university of groningen / faculty of mathematics and natural sciences / artificial intelligence

# Training a Physics-Based Osseointegrated Transfemoral Amputee Model with a Reduced State Observation using Deep Reinforcement Learning

**Brown N. Ogum**
**s4056728**

A thesis presented for the degree of
Master of Science in Artificial Intelligence

# Training a Physics-Based Osseointegrated Transfemoral Amputee Model with a Reduced State Observation using Deep Reinforcement Learning

Brown N. Ogum
s4056728

Internal Supervisor(s): Prof. Dr. Raffaella Carloni
Prof. Dr. Lambert Schomaker
(Bernoulli Institute for Mathematics, Computer Science, and Artificial Intelligence, Faculty of Science and Engineering, University of Groningen)
August 31, 2022

**Artificial Intelligence / Human-Machine Communication**
**University of Groningen, The Netherlands**

# Contents

# Abstract

This project leverages a physics-based simulated environment (OpenSim) for the implementation of an intelligent control framework of an osseointegrated transfemoral amputee model, for the task of normal walking. A Deep Reinforcement Learning (DRL) algorithm - Proximal Policy Optimization (PPO) (with imitation learning) is used to optimize a policy for the walking task. Training the model in a simulation gives us access to additional muscle information which are not readily observable in the real world. The main aim of this research is to observe if a transfemoral amputee model can be trained to walk using Deep Reinforcement Learning while using a reduced number of state observers. To this end, a transfemoral amputee agent is trained to walk using the complete observation state - which contains kinematic data of the agent, including the force, length, and velocity of the agent's muscles. The agent is also trained by observing a reduced state. This reduced state only contains data that can readily be obtained in the real world with devices such as Inertia Measurement Units (IMUs) and rotary encoders. Hence, the muscle information is not included in this state representation. The effects that the lack of muscle forces and fiber length and velocity information have on the generation of gait are observed using three symmetry measures - RMSE, symmetry angle, and trend symmetry. Several Deep Neural Network architectures were trained in a supervised manner to predict the missing muscle information of the reduced state of the prosthesis model and their results were appraised using 5-fold cross-validation. It was observed that a fully-connected feed-forward neural network with 3 hidden layers had the best performance on the prediction task. Lastly, empirical results using an observation state that was augmented with the predicted muscle information showed that the transfemoral amputee model can be trained to walk using this framework with comparable rewards and symmetry to using the complete observation.

***Keywords*** – Deep Reinforcement learning, Transfemoral Amputee Model, Gait Symmetry, Feature Approximation

# Acknowledgements

This thesis has been supported by a number of people.

First, I want to extend tremendous gratitude to my supervisors. I would like to thank my first supervisor, Prof. Raffaella Carloni for her constant support and academic guidance in the course of this research. I also want to thank her for the computational support granted, to train the AI models. I'm extremely grateful to my second supervisor, Prof. Lambert Schomaker for providing useful feedback and suggestions on technical aspects of this thesis.

I am also thankful to members of the Robotics Research Lab - *Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence*, especially Vishal, Chandan, and Robin, for their contribution in discussions about this project.

I could not have performed this research without the ceaseless support of my parents, Prof. Daniel Ogum and Dr. Lovenda Ogum, and my caring siblings; Albert, Favour, and Victor.

Thank you Sarima, for all the love and support.

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Human gait refers to a person's manner of movement. It has to do with the pattern of movement of limbs during locomotion. The study of human gait has been of great importance for a multitude of reasons. Not only has it helped advice on ways to reduce the risk of injury during movement, but it has also helped in diagnosing issues and injuries based on an individual's manner of movement. Gait analysis has also provided a very detailed description of the complexities associated with bipedal locomotion. Humans have become so efficient at bipedal locomotion that it is easy while walking on an aisle, shopping for groceries, to forget just how complex the task of walking actually is. The brain coordinates full-body movements by sending excitation signals to the nervous system which activates or deactivates muscle groups; leading to excitation or relaxation of the muscle groups. Research has provided information on what muscle groups get activated during the different phases of the human gait cycle for prostheses and non-prostheses users (Huang & Ferris, 2011; Feger et al., 2015). Even with this knowledge, it will be an arduous task to decide on what muscles have to be contracted and which ones have to be relaxed in order to generate a normal gait pattern. This means that it would be a programmer's nightmare to hard-code a set of rules for gait generation, given the state of the individual. Luckily, we do not have to. Advancements in the field of Artificial Intelligence and Machine Learning, and in particular, Deep Reinforcement Learning, have made it possible to use the computational power of computers, as well as intelligent algorithms, to solve daunting problems such as the above-mentioned policy for gait generation. These algorithms train a policy to produce actions given the state of the model (the human in this case) in a way that maximizes a predefined reward system. They have a wide variety of applications from robotic automation and self-driving cars to personalized healthcare, and many more.

This research trains a simulated amputee model (Raveendranathan & Carloni, 2020) in Opensim (Delp et al., 2007) to have a regular human gait using a deep rein-

forcement learning algorithm - proximal policy optimization (PPO) Schulman et al. (2017). The trained agent is a transfemoral amputee model with a prosthetic limb osseointegrated into the left leg. De Vree & Carloni (2021) successfully trained a healthy human model to walk using PPO and imitation learning (Hussein et al., 2017). There has, however, not been much research done on training an osseointegrated transfemoral amputee agent for gait generation. Furthermore, there has been no research that the author could find which investigates the effects of training such an agent with a reduced observation space. This underlies the novelty and importance of this study.

In this research, comparisons of the quality of gait will be made for training the described amputee model with, and without access to its muscle information. An approach for handling the reduced observation is presented which approximates the muscle excitation of the amputee model.

## 1.1 Research Questions

This thesis is motivated by the following research questions:

- Can a transfemoral amputee model be trained to have a regular gait using Deep Reinforcement Learning without muscle information?
- Does muscle information approximation yield features that are useful for learning locomotion?
- What are the benefits (if any) of utilizing the augmentation of muscle information in producing a gait?

## 1.2 Significance of Study

This research is part of an ongoing project - MyLeg (Smart and intuitive osseointegrated transfemoral prosthesis embodying advanced dynamic behaviors) at the Bernoulli Institute's Robotics Research Lab. The contribution of this research is to train an osseointegrated transfemoral amputee model to mimic a natural human gait in a simulation whilst only having access to data that can be obtained using IMUs and encoders. This is particularly important for building the control architecture of a physical transfemoral prosthesis by helping the user achieve intuitive control of the prosthesis and providing a benchmark or an expectation of the robotic prosthetic limb's performance with the available data. To this effect, this study provides a way to achieve human-like movement by building an intelligent control system for a physics-based amputee model based on muscle contraction and relaxation, as well as the control of actuators on a prosthetic leg while not having access to muscle information (a reduced state observation).

## 1.3 Thesis Outline

The remainder of this thesis report is divided into the following chapters:

- **Theoretical Background**
  This chapter introduces the theory behind the methods used and other similar methods.

- **Research Methods**
  In this chapter, the experimental setup for the research carried out is described in detail. It explains the utilized tools which include the simulation environment and agent. The chapter also describes the experiments conducted, utilized architectures, and hyperparameters used. In addition, the algorithmic framework and implementation, as well as the measures of performance used in conducting this research are elucidated in this chapter.

- **Results**
  This chapter provides the results of the conducted experiments. The presented results include the obtained reward of the models as well as the symmetry measures for the amputee agent.

- **Discussion and Conclusion**
  This chapter will conclude the research paper by summarizing the research done in this thesis. It also discusses the implications of the obtained results and provides answers to the stated research questions and puts forward recommendations for future research within this line of research.

# Chapter 2

# Theoretical Background

## 2.1 Machine Learning

Machine learning is a rapidly growing field that has garnered a lot of interest due to its wide application domain and its high usage in our daily lives. It allows us to solve problems that would be incredibly complex to solve using rule-based programming. But what is machine learning? When can learning be said to have occurred in a machine? Mitchell (1997) provides a succinct definition of the term. According to him, a computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$.

