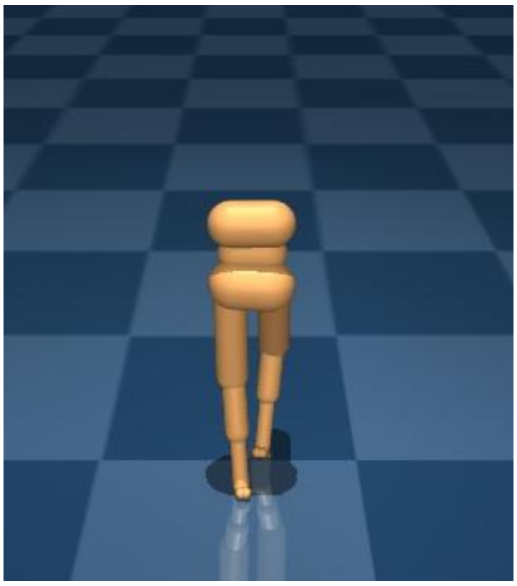
Emergence and Imitation of Locomotion in 2D and 3D Environments



Sahan Ayvaz

Semester Thesis
July 2018

Supervisors:
Emre Aksan
Prof. Dr. Otmar Hilliges

Abstract

Deep Reinforcement Learning (DRL) is an appealing approach to build robust controllers in continuous action spaces. Yet the pure DRL solutions usually yield unnatural gaits in locomotion. Imitation Learning provides a more applicable framework to build an intelligent agent with human-like locomotion. We implement the state-of-art methods in both approaches, Proximal Policy Optimization and Generative Adversarial Imitation Learning respectively, with the ultimate goal to produce such an agent with human-like walking style.

Contents

List of Figures

List of Tables

1 Introduction

Human locomotion is highly efficient, energy preserving and adaptive to multiple terrains. The combination of all those capabilities with high-dimensionality of human body exponentially increases the difficulty of building robust controllers for humanoid agents. The main objective of this semester thesis is to investigate possible solutions which could produce such an intelligent agent learning to move with human-like gaits on its own.

Reinforcement Learning provides a framework to formulate a goal-oriented learning where an agent could interact with its environment by taking actions to maximize its expected reward given by the environment. In recent years, using neural networks to approximate non-linear functions within RL framework, an approach known as Deep Reinforcement Learning, has demonstrated remarkable results for control of high dimensional systems.

Deep Reinforcement Learning has shown a rapid progress since its introduction, and various methods have been proposed. This semester thesis focuses on Actor-Critic Proximal Policy Optimization (PPO), the state-of-art policy gradient method. Chapter 2 gives a formal introduction to policy gradient methods and PPO.

Despite the effectiveness of methods relying on (Deep) Reinforcement Learning, their dependence on explicit reward functions could lead to insufficiently human-like behaviors especially in locomotion tasks. Designing hand-crafted reward functions to reflect human-like gaits requires a great care, yet the reward engineering is known to be brittle and could lead to unexpected results.

Imitation Learning allows an agent to learn the desired behaviors from expert demonstrations without the explicit knowledge of rewards. Since Imitation Learning removes the explicit dependence on reward functions, and thus avoid the challenging reward engineering, it could be utilized to produce human-like behaviors with the use of human demonstrations such as motion capture data of human walking.

There has been various proposed methods for imitation task, and this semester thesis uses the state-of-art method known as Generative Adversarial Imitation Learning (GAIL) where the rewards are inferred from the adversarial training between the expert and the imitator agent. Chapter 2 gives a formal introduction to GAIL and provides minor modifications to the originally proposed algorithm.

This semester thesis focuses on implementing and reproducing the results of PPO and GAIL for locomotion task, and aims to provide an extension to GAIL framework incorporating motion capture data of walking to produce an intelligent agent possessing similar human-like gaits for locomotion. The structure of thesis is as follows:

Chapter 2 formally introduces Reinforcement Learning by explaining mathematical preliminaries along with policy gradient methods. The chapter also discusses difference between model-free and model-based Reinforcement Learning, explains Proximal Policy Optimization, Actor-Critic method and Generative Adversarial Imitation Learning.In addition, it provides an overview of related work and core papers which are used to achieve the desired goal of this semester thesis.

Chapter 3 presents the reproduced and improved results for PPO and GAIL. The chapter also presents the results for a human-like walking agent, and discusses the reasons for the proposed modifications to the original papers.

Chapter 4 presents a summary and conclusion of this semester thesis with the additional discussion of possible future work.

Appendices provide additional experimental details including hyperparameters of the neural network architectures.

2

Background & Related Work

We will briefly overview the mathematical preliminaries of the core concepts and methods used in this semester thesis. We begin with the definition of Reinforcement Learning and explain Deep Reinforcement Learning. We also look at Policy Gradient methods, describe Proximal Policy Optimization with variance reduction technique called Generalized Advantage Estimation and concisely look at Actor-Critic methods. This chapter ends with the presentation of Imitation Learning and a short overview of Generative Adversarial Imitation Learning.

2.1. Reinforcement Learning

Reinforcement Learning is a branch of machine learning dealing with sequential decision making. The RL problem consists of an agent interacting with its environment where the agent acts upon and receives a scalar reward. The goal of the RL problem is to find an optimal policy mapping from states to actions which maximizes the expected discounted sum of rewards. The problem formulation is defined in terms of optimal control and Markov Decision Process. [Sutton and Barto 1998]

2.1.1. Markov Decision Process

A Markov Decision Process is defined by:

- A set of states, $s \in S$, where S denotes the state space
- A set of actions, $a \in A$, where A denotes the action space

- A scalar reward, r, and a reward function, $R : S \times S \times A \rightarrow \mathbb{R}$,

$$R(s, a, s') = \mathbb{E}(r_t | s_t = s, a_t = a, s_{t+1} = s')$$

- Transition probability

$$p(s'|s, a) = Pr(s_{t+1} = s'|s_t = s, a_t = a)$$

- Discount factor γ

The algorithms which assume the transition probabilities are known or could be approximated are *model-based* methods whereas the algorithms which does not assume the transition probabilities are known nor attempt to approximate them are *model-free* methods. This semester thesis focuses on *model-free* algorithms.

2.1.2. Policy and Value Functions

A policy function maps states (or observations) to actions. It could be deterministic or stochastic. This semester thesis focuses on stochastic policies, $\pi(a|s)$, such that a probability distribution over actions given states are assumed to be a multivariate Gaussian distribution. The discount factor is used to calculate the discounted sum of rewards. For a finite-horizon trajectory, $t = 0...T$, the discounted sum of rewards are calculated as following[1]

$$G_T = r_0 + \gamma r_1 + \gamma^2 r_2 + ... = \sum_{t=0}^{T} \gamma^t r_t$$

A value function, $V_\pi(s)$, predicts the expected discounted sum of rewards following a particular policy, π, from a particular state s.

$$V_\pi(s) = \mathbb{E}\big(\sum_{t=0}^{T} \gamma^t r_t | s_t = s\big)$$

Similarly, a Q-function, $Q_\pi(s, a)$, predicts the expected discounted sum of rewards following a particular policy, π, from a particular state s and action a.

$$Q_\pi(s, a) = \mathbb{E}\big(\sum_{t=0}^{T} \gamma^t r_t | s_t = s, a_t = a\big)$$

An advantage function, $A_\pi(s, a)$, is defined as

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$$

[1]Instead of using infinite-horizon formulations, we use finite-horizon formulations for both the discounted sum of rewards and value function.

2.1.3. Deep Reinforcement Learning

In the classical Reinforcement Learning setting, the policy and value functions are approximated by using linear functions. Deep Reinforcement Learning introduces non-linear functions to the policy and value functions by using neural networks. The non-linear functions generalize a larger subset of functions with better approximation especially for high dimensional problems. However, using non-linear function approximators no longer yields theoretical convergence guarantees as linear function approximators. DRL provides soft theoretical bounds on convergence, and for most problems, the introduced non-linear function approximators do not have a convergence problem in practice. [Duan et al. 2016]

2.2. Policy Gradient Methods

Policy Gradient methods [Sutton et al. 1999] directly try to optimize the policy function to maximize the expected discounted sum of rewards. We let τ to denote a state-action sequence $(s_0, a_0), (s_1, a_1), ..., (s_T, a_T)$ and θ the parameters (or weights) of the policy function. The expected sum of rewards for this sequence could be written as[2] [Abbeel and Schulman 2016]

$$U(\theta) = \mathbb{E}\big(\sum_{t=0}^{T} R(s_t, a_t)|\pi_\theta\big) = \sum_{\tau} P(\tau;\theta)R(\tau)$$

where $P(\tau;\theta)$ denotes the transition and policy probabilities parametrized by θ.

