distribution/density of sampling features ϕ from the expert's demonstrations, say $p_{\phi \sim \text{expert}}(\phi)$ and the probability distribution of sampling ϕ from the agent's learnt policy $p_{\phi \sim \text{agent-policy}}(\phi)$ [1]. The methods detailed in the following sections achieve the same objective in various ways.

$$\pi_{IL} = \arg\min_{\pi} \texttt{dist}(p_{\phi \sim \texttt{expert}}(\phi), p_{\phi \sim \pi}(\phi))$$

Imitation learning is generally formulated in either of the two following branches. One, as the problem of directly replicating the given demonstrations and two, as the problem of recovering the underlying reward function from the given expert demonstration and then optimally behaving under this reward function. The former family of methods is akin to *supervised learning* and is often known as *behaviour cloning*. The latter is commonly known as *inverse reinforcement learning (IRL)* followed by optimal control (via a host of methods). While this literature research primarily focuses on IRL, section 3.1.1 also provides a brief introduction to behaviour cloning and some challenges of directly replicating expert demonstrations.

3.1.1. Behaviour Cloning & Its Challenges

Behaviour Cloning (BC) [67] is a subfield of imitation learning where the agent aims to directly recover a policy $\pi(s)$ as a function of the agent's current state from a set of expert demonstrations of intended actions $a_E \in A$ in states $s \in S$. The dataset can be written as $\mathcal{D}\{(a_E, s)\}$. This is generally done via the standard supervised learning procedure.

It is important to note the fact that behaviour cloning operates under the premise that the learner has access to trajectories of the expert's policy that **include the actions that were executed by the expert**. This fact is highlighted here as it is one of the main shortcomings of this family of methods (more on partial observability in section 3.2). Behaviour cloning also assumes that the expert trajectories show optimal behaviour. Assuming that behaviour cloning can perfectly capture the expert's demonstrations, such a policy can in theory never outperform the expert. It is hence crucial that the expert demonstrations are indeed close to the behaviour that is actually desired in the system. Depending on the class of the BC algorithm, it might either be prone to errors caused by the *correspondence problem* or might fail to learn corrective behaviour (if the distribution of states from the expert dataset is not the same as the distribution of states encountered by the agent, non i.i.d). [68], [69] show that the worst-case performance gap between the expert and the imitator grows quadratically with the number of decision steps in the environment (given that the imitator has not perfectly captured the expert's action distribution, which is almost impossible to avoid). Hence, policies that are still close in terms of a supervised learning loss can be drastically different on application in an environment. Behaviour Cloning is often seen as a rudimentary first step in the problem of imitation and can be superseded in performance and sample efficiency by alternative approaches. For instance [70] present a BC-like method to imitation learning that simply uses reinforcement learning but augments the reward function to provide the imitator with additional incentive when taking actions that were also chosen by the expert. This simple addition already greatly improves over the basic BC formulation.

3.1.2. Imitation via Inverse Reinforcement Learning

The second major family of imitation learning algorithms employs a procedure to recover the unknown reward function that the expert trajectory seems to have maximised and then maximise the expectation over the return computed using this reward function (using an optimal control method) to obtain the learner's policy. The algorithms to recover the reward function are termed inverse reinforcement learning (IRL).

Formally, IRL makes the assumption that the expert is operating under the framework of a Markov Decision Process with the reward formulated as a function of the expert's trajectory $R(\tau)$ (since that is what the imitating agent has access to). The general outline of an IRL algorithm, as described in [1] is shown in Algorithm 3. These algorithms use parametric representations of the reward function and policy, keep track of the visitation frequency of state-action pairs following the learnt policy, and then match this with the trajectories found in the expert dataset through some optimisation objective. The policy and reward function is then updated iteratively with gradient-based optimisation.

Algorithm 3 General Feature Matching IRL Algorithm [1]

Require: Expert trajectories $\mathcal{D} = \{(\tau_i)\}_{i=1}^{N}$
Require: Parameterise the learnt reward function and imitator policy R, π with parameters w, θ
repeat
  Compute state-action visitation frequency μ for the current version of the policy π_θ
  Evaluate objective function $\mathcal{L}$ by comparing μ and the distribution implied by $\mathcal{D}$
  Compute the gradient of the objective in terms of the reward function parameters $\nabla_w \mathcal{L}$
  Update parameters w in the direction of the gradient
  Update parameters of π_θ using some RL algorithm
until until convergence criteria met
  return Reward function and imitator policy R_w, π_θ

3.2. Partial Observability: Imitation In The Absence Of Actions

When discussing imitation learning algorithms, it is important also to note their dependence on the modality of information available in the expert demonstrations. Until recently, it was a general assumption in most IRL studies that the imitating agent has access to all aspects of the expert's trajectory, i.e. the complete sequence of **states and actions**. However, recent works such as [4], [71] highlight the realities of most applications of imitation learning in robotics. It is quite often the case that the imitating agent simply can not have access to the specific actions executed by the expert.

First, in many of the cases where imitation learning might be an intuitive solution, the expert does not have an explicit representation of its policy. Imagine an expert surgeon trying to teach a robot assistant. The surgeon might be able to demonstrate high-precision manoeuvres as a set of tool trajectories (states) with ease, but it might not be possible for them to point out the exact torques and forces exerted at their joints (actions). Another interesting case of imitation

where action information could be unavailable is trying to learn a skill by viewing videos of an expert (with frames being the states). Secondly, even in situations where recording actions is technically possible, it might still pose additional constraints and challenges. One could argue that instead of moving their own body to teach the robot, the surgeon could instead just move the robot's arm. While this is possible on paper, it is both drastically slower and cumbersome than if the surgeon could simply demonstrate the motion themselves. It further requires the expert to have a certain level of skill in the process of demonstration (the surgeon would have to be comfortable with robots and have the skill to move the robot's arm in the same way as they would theirs). Finally, requiring actions in the expert trajectory freezes the possible embodiments of agents that could imitate the expert. The actions recorded by the surgeon on one kind of robot will probably not transfer over to another which has a different configuration of actuators. This is defined as the *correspondence problem* [66], [72]. The correspondence problem occurs when the embodiments of the expert and the demonstrator are different. It could also occur in situations where both have the same embodiment, but the trajectories learnt by the expert do not comply with the dynamics model of the imitating system. Algorithms that rely on the availability of actions would hence be heavily restricted in such real-world scenarios.