Many kinds of tasks have been tackled using machine learning approaches. Some of the common groups of tasks in machine learning are listed below.

- Classification: this group of learning tasks requires the assignment of observations into one of several categories or classes. To solve a classification problem, the learning algorithm learns a function that maps a vector input to a categorical output. The classification task is quite common in machine learning. It can include various tasks from identifying objects and animals in images to identifying spam mails if the true values of the objects, animals, or mails are used in the learning process by the algorithm.

- Regression: these tasks usually deal with the prediction of a continuous numerical value, given some input vector. To solve this type of task, the learning algorithm learns a function $f : \mathbb{R}^n \to \mathbb{R}$. Some examples of regression tasks include estimation of housing price, stock price, a person's age, etc., given some relevant features.

- Transcription: these tasks require the machine learning algorithm to observe

an unstructured representation of data (e.g. audio or video recording, or image with text) and transcribe the data into a discrete textual form. Machine learning models have been trained to perform transcription tasks with a high degree of accuracy (Dahl et al., 2012; Hinton et al., 2012).

- Machine Translation: this learning task aims to train a machine learning system to be proficient at translating text from one language to another. Strides in Deep learning algorithms and systems capable of training such models have been especially beneficial in this group of learning tasks as it has helped machine learning experts achieve near human-like accuracy in this type of machine learning task

- Anomaly Detection: in this type of task, the learning algorithm is tasked with finding a solution that sifts through data and identifies unusual patterns that do not conform to expected behavior.

- Synthesis and sampling: in this type of learning problem, the learning algorithm is tasked with generating new examples that are similar to those in the training data.

The performance measure $P$ of a machine learning program gives insight as to how well the algorithm is learning. This measure has to be carefully chosen so that the algorithm learns to do what is actually intended by the developer. The performance metric utilized strongly depends on the machine learning task. For example, in a classification task, the accuracy of the classifier can be a suitable performance metric. In a regression task, the mean squared error can be a suitable error metric. However, in an unsupervised clustering task in which the ground truth is not known, such metrics might not be suitable.

Lastly, the experience $E$ denotes the information the learning algorithm is allowed to have in the training process. Does the learning algorithm make use of the target values (ground truth), or does it only go through a dataset and try to draw some inferences only using the available data? The learning algorithm could also gain experience by interacting with an environment. The gaining of experience is the basis of grouping learning algorithms into categories such as supervised, unsupervised, semi-supervised, and reinforcement learning. These categories of machine learning algorithms will be further discussed in the coming sections.

### 2.1.1 Unsupervised Learning

Unsupervised learning refers to machine learning methods that learn useful properties from unlabelled data sets. The lack of context in the training examples implies that a comparison of the learning algorithm's performance with the ground-truth is not possible. This is because there is no *teacher* to facilitate the evaluation of the

system's performance. Rather than having a *teacher* capable of denoting the ground truth given an example, unsupervised learning makes use of a task-independent measure of the quality of the representation to be learned (Becker, 1991). Unlike supervised learning, unsupervised learning methods cannot be directly applied to a regression or classification problem as one has no idea of what the values of the output may be. Hence, the main goal in unsupervised learning is to discover hidden and interesting patterns in unlabelled data (El Bouchefry & de Souza, 2020). The goal of unsupervised learning tasks could be to discover groups of similar examples within a data set (clustering), or to determine the distribution of data within the input space; known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization (Bishop, 2006).

### 2.1.2 Supervised Learning

Supervised learning algorithms experience data sets that have input vectors along with their corresponding target values. These machine learning methods analyze a data set of examples and their target values in order to make predictions on novel examples. Supervised learning is also referred to as learning with a teacher. Conceptually, the teacher can be thought of as having knowledge of the environment. This environment is, however, unknown to the learning model. By the virtue of this knowledge, the teacher is able to give an evaluation of the model's performance on training examples (Haykin, 2009).

### 2.1.3 Semi-supervised Learning

In a number of cases, only a portion of the available training data is labeled. These types of cases can make use of semi-supervised learning methods. In addition to unlabelled data, algorithms of this type are provided with some supervision information. This information often is targets associated with some of the examples (Chapelle et al., 2010).

### 2.1.4 Reinforcement Learning

When we ponder the nature of learning, one of the first thoughts to cross one's mind is perhaps, the idea that we learn by interacting with our environment. Throughout our lives, a major source of knowledge about our environment and ourselves comes from the wealth of information about cause and effect, about the consequences of our actions. This knowledge obtained from experiencing our environment helps us make decisions on what to do in order to achieve goals (Sutton & Barto, 2018). In the Reinforcement learning framework, the learning agent does not have access to a teacher that gives examples of good or bad behavior. Rather, the learning agent

experiences its environment by trial and error in order to discover a policy or pattern of behavior, which maximizes its reward.

Computational usage of reinforcement learning frameworks has greatly increased in recent years. Reinforcement learning algorithms have been utilized for many decades, however, the introduction of deep reinforcement learning algorithms - classic reinforcement learning frameworks combined with deep neural networks, has gained popularity in recent years (Figure 2.1). This additional attention to the class of learning algorithms was in large, due to breakthrough articles by Deepmind (Mnih et al., 2013, 2015).

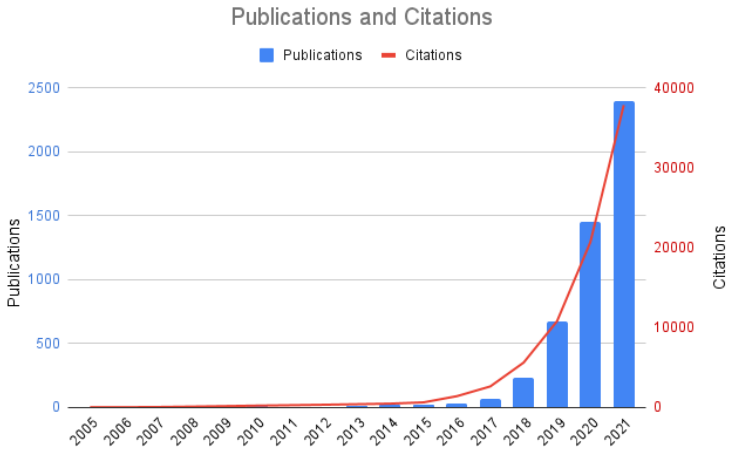



Figure 2.1: Web of Science topic search for "Deep Reinforcement Learning"

Reinforcement learning and in particular, deep reinforcement learning algorithms have been utilized in a multitude of cases. One of the common uses of these learning algorithms in robotics is for learning manipulation skills. Skills such as shooting hockey pucks (Chebotar, Hausman, Zhang, et al., 2017), opening doors (Chebotar, Kalakrishnan, et al., 2017), putting caps on bottles (Levine et al., 2016), and lego block stacking (Levine et al., 2015; Haarnoja et al., 2018) have been learnt using reinforcement learning algorithms. Reinforcement learning in robotics has also been hugely utilized for grasping tasks (Chebotar, Hausman, Kroemer, et al., 2017; Levine et al., 2018; Kalashnikov et al., 2018). Another aspect of robotics that deep reinforcement learning has significantly contributed to is legged locomotion. This paper falls within this scope and further details of reinforcement learning as it relates to locomotion will further be discussed in section 2.4.

## 2.2 Deep Reinforcement Learning Policy Optimization

Deep reinforcement learning algorithms are generally grouped into two categories; value-based methods and policy-based methods. The DRL algorithm utilized in this project is a policy-based DRL algorithm. In order to fully understand and appreciate the benefits of utilizing a policy-based method, it is perhaps important to first briefly analyze an alternative DRL schema; value-based learning.