In this notation, our new objective is to find θ such that

$$\max_\theta U(\theta) = \max_\theta \sum_\tau P(\tau;\theta)R(\tau)$$

Taking the gradient with respect to θ and utilizing the derivative of logarithm trick (*) yields the following

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau) \\
&= \sum_\tau \nabla_\theta P(\tau;\theta)R(\tau) \\
&= \sum_\tau \frac{P(\tau;\theta)}{P(\tau;\theta)}\nabla_\theta P(\tau;\theta)R(\tau) \\
&= \sum_\tau P(\tau;\theta)\left(\frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)}\right)^{*} R(\tau) \\
&= \sum_\tau P(\tau;\theta)\nabla_\theta \log P(\tau;\theta)R(\tau) \\
&= \mathbb{E}\Big[\nabla_\theta \log P(\tau;\theta)R(\tau)\Big]
\end{aligned} \tag{2.1}$$

[2] We discard the discount factor to simplify the explanation; however, the discount factor could simply be added by multiplying the reward function with the discount factor.

We can empirically estimate the expectation by using m state-action sequences (also called trajectories). Therefore, we can write the empirical estimate for the gradient of $U(\theta)$ as

$$\nabla_\theta U(\theta) \approx \frac{1}{m} \sum_{i=0}^{m} \nabla_\theta \log P(\tau^{(i)};\theta) R(\tau^{(i)})$$

We then decompose the transition and policy probabilities.

$$\begin{aligned}
\nabla_\theta \log P(\tau^{(i)};\theta) &= \nabla_\theta \log \Bigg[\prod_{t=0}^{T} P(s_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)} \pi_\theta(a_t^{(i)}|s_t^{(i)}) \Bigg] \\
&= \nabla_\theta \Bigg[\sum_{t=0}^{T} \log P(s_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)} + \sum_{t=0}^{T} \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \Bigg] \\
&= \nabla_\theta \sum_{t=0}^{T} \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \\
&= \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})
\end{aligned} \tag{2.2}$$

Therefore, the empirical estimate for the gradient of $U(\theta)$ becomes independent of the transition probabilities.

$$\begin{aligned}
\nabla_\theta U(\theta) &\approx \frac{1}{m} \sum_{i=0}^{m} \Bigg[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) R(\tau^{(i)}) \Bigg] \\
&\approx \frac{1}{m} \sum_{i=0}^{m} \nabla_\theta \log \pi_\theta(\tau^{(i)}) R(\tau^{(i)})
\end{aligned} \tag{2.3}$$

This policy gradient method is known to suffer from variance; therefore, we can introduce an unbiased baseline independent of θ. [Abbeel and Schulman 2016]

$$\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \big[R(\tau^{(i)}) - b \big] \tag{2.4}$$

If we use our value function as our baseline,

$$\begin{aligned}
&\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \big[R(\tau^{(i)}) - V_\pi(\tau^{(i)}) \big] \\
&\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \Bigg[\sum_{k=t}^{T} R(s_k^{(i)}, a_k^{(i)}) - V_\pi(s_k^{(i)}) \Bigg]
\end{aligned} \tag{2.5}$$

We can see that

$$\frac{1}{m} \sum_{i=0}^{m} \sum_{k=t}^{T} R(s_k^{(i)}, a_k^{(i)}) = Q_\pi(s_k^{(i)}, a_k^{(i)})$$

and thus

$$\frac{1}{m}\sum_{i=0}^{m}\sum_{k=t}^{T} R(s_k^{(i)}, a_k^{(i)}) - V_\pi(s_k^{(i)}) = A_\pi(s_t^{(i)}, a_t^{(i)}) = \hat{A}_t$$

This formulation is called vanilla policy gradient method where the objective becomes

$$\max_\theta U(\theta) \approx \max_\theta \mathbb{E}\Big[\sum_{t=0}^{T} \log \pi_\theta(a_t|s_t)\hat{A}_t\Big] \tag{2.6}$$

2.2.1. Proximal Policy Optimization

The vanilla policy gradient might make large updates to the policy function, and this large step could worsen the policy optimization. [Schulman et al. 2015a] suggests using KL divergence between the old and new policy to define a trust region for the policy update step. They also argue that the following differentiations are equivalent because of the derivative of logarithm trick.

$$\mathbb{E}\Big[\sum_{t=0}^{T} \log \pi_\theta(a_t|s_t)\hat{A}_t\Big] = \mathbb{E}\Big[\sum_{t=0}^{T} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\Big] \tag{2.7}$$

Using this equivalence, they define a new objective with trust region optimization.

$$\begin{aligned} &\max_\theta \mathbb{E}\Big[\sum_{t=0}^{T} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\Big] \\ &\text{subject to } \mathbb{E}\big[KL[\pi_{\theta_{old}(|s_t)}|\pi_{\theta(|s_t)}]\big] < \delta \end{aligned} \tag{2.8}$$

Proximal Policy Optimization makes a first-order approximation to the constrain of the new objective and uses the KL divergence as a regularization term. [Schulman et al. 2017]

$$\max_\theta \mathbb{E}\Big[\sum_{t=0}^{T} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\Big] - \beta\mathbb{E}\big[KL[\pi_{\theta_{old}}(\cdot|s_t)|\pi_\theta(\cdot|s_t)]\big] \tag{2.9}$$

Even though the theory suggests using the KL divergence as a regularization term should work, choosing a fixed parameter β is not sufficient. Therefore, an adaptive β is introduced such that

$$\begin{aligned} &\text{compute } d = \mathbb{E}\big[KL[\pi_{\theta_{old}(|s_t)}|\pi_{\theta(|s_t)}]\big] \\ &\text{if } d < d_{targ}/1.5, \beta \leftarrow \beta/2 \\ &\text{if } d > d_{targ} \times 1.5, \beta \leftarrow \beta \times 2 \end{aligned} \tag{2.10}$$

2.2.2. Generalized Advantage Estimation

One of the main challenges of the policy gradient methods is the large number of samples required. [Schulman et al. 2015b] introduces an exponentially weighted estimator of the advantage function with the cost of some bias. This method is called Generalized Advantage Estimation, and can substantially improve empirical results on high dimensional DRL problems.

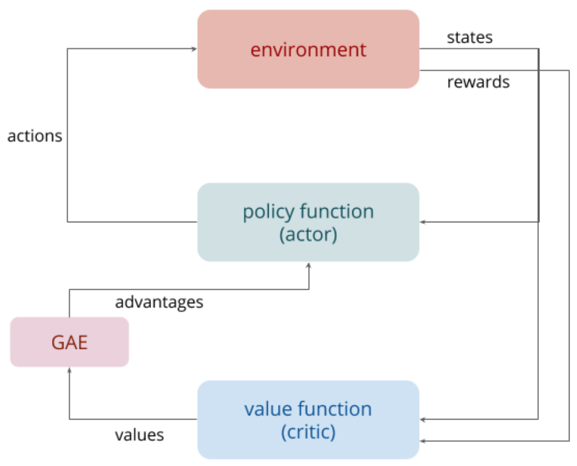


Figure 2.1.: *Actor-Critic Framework*

We first define the Temporal Difference (TD) residual of the value functions with the discount factor γ as $\delta_t^V == r_t + \gamma V(s_{t+1})V(s_t)$. We will denote the discounted $TD(k)$ residuals by $\hat{A_t}^{(k)}$.

$$\begin{aligned} \hat{A_t}^{(1)} &:= \delta_t^V && = r_t + \gamma V(s_{t+1}) - V(s_t) \\ \hat{A_t}^{(2)} &:= \delta_t^V + \gamma \delta_{t+1}^V && = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t) \\ \hat{A_t}^{(k)} &:= \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V \end{aligned} \tag{2.11}$$

The generalized advantage estimator GAE(γ, λ) for k steps is defined as [3]:

$$\begin{aligned} \hat{A_t}^{GAE(\gamma,\lambda)} &= (1-\lambda)(\hat{A_t}^{(1)} + \gamma \hat{A_t}^{(2)} + \gamma^2 \hat{A_t}^{(3)} + ...) \\ &= \sum_{l=0}^{k} (\gamma\lambda)^l \delta_{t+l}^V \end{aligned} \tag{2.12}$$

2.2.3. Actor-Critic PPO

Actor-Critic methods are a natural extension to RL problems consisting of various different implementations. In this semester thesis, the actor is the policy network, and the critic is the value network. The value network minimizes the Euclidean distance between the value estimates and the discounted sum of rewards. The main task of the value network is to predict the values more accurately, since the value estimates are used to calculate the advantages using GAE. The advantages are then fed into the PPO loss to update the policy network. This feeding loop between the actor and the critic continues while training, and could learn complicated stochastic policies. [Wang et al. 2016] Figure 2.1 shows the sketch of Actor-Critic framework used in our implementations.