Apart from the above-mentioned practical considerations, [73] argue that in long-horizon situations, it might also be better to plan a sequence of states than actions. The actions can then be induced from these state trajectories by following gradient-free optimisation methods like CEM/CMA, or model-based control techniques like MPC/MPPI. In long-horizon tasks, they find action-based methods to produce only sub-optimal plans. This is also seen in recent imitation learning methods predicting action sequences like Diffusion Policy [74] where temporal inconsistency is mitigated by using receding horizon control (only applying a small fraction of predicted actions to the environment at each step). These insights are especially relevant to robotics which often involves long-running tasks where frequent replanning is computationally infeasible.

The inability to operate under the constraints of partial observability is a major shortcoming of a large portion of imitation learning methods in recent literature. Having identified this crucial gap, this report also focuses on achieving imitation learning under partial observability.

4

Imitation Learning Through Generative Modelling

Having discussed generative machine learning models (chapter 2) and imitation learning (chapter 3) in the previous chapters, the current chapter presents arguments in favour of their combination. Imitation learning algorithms stand to gain significantly by incorporating generative models like GANs within their reward learning sub-routine. This chapter draws parallels between the inner workings of generative modelling and imitation learning and discusses the potential benefits of their combination. The following sections then discuss some state-of-the-art generative imitation learning algorithms and draw attention to their benefits and shortcomings.

Generative models have the ability to induce probability distributions from large unstructured datasets to produce new data samples that, in theory, just seem to have come from the original unknown data distribution. To achieve this, generative models minimise the divergence between the data and learnt distributions. Similarly to generative models, imitation learning also aims to minimise the diverge between two probability distributions. In the case of IL, these are distributions of the agent's and the expert's motions. Viewing the demonstration motions as i.i.d samples from the expert (data) distribution, it follows that the underlying objectives of imitation learning are essentially the same as those of generative modelling [1]. The only distinction is that unlike generative modelling, where the goal is to just return new samples, the goal with imitation learning is to instead learn a policy that maps an agent's state to some desired action. Given the aligned objectives of these methods, it stands to reason that their combination might lead to newer techniques that benefit from their individual merits.

The key idea with generative imitation learning is to leverage the many advantages of generative objectives (sample diversity, better expressiveness etc.) to better guide an agent towards the intended policy. In usual circumstances, the probability distributions learnt by generative

models are not their direct output and are just used to sample the new data points that the model returns as outputs. While different generative techniques model the distributions differently (EBMs use energy functions while GANs use classifiers to iteratively update a sampling function), in all cases of generative modelling, the probability distributions inherently specify the desirability of the generated sample. The higher the probability of a sample, the more likely it is that this sample might have come from the data distribution. In the case where the data points are the expert's motions, the learnt probability distribution can be leveraged as cost functions that might drive a learning agent to take desirable actions. Hence, using generative objectives to learn reward functions, a learning algorithm could easily distil the motivations of the expert and the dynamics of the environment just from demonstrations, while having the added benefit of generating diverse samples. It could also be argued that these features make generative modelling based IL more sample-efficient and better at managing exploration. It must also be noted that the inclusion of generative models in the IRL pipeline does not impose additional restrictions on the other parts of IRL. In fact, recent works (section 4.8) also use these models to learn rewards and subsequently use traditional IRL approaches like maximum entropy/ maximum occupancy entropy to then learn policies from these rewards.

The following sections discuss key works in recent literature that attempt to use generative modelling techniques like GANs and EBMs within imitation learning. The general theme across all methods discussed here is to use a generative model either within the IRL pipeline shown in Algorithm 3 to learn a reward function or to instead use generative modelling to directly predict actions (similar to behaviour cloning). The next few sections discuss the algorithms, their theoretical foundation, and claimed benefits over the state-of-the-art. This is followed by a broader, comparative discussion (section 4.9) that highlights their limiting factors and some unexplored gaps that, if addressed, could improve their performance.

4.1. Generative Adversarial Imitation Learning (GAIL)

[2] introduce GAIL, a method to iteratively **learn** and **optimise the return from** the underlying reward function implied in expert demonstrations through an optimisation objective similar to that seen in *Generative Adversarial Networks* [35]. GAIL proposes to "directly" learn a policy instead of first learning a reward function and then optimising it (although internally it still does learn a reward function). While much of the theoretical work discussed in [2] is beyond the scope of this report, the main theoretical contributions are described below.

In general, IRL can be formulated as a procedure which finds a reward function under which the expert outperforms all other policies. Note that this does not directly imply that the expert's policy is optimal at solving the problem but instead defines IRL as a problem where a reward function is found under which the expert policy is the best. In this formulation, the reward function is defined to be regularised by a function ψ. An RL problem can then be defined to optimise the reward function returned by the inner IRL problem. [2] provide a different perspective on IRL by first defining a policy in terms of its occupancy measure (ρ_π) – the distribution of state-action pairs that an agent encounters when navigating the environment

with the policy – and then viewing IRL as a procedure that tries to induce a policy that matches the expert's occupancy measure. In doing so they show that various settings of ψ lead to various imitation learning algorithms with varying degrees of similarity between the expert and the imitator. With no reward regularisation, one can theoretically recover a policy that exactly matches the expert's occupancy measure. While this is enticing at first, it is not very practical as the expert dataset is finite and can not cover all possible ways of acting in a very large state-action space. They then show various examples of regularisation and its effect on the ability of the recovered policy to match the expert's occupancy measure.