The value-based DRL methods such as deep Q-learning, learn an approximation of the Q-value of state-action pairs using deep neural networks. The Q-value of a state-action pair $(s,a)$ is the expected discounted return (i.e. sum of all rewards received from time step t onwards) received if the agent takes an action $a$ from a state $s$ and follows the policy distribution $\pi$ thereafter (Equation 2.1). This is very similar to the value of a state (V-value). The V-value of a state is the expected discounted return if the agent is in state $s$ at time $t$ and thereafter, follows its policy $\pi$ (Equation 2.2). A value-based DRL algorithm selects actions using an epsilon-greedy policy or a softmax policy. This allows the learning algorithm to select the most suitable action (state transition). To allow for more exploration of the environment, an epsilon-greedy approach is usually used. This selects actions based on the greedy policy and a predefined probability for selecting the greedy policy's action or a random action. During the training of a value-based DRL method, a neural network with weights and bias parameters $\theta$ is trained to minimize the loss $\mathcal{L}(\theta)$ for all states $(s)$ or states and action pairs $(s,a)$ so that the estimated V-values or Q-values converge.

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[R_t|s_t = s, a_t = a] \tag{2.1}$$

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R_t|s_t = s] \tag{2.2}$$

Policy-based DRL methods approach the reinforcement learning task very differently. Instead of using the learnt value of a state or state-action pair to determine suitable actions, this type of algorithms directly learn a policy $\pi_{\theta}(a|s)$ that maximizes the expected return $R_t$ for trajectories $\tau = (s_0, a_0, s_1, a_1, ..., s_T, a_T)$ generated by the actions selected by the policy. The objective function that is maximized in these methods is defined by:

$$J(\theta) = \mathbb{E}_{\tau\sim\rho\theta}[R(\tau)] = \mathbb{E}_{\tau\sim\rho\theta}[\sum_{t=0}^{T} \gamma^t r(s_t, a_t,)] \tag{2.3}$$

$\rho\theta(\tau)$ denotes the likelihood that a trajectory $\tau$ was generated by the policy function $\pi$ with parameters $\theta$.

$$\rho_\theta(\tau) = p_\theta(s_0, a_0, ..., s_T, a_T) = p_0(s_0) \prod_{t=0}^{T} \pi_\theta(s_t, a_t) p(s_{t+1}|s_t, a_t) \tag{2.4}$$

Policy-gradient methods are the most commonly used technique for the optimization of a policy $\pi_\theta(a|s)$. This optimization technique and the specific optimization strategy (proximal policy optimization) utilized for this research are discussed in the succeeding subsections.

### 2.2.1 Policy Gradient Methods

Policy gradient methods work by computing an estimator of the policy gradient and plugging it into a gradient ascent algorithm (Schulman et al., 2017). A popular early policy gradient method is the REINFORCE algorithm (Williams, 1992). The gradient estimation is characterized by:

$$\nabla_\theta J(\theta) = \nabla_\theta \log \pi(a_t|s_t;\theta) R_t \tag{2.5}$$

This allows the weights of the policy to be updated using gradient ascent; $\theta \leftarrow \theta + \eta \nabla_\theta J(\theta)$. This approach to reinforcement learning is beneficial because it is model-free. Model-free RL algorithms do not need the transition probability distribution and the reward function associated with the Markov Decision Process (MDP). This means that the reinforcement learning model can learn by trial-and-error through exploring its environment and generating trajectories/episodes. The REINFORCE algorithm does, however, have its drawbacks. Firstly, the returns $R_t$ can have a very high variance which is problematic for neural networks. Another issue with this learning algorithm, along with other online learning algorithms is that they are sample inefficient because new trajectories have to be sampled frequently to update the policy. The sample inefficiency of online learning algorithms can, unfortunately, not be changed because it is an essential part of the learning update. A technique used for improving sample efficiency is described in chapter 3. Many ways of improving the performance of the classical policy gradient method have been introduced. These methods try to handle the instability resulting from the high variance mentioned earlier.

### 2.2.2 Trust Region Methods

Trust region methods maximize an objective function with respect to a constraint on the size of the policy update. Kakade & Langford (2002) introduced the possibility to relate the expected return $\eta(\pi)$ of two policies $\pi_\theta$ and $\pi_{\theta_{\text{old}}}$ using the advantage $A$. The advantage represents a comparison of the expected return when using a new

policy $\pi_\theta$ to the expected return from a previous policy $\pi_{\theta_{\text{old}}}$. TRPO (Schulman et al., 2015), a common trust region method performs its policy update by:

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \tag{2.6}$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta\text{old}}, \pi_\theta]] \leq \delta \tag{2.7}$$

Alternatively, regularizing the objective function with the KL divergence can be done (Equation 2.8) instead of utilizing the hard constrained. It is however challenging to select a single value of $\beta$ that performs well for different tasks.

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta\text{old}}(a_t|s_t)} \hat{A}_t - \beta \text{KL}[\pi_{\theta\text{old}}, \pi_\theta] \right] \tag{2.8}$$

The authors (Schulman et al., 2015) introduced a trust region constraint which is defined by the KL divergence between the new policy and the old policy. The TRPO method performed well on simulated robotic tasks of swimming, hopping, and walking, as well as playing Atari games given raw images. Trust-region methods are still being developed and optimized. Wu et al. (2017) proposed a scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. Duan et al. (2016) compared different DRL algorithms and found that *TRPO*, DDPG, and TNPG Schulman et al. (2015) are effective in training deep neural network policies. The robustness of TRPO has, however, been called to question.

### 2.2.3 Proximal Policy Optimization

Proximal policy optimization (PPO) is a variant of the TRPO algorithm. Unlike TRPO, PPO formulates the constraint as a penalty/clipping objective, instead of using the Kullback–Leibler constraint. PPO is preferred to TRPO because of its simplicity of implementation, and its widely similar/superior performance on reinforcement learning tasks. The ease of implementation is due to the algorithm's ability to maximize its clipped objective using first-order methods like stochastic gradient ascent or Adam. There is a variant of PPO which uses an adaptive KL penalty to control the change of the policy. However, this hardly improves the performance of the algorithm hence; the widely used variant of the algorithm uses the objective function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)] \tag{2.9}$$

where;

- $\theta$ is the policy parameter
- $\hat{\mathbb{E}}_t$ is the empirical expectation over timesteps
- $r_t$ is the ratio of the probability under the new and old policies, respectively
- $\hat{A}_t$ is the estimated avantage at time $t$
- $\varepsilon$ is a hyperparameter, usually 0.1 or 0.2

The left part of the *min* operator in Equation 2.9 is the same as the surrogate objective of TRPO. The right-hand part restricts the algorithm from making drastic updates to the policy. This reduces the instability associated with policy-based reinforcement learning methods resulting from the huge variance in the state-action space. The effect of using this clipped sampling weight in PPO is illustrated in Figure 2.2

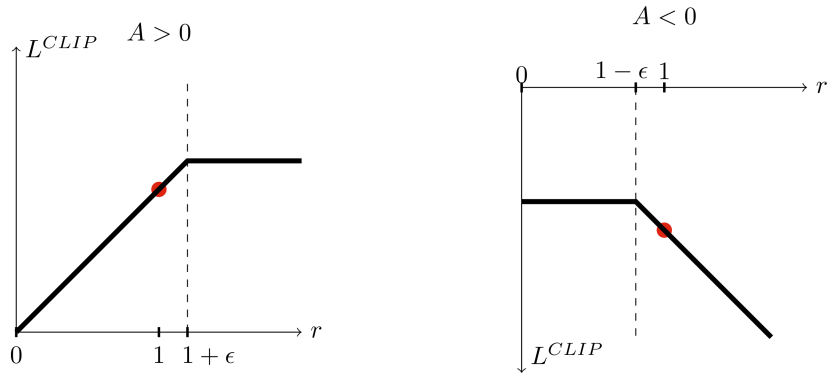



Figure 2.2: Plots showing one timestep of the surrogate function $L^{CLIP}$ as a function of the probability ration $r$, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. (retrieved from Schulman et al. (2017))

## 2.3 Artificial Neural Networks

The human brain is capable of organizing its structural constituents (neurons) in order to perform certain computations (e.g., motor control and visual perception) (Haykin, 2009). This seamless ability of the human brain to perform tasks that are incredibly complex even for very powerful computational systems has been a great inspiration to researchers for many decades. Like other computational techniques that take inspiration from the real world (Holland, 1992; De Jong, 1975; Yang, 2010), artificial neural networks have been used to solve complex problems

by modelling low-level activities that are performed in the brain. Haykin (2009) defines an artificial neural network viewed as an adaptive machine:

> *A(n) (artificial) neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:*
>
> 1. *Knowledge is acquired by the network from its environment through a learning process.*
> 2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

This section goes over some of the relevant low-level inspirations utilized by artificial neural networks as well some of the computational techniques utilized for the execution of these networks.