[3]The complete derivation could be found in [Schulman et al. 2015b]

2.3. Generative Adversarial Imitation Learning

Imitation Learning is a broad set of algorithms which the learner agent tries to mimic the expert in order to achieve the same or similar behaviors. Generative Adversarial Imitation Learning is a model-free imitation learning algorithm which draws an analogy between imitation learning and generative adversarial networks. [Ho and Ermon 2016] In this section, we will only present the GAIL algorithm without proof, and comment on the general training routine for GAIL.

The main objective of the GAIL framework is to construct Inverse Reinforcement Learning problem as the dual of Reinforcement Learning. Inverse Reinforcement Learning tries to find the reward function of a trained expert policy. [Abbeel and Ng 2004] GAIL aims to avoid first training IRL objective to find the reward function, and then use this inferred reward function to train an agent with RL. Therefore, GAIL framework combines IRL and RL into the same formulation to obtain an efficient imitation learning solution especially for large, high dimensional environments.

In order to combine IRL and RL into the same formulation, [Ho and Ermon 2016] suggests using adversarial loss for training. As a result, the discriminator tries to discriminate if the **(state, action)** tuples are coming from the expert or the generator policy trajectories. In addition, the discriminator gives an estimate of the reward function which is used to update the policy function using a policy gradient method. The min-max game between the discriminator and the generator policy continues until convergence. In our implementation, we use GAIL with Actor-Critic PPO to update the policy network and estimate the values, and GAE to estimate advantages. Algorithm 1 summarizes our training approach.

```
1 Input: Expert trajectories $\tau_E \sim \pi_E$, initial policy, value and discriminator parameters
      $\theta_0, \omega_0, \phi_0$
2 for $i = 0$ to $n$ do
3     Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4     Obtain estimated trajectory rewards from the discriminator using
        $-log(1.0 - D_{\phi_i}(s,a))$
5     Calculate advantages $\hat{A}_\tau$ using GAE
6     Take a policy step from $\theta_i$ to $\theta_{i+1}$ using PPO by calculating the gradient of

        $$\mathbb{E}_{\tau_\square}\Big[\sum_{t=0}^{T}\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\Big] - \beta\mathbb{E}\big[KL[\pi_{\theta_{old}}(\cdot|s_t)|\pi_\theta(\cdot|s_t)]\big]$$

        to take a gradient step
7     Take a value step from $\omega_i$ to $\omega_{i+1}$ by minimizing the l2-loss between the discounted
        sum of estimated rewards and the predicted values
8     Update the discriminator parameters from $\phi_i$ to $\phi_{i+1}$ by calculating the gradient of

        $$\mathbb{E}_{\tau_i}\big[\log(D_\phi(s,a)) + \mathbb{E}_{\tau_E}\big[1 - \log(D_\phi(s,a))\big]$$

        to take a gradient step
  end
```

Algorithm 1: GAIL with Actor-Critic PPO and GAE

2.4. Related Work

In this section, we investigate the two core DeepMind papers which we build upon our ideas and implementations to build an intelligent agent with human-like locomotion. We remind that our core task is to build a robust controller for this desired agent in continuous action spaces using DRL. [Lillicrap et al. 2015]

2.4.1. Emergence of Locomotion in Rich Environments

[Heess et al. 2017] examines the effect of training in diverse environments to the complexity of learning behaviors even for simple reward functions such as forward progress.

In order to introduce the agents to the set of challenging environments, they make use of Curriculum Learning where the difficulty of the environment is incrementally increased to speed up the training. Curriculum Learning [Bengio et al. 2009] is a long-standing idea, yet combination of such diverse curricula with simple reward functions is first presented in this paper. We test the idea of Curriculum Learning with the simple reward of forward progress to build an agent completing obstacle courses in the next section.

[Heess et al. 2017] suggests that it is necessary to have a scalable reinforcement learning algorithm to be able to train the agents in rich and diverse environments. Therefore, the main contribution of the paper is the Distributed PPO. Although we do not train in distributed setting, we make use of the additional KL trust region constraint in our implementation of PPO. Based on our empirical results, we increase the penalty term from 2 to 4 though. The motivation behind early stopping policy update if $KL > 4 * d_{target}$ is to stabilize the training in policy epochs and to constraint the adaptive beta updates.

Since the agents are trained in environments with different terrains and obstacles, we can separate the policy network into two subnetworks. One of the subnetworks only receives the agent-related information, and the other one only receives the terrain-related information. [Heess et al. 2016] We also make use of this policy separation while testing for Curriculum Learning in the next chapter.

2.4.2. Learning Human Behaviors from Motion Capture by Adversarial Imitation

[Merel et al. 2017] shows that GAIL could sufficiently work by using only **states** instead of **(state, action)** tuples which the original GAIL framework uses. This is a very important result that leads to using the motion capture data for imitating human-like behaviors.

Apart from the extended GAIL using only **states**, [Merel et al. 2017] indeed follows the same desire as [Heess et al. 2017]. To build a complex behavior, we usually lack of good reward functions. Simple rewards could lead to complex behaviors if combined with a curriculum; however, producing a human-like behavior requires more than just curriculum. Instead of hand-crafting reward functions to obtain the desired behavior, they focus on imitation learning to infer rewards implicitly.

In addition, [Merel et al. 2017] explains a handful of insights to fix the unnatural gaits of the Humanoid agent. We make use of episode initializations from the motion capture data and get inspired by their discussion about feeding custom observations to the generator policy and the discriminator.

Most significantly, we acquire the same motion capture database (the CMU Motion Capture Database), and simplify their full Humanoid agent focusing only to the lower half.

3

Results

The previous chapter formally introduced Proximal Policy Optimization (PPO), Generalized Advantage Estimation (GAE) and Generative Adversarial Imitation Learning (GAIL), and presented the related work. In this chapter, we will first demonstrate the reproduced results for Actor-Critic PPO with GAE, Curriculum Learning and GAIL on simpler environments. We will then present an intelligent agent learning to walk like human using GAIL with motion capture data.

3.1. Proximal Policy Optimization

Despite fairly straightforward implementations of Actor-Critic PPO with GAE, the complexity of the system arising from the nature of the problem and the large numbers of hyperparameters suggests first to test the correctness of the implementations on a simpler environment. Walker2d-v2 (see Figure 3.1) is chosen as our testing environment for our implementations since it is very similar to an humanoid agent in terms of continuous action space yet only in 2D with a relatively low action space dimension. After we test the correctness of our PPO implementation on Walker2d-v2, we also investigate the possibility of using terrain information to complete an obstacle course with Curriculum Learning and perceptual encoding.

Our experiments are simulated in MuJoCo [Todorov et al. 2012] with environments built in OpenAI Gym [Brockman et al. 2016]. For pure RL tasks, we do not change the default reward function implemented in OpenAI Gym. Most importantly, we fix the environment seed to 0 for all experiments. Additional details about the Walker2d-v2 and Simplified Humanoid environments could be found at Appendix A1.