Under this interpretation of IRL, [2] propose a new form of regularisation ψ under which the imitator policy accurately matches the expert's occupancy measure while also being tractable in large environments. The regulariser from GAIL is

$$\psi_{GA} = \begin{cases} \mathbb{E}_{\pi_E}[g(r(s,a))] & if r < 0 \\ +\infty & otherwise \end{cases}$$

where $r(s,a)$ is the cost function learnt via IRL and

$$g(x) = \begin{cases} -x - \log(1 - e^x) & if x < 0 \\ +\infty & otherwise \end{cases}$$

This regulariser places low penalties on reward functions that return high rewards to expert state-action pairs and heavily penalises those that assign low rewards to the expert. ψ_{GA} is also dependent on the expert dataset and is hence problem agnostic. Based on this definition, $\psi_{GA}(\rho_\pi - \rho_{\pi_E})$ equates to the Jensen Shannon divergence (section 1.3.1) between ρ_π, ρ_{π_E}. Minimising ψ_{GA} is roughly equivalent to minimising the difference between the imitator and expert. This interpretation of the regulariser is analogous to generative adversarial networks where a generator network G attempts to generate samples that "fool" a discriminatory D. In the case of GAIL, the learner's occupancy measure ρ_π is analogous to the distribution of the generator while the true data distribution is represented by the expert's occupancy measure ρ_{π_E}. GAIL essentially boils down to learning a discriminator (classifier) to distinguish between these two and finding a saddle point as a policy that minimises the classification error. The GAIL optimisation problem is (the causal entropy term is dropped as it does not change the theoretical or empirical results [4])

$$\min_{\pi \in \Pi} \max_{D \in (0,1)} \mathbb{E}_{s,a \sim \pi}[\log(D(s,a))] + \mathbb{E}_{s,a \sim \pi_E}[\log(1 - D(s,a))]$$

Intuitively GAIL simply uses the learnt discriminator as an implicit "reward" function and iterates between using RL to update the policy and updating the discriminator parameters by

maximising the difference between the distributions of the expert and imitator trajectories.

Algorithm 4 General Adversarial Imitation Learning [2]

Require: Expert trajectories $\tau_E \sim$ `unknown` π_E
Require: Parameterise the imitator policy and discriminator parameters by θ, w as π_θ, D_w
1: **repeat**
2: Sample trajectory $\tau \sim \pi_\theta$
3: Update D_w to maximise $\mathbb{E}_{s,a\in\tau}[\log(D(s,a))] + \mathbb{E}_{s,a\in\tau_E}[\log(1 - D(s,a))]$
4: Update π_θ via policy gradient algo (TRPO here) with reward $\log(D_w(s,a))$
5: **until** until convergence criteria met
return Imitator policy π_θ

4.2. Generative Adversarial Imitation from Observation (GAIfO)

As described in section 3.2, requiring actions in the expert dataset drastically reduces the applicability of imitation learning in real-world situations. [4] propose a modification on GAIL that does not rely on the actions executed by the expert to imitate behaviour. GAIfO uses state-only trajectories with a modified goal of not directly imitating the expert's actions but instead taking actions that have the same effect in the environment as the (unknown) actions that were executed by the expert. This modification means that reward functions are learnt not as a function of states and actions but as a function of state transitions. [4] posit that the desirable state transitions seen in expert demonstrations form a low dimensional manifold in the $S \times S$ space and the reward functions learnt via GAIfO penalise state transitions observed in imitator trajectories based on their closeness to this manifold.

Similar to GAIL, [4] show theoretically that performing reinforcement learning on a reward function learnt by inverse reinforcement learning from observation with regulariser ψ and unknown expert policy π_E, $RL \circ IRLfO_\psi(\pi_E)$ is equivalent to finding a policy that minimises the regulariser over the difference in the occupancy measures of the expert and imitator, $\arg\min_{\pi\in\Pi} \psi * (\rho_\pi - \rho_{\pi_E})$.

They then define a regulariser that is quite similar to the one from GAIL but instead uses a reward function on state transitions

$$\psi_{GA} = \begin{cases} \mathbb{E}_{\pi_E}[g(r(s,s'))] & if\, r < 0 \\ +\infty & otherwise \end{cases}$$

GAIfO ultimately amounts to the exact setup of GAIL but with a discriminator (classifier) that is instead a function of state transitions. Similarly to GAIL, here, the discriminator attempts to distinguish the source of the observed data to identify whether the given transitions are from expert actions in the environment or from a generator's (imitator policy) actions in the environment. This process can be viewed as attempting to learn an imitator that leads to the same state transitions as done by the expert. GAIfO works via the same procedure as

Algorithm 4 with the discriminator maximising

$$\mathbb{E}_{s,s' \in \tau}[\log(D(s,s'))] + \mathbb{E}_{s,s \in \tau_E}[\log(1 - D(s,s'))]$$

and the imitator policy using $\log(D_w(s,s'))$ as a reward function. GAIL and GAIfO form the basis for the next few algorithms that have shown significant empirical results in imitating complex actions such as humanoid walking, running, and even some forms of combat.

4.3. Adversarial Inverse Reinforcement Learning (AIRL)

Adversarial Inverse Reinforcement Learning (AIRL) [75] is an algorithm that uses an adversarial procedure to learn explicitly formulated reward functions. AIRL claims to improve over methods like GAIL by explicitly learning a reward function (as a discriminator with a known functional form that depends on the policy and a value function [76]) and operating at action-level granularity (as opposed to GAIL which takes in trajectories), thus allowing use in more difficult problem settings. One of their main contributions is to also learn reward functions that are robust to changes in the underlying environment – thus facilitating "knowledge transfer". Their experiments show that AIRL achieves similar performance levels as GAIL on tasks that do not require knowledge transfer and outperforms other IRL methods by even larger margins. When compared with GAIL in an experimental setup with a larger variance between the demonstration environment and the inference environment, AIRL is also shown to outperform GAIL.

[75] claim that operating with complete trajectories (as done in GAIL), results in a high variance in the discriminator's estimates. Their main contribution is to then modify the discriminator as a function of single state-action pairs. In this modification, they propose to solve the issue of reward ambiguity. Reward ambiguity is the problem that for any given set of expert demonstrations, there is more than one optimal reward that describes the expert's motives. Given a set of rewards that denote the same expert motives but only differ in formulation [77]

$$\hat{r}(s,a,s') = r(s,a,s') + \gamma\Phi(s') + \Phi(s)$$

[75] argue that not all shaped rewards (shaped by some function $\Phi : S \rightarrow \mathcal{R}$) are robust to changes in the dynamics of the underlying MDP. This means that given two problems with the same reward function but different dynamics, a policy learnt to optimise the reward in one problem is not guaranteed to work well in the other. This implies that a reward function learnt via IRL in one problem also carries some influence of the dynamics of the problem. A change in the dynamics of the problem would result in the policy learnt from that reward function being suboptimal (even if the change only influences the dynamics and not the true reward function).