### 2.3.1 McCulloch-Pitts Neuron

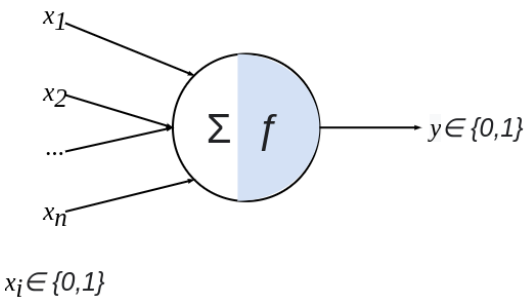



Figure 2.3: McCulloch-Pitts neuron with n inputs

In 1943, a neuroscientist; Warren McCulloch, and a logician; Walter Pitts, proposed the first computational model of a neuron (McCulloch & Pitts, 1943). The McCulloch-Pitts neuron, as it is commonly known, led to the first wave of interest in artificial neural networks. This mathematical neuron model consists of multiple binary inputs which are inspired by biological dendrites that propagate electrochemical stimulation received from other neural cells. The McCulloh-Pitts neuron model aggregates the received binary inputs (see Equation 2.10) and then, utilizes an activation function $f$ to generate a binary output $y$ based on a predefined threshold $\theta$ (Equation 2.11). This process can be seen in Figure 2.3.

$$a = \sum_{i=0}^{n} x_i \tag{2.10}$$

$$y = f(a) = \begin{cases} 0 & x < \theta \\ 1 & x \geq \theta \end{cases} \tag{2.11}$$

### 2.3.2 Perceptron

The Perceptron is a single McCulloch-Pitts neuron-like unit. Frank Rosenblatt pioneered the Perceptron (Rosenblatt, 1958, 1962), which is one of the earliest, highly successful, machine learning device and concept. The perceptron is simply an artificial neural network having several input nodes $x$ (where $x \in \mathbb{R}^N$). Figure 2.4 shows a visualization of a perceptron. It shows an output node connected to $n$ weighted input nodes and a bias node. The perceptron generates an output $y$ based on an input vector $x$ (see Equations 2.12 & 2.13).

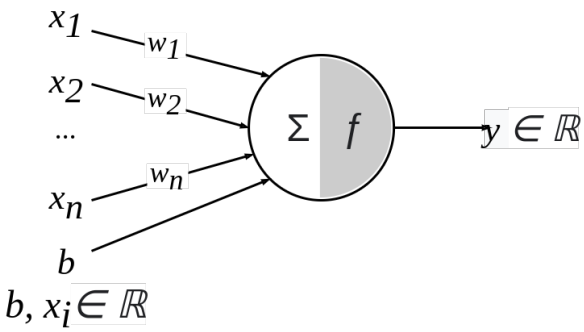



Figure 2.4: A Perceptron with a bias term

$$a = \sum_{i=0}^{n} x_i w_i + b \tag{2.12}$$

$$y = f(a) \tag{2.13}$$

This allows the perceptron to find a solution for linearly separable datasets. The bias term lets the perceptron find a solution for not only homogeneous linearly separable datasets, but also inhomogeneously linearly separable datasets. Using this perceptron model and a learning algorithm such as Hebbian learning (Hebb, 1949) or gradient descent, the perceptron is able to learn linearly separable dichotomies such as the AND, and OR functions by iteratively changing its weights using an intelligent algorithm. The perceptron eventually converges at a solution which is essentially a hyperplane that dichotomizes the dataset. This implies that the perceptron is, therefore, unable to find solutions for non-linearly separable problems such as the XOR problem. For this, a more complex artificial neural network structure is required.

### 2.3.3 Multi-Layer Perceptron

The multi-layer perceptron (commonly referred to as a fully-connected feed-forward neural network) surpasses the capabilities of the classic perceptron. This network architecture consists of an input layer and output layer as well as hidden layer(s) that contain hidden nodes as well as differentiable non-linear activation functions. In this type of artificial neural network architecture, the outputs of multiple perceptrons in a previous layer are connected as inputs for a node in the next layer. Some feed-forward neural network architectures can have skip-connections which are inputs from nodes that are more than one layer behind in the network.

Multi-layer perceptrons have been widely used in artificial intelligence and machine learning. One of the reasons for the popularity of these networks is their ability to be utilized as universal function approximators. Numerous research has been done in analyzing the use of feed-forward networks as universal function approximators (Hornik et al., 1989; Friedman, 1994; Leshno et al., 1993; Sonoda & Murata, 2017; Mhaskar & Micchelli, 1992; Castro et al., 2000). While MLPs are in principle, universal approximators, training them to achieve an adequate performance on machine learning tasks can be challenging. Several factors have to be taken into account such as; the network's architectural complexity, activations, as well as the learning algorithm. Such factors make training an MLP to approximate an arbitrary function, an arduous task. Thankfully, there is a wealth of knowledge about the optimization and regularization of these artificial neural networks, obtained from a plethora of research done in the field of machine learning.

### 2.3.4 Optimization

In machine learning, the term optimization is often used to refer to methods used to improve a learning model's empirical performance. In a supervised learning scenario, this would be the performance on a *training dataset*. The error associated with the model's empirical performance is called the empirical error/loss. While training a learning model, it is desirable to reduce this empirical loss however, this is not the ultimate goal in machine learning. The goal is to minimize the true loss. The distribution of the true loss of the trained observation is however not usually known therefore, a validation dataset is often used as an approximation of the learning model's true risk.

## 2.4 Related Work

Simulations have long been used as an efficient tool for modeling human gait (Chow & Jacobson, 1971; Davy & Audu, 1987; Marshall et al., 1989; Koopman et al., 1995). Advancements in the field of Reinforcement Learning have birthed the introduction of algorithms such as the Actor-Critic methods; Advantage Actor-Critic

(A2C), and Asynchronous Advantage Actor-Critic (A3C) (Mnih et al., 2016). These algorithms have further been optimized to reduce the instability that plagues them. More recent improvements to these algorithms which fall under the broader category of policy-gradient methods include Proximal Policy Optimization (Schulman et al., 2017) and Trust Region Policy Optimization (Schulman et al., 2015), with the former producing comparable or better performances while being much easier to implement.

Table 2.1 shows some relevant applications of DRL for various locomotive tasks. It has been observed that PPO produces better performance than DDPG - another policy-gradient method, for the task of generating a healthy gait in a simulated environment (Salwan & Kant, 2020; Ananthakrishnan et al., 2018). (Melo & Maximo, 2019) and (Anand et al., 2019; De Vree & Carloni, 2021) observed that using PPO and PPO with Imitation Learning (respectively) performs impressively for robotic locomotion tasks. Another interesting approach for a locomotion task was implemented by (Peng et al., 2020). The authors utilized an online-learning-based Actor-Critic Network for walking assistance control of a lower limb exoskeleton with hemiplegic patients. The patients were modeled as a Leader-Follower Multi-Agent System (LFMAS) framework which allowed the authors to have the affected leg track the trajectory of the unaffected leg after every half gait cycle. Real and simulated experiments showed that the approach was effective on a 2DoF system.