All of the policy networks used in this section are multi-layered perceptrons parameterizing

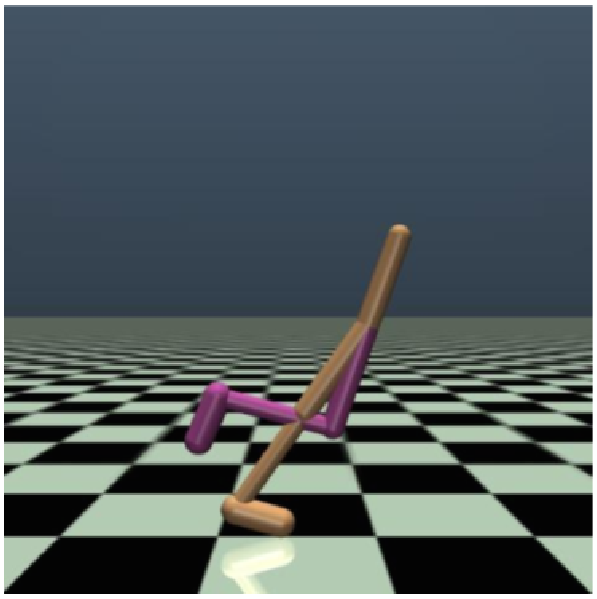


Figure 3.1.: *Walker2d-v2*

the means of a multivariate Gaussian distribution. We use diagonal covariance matrix with a separate set of parameters specifying the log standard deviations. Similarly, all of the value networks used in this section are multi-layered perceptrons minimizing the l2-norm between the predicted values and the discounted sum of rewards.

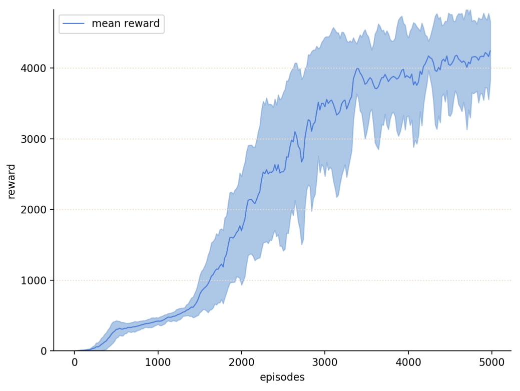
3.1.1. Testing Implementation on Walker2d-v2

Experiment Details

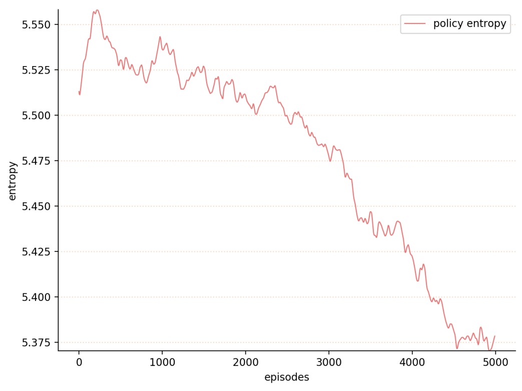
There are two proposed methods to collect **(state, action)** tuples to update the policy and value networks: using a fixed time horizon where the agent collects a fixed number of tuples versus using a fixed number of episodes where the agent collects a varying number of samples due to early termination in the early stages of training. The former method could help to stabilize training; however, the latter method could speed up training because of less training samples generated in the early stages. While testing the implementation of PPO on Walker2d-v2, we use the latter method by collecting 20 episodes per iteration.

While training, we normalize states by subtracting the mean and dividing by the standard deviation using the statistics aggregated over the course of entire experiment. The running mean and standard deviation are updated in every 20 episodes (per iteration). We scale rewards by the running standard deviation aggregated over the course of entire experiment. We also normalize advantages per batch where one batch consists of 20 episodes.

We use an experience replay buffer for the value function where the previous batch is concatenated to the current batch and the concatenated batch is fed into the network. The policy function only uses the current batch. Both policy and value functions shuffle their inputs before loss and gradient calculations.



(a)



(b)

Figure 3.2.: *Progress of Learning in PPO*
(a) 20-episode mean reward versus iteration (b) Policy entropy versus iteration

We use a simple reward function encouraging forward motion: $r = 1.0 + v_x + (1e-3)||u||$ where 1.0 is being alive bonus, v_x the linear velocity, and u the control inputs. We also early terminate if the height of the agent is less than 0.8 or greater than 2.0. The reward function and the early termination criterion are directly taken from OpenAI Gym.

Additional experiment details about network structures and hyperparameters can be found at Appendix A1.

Results

We update our policy and value networks for 250 iterations collecting 20 episodes per iteration (in total 5,000 episodes). Figure 3.2 (a) shows the progress of PPO by comparing 20-episode mean rewards with standard deviations against iterations.

Another important metric to check while training is policy entropy (Figure 3.2 (b)). A healthy PPO training should demonstrate a steady decrease in policy entropy.

The fully trained Walker2d-v2 agent is able to run forward as fast as possible; however, we observe that the running behavior does not show human-like gaits. (see video)

3.1.2. Curriculum Learning

Curriculum Learning is a meta-learning algorithm describing the process of learning complex tasks by iteratively increasing the environment difficulty. [Bengio et al. 2009]
[Karpathy and van de Panne 2012] In this section, we focus on completing obstacle courses with blocks of varying lengths requiring the Walker2d-v2 agent to learn running and jumping over blocks.

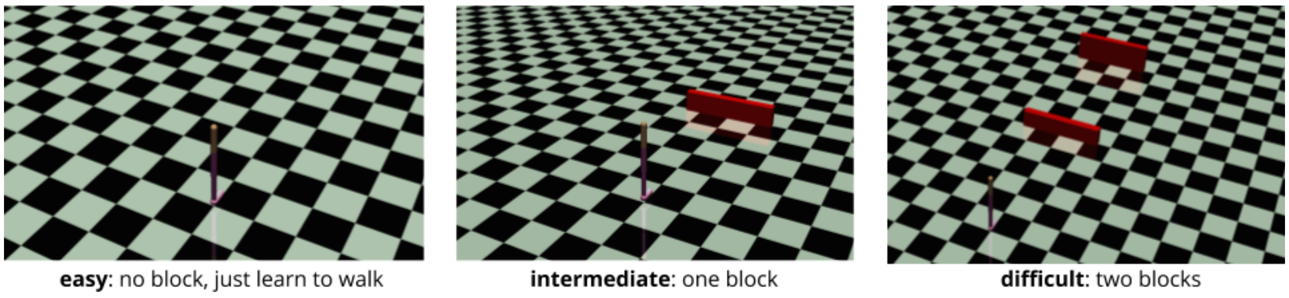

easy: no block, just learn to walk
intermediate: one block
difficult: two blocks

Figure 3.3.: *Sample of Curriculum Environments from easy to difficult*

Experiment Details

We procedurally generate environments consisting of different number of blocks with varying lengths. The agent begins training in an environment with no blocks to be able to learn running. The difficulty of the environment is increased as the agent progress within the curriculum from only one block to two blocks. Figure 3.3 shows a sample of curriculum environments used in training.

We use the same experiment details described in Experiment Details of the subsection 3.2.1 Testing Implementation on Walker2d-v2. In addition, we make minor modifications to the policy network to process the extra terrain information. The agent receives two sets of information: (1) the same set of egocentric features used in testing PPO, joint positions and velocities, (2) 50 equally spaced points of heightfield lying ahead of the agent [1]. We separate our policy network into two subnetworks where one of the subnetworks only receives the egocentric features while the other one only receives the terrain information. Similar to [Heess et al. 2016], the subnetwork receiving the terrain information acts as a perceptual encoder. The outputs of the subnetworks are concatenated, and processed by a final layer to obtain the means of a multivariate Gaussian distribution. The log variances are separately parametrized independent of the means.

Results

When the Walker2d-v2 agent completes the curriculum, it can run and jump over previously seen blocks with ease. (see video) Similar to the agent trained with PPO, however, running and jumping do not possess human-like gaits.

As an interesting side note, we also test the generalization ability of the agent to the blocks which it did not encounter during training. For the blocks with heights lower than the maximum height encountered during training), the agent is still able to complete obstacle courses with ease (see video showing an agent completing a 2-block course with block heights of 0.4); however, the agent shows partially successful attempts in environments with blocks with heights even slightly higher than the maximum height encountered during training. (see video showing an agent completing a 2-block course with block heights of 0.2 and 0.55)

[1][Heess et al. 2017] suggests using 10 points behind the agent and extending 40 points ahead; however, our experiments show that only giving information about points lying in front of the agent speed up training and produce similar results for our curriculum.