AIRL proposes a discriminator function that is disentangled from the environment dynamics by formulating it as a summation of a reward approximator g_θ and a shaping function h_ϕ. They show that the reward approximator can be formulated as a function of only the environment state, thereby disentangling it from the dynamics.

$$D_{\phi,\theta}(s,a,s') = \frac{\exp\left(f_{\phi,\theta}(s,a,s')\right)}{\exp\left(f_{\phi,\theta}(s,a,s')\right) + \pi(a|s)}$$
$$f_{\phi,\theta}(s,a,s') = g_\theta(s,a) + h_\phi(s') - h_\phi(s)$$

The final AIRL algorithm very closely resembles GAIL, however, due to the modified discriminator function, it is now capable of explicitly learning a reward formulation that is also uninfluenced by changes in the environment dynamics.

4.4. Adversarial Motion Priors (AMP)

Adversarial motion priors [3] is an imitation learning algorithm that combines goal-conditioned reinforcement learning with a GAIfO [4] style reward function learning scheme. AMP is built on the idea that while it is quite challenging to assign an objective for the “style" of motion that the imitator is supposed to learn, it is still straightforward to define higher-level objective functions like symmetry and effort. AMP learns this style reward by learning to discriminate (classify) motions in the expert’s dataset from those executed by the learner’s policy. Similarly to other adversarial imitation learning methods, AMP then trains the imitator via reinforcement learning using the classifier’s output as a reward function.

The key contributions of AMP are the incorporation of goal-conditioned RL within state-only adversarial imitation learning and the excellent empirical results shown on full-body humanoid motion imitation. Since AMP only uses reference motions (and not actual actions) executed by the expert, the style-reward learnt can also be generalised across different agent embodiments. In doing so, AMP also sidesteps the correspondence issue discussed in section 3.2.

Formally, AMP uses a linear combination of task reward $r^G(s,s',a,goal)$ and a style-reward $r^S(s,s')$ that is learnt as a discriminator $D(s,s')$ learnt through a modified GAIfO objective.

$$r(s,s',a,g) = w^G r^G(s,s',a,goal) + w^S r^S(s,s')$$
$$r^S(s,s') = \max[0, 1 - 0.25(D(s,s') - 1)^2]$$

The offset, scaling, and clipping are done to bind the learnt reward function between $[0,1]$ as [3] claim that this aids training. The discriminator is learnt by slightly modifying the GAIfO objective function to match the least squares GAN objective proposed by [78]. This is done to simplify the optimisation challenges that are typically seen in GAN-like training (discussed in

detail in section 2.2). The discriminator objective in AMP is shown below. Proximal policy optimisation is used within the algorithm to learn the final policy.

$$\arg\min_D \mathbb{E}_{s,s'\sim\tau_E}[(D(s,s')-1)^2] + \mathbb{E}_{s,s'\sim\tau_\pi}[(D(s,s')+1)^2]$$

4.5. Conditional Adversarial Latent Models (CALM)

[5] propose CALM, an adversarial imitation learning technique based on GAIfO (section 4.2) for learning lower-level motion policies that can then be directed using another learnt higher-level policy. To achieve this, CALM learns an encoder $E()$ to transform a series of state-only trajectories of joint locations (called motions M) into a latent representation z in some space Z. Simultaneously, it learns a policy as a decoder $\pi(a|s,z)$ to take in the latent space motion along with the current state of the agent s and return actions. This policy learns to decode a variety of motions in the same function (hence learning a multimodal motion distribution). It however does not learn the directionality of the motion. To influence the directionality of the motion, [5] propose to train another high-level policy to select the latent space vectors z based on which the policy generates motions. The selection of latent space vectors is done based on a user-given input (from which the encoder returns $\hat{z}$) that indicates the desired behaviour style. When combined, this system can produce motions such as “moving in a given direction while crouch-walking". The combination of these two learnt policies is handled by a finite state machine.

Similar to GAIfO, CALM also uses a state-only dataset and aims to minimise the Jensen-Shannon divergence between the policy distribution and the motion dataset distributions of state transitions, (s,s'). The discriminator D aims to minimise an objective similar to GAIfO that additionally uses $z = E(M)$ as conditioning.

$$\pi_{CALM} = \arg\min_\pi \mathbb{E}_{M\sim motiondataset}[D_{JS}(p_\pi(s,s'|z)||p_{dataset}(s,s'))]$$
$$\mathcal{L}_{decoder} = \mathbb{E}_{M\sim motiondataset}[\mathbb{E}_{(s,s')\sim p_{dataset}}[\log D(s,s'|z)] + \mathbb{E}_{(s,s')\sim p_\pi}[\log(1-D(s,s'|z))]]$$

The higher-level policy that provides directionality information is trained in a standard RL fashion with a reward for latent similarity. This function rewards latent space vectors similar to the user-requested direction and penalises those that are not. The final inference procedure is as follows

1. Given a direction $d*$, and a motion type M_i, the higher level policy returns $\hat{z}$
2. The lower level policy then returns $a = \pi(s,\hat{z})$

CALM claims to fix the issue of mode-dropping that is seen in GAIfO by using the two-step procedure of first training an encoder-decoder pair, explicitly conditioning the output of the

decoder on the latent space vector, and then using a separate higher-level policy for selecting a hyperplane in the latent space which depicts the user-desired motions. They posit that directly passing the outputs of an encoder to a decoder (policy) does not produce the desired motions as the encoder only learns an imperfect mapping from the motion space to the latent space. Instead, separately conditioning the decoder on a diverse set of latent space vectors forces it to reproduce a diverse set of motions from the original dataset.