Table 2.1: State-of-the-art DRL algorithms and some relevant applications

| DRL Algorithm | Algorithm Type | Article | Application Domain |
|---|---|---|---|
| DDPG | off-policy | (Salwan & Kant, 2020; Ananthakrishnan et al., 2018) | Healthy human gait generation |
| Q-Learning | off-policy | (Liu & Hodgins, 2017) | Humanoid locomotion for highly dynamic behaviors |
| A3C | off-policy | Sartoretti et al. (2019) | Learning hexapod locomotion stochastically |
| Least Square Policy Iteration | off-policy | Tu et al. (2020) | Adaptive personalized torque assistance for walking |
| Neural fitted Q with Continuous Action | off-policy | Wen et al. (2017) | Adaptive human-prosthesis control |
| PPO | on-policy | (Melo & Maximo, 2019) | Robotic locomotion |
| | | (Salwan & Kant, 2020; Ananthakrishnan et al., 2018) | Healthy human gait generation |
| | | (Park et al., 2020) | Robustness of DRL policy for biped locomotion |
| PPO+Imitation Learning | on-policy | (Anand et al., 2019; De Vree & Carloni, 2021) | Physics-based model locomotion |
| TRPO | on-policy | (Schulman et al., 2015) | Learning robotic swimming, hopping, and walking gaits |
| Actor-Critic + LFMAS | on-policy | Peng et al. (2020) | Walking Assistance Control of a Lower Limb Exoskeleton |
| Direct Heuristic Dynamic Programming (dHDP) | on-policy | (Wen, Si, et al., 2020) | Online adaptive control of knee prosthesis |
| | | (Wen, Li, et al., 2020) | Impact of robotic knee prosthesis mechanics on human gait symmetry |
| Policy Iteration with Constraint Embedded | on-policy | Li et al. (2021) | Robotic knee prosthesis impedance control |

# Chapter 3

# Research Methods

This chapter describes the methodology utilized to conduct simulations, and train a transfemoral amputee model to walk using deep reinforcement learning. It also describes the methods used to answer the stated research questions (section 1.1). It includes a description of the learning agent and the simulation environment. This chapter also describes and justifies the selected hyperparameters in the learning algorithm (Proximal Policy Optimization). Furthermore, the method of reward injection in the Deep Reinforcement Learning framework is explained. A proposed solution for learning with a reduced state observation is then introduced and finally, the conditions that constitute a good performance by the learning agent are explained.

## 3.1 Materials

### 3.1.1 Opensim

To efficiently simulate an osseointegrated transfemoral amputee model, and its complex interaction with its environment, Opensim (Delp et al., 2007) was utilized. This is an open-source software system that allows the modeling, simulation, and analysis of biomechanical agents. Opensim is suitable for the purpose of this research because it provides a dynamic simulation of movements for the designed agent (subsection 3.1.2). It is also adequate because it incorporates models describing the anatomy and physiology of the elements of the musculoskeletal system and the mechanics of multi-joint movement thus, providing an adequate framework for the visualization and analysis of the transfemoral amputee model. Leveraging this simulated environment gives us access to a wealth of information about the dynamics of the agent such as muscular, spatial, and temporal information. This insight into the system's dynamics can then be used to train an intelligent control system for the walking task of the transfemoral amputee agent, as well as perform analyses on the

resulting gait pattern.

To conduct this research, *osim-rl* (Kidziński et al., 2018) is used. This is a package that allows the synthesis of physiologically accurate movement by combining biomechanical expertise embedded in OpenSim simulation software with state-of-the-art control strategies using Deep Reinforcement Learning. This package was used for controlling the actions and observing the state of the model.

### 3.1.2 The Agent

The transfemoral amputee agent utilized in the forthcoming simulated experiments was developed by Raveendranathan (Raveendranathan, ongoing). This prosthesis model is osseointegrated into the femur of the left leg (Raveendranathan & Carloni, 2020) and can be seen in Figure 3.1. This agent weighs 67.4 kilograms and is composed of a skeletal and prosthetic base. The structure of the agent is controlled by muscles and actuators which grant control to the joints of the agent; thus, allowing a range of motion (flexion, adduction, etc.) which when utilized with an intelligent policy, grants the agent the ability to achieve locomotion.

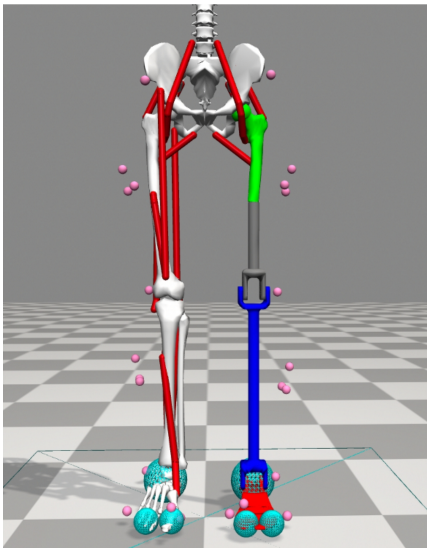


Figure 3.1: Osseointegrated transfemoral amputee model developed by Raveendranathan (Raveendranathan & Carloni, 2020). The red lines are the muscles. The pink balls are markers and the blue spheres on the feet are the contact meshes. The model has 11 muscles on the right leg. On the left, there are 4 muscles, and an actuator at the knee and at the ankle joint.

The transfemoral amputee agent consists of simulated biological muscles. These muscles are based on a non-linear first-order dynamic Hill-type muscle model be-

tween excitation and activation (Thelen, 2003). The Hill-type muscle model includes a contractile element (CE), a parallel elastic element (PE), and a series elastic element (SE), as can be seen in Figure 3.2. The generated muscle force is a function of three factors: the fiber-length, the fiber-length-velocity, and the muscle activation level, which can range between 0% and 100%. The muscle activations generate a movement as a function of muscle properties, such as the maximum isometric force, the muscle fiber length $L^M$, the tendon slack length $L^T$ , the maximum contraction velocity, and the pennation angle $\alpha^M$. Although the Hill-type muscle model does not realistically represent the human muscle architecture, its performance in simulating the gross biomechanical behaviour of a muscle-tendon unit in a computationally inexpensive manner is quite precise (Arslan et al., 2019).

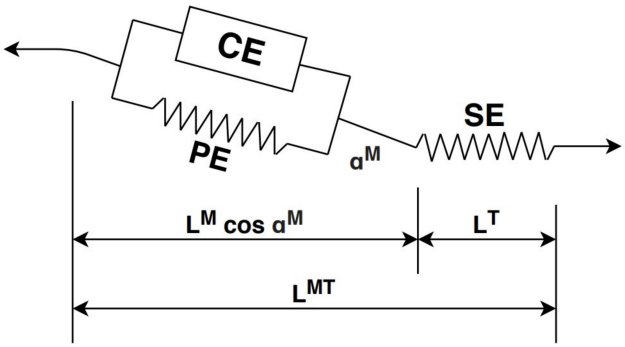



Figure 3.2: Hill-type muscle model that describes the musculo-tendon contraction mechanics in the transfemoral amputee model. It includes a contractile element (CE), and two elastic elements (one parallel and one series). The elements generate a force on the tendon. Figure from (De Vree & Carloni, 2021; Thelen, 2003)

Activation Coordinate Actuators are used to actuate the knee and ankle joints of the prosthesis. This type of actuator produces a generalized force using first-order linear activation dynamics. The activation of the actuator is represented by one state variable $\dot{a}$, represented by $\dot{a} = \frac{x-a}{\tau}$. Here $x$ is the actuator excitation, a is the activation constant set to 0.01, and $\tau$ is the activation time constant; set to 0.02.