3.2. Generative Adversarial Imitation Learning

The originally proposed GAIL method uses **(state,action)** tuples to imitate the expert trajectories. While using motion capture data, on the other hand, the raw control actions are unknown. Therefore, we have to present that GAIL could be extended to the setting where only **states** are presented. Similar to the section 3.2 Proximal Policy Optimization, we first test our implementation of GAIL in the simple Walker2d-v2 environment with **(state, action)** tuples and then demonstrate in the same environment that **states** are sufficient to imitate the expert. Finally, we use motion capture data of human walking in Simplified Humanoid environment to produce an agent walking with human-like gaits.

3.2.1. Testing Implementation on Walker2d-v2

Experiment Details

We use the trained Walker2d-v2 agent as our expert, and collect different numbers (5, 10, 25 and 50) of trajectories consisting of **(state, action)** tuples from the stochastic expert policy. We also subsample the collected trajectories with subsampling rate of 20. Most significantly, the collected trajectories consists of unnormalized **(state, action)** tuples.

We feed into our generator policy the normalized **states**, whereas the discriminator receives the unnormalized **(state, action)** tuples from the generator and expert trajectories. The discriminator also receives the same number of generator and expert samples, and if necessary, expert samples are downsampled to enforce this constraint.

We perform a small sweep over policy and discriminator hyperparameters - learning rates for both policy and discriminator [3e-3, 3e-4, 3e-5], policy epochs [1, 5, 10] and discriminator epochs [1, 2, 5].

We update the policy with a reward proportional to $-log(1 - D(.))$ instead of the originally suggested $-log(D(.))$. [Merel et al. 2017] We do not normalize rewards obtained from the discriminator. In addition, compared to [Merel et al. 2017] and [Ho and Ermon 2016], we use PPO to perform policy updates instead of Trust Region Policy Optimization (TRPO). We first update our policy and value networks and then update our discriminator.

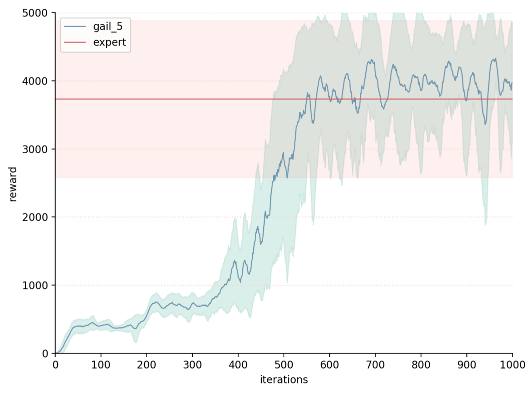
Additional experiment details about network structures and hyperparameters can be found at Appendix A2.

Results

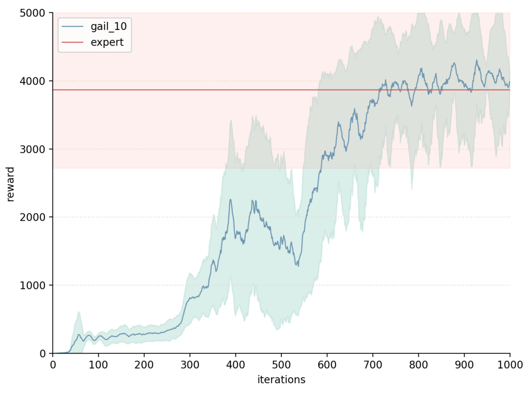
We update our policy, value and discriminator networks for 1000 iterations collecting 20 episodes per iterations. Figure 3.4 and 3.5 shows the training progress by comparing 20-episode mean true rewards [2] with standard deviations versus iterations for different number of trajectories.

Figure 3.4 (a) indicates that GAIL can successfully learn to imitate the expert even in the

[2]True rewards are obtained from the reward function used for training the Walker2d-v2 agent with PPO

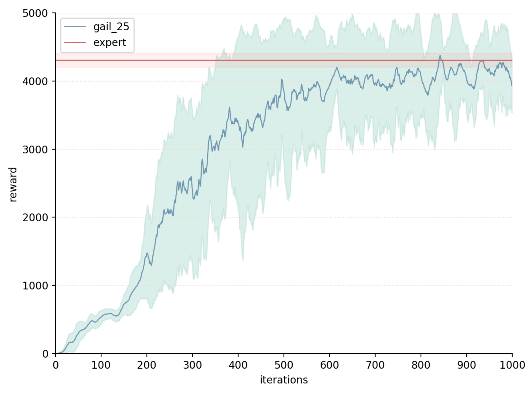


(a)

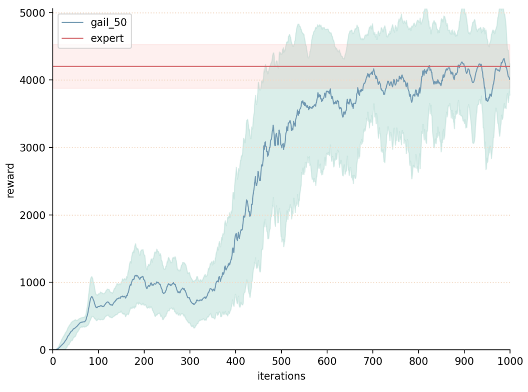


(b)

Figure 3.4.: *Progress of Learning in GAIL for scarce expert trajectories*
(a) 20-episode mean reward versus iteration for 5 expert trajectories (b) 20-episode mean reward versus iteration for 10 expert trajectories



(a)



(b)

Figure 3.5.: *Progress of Learning in GAIL for abundant expert trajectories*
(a) 20-episode mean reward versus iteration for 25 expert trajectories (b) 20-episode mean reward versus iteration for 50 expert trajectories

setting of scarce expert trajectories. The reproduced results demonstrating the sample efficiency of GAIL in terms of expert data also align with the results of the original paper. We also benchmark our GAIL results against Behavioral Cloning (BC). Comparison between BC and GAIL using different numbers of expert trajectories and additional details about BC can be found at Appendix A2.

We also visually compare the expert and generator policies to confirm that the generator displays similar characteristics in running. (see video showing a trained generator policy with 25 expert trajectories). The visual comparison between the expert and the generator suggests that GAIL could produce a generator policy possessing similar gaits with the expert.

3.2.2. Walker2d-v2 with only states

Experiment Details

We use the trained Walker2d-v2 agent as our expert, and only 50 trajectories consisting of **states** from the stochastic expert policy. Therefore, the comparison between the original GAIL with **(state, action)** tuples and the extended GAIL with **states** is only done for 50 expert trajectories.

We perform a small sweep over discriminator hyperparameters - learning rate [3e-4, 3e-5] and discriminator epochs [2, 5]. The rest of the experiment details employ the same setup described in Experiment Details of subsection 3.3.1 Testing Implementation on Walker2d-v2.

Results

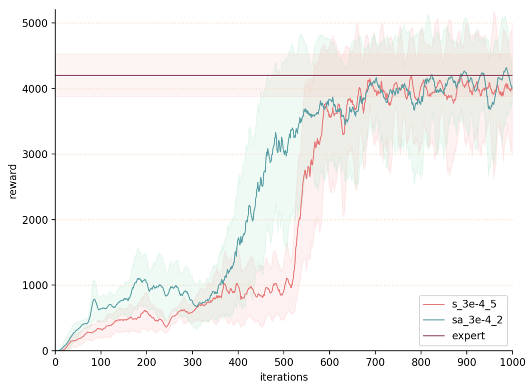
The properly tuned extended GAIL yields the same result in Figure 3.6 (a) as the original GAIL where feeding **(state, action)** tuples to the discriminator provides no benefit for training the generator policy.

We also visually compare the expert and the generator policy trained under the extended GAIL setting to confirm that the generator displays similar characteristics in running. (see video) The attached video suggests that the extended GAIL is able to produce visually similar gaits to the expert.