4.6. Diffusion Policy

While algorithms in the previous few sections used GAN-like optimisation objectives to learn rewards, this and the next few methods employ diffusion models (section 2.3) to learn the underlying imitation signals. Diffusion policy [74] employs a procedure to directly learn a mapping from states to actions as a conditional denoising diffusion process (DDPM [34]). In doing so, they learn a policy in a behaviour cloning-like fashion (by matching the expert demonstration and policy distributions) but also maintain the generalisation capabilities of diffusion models. They propose to make two main modifications to DDPM that enable its use in robotics problems with visual state information. First, the data modality from DDPM is changed from images to robot action trajectories. Second, the denoising process is conditioned on the visual state information.

The diffusion model is trained to predict temporally consistent, and reactive sequences of actions while the predicted actions are applied in the environment via receding horizon control [79]. Concretely, at every timestep t, the diffusion model takes in the previous T_o timesteps of observations O_t and returns a sequence of actions of size T_p. From this sequence, the first T_a actions are actually applied to the environment. [74] claim that the use of sequences of observations and the application of the receding horizon principle encourages temporal consistency in actions as the next set of actions is always conditioned on the previous observation sequence. The conditioning of model outputs is naturally facilitated by diffusion models and the FiLM [80] layer is used to compute conditioned latent space samples of inputs from which the model learns $p(a_t|O_t)$. Once a diffusion model is trained, action sequences are generated by K steps of inference at each environment timestep t. The denoising update at each inference step is

$$a_t^{k-1} = \alpha(a_t^k - \gamma\epsilon_\theta(O_t, a_t^k, k) + \mathcal{N}(0, \sigma^2 I))$$

The paper presents varied experiments on both real and simulated environments to show that diffusion policy can learn multimodal action distributions better than other state-of-the-art methods, and is capable oh high-dimensional representation learning (predicting sequences of actions rather than single actions). They also show detailed comparisons of neural network architectures like CNNs and Transformers and propose general guidelines for visual encoder modelling. Diffusion policy is compared against Implicit Behaviour Cloning [81], a general

procedure to learn an energy function and then minimise this via a variety of procedures. Being an EBM, implicit behaviour cloning in theory must also possess the same advantages as diffusion. However, through empirical results, comparisons of learning curves, and theoretical argumentation, [74] show that using diffusion models leads to substantially more stable training than IBC. They argue that the main reason for the stability of diffusion policy is because diffusion models avoid the computation of the normalisation constant Z_θ – something that the energy-based training of IBC approximates through a sampling process.

4.7. SE(3) Diffusion Fields

[82] present a diffusion-based procedure to learn smooth cost functions for combined grasp and motion optimisation for tasks in robotic manipulation. The smoothness of a cost function refers to the ability to compute informative gradients for all points, from which a downstream motion optimisation problem can be solved. The main contributions of this work are to first present a modification to diffusion models that enables their use in the SE(3) lie group (as opposed to Euclidean space) and secondly to use this model to learn cost function for downstream optimisation tasks like 6-degree-of-freedom grasping. The learnt cost functions can also be combined with additional optimisation objectives due to the composability (section 2.4) offered by diffusion models. While the changes made to enable the use of diffusion models in the SE(3) lie group are out of scope for this report, it is useful to note that [82] formulate the diffusion model as an EBM to return energies as costs. The gradient of this energy w.r.t the input sample is then used to explicitly compute the denoising score which is used within a denoising score matching objective that they modify for SE(3). This two-part formulation of diffusion leads to an intuitive application of diffusion models as

1. Techniques for learning to denoise an input and thereby facilitating downstream sample generation via Langevin MCMC
2. Techniques for learning functions that are representative of the “quality" of an input and thereby find use as optimisation targets

4.8. Neural Density Imitation (NDI)

The section on GAIL (section 4.1) discusses how [2] define the policy in terms of its occupancy measure. The occupancy measure ρ_π is the distribution of state-action pairs induced by a policy. GAIL is an imitation learning framework that aims to match the occupancy measures of the expert and imitator policies via an adversarial training procedure, thereby learning an imitator policy that performs similarly to the expert.

Neural Density Imitation [83] is an imitation learning framework that uses the same idea but substitutes the adversarial objective with a non-adversarial one that provably maximises (a lower bound of) the reverse KL divergence. [83] reason that adversarial imitation learning suffers from training instability and poor convergence due to the various issues of the alternating min-max optimisation (section 2.2.1). They argue that adversarial imitation learning further

suffers because the generator (learnt policy) is updated via high-variance performance estimates that are obtained from rollouts in simulation.

Formally, the occupancy measure is defined as a function of the parameterised policy π_θ, and $p_{\theta,t:t+k}$, the joint distribution of state trajectories $\{s_t, s_{t+1}, ..., s_{t+k}\}$. Intuitively, it captures the frequency of visiting (s, a) if π_θ is run infinitely.

$$\rho_{\pi_\theta}(s, a) = \sum_{t=0}^{\infty} \gamma^t p_{\theta,t}(s)\pi_\theta(s, a)$$

In general, imitation learning minimises the divergence between the expert and imitator occupancy measures. [83] show that the reverse KL divergence is in turn dependent on the *entropy* of a policy, a term that ensures exploration (occupancy measure diversity). While GAIL and other adversarial methods aim to minimise $KL(\rho_{\pi_E}||\rho_{\pi_\theta})$ (or other divergence measures like the JS divergence), NDI uses a non-adversarial objective. The two main challenges that [83] address are

1. Since the expert policy is not explicitly known, ρ_{π_E} needs to be estimated from samples from the demonstration dataset. They use energy-based models for this. They propose to learn an energy-based model $q_\phi(s, a)$ to approximate ρ_{π_E} and argue that EBMs are appropriate since the reverse KL divergence is sufficiently minimised by unnormalised density estimates of the policy's occupancy measure (policy optimality is invariant to a constant change in the reward function). Hence, score-matching is used to directly estimate (Z is a non-factor due to the definition of the score section 2.3)

$$q_\phi(s, a) = \frac{\exp\left(-E_\phi(s, a)\right)}{Z}$$

2. Using non-adversarial objectives to maximise the entropy of the implicitly defined imitator policy. For this, they use maximum occupancy entropy RL (for entropy regularisation) [84], [85] with the reward function defined from the estimated policy occupancy measure $\log q_\phi(s, a) + misc.$.