The transfemoral amputee model comprises 15 muscles and 2 actuators (Figure 3.3) which are controlled by 17 control signals. There are 11 muscles on the right leg. On the left leg of the model, there 4 muscles and 2 actuators. A summary of the primary function of these muscles can be seen in Table 3.1. These control signals for the model's muscles and actuators, give the model 14 degrees of freedom; the pelvic tilt, list and rotation, lumbar extension, (left and right) knee flexion, ankle flexion, hip flexion, adduction, and rotation. The range of motion for each of these degrees of freedom can be seen in table Table 3.2.

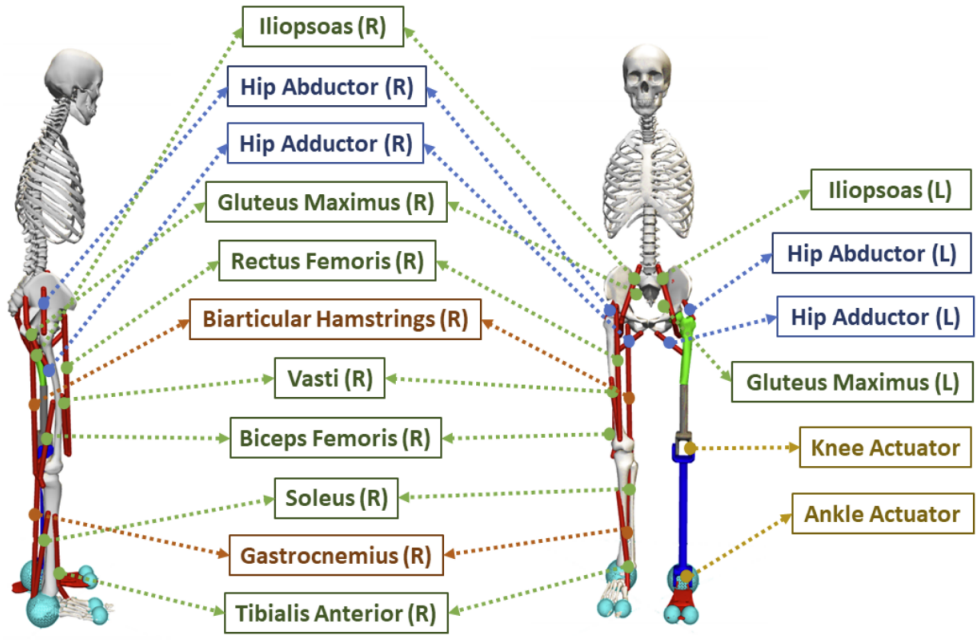



Figure 3.3: Osseointegrated tranfemoral amputee model with labeled muscles and actuators. It shows the 15 muscles and 2 actuators possessed by the agent. The uniarticular muscles are labeled in green, biarticular muscles in red, hip adduction and abduction in blue, and actuators in blue.

At each time-step in the course of the simulation, the agent receives 17 control signals corresponding to the described muscles and actuators. The agent then acts appropriately with respect to the signals, resulting in a change in the observed state of the agent. The action performed by the agent is bounded by laws of physics in Opensim which mimics realistic movement. The complete observed state of the agent is represented by a vector of size 91. This observation state can be seen in Table A.1. The agent has a second observation state which is a subset of the complete observation space. This second observation state will be subsequently referred to as the reduced state observation and it is more akin to real-world observable data. The force, velocity, and length of the muscle fibers in Table 3.1 are not a part of this reduced observation state.

### 3.1.3 The Imitation Dataset

The imitation data used in training the DRL algorithm to have a human-like gait was gathered from (Camargo et al., 2021). The dataset contains data from 22 able-bodied adults, age 21 ± 3.4 yr, height 1.70 ± 0.07 m, and mass 68.3 ± 10.83 kg. The subjects were instrumented unilaterally on their right side with 11 EMG (Biometrics. Ltd. Newport, UK), 3 goniometers (Biometrics. Ltd. Newport, UK), 4

Table 3.1: Primary functions of the 15 muscles and 2 actuators, present in the transfemoral amputee model

| **Muscle or Actuator** | **Primary Function** | **Leg** |
|---|---|---|
| Biarticular Hamstrings | Hip extension, knee flexion | Right |
| Rectus Femoris | Hip flexion, knee extension | Right |
| Vasti | Knee extension | Right |
| Biceps Femoris | Knee flexion | Right |
| Gastrocnemius | Knee flexion, ankle extension | Right |
| Soleus | Ankle extension | Right |
| Tibialis Anterior | Ankle extension and flexion | Right |
| Hip Abductor | Away from body's vertical midline | Both |
| Hip Adductor | Towards body's vertical midline | Both |
| Iliopsoas | Hip flexion | Both |
| Gluteus Maximus | Hip extension | Both |
| Knee Actuator | Knee flexion and extension | Left |
| Ankle Actuator | Ankle flexion and extension | Left |

six-axis inertial measurement units (Yost, Ohio, USA), and bilaterally with 32 motion capture markers following the Helen Hayes Hospital marker set (Vicon. Ltd., Oxford, UK). The ground reaction forces were recorded using force plates (Bertec, Ohio, USA).

This research makes use of the processed data from de Boer (2021) which selected one subject (AB06) from the above-described dataset. The subject has a height of 1.80m and weighs 74.8kg. In de Boer's research, the amputee model was scaled to fit the data using the motion caption markers that are present in the dataset. The dimensions of each segment in the model were scaled so that the distances between the virtual marker (on the amputee model) match the distances between the markers (from AB06). Opensim's (Delp et al., 2007) Scale tool was used to perform the scaling of the data. Figure 3.1 shows motion capture markers (represented by pink balls), fixed to the joints that they belong to, by specifying the location and the body to which the marker is attached. The model was scaled to fit the data by utilizing two consecutive timestamps from the data while the subject stood idly.

A transformation of the data was then performed in order to fit the orientation of the model, using the marker data in the dataset and the experimental data preview tool of Opensim. The data was rotated 270 degrees around the Y-axis. This matched the model's orientation and the orientation of the subject in the dataset. The angles of the joints were retrieved after this rotation process using Opensim's inverse kinematic tool. This tool produces a motion file with the joint angles. The maximum marker error was below 2-4 cm and the RMSE was under 2cm. These are acceptable error margins according to the OpenSim user's guide (OpenSim, 2012)

Table 3.2: The range of motion for the degrees of freedom of the transfemoral amputee model.

| **Joint** | **Range of motion (in degrees)** |
|---|---|
| Pelvis tilt | -90 to 90 |
| Pelvis list | -90 to 90 |
| Pelvis rotation | -90 to 90 |
| Lumbar extension | Locked to -5 |
| Hip flexion (left & right) | -120 to 120 |
| Hip adduction (left & right) | -45 to 45 |
| Hip rotation (left & right) | Locked to 0 |
| Knee flexion (left & right) | -120 to 10 |
| Ankle flexion (left & right) | -60 to 30 |

## 3.2 Proximal Policy Optimization

To train the described agent to have a human-like gait, an intelligent policy is needed which receives the state of the agent as input and produces as output; an action that controls the agent at each time-step of the simulation. This action is a vector that corresponds to the control signals of the agent and its values are activations for the 15 muscles and 2 actuators of the agent. The policy utilized in this study is a feed-forward artificial neural network with four layers; one input layer (observed state of the agent), two hidden layers, and an output layer. To train this policy network for the described locomotion task, the deep reinforcement learning algorithm; proximal policy optimization is used. PPO has already been introduced in subsection 2.2.3 in terms of its optimization goal. This section describes the algorithmic implementation of the optimization algorithm as it was used in this project as well as the reason behind the selected hyperparameters.