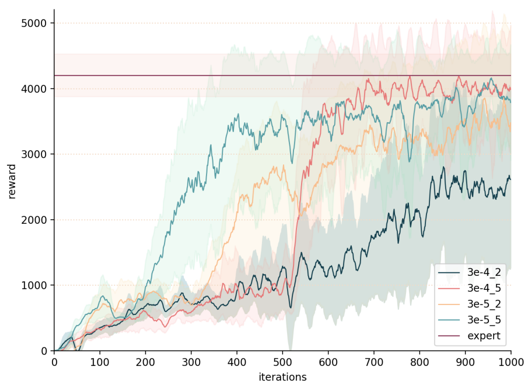
[Merel et al. 2017] interprets the success of the extended GAIL with deterministic inverse mapping between successive states to actions. This interpretation comes from the fact that the dynamics of the system are governed by second-order differential equations which the simulation environment MuJoCo utilizes to generate successive states. A further discussion of the dynamics of the model and the underlying second-order differential equations can be found at Section 5: Conclusion.

3.2.3. Human-Like Locomotion on Simplified Humanoid

Having validated our implementation of the extended GAIL on the Walker2d-v2 environment, we focus on applying the extended GAIL to Simplified Humanoid (Figure 3.7) with motion



(a)



(b)

Figure 3.6.: *The Extended GAIL with only states*
(a) Comparison between the original and extended GAIL using 50 expert trajectories
(b) Hyperparameter search for the extended GAIL

capture data to produce an agent with human-like walking style.

Simplified Humanoid is actually the truncated lowerhalf of the Humanoid CMU in DeepMind Control Suite [Tassa et al. 2018] excluding the toes. The Simplified Humanoid with 3D environment space and increased action space dimension still poses substantially complex problem compared to the Walker2d-v2.

Experiment Details

Up to now, we collect a fixed number of episodes to update the policy, value and discriminator networks. To stabilize the training and account for the high dimensionality of states and actions in the Simplified Humanoid environment, we collect a fixed time horizon consisting of 20,000 timesteps per iteration.

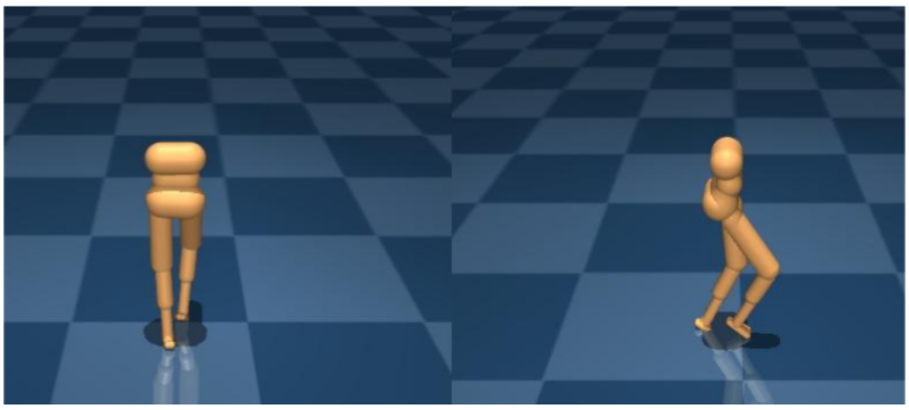


Figure 3.7.: *Simplified Humanoid*

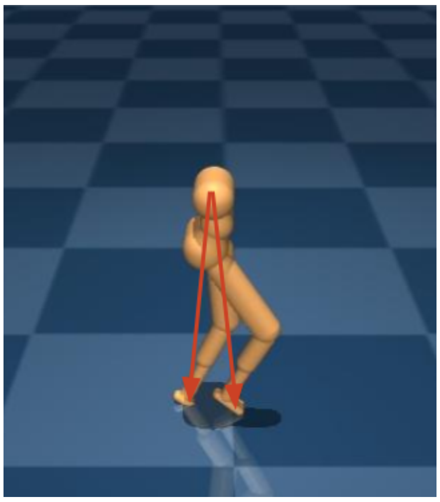


Figure 3.8.: *3D Extremity Vectors from Thorax to Feet in Red*

Similarly, up to now, states only involve the egocentric features of the Walker2d-v2 environment: the joint positions and velocities. States available to the generator policy consist of those normalized egocentric features of the Walker2d-v2 environment, and similarly the discriminator receives the unnormalized egocentric features of the expert and the generator policy. For Simplified Humanoid, we have two sets of state features: (1) full observations consisting of joint positions, joint velocities, body inertias, body linear velocities, actuator forces and external forces exerted by the floor, (2) egocentric observations consisting of height and orientation of thorax, center of mass position and velocity, torso orientation, sensor data of the root about velocity, orientation and acceleration, and 3D extremity vectors from thorax to feet (see Figure 3.8, for extremity vectors). We try different input pairs to the generator policy and the discriminator to achieve human-like walking.

We observe that randomly initialized poses of the Simplified Humanoid body lead to unnatural gaits. Therefore, we initialize the pose of the Simplified Humanoid body with a randomly sampled joint positions and velocities from the motion capture data.

The motion capture data come from The CMU Motion Capture Database Subject 8 clips 1-11 providing Acclaim Skeleton Files (ASF) and Acclaim Motion Capture (AMC). We use AMC parser provided by DeepMind Control Suite to extract joint positions and velocities from native AMC. We then load the extracted joint positions and velocities to the Simplified Humanoid environment fixing the joints with the extracted features. The full and egocentric observations are collected by simulating the Simplified Humanoid environment with the fixed joints.

Results

We first test the pure Deep Reinforcement Learning (DRL) solution to visually inspect the naturalness of the walking style. We make use of the normalized full observations feeding into the policy and value networks. As expected, the pure DRL solution shows insufficient human-like gait. (see video) In addition, we observe that the Simplified Humanoid body has a balancing problem, and thus the agent spends most of its training to find a balance instead of moving forward. Despite our attempt to simplify the Humanoid body by reducing the dimensionality, we create yet another difficulty to attain the DRL solution. Luckily, the balancing problem does not persist for the imitation task mainly because of initializing joint positions and velocities from the motion capture data.

video demonstrates the desired behavior of one of the motion capture clips.[3] To achieve a similar desired behavior, our first attempt at imitation learning made use of full observations same as our attempt for pure DRL. Both the generator policy and the discriminator receive the full observations. However, this naive application does not work well for imitation human-like style. (see video for representative result unsatisfactorily trained with the naive adversarial imitation)

[Merel et al. 2017] suggests using only egocentric observations for both of the networks; however, we cannot make this application work properly, and obtain similar results. Therefore, we decide to try different input combinations feeding into the generator policy and the discriminator to achieve the desired behavior.

Instead of using only egocentric observations, we append joint positions and velocities and create egocentric + joint information (egoplus) observations. We first attempt feeding the generator policy and the discriminator the egoplus observations. This application improves the visual quality slightly; however, it still does not result in human-like style. (see video)

We finally attempt to feed the generator policy and the discriminator different sets of observations. The motivation behind this comes from combining the pure DRL solution with invariant features suggested in []. Therefore, the generator policy receives only the full observations while the discriminator receives only the egoplus observations. The visual quality of the imitation improves, and the agent at least uses both of its legs to make forward progress. (see video)

Although the final attempt moderately improves the naive application, it is still not perfect. We observe that the jitter persists, and the control inputs (joint torques) seem to be overloading the actuators. We try to clip the control inputs to [-1, 1] in training and in testing. The clipped training result in failure (Figure 3.9) while the clipped testing does not improve the visual quality.

[Merel et al. 2017] accounts the jerky fashion to the noise in the motion capture data and dynamical differences between the simulated agent and the actual human body. We interpret the jerky fashion in two ways: (1) observing the jitter and the balancing problem in the pure DRL solution imply that the dynamical inconsistency of the Simplified Humanoid might compound errors affecting the simulation physics, (2) having two different suites utilizing MuJoCo (using DeepMind Control Suite to extract joint positions and velocities and OpenAI Gym to collect expert observations) could result in unexpected noise, since we have experienced some floating point precision errors while training.