4.9. Discussion: What Can We Learn?

Chapter 4 can be split into two halves, imitation learning algorithms that (i) derive from generative adversarial networks (section 4.1 - section 4.5) and (ii) derive from EBMs and particularly diffusion models (section 4.6 - section 4.8). The goal of this section is to briefly summarise their contributions, and highlight their relative strengths and weaknesses. Table 4.1 presents a comparison of these algorithms based on their technical characteristics.

Table 4.1: A comparison of different imitation learning methods that use generative modelling

Features	Underlying Generative Model	Required Data Modality	Learnt Function
IL Algorithm			
GAIL	GAN	s-a pairs	Implicitly learnt reward fn. + policy
GAIfO	GAN	s-s′ pairs	Implicitly learnt reward fn. + policy
AIRL	GAN	s-a pairs	Explicitly learnt reward fn. + policy
AMP	GAN	s-s′ pairs	Implicitly learnt reward fn. + policy
CALM	GAN	s-s′ pairs	Implicitly learnt reward fn. + low-level/high-level policy
Diffusion Policy	Diffusion	s-a pairs	Policy
SE(3) Diffusion	EBM + Score Matching	grasp poses	Reward fn. as learnt energy function + policy
NDI	EBM + Score Matching	s-a pairs	Reward fn. as energy function of occupancy measure + policy

Before diving into a detailed technical analysis of each of these algorithms, it is important to first identify the desired features of an imitation learning algorithm. The ideal generative modelling based IL algorithm

1. Has stable training characteristics
2. Is capable of learning diverse, multi-modal expert distributions or a policy that depicts multi-modal behaviour
3. Is admissible to the process of conditioning (thereby allowing some flexibility over the features of the generated motion)
4. Learns a smooth distribution that can provide informative gradients at all points in the sample space
5. Operates with partially observable demonstrations

Algorithms like GAIL, GAIfO, AMP, and CALM have shown markedly good empirical performance. It can be argued that Adversarial Motion Priors (AMP) is the best from this set because of the combination of learnt “style" rewards with goal-conditioned RL rewards obtained from the environment. This allows the agent to mimic the style of the expert motion while still separately optimising the goals implied by the environment’s reward function. In comparison to CALM, which also uses a very similar procedure, AMP is arguably better as it does not need to learn two independent policies which are then connected by another user-configured finite state machine. Finally, AMP also uses state-only demonstrations and is hence easier to apply in the real world.

Unfortunately, most of the GAN-derived algorithms (including AMP) fail to satisfy the requirements discussed above. The simultaneous min-max optimisation in these algorithms has been studied in detail in several past studies [35], [48]–[50] and is known to be quite unstable. The policy (generator) update in these algorithms suffers further instability as it requires the estimation of the performance measure by the computation of an expectation over complete trajectories – a high-variance process due to the inherent stochasticity of environments and the inability of the expert dataset to cover all possible trajectories in the trajectory space. The GAN-like optimisation objective also suffers from issues like mode-dropping that render the learnt distributions inadequate at generating diverse samples. It is further inadmissible to conditioning. When used as a reward function, the non-smooth discriminator also fails

to provide informative gradients from which a policy could be learnt. While AIRL does outperform GAIL in transfer learning scenarios, avoids some of the instability by using singular state-action pairs, and has a discriminator with a known functional form, it still uses the GAN-like training objective. AIRL is hence prone to mode-dropping and instability arising from the simultaneous min-max optimisation. GAIL and AIRL also require the actions taken by the expert and are restricted in their practical applicability.

In contrast, the energy-based density estimation in algorithms like Diffusion Policy, NDI, and SE(3)-Diffusion sidesteps the issues of instability and restricted output modality. It is also naturally admissible to conditioning on features that affect sample characteristics. More importantly, energy functions are smooth in the sample space. When used as reward functions they can provide informative gradients based on which a policy can be optimised. It is important to highlight that although Diffusion Policy uses complete trajectories of expert demonstrations (and subsequently also predicts a sequence of actions), it is not prone to instability arising from the variance in trajectories. This is because Diffusion Policy is a one-shot approach and does not involve subsequent reinforcement learning in the loop to optimise a policy. NDI avoids this kind of instability by working with single state-action-next state pairs instead of trajectories.

However, both Diffusion Policy and NDI require the expert's actions and are somewhat restricted in their practical applicability. Diffusion Policy can also be seen as a generative version of behaviour cloning as it aims to directly capture the distribution of expert trajectories and produces action sequences as a function of observation history (conditioning the diffusion process on observations). Under this interpretation, it can be argued that Diffusion Policy also suffers from the correspondence problem (section 3.2). Even though diffusion can generate diverse samples and the receding horizon control used in Diffusion Policy ensures some degree of temporal consistency, Diffusion Policy might still fail to reliably replicate the expert. The main challenge might be to learn corrective behaviour (section 3.1.1) once the agent has already slightly deviated from the trajectory recommended by the expert policy. This could be worsened if the demonstration distribution is not the same as the distribution of trajectories encountered by the agent.

In conclusion, this chapter performs a theoretical and practical comparison of several state-of-the-art imitation learning algorithms that employ generative modelling. From this, it appears that there is a mixed set of benefits and drawbacks to these methods and a single algorithm cannot claim to dominate others. A combination of the beneficial features of algorithms like Adversarial Motion Priors (partial observability, and reward combining) and Neural Density Imitation (energy-based reward learning and using individual s-a pairs instead of trajectories) might lead to a better imitation learning algorithm that does supersede the current state-of-the-art.

5

Conclusions & Future Work

This review explores the current state-of-the-art at the intersection of imitation learning and generative machine learning. It starts with an explanation of some fundamental concepts in statistical machine learning (section 1.3), discussing frameworks like reinforcement learning (RL), popular RL methods, and other gradient-free techniques commonly seen in the literature.