Firstly, let us go over the objective function of the PPO algorithm. There are two variants of PPO as introduced by (Schulman et al., 2017). One of these variants is closer to TRPO because it performs a KL-constrained update. This version of PPO maximizes a surrogate objective function $L^{KLPEN}$ (Equation 3.1). This variant differs from TRPO because there is no hard constraint on the KL-divergence. Rather, the KL-divergence is penalized in the objective function and the penalty coefficient $\beta$ is adjusted over the course of training. This implementation of PPO is more computationally costly than the clipped surrogate objective function (Equation 2.9) and scaling the penalty coefficient appropriately during training can be challenging. This optimization method does not have a KL-divergence term in the surrogate objective function and does not have a constraint parameter to be appropriately scaled. Rather, it works by using a different objective function $L^{CLIP}$ (Equation 2.9) which specifically performs clipping to remove the incentive for the new policy to deviate

far from the old policy in the training step.

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t - \beta \mathrm{KL}[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] \right] \tag{3.1}$$

The algorithm for PPO can be seen in Algorithm 1. This algorithm makes use of $N$ parallel actors (or workers) in a bid to take advantage of the parallel processing of processors. The PPO algorithm requires a data collection facility from the simulation which; collects data by generating stochastic actions, and calculates the advantage estimate using Generalized Advantage Estimate (GAE). This part of the algorithm is parallelized using $N$ workers and is represented in lines $3-9$ of Algorithm 1. Line 10 is a separate major thread of the algorithm. Here, the collected trajectories (state, action pairs) are sampled into minibatches of size $M$. These collected minibatches are then used to optimize the surrogate objective function $L^{CLIP}$. The optimization update is performed for K epochs on the same minibatch of trajectory samples. This allows a more sample-efficient manner of optimization and is possible because PPO nullifies the risk of taking large update steps on the trajectory samples.

**Algorithm 1** PPO algorithm

1: $\theta_0 \leftarrow$ inital policy parameters
2: $\phi_0 \leftarrow$ inital value function parameters
3: **for** $k = 0, 1, 2, ...$ **do** ▷ number of iterations
4: $\mathcal{D}_k \leftarrow \{\}$
5: **for** $worker = 0, 1, 2, ...N$ **do**
6: Collect trajectories $\{\tau_{worker}\} \subset \mathcal{D}_k$ by running policy $\pi_{\theta_k}$ in the environment for $T$ timesteps
7: Compute rewards $\hat{R}_t$
8: Compute advantage estimates, $\hat{A_1}, ..., \hat{A}_T$ using value function $V_{\phi,k}$
9: **end for**
10: Update policy parameters $\theta_{k+1}$ by maximizing PPO surrogate objective $L$, with K epochs and minibatch size $M < NT$
11: **end for**

Table 3.3 summarizes the hyperparameter and specifications used in the deep reinforcement learning algorithm. In the course of training the transfemoral amputee to walk, 10 workers were used for parallelization, in order to gather trajectory data (state, action pairs). The stopping criteria for the training process of PPO was a defined number of iterations. A suitable value for this was determined in chapter 4. Each worker collects trajectories $\tau_{worker}$ of size 1,536 timesteps. A timestep is 10 milliseconds of the simulated gait. This is consistent with the hyperparameter selection for De Vree & Carloni (2021) on a healthy subject model. Some important hyperparameters for the PPO algorithm also include the clip parameter $\varepsilon$

Table 3.3: Summary of the selected hyperparameters used for training models with PPO for the locomotion task.

| **Parameter** | **Value** | **Parameter** | **Value** |
|---|---|---|---|
| workers ($N$) | 10 | PPO clip parameter ($\varepsilon$) | 0.2 |
| iterations | $\geq 700$ | optimization mini-batch size | 512 |
| episodes | N/A | number of policy updates per mini-batch ($K$) | 4 |
| total steps | N/A | discount factor ($\gamma$) | 0.99 |
| steps per worker ($T$) | 1,536 | GAE ($\lambda$) | 0.95 |
| steps per action | 1 | hidden layers in neural network | 2 |
| entropy coefficient | 0.01 | hidden layer size | [100, 228, 312] |
| stochastic policy | true | activation function | tanh |
| prediction categories | $\geq 2$ | optimization stepsize | 0.001 |

which was set to 0.2. This clip parameter value produced the best benchmark score in Schulman et al. (2017) and has been utilized in previous research for healthy and amputee models for locomotion tasks. The activation function for the policy network, as well as the value network, was set to be the hyperbolic tangent function (*tanh*). Four policy updates were performed on each sampled mini-batch from the collected trajectories and mini-batches were sampled with a size of 512. The discount factor $\gamma$ determines the extent the agent considers future rewards. If the discount factor $\gamma$ is set to 0, the agent only considers the current reward. A $\gamma$ value close to 1 implies that in determining the value of a state, the learning algorithm also takes into account rewards in the distant future. Another important hyperparameter that had to be carefully selected is the entropy coefficient. The entropy coefficient is a regularizer. It is multiplied by the maximum possible entropy and added to the surrogate objective $L^{CLIP}$ and the loss of the value function. This constitutes the actual object function that is minimized. This entropy term can also be utilized as a regularizer using early stopping in the training update step. In that approach, the policy update is performed if the KL divergence is less than or equal to a target KL value (usually between 0.003 and 0.03). However, regularizing using the KL divergence was implemented by adding to the surrogate objective function because this implementation has been utilized in previous research with impressive results.

The generalized advantage estimate (GAE) was used to obtain a value corresponding to the benefit of performing an action $a$, given the current state $s$. It does this by calculating the difference between the expected discounted $\lambda$-return, and the current state value. This is given described in Equation 3.2. Utilizing the $\lambda$-return of the GAE has been shown to yield better performance than using the simple $n$-step return (Schulman et al., 2015; Sutton et al., 1998). The $\lambda$-return calculates an exponentially weighted average of $n$-step returns with decay parameter $\lambda$ (see Equation 3.3). A $\lambda$ value of 0 reduces the return to a single time-step, while a $\lambda$ value of 1 implies there is no discount in rewards, hence, all future rewards are considered.

$$\hat{\mathcal{A}} = R_t(\lambda) - V(s_t) \tag{3.2}$$

$$R_t(\lambda) = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^n + \lambda^{T-t-1} R_t^{T-t} \tag{3.3}$$

The value network, as well as the policy network, has four layers; one input layer with dimensions equal to the observation space, two hidden layers whose size will be discussed in the experimental setup (section 3.5), and an output layer. The output layer of the policy network is composed of 17 units, corresponding to the action space of the transfemoral amputee agent while the output layer of the value network is made up of just one unit; corresponding to the value of the observed state. Both networks use a hyperbolic tangent as the activation function for the hidden units. The value function is optimized using stochastic gradient descent. The policy network is optimized using Adam optimization with a stepsize of 0.001.

The policy network outputs actions for the agent given the observed state. These generated actions were implemented to be a discrete set of values. An additional hyperparameter - the number of prediction categories was utilized. PPO can be trained with a continuous action space by creating a policy network that predicts a mean and standard deviation value. These values are then used to sample an action from a normal distribution. This project utilizes a simpler discrete implementation. The given *number of prediction categories* determines the discrete values that the action space consists of. The values are chosen to have equal spacing and cover up the complete action space. For example, an action space ranging from 0 to 1 with three discrete prediction categories will result in the discrete action space; [0, 0.5, 1.0]. The same action space with five prediction categories with result in the discrete action space; [0, 0.25, 0.5, 0.75, 1.0].

## 3.3 Reward Injection

The manner of reward injection is perhaps, the most significant aspect of training a deep reinforcement learning model. There is a multitude of ways to design a reward function for the learning task. These different manners of reward injection have their benefits as well as their drawbacks. The reward function can be deterministic or stochastic. It can also be sparse or dense. Deterministic rewards can be obtained using the current state, action and consequent state of the agent while stochastic rewards have some degree of randomness embedded. This randomness can be a useful regularization method for the training of the learning agent. In a sparse reward system, the learning algorithm does not have access to the instantaneous reward of the agent's actions, rather, there is a reward value provided after a number of actions are

performed. If the sparseness of the reward function is high - meaning almost all of the instantaneous rewards are 0, this can lead to a common DRL challenge known as the credit-assignment problem. In this problem, due to the rare nature of nonzero instantaneous rewards, the sequence of actions that generated that reward will be very long. It would not be clear to the learning agent, what actions were beneficial in attaining that reward. This can also cause an extremely long training time for the learning agent due to possible mini-batches of trajectories with uninformative zero rewards.