[3]The CMU Motion Capture Database Subject 8 clip 1

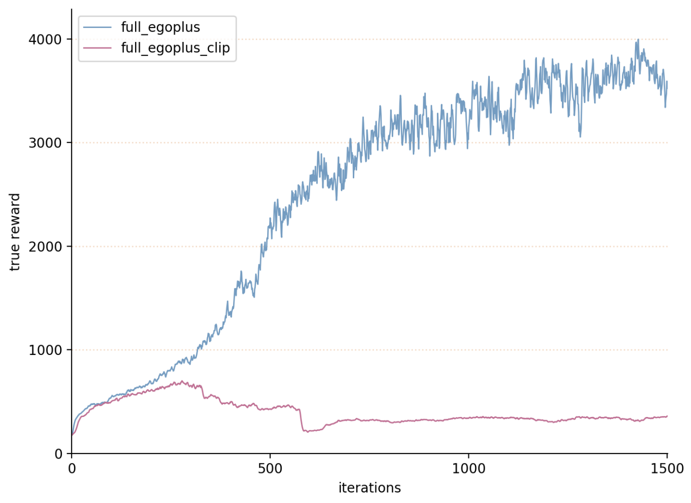


Figure 3.9.: *Comparison between clipped and unclipped training (Policy Input: Full Observations, Generator Input: Egoplus Observations)*

4 Conclusion and Outlook

This semester project investigates the state-of-art Deep Reinforcement and Imitation Learning methods to produce an intelligent agent learning to walk with similar human-like gaits. We implement Actor-Critic Proximal Policy Optimization with Generalized Advantage Estimation, and Generative Adversarial Imitation Learning. We also reproduce the results in the Walker2d-v2 environment to test the correctness of our implementations. After making sure that, our final training with the Simplified Humanoid results in human-like walking despite moderate unnaturalness in the gait.

While testing Proximal Policy Optimization, we examine the Curriculum Learning to build an agent capable of completing obstacle courses with running and jumping behaviors. In a possible future work, we want to combine the Curriculum Learning with similar obstacle courses and GAIL with motion capture data; therefore, we could produce an agent with diverse human-like behaviors capable of completing obstacle courses.

All of the DRL algorithms used in this semester thesis are model-free reinforcement learning. In DRL community, there is an increased interest in model-based methods. In model-based reinforcement learning setting, most approaches focus on defining the model with the transition probabilities, since the RL problem is formulated under Markov Decision Process. While testing for the extended GAIL using only states, however, we note that the second-order differential equations describe the dynamics of the system. Solving differential equations with neural networks have been studied lightly [Lagaris et al. 1998], [Ramuhalli et al. 2005] and as far as we know, approximating second-order differential equations to describe the model is still not attempted. In a possible future work, we could focus on solving differential equations to describe the model by using the successive states obtained from collected trajectories. Once we obtain the dynamics model, we could use cross-entropy method (CEM) to find a policy maximizing a reward-related objective such as the expectation of discounted sum of rewards. [Heyendaalseweg and Oliehoek 2008]

Apart from model-based RL, imitating directly from raw visual inputs has gained traction in both DRL and imitation learning communities. Context translation learning, for example, shows promising results for 2D-planar environments. [Liu et al. 2017] The one caveat of context translation is that the expert policies are collected from the same environment which the imitator is trained with changed contexts in birds-eye view. A more complicated problem is to imitate directly from raw visual inputs where the expert and the imitator do not share the environment. Most importantly, the problem could be defined as learning the imitation from real-life video clips of human locomotion. In a future work, this problem could be studied for 2D planar motions for the beginning, and hopefully for 3D motions in later stages.

Deep Reinforcement and Imitation Learning still have a quite large number of open problems waiting to be answered. We only glanced a couple of methods that gained popularity recently. Although some of our experiments did not yield satisfactory results, we will continue to test different proposed solutions and hopefully develop improved methods in the future.

A Appendix

A.1. Environment Details

Table A.1.: *Walker2d-v2 Details*

dim(S)	17
dim(A)	6
states	joint positions and velocities
actions	joint torques

Table A.2.: *Simplified Humanoid Details*

dim(Full Observations)	314
dim(Egocentric Observations)	26
dim(Egoplus Observations)	59
dim(A)	11
Full Observations	Joint Positions (16)
	Joint Velocities (17)
	Body Inertias (120)
	Body Linear Velocities (72)
	Actuator Forces (17)
	Forces exerted by the floor (72)
Egocentric Observations	Thorax Upright (1)
	Thorax Height (1)

	Center of Mass Position (3)
	Center of Mass Velocity (3)
	Torso Orientation (3)
	Sensordata (9)
	Extremities (6)
actions	joint torques

A.2. PPO Additional Details

Network Structures and Hyperparameters

Table A.3.: *PPO with Walker2d-v2*

using the same architecture for policy and value networks

number of hidden layers	2
number of hidden units	(100, 100)
nonlinearity	tanh
policy learning rate	3e-4
policy epochs	20
policy KL target	0.01
policy initial log variance	-1.0
value learning rate	1e-3
value epochs	10
value batch size	256

Table A.4.: *Curriculum Learning*

using the same hyperparameters as PPO with Walker2d-v2

policy hidden layers	2
policy hidden units	(300, 200)
policy nonlinearity	tanh
value hidden layers	2
value hidden units	(300, 200)
value nonlinearity	tanh
perceptual encoder hidden units	(30, 40, 100)
perceptual encoder nonlinearity	tanh

A.3. GAIL Additional Details

Table A.5.: *GAIL with Walker2d-v2*

using the same architecture for policy, value and discriminator networks
and the same network structure is used to train the extended GAIL

number of hidden layers	2
number of hidden units	(100, 100)
nonlinearity	tanh
policy learning rate	3e-4
policy epochs	5
policy KL target	0.01
policy initial log variance	-1.0
value learning rate	1e-3
value epochs	10
value batch size	256
discriminator learning rate	3e-4
discriminator epochs	2
Entropy regularization weight	1e-3

Table A.6.: *GAIL with Simplified Humanoid*

number of hidden layers for policy and value	3
number of hidden units for policy and value	$(10 \times dim(S), 10 * \sqrt{(dim(S) \times dim(A)}, 10 \times dim(A))$
nonlinearity for policy and value	tanh
policy learning rate	3e-5
policy epochs	5
policy KL target	0.01
policy initial log variance	-1.0
value learning rate	9e-4
value epochs	10
value batch size	256
number of hidden layers for discriminator	2
number of hidden units for discriminator	(300, 200)
nonlinearity for discriminator	tanh
discriminator learning rate	3e-4
discriminator epochs	2
Entropy regularization weight	1e-3

[1]

[1]The network hyperparameters are chosen according to the heuristics suggested in the blog post by Pat Coady

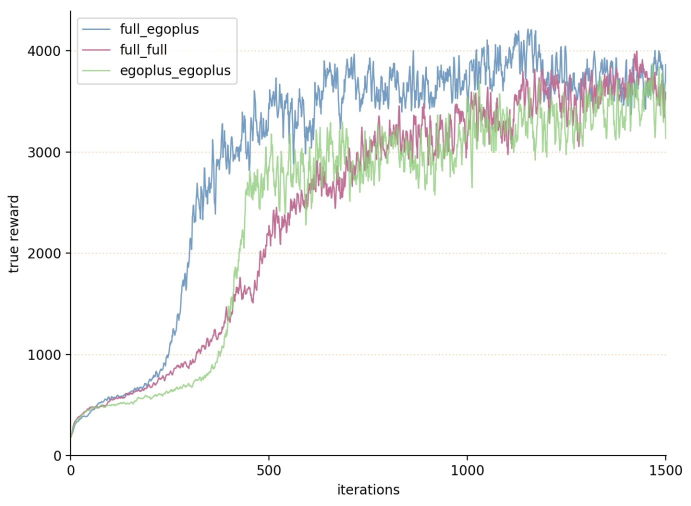


Figure A.1.: *Comparison between different input pairs to the generator and the discriminator for Simplified Humanoid respectively*

Comments on GAIL with Simplified Humanoid Results

Figure A.1 shows the training progress of the extended GAIL using different input pairs to the generator and the discriminator respectively. We observe that the training progress based on 20,000 timesteps mean true rewards does not correlate with visual similarity to the human-like motion. The lack of meaningful metric to measure this visual similarity causes inefficiency in training a successful policy, since we have to try many different models and check every single trained generator policy by visual inspection.