The main focus of this review begins with an explanation of generative modelling (chapter 2) and the two commonly seen types of modelling frameworks – likelihood-based and implicit. The characteristics of these two techniques are compared and some challenges like the tractability of the normalising constant are noted. This is followed by a detailed analysis of generative adversarial networks (GANs) and their problems such as training instability, mode-dropping, conditioning difficulties, and non-smoothness. The pertinence of these technical challenges to the field of imitation learning is also discussed. The discussion then moves on to energy-based models (EBMs) – an alternative approach to generative modelling. We review the theoretical background to EBMs, the benefits of modelling probability distributions through the unnormalised negative log probability, and the definition of the score. Further, techniques like score-matching and Langevin MCMC are discussed. Finally, we discuss diffusion models that can be viewed as a combination of score-based EBMs and Langevin MCMC. This is followed by a detailed comparison of the benefits of EBMs and some of the issues of GANs (and adversarial imitation learning) that were previously highlighted.

The discussion then moves on the imitation learning (IL) (chapter 3). We explain the intuition and reasoning behind the idea of leveraging expert demonstrations and then pose imitation learning as the problem of minimising the divergence between the learnt and expert probability distributions. Like the previous chapter, the two main families of imitation learning algorithms – behaviour cloning (BC) and inverse reinforcement learning (IRL) – are introduced and some challenges of BC are noted. The rest of this report focuses primarily on IRL. We then introduce some practical challenges in imitation learning and argue about the necessity of being compliant

with partially observable demonstrations. Having seen both generative modelling and IL, we then compare the similarity in their objectives (matching a learnt distribution to another from which i.i.d samples are available) and discuss the benefits of combining generative modelling and imitation learning.

Finally, the review presents some relevant works from recent literature (chapter 4) that achieve this. We review a variety of methods that use both GANs and EBMs within their IL framework and explain their theoretical background and claimed benefits. We then contrast their advantages and drawbacks and conclude that a single algorithm can not be shown to confidently dominate others. This leads to a proposal for a new algorithm that combines the best features of some of these methods.

To conclude, we look back at the research questions posed in section 1.1. Based on the discussions and analysis in the previous chapters, it can be argued that energy-based models such as diffusion models can indeed have a substantial impact on the current state-of-the-art generative imitation learning algorithms. The improvements could primarily come in the form of better stability and expressiveness, and smoother reward functions leading to faster (and potentially better) policy learning. The review also identifies some desirable features such as operation under partial observability and the use of entropy regularisation, that might improve the practical applicability of these methods. Considering the findings of this review, the following research direction is recommended for exploration via a thesis.

1. Propose a new generative imitation learning algorithm based on EBMs/diffusion. The use of EBMs ensures training stability and multi-modality. These features improve the learnt reward functions that in turn improve the policies learnt by the new algorithm.
2. Pose a theoretically sound reward formulation as a function of the learnt energy. Given the smoothness of the learnt energy function, this would allow the computation of informative gradients for optimising the policy. Can the energy function be seen as the unnormalised log probability of the occupancy measure of the policy [83]? Does using the occupancy measure as a reward function capture the motivations of the expert?
3. Study the difference between directly optimising the reward (perhaps with PPO/TRPO) and using additional exploration-inducing methods like maximum occupancy entropy with reward optimisation.
4. Ensure that the proposed algorithm operates under partial observability (given only a trajectory of state transition pairs). Since it is difficult to get accurate action data (as well as physical state information like forces and accelerations) in expert demonstrations of contact-rich tasks, the ability to work in such conditions is a crucial feature for the proposed algorithm.
5. Compare the performance of the proposed algorithm against baselines such as AMP, GAIL, NDI, and Diffusion Policy [2], [3], [74], [83]. Is there a performance reduction compared to algorithms that also have access to the expert's actions?

The field of imitation learning seems to be at a promising phase, given the large number of papers in recent literature. The rapid rise in generative modelling aided robotics research surely points towards exciting avenues for the use of highly skilled robots in all kinds of new areas. This literature review and the following thesis aim to contribute towards robots that might someday achieve the same level of dextrous manipulation that currently only has a place in science fiction.

References

[1] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[2] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[3] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–20, 2021.

[4] F. Torabi, G. Warnell, and P. Stone, "Generative adversarial imitation from observation," *arXiv preprint arXiv:1807.06158*, 2018.

[5] C. Tessler, Y. Kasten, Y. Guo, S. Mannor, G. Chechik, and X. B. Peng, "Calm: Conditional adversarial latent models for directable virtual characters," in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023, pp. 1–9.

[6] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are gans created equal? a large-scale study," *Advances in neural information processing systems*, vol. 31, 2018.

[7] M. Krenn, L. Buffoni, B. Coutinho, *et al.*, "Forecasting the future of artificial intelligence with machine learning-based link prediction in an exponentially growing knowledge network," *Nature Machine Intelligence*, vol. 5, no. 11, pp. 1326–1335, 2023.

[8] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Compute and energy consumption trends in deep learning inference," *arXiv preprint arXiv:2109.05472*, 2021.

[9] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Trends in ai inference energy consumption: Beyond the performance-vs-parameter laws of deep learning," *Sustainable Computing: Informatics and Systems*, vol. 38, p. 100 857, 2023.

[10] A. Ramesh, M. Pavlov, G. Goh, *et al.*, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8821–8831.

[11] Y. Liu, T. Han, S. Ma, *et al.*, "Summary of chatgpt-related research and perspective towards the future of large language models," *Meta-Radiology*, p. 100 017, 2023.

[12] N. Stiennon, L. Ouyang, J. Wu, *et al.*, "Learning to summarize with human feedback," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3008–3021, 2020.

[13] A. Van Wynsberghe, "Sustainable ai: Ai for sustainability and the sustainability of ai," *AI and Ethics*, vol. 1, no. 3, pp. 213–218, 2021.

[14] C.-J. Wu, R. Raghavendra, U. Gupta, *et al.*, "Sustainable ai: Environmental implications, challenges and opportunities," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 795–813, 2022.

[15] P. Dhar, "The carbon impact of artificial intelligence.," *Nat. Mach. Intell.*, vol. 2, no. 8, pp. 423–425, 2020.

[16] B. C. Stahl and D. Eke, "The ethics of chatgpt–exploring the ethical issues of an emerging technology," *International Journal of Information Management*, vol. 74, p. 102 700, 2024.

[17] O. Bendel, "Image synthesis from an ethical perspective," *AI & SOCIETY*, pp. 1–10, 2023.