The most important aspect to consider in designing a good reward function is that it should be informative for the goal of the learning task. A simplistic reward for this project can be designed that only takes into account, the position of the pelvis and issues a constant deterministic reward if the pelvis position moves forward in the next timestep after an action is performed. While simple to implement, this manner of reward injection will in all likelihood, result in a policy that learns a unique manner of maximizing the reward. There is no incentive for the agent to swing its legs. The manner of swinging locomotion that humans utilize is very efficient and much more difficult to learn with a control policy hence, the policy would likely converge to a parameter value that achieves forward motion using a much simpler control system.

For the stated reasons, the dense, deterministic reward function in Equation 3.4 was utilized in this project. This reward function is composed of a goal reward $r_{goal}$, an imitation reward $r_{imitation}$ and a penalty term $p$. At every timestep $t$, the instantaneous reward $r_t$ is calculated for given the state $s_t$, action $a_t$, and consequent state $s_{t+1}$. The following subsections describe each aspect of the reward function (Equation 3.4).

$$r_t = 0.1 \cdot r_{goal,t} + 0.9 \cdot r_{imitation,t} - p_t \tag{3.4}$$

De Vree & Carloni (2021) used a similar structure for the computation of the agent's reward - with a goal and an imitation reward, as well as a penalty term. However, the definition of these partial rewards functions in this study is different from that of De Vree's research. A notable difference occurs in the weights used for the partial rewards. De Vree's research made use of 0.4 and 0.6 goal and imitation reward weights respectively. Using that reward scheme results in slower learning of a natural gait pattern for the agent. The agent quickly learns to improve its reward $r_t$ by maximizing the goal reward. The imitation reward requires much more complex muscle activation coordination to achieve a higher reward. Increasing the weight on the imitation reward increases the incentive for the reinforcement learning policy to learn a function that achieves a human-like gait. This, in turn, leads to faster learning of the reinforcement learning task.

### 3.3.1 Goal Reward

A goal reward $r_{goal}$ at timestep $t$ is issued to create an incentive for the agent to move in a continuous straight direction. This partial reward has three components corresponding to the x, y, and z coordinate axes of the agent and it is obtained by computing the proximity $p$ between the value of these coordinates and the desired value for the agent. Interpolating these values correctly sets the desired velocity for the agent which is about 1.4m/s. The defined partial reward can be seen in Equation 3.5.

$$r_{goal,t} = e^{-(p_x+p_y+p_z)} \tag{3.5}$$

### 3.3.2 Imitation Reward

As eluded to earlier in this section, the deep reinforcement learning algorithm needs guidance in order to generate an acceptable solution for the locomotion task. In this project, this was implemented by a technique called imitation learning. Imitation learning injects additional rewards to the learning agent to guide its learning process. This additional data is beyond what is available in the course exploration. In imitation learning, data is provided to the learning agent which contains what is considered good state trajectories. This data is referred to as imitation data. The learning agent should then create a policy that mimics the solution provided by the imitation data. The imitation reward is obtained by the function in Equation 3.6.

$$r_{imitation}(t) = 0.9 \cdot r_{position}(t) + 0.1 \cdot r_{velocity}(t) \tag{3.6}$$

The choice of weights for the position and velocity rewards of the imitation reward $r_{imitation}$ in Equation 3.6, similar to $r_t$ nudges the function to learn a human-like gait quicker. If both weights are equal, the function could first learn to maximize the velocities of different joints of the agent at each timestep. This would appear to be a very impressive learning curve, however, the position of those joints could be highly unlike a human-like gait. Hence, the optimization choice of assigning a higher weight to the position reward is made in a bid to achieve faster learning of the human-like gait.

The imitation reward $r_{imitation}$ at timestep $t$ is characterized by the imitation positional reward $r_{position}$ and the imitation velocity reward $r_{velocity}$. These are obtained by comparing the position or velocity of different parts of the agent. The partial reward is obtained by the summation of the squared difference of the agent's observed hip, knee and ankle position/velocity at timestep $t$ and the corresponding imitation data feature. Equation 3.7 and Equation 3.8 describe the function for the partial imitation reward $r_X$ where $X$ is either the position or velocity attribute, Y is the agent's

joint being observed and t is the timestep of the simulation. The observed state of the agent is denoted by $s$ and the imitation dataset is represented by $im\_data$.

$$loss(X,Y,t) = (s(X,Y,t) - im\_data(X,Y,t))^2 \tag{3.7}$$

$$r_X(t) = e^{-(loss(X,ankle,t)+loss(X,knee,t)+loss(X,hip,t))} \tag{3.8}$$

Note that the partial reward for the imitation dataset calculates both the hip flexion and adduction. Also, the loss for the above-listed features is computed for both the healthy leg and for the leg with the prosthesis.

### 3.3.3 Penalty

The penalty constraint $p_t$ on the instantaneous reward $r_t$ (Equation 3.4) encourages the learning agent to find a more energy-efficient manner of solving the optimization task. This is done by penalizing the agent's actions which correspond to the activation of the agent's muscles and actuators at each timestep. Thereby, limiting the reward obtained for actions with high activation values.

## 3.4 Deep Reinforcement Learning with a Reduced State Observation

The goal of this project is to train the described transfemoral amputee model to have a healthy gait pattern, using the reduced state representation. This project utilizes the described framework in Figure 3.4 to achieve this. The figure shows three different manners of training the amputee agent. These training types are defined by the observation that the DRL algorithm is allowed to receive as input to the policy network. The observation can either be the complete state of the agent, a reduced representation of the state, or an augmented state representation. The technique that was utilized for the training of the DRL model using these observation types is discussed in this section.

When the agent acts in the simulation environment, it changes its state. This new observed state is the complete state of the agent. The reward is then computed based on the agent's state and the action vector utilized. To train using the reduced state observation, the observed state of the agent is decreased by removing the muscle forces, length, and velocity. In order to leverage the muscle information absent from the reduced state representation, an additional step is required. In this case, the reduced state representation is then fed to a pre-trained artificial neural network

Table 3.4: Overview of feed-forward network parameters used for muscle information prediction.

| **Parameter** | **Value(s)** |
|---|---|
| Optimization algorithm | Adam |
| Loss function | Mean absolute error |
| Number of hidden layers | 1,2,3 |
| Number of hidden layer units | 64, 256, 512 |
| Activation function | relu, tanh |
| Batch normalization | with, without |
| Drop out | with, without |
| Early Stopping | yes |

that predicts the missing muscle forces, velocities, and lengths. The predicted muscle information is used to augment the reduced state observation hence, obtaining an approximation of the complete state of the agent. The state of the agent obtained by applying this technique of completing the reduced state observation by predicting the missing muscle information values, using a pre-trained neural network is referred to as the augmented state in this paper.

During the training of the DRL policy, the augmented state representation is generated without access to the missing values of the reduced state space. Using collected augmented state representation, action pairs, as well as computed rewards for those actions, the policy network is optimized using the proximal policy optimization algorithm.

## 3.5 Experimental Setup

### 3.5.1 Predicting the Muscle Information

In order to predict the missing muscle information using the reduced state observation as described in Figure 3.4, various artificial neural network architectures are trained. These networks are multi-layer perceptrons and are chosen because of their universal function approximation capability. The network architecture, method of data collection, and training techniques are described in this section.

A feed-forward artificial neural network with fully-connected layers is trained for the prediction task. In order to select a suitable network architecture and hyperparameter values, 5-fold cross-validation is used. Two million observation steps are collected using random initialization and action sampling for the agent. These collected observation steps are then split into a reduced state representation and muscle information. The reduced state of the model serves as input to the artificial neural network. The network is trained in a supervised manner to predict this muscle