Behavioral Cloning

Behavioral Cloning (BC) is an appealingly simple supervised approach to imitation learning algorithm. The policy directly maps the states to the actions by minimizing l2-loss between the expert and the predicted actions. Despite its simplicity, BC works quite well if large amounts of data present. In addition to requiring large amounts of data, BC needs **(state,action)** tuples while a recently proposed model-based BC claims to use only actions. [ref] Compared to GAIL, BC does not interact with environment and perform online updates to different batch of trajectories in each iteration; therefore, BC usually has a faster and more stable training.

For our experiments, we use a deterministic policy to train BC. Our policy is a multi-layered perceptron with the hyperparameters shown in Table A.7

Table A.7.: *BC Hyperparameters*

number of hidden layers	2
number of hidden units	(100, 100)
nonlinearity	tanh

policy learning rate	1e-3
policy epochs	1
policy batch size	256
policy iterations	5000

Figure A.2 shows the comparison between BC and the original GAIL for Walker2d-v2 environment[2] where BC simply fails if limited amount of expert trajectories are presented. We additionally provide the training progress of BC versus iterations in Figure A.3

[2]Normalized scores are calculated by normalizing the maximum rewards of 20-episode mean true rewards with the mean of the expert

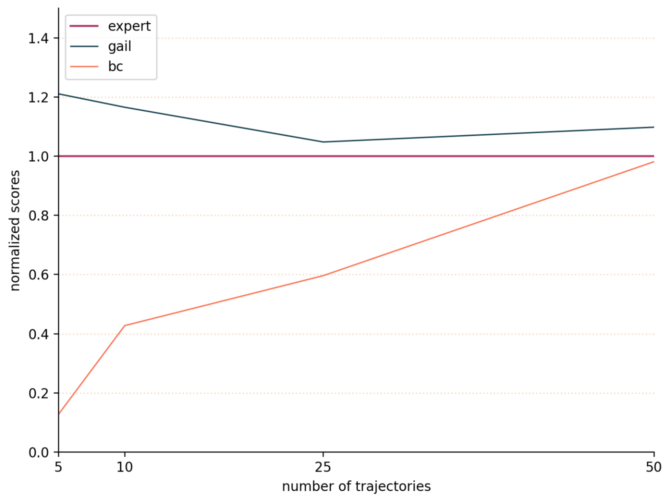
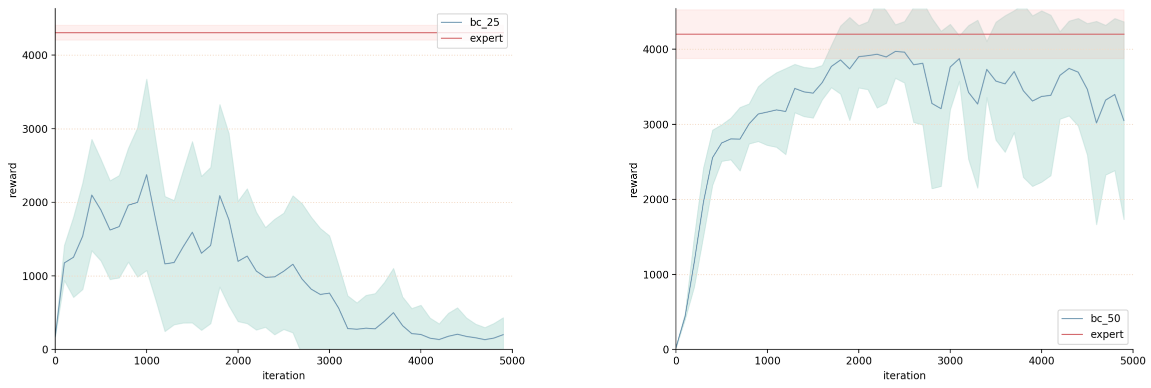


Figure A.2.: *Comparison between BC and GAIL using (5, 10, 25, and 50) trajectories*



(a) (b)

Figure A.3.: *BC training progress versus iterations*
(a) BC using 25 expert trajectories (b) BC using 50 expert trajectories

Bibliography

ABBEEL, P., AND NG, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *ICML*.

ABBEEL, P., AND SCHULMAN, J. 2016. Deep reinforcement learning through policy optimization. In *Tutorial at Neural Information Processing Systems*.

BENGIO, Y., LOURADOUR, J., COLLOBERT, R., AND WESTON, J. 2009. Curriculum learning. In *ICML*.

BROCKMAN, G., CHEUNG, V., PETTERSSON, L., SCHNEIDER, J., SCHULMAN, J., TANG, J., AND ZAREMBA, W. 2016. Openai gym. *CoRR abs/1606.01540*.

DUAN, Y., CHEN, X., HOUTHOOFT, R., SCHULMAN, J., AND ABBEEL, P. 2016. Benchmarking deep reinforcement learning for continuous control. In *ICML*.

HEESS, N., WAYNE, G., TASSA, Y., LILLICRAP, T. P., RIEDMILLER, M. A., AND SILVER, D. 2016. Learning and transfer of modulated locomotor controllers. *CoRR abs/1610.05182*.

HEESS, N., DHRUVA, T., SRIRAM, S., LEMMON, J., MEREL, J., WAYNE, G., TASSA, Y., EREZ, T., WANG, Z., ESLAMI, S. M. A., RIEDMILLER, M. A., AND SILVER, D. 2017. Emergence of locomotion behaviours in rich environments. *CoRR abs/1707.02286*.

HEYENDAALSEWEG, S. K., AND OLIEHOEK, S. A. F. A. 2008. Cross-entropy method for reinforcement learning.

HO, J., AND ERMON, S. 2016. Generative adversarial imitation learning. In *NIPS*.

KARPATHY, A., AND VAN DE PANNE, M. 2012. Curriculum learning for motor skills. In *Canadian Conference on AI*.

LAGARIS, I. E., LIKAS, A., AND FOTIADIS, D. I. 1998. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks 9 5*, 987–1000.

LILLICRAP, T. P., HUNT, J. J., PRITZEL, A., HEESS, N., EREZ, T., TASSA, Y., SILVER, D., AND WIERSTRA, D. 2015. Continuous control with deep reinforcement learning. *CoRR abs/1509.02971*.

LIU, Y., GUPTA, A., ABBEEL, P., AND LEVINE, S. 2017. Imitation from observation: Learning to imitate behaviors from raw video via context translation. *CoRR abs/1707.03374*.

MEREL, J., TASSA, Y., DHRUVA, T., SRINIVASAN, S., LEMMON, J., WANG, Z., WAYNE, G., AND HEESS, N. 2017. Learning human behaviors from motion capture by adversarial imitation. *CoRR abs/1707.02201*.

RAMUHALLI, P., UDPA, L., AND UDPA, S. S. 2005. Finite-element neural networks for solving differential equations. *IEEE Transactions on Neural Networks 16*, 1381–1392.

SCHULMAN, J., LEVINE, S., ABBEEL, P., JORDAN, M. I., AND MORITZ, P. 2015. Trust region policy optimization. In *ICML*.

SCHULMAN, J., MORITZ, P., LEVINE, S., JORDAN, M. I., AND ABBEEL, P. 2015.

High-dimensional continuous control using generalized advantage estimation. *CoRR abs/1506.02438*.

SCHULMAN, J., WOLSKI, F., DHARIWAL, P., RADFORD, A., AND KLIMOV, O. 2017. Proximal policy optimization algorithms. *CoRR abs/1707.06347*.

SUTTON, R. S., AND BARTO, A. G. 1998. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks 16*, 285–286.

SUTTON, R. S., MCALLESTER, D. A., SINGH, S. P., AND MANSOUR, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

TASSA, Y., DORON, Y., MULDAL, A., EREZ, T., LI, Y., DE LAS CASAS, D., BUDDEN, D., ABDOLMALEKI, A., MEREL, J., LEFRANCQ, A., LILLICRAP, T. P., AND RIEDMILLER, M. A. 2018. Deepmind control suite. *CoRR abs/1801.00690*.

TODOROV, E., EREZ, T., AND TASSA, Y. 2012. Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033.

WANG, Z., BAPST, V., HEESS, N., MNIH, V., MUNOS, R., KAVUKCUOGLU, K., AND DE FREITAS, N. 2016. Sample efficient actor-critic with experience replay. *CoRR abs/1611.01224*.