[18] F. Kraemer, K. Van Overveld, and M. Peterson, "Is there an ethics of algorithms?" *Ethics and information technology*, vol. 13, pp. 251–260, 2011.

[19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[20] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[21] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[22] S. J. Russell and P. Norvig, *Artificial intelligence a modern approach*, 4th ed. London, 2010.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[24] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[25] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[26] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[28] M. Kobilarov, "Cross-entropy motion planning," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012.

[29] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "The cross-entropy method for optimization," in *Handbook of statistics*, vol. 31, Elsevier, 2013, pp. 35–59.

[30] N. Hansen, "The cma evolution strategy: A comparing review," *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pp. 75–102, 2006.

[31] K. Wampler and Z. Popović, "Optimal gait and form for animal locomotion," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, pp. 1–8, 2009.

[32] V. Gabillon, M. Ghavamzadeh, and B. Scherrer, "Approximate dynamic programming finally performs well in the game of tetris," *Advances in neural information processing systems*, vol. 26, 2013.

[33] I. Szita and A. Lörincz, "Learning tetris using the noisy cross-entropy method," *Neural computation*, vol. 18, no. 12, pp. 2936–2941, 2006.

[34] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[36] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[37] A. v. d. Oord, S. Dieleman, H. Zen, *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[38] M. Kumar, M. Babaeizadeh, D. Erhan, *et al.*, "Videoflow: A flow-based generative model for video," *arXiv preprint arXiv:1903.01434*, vol. 2, no. 5, p. 3, 2019.

[39] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *Advances in neural information processing systems*, vol. 29, 2016.

[40] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[41] Y. Song and D. P. Kingma, "How to train your energy-based models," *arXiv preprint arXiv:2101.03288*, 2021.

[42] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," *Predicting structured data*, vol. 1, no. 0, 2006.

[43] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 29–37.

[44] L. Weng, "Flow-based deep generative models," *lilianweng.github.io*, 2018. [Online]. Available: https://lilianweng.github.io/posts/2018-10-13-flow-models/.

[45] L. Weng, "From autoencoder to beta-vae," *lilianweng.github.io*, 2018. [Online]. Available: https://lilianweng.github.io/posts/2018-08-12-vae/.

[46] L. Weng, "What are diffusion models?" *lilianweng.github.io*, Jul. 2021. [Online]. Available: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

[47] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[48] D. Saxena and J. Cao, "Generative adversarial networks (gans) challenges, solutions, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–42, 2021.

[49] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of gans," *arXiv preprint arXiv:1705.07215*, 2017.

[50] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[51] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016.

[52] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[53] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[54] P. Langevin, "Sur la théorie du mouvement brownien," *Compt. Rendus*, vol. 146, pp. 530–533, 1908.

[55] G. Parisi, "Correlation functions and computer simulations," *Nuclear Physics B*, vol. 180, no. 3, pp. 378–384, 1981.

[56] U. Grenander and M. I. Miller, "Representations of knowledge in complex systems," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 56, no. 4, pp. 549–581, 1994.

[57] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural computation*, vol. 23, no. 7, pp. 1661–1674, 2011.

[58] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, PMLR, 2015, pp. 2256–2265.

[59] Y. Du and I. Mordatch, "Implicit generation and modeling with energy based models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[60] K. Kurach, M. Lučić, X. Zhai, M. Michalski, and S. Gelly, "A large-scale study on regularization and normalization in gans," in *International conference on machine learning*, PMLR, 2019, pp. 3581–3590.

[61] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[62] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[63] J. Skalse, N. Howe, D. Krasheninnikov, and D. Krueger, "Defining and characterizing reward gaming," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9460–9471, 2022.

[64] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell, "Deeply aggrevated: Differentiable imitation learning for sequential prediction," in *International conference on machine learning*, PMLR, 2017, pp. 3309–3318.

[65] S. Schaal, "Learning from demonstration," *Advances in neural information processing systems*, vol. 9, 1996.

[66] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[67] M. Bain and C. Sammut, "A framework for behavioural cloning.," in *Machine Intelligence 15*, 1995, pp. 103–129.

[68] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[69] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[70] S. Reddy, A. D. Dragan, and S. Levine, "Sqil: Imitation learning via reinforcement learning with sparse rewards," *arXiv preprint arXiv:1905.11108*, 2019.

[71] F. Torabi, G. Warnell, and P. Stone, "Recent advances in imitation learning from observation," *arXiv preprint arXiv:1905.13566*, 2019.

[72] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Survey: Robot programming by demonstration," Springrer, Tech. Rep., 2008.

[73] O. Rybkin, C. Zhu, A. Nagabandi, K. Daniilidis, I. Mordatch, and S. Levine, "Model-based reinforcement learning via latent-space collocation," in *International Conference on Machine Learning*, PMLR, 2021, pp. 9190–9201.

[74] C. Chi, S. Feng, Y. Du, *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *arXiv preprint arXiv:2303.04137*, 2023.

[75] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248*, 2017.

[76] C. Finn, P. Christiano, P. Abbeel, and S. Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," *arXiv preprint arXiv:1611.03852*, 2016.

[77] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, 1999, pp. 278–287.

[78] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2794–2802.

[79] D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," in *Proceedings of the 27th IEEE Conference on Decision and Control*, IEEE, 1988, pp. 464–465.

[80] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[81] P. Florence, C. Lynch, A. Zeng, *et al.*, "Implicit behavioral cloning," in *Conference on Robot Learning*, PMLR, 2022, pp. 158–168.

[82] J. Urain, N. Funk, J. Peters, and G. Chalvatzaki, "Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 5923–5930.

[83] K. Kim, A. Jindal, Y. Song, J. Song, Y. Sui, and S. Ermon, "Imitation with neural density models," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5360–5372, 2021.

[84] L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov, "Efficient exploration via state marginal matching," *arXiv preprint arXiv:1906.05274*, 2019.

[85] R. Islam, R. Seraj, P.-L. Bacon, and D. Precup, "Entropy regularization with discounted future state distribution in policy gradient methods," *arXiv preprint arXiv:1912.05104*, 2019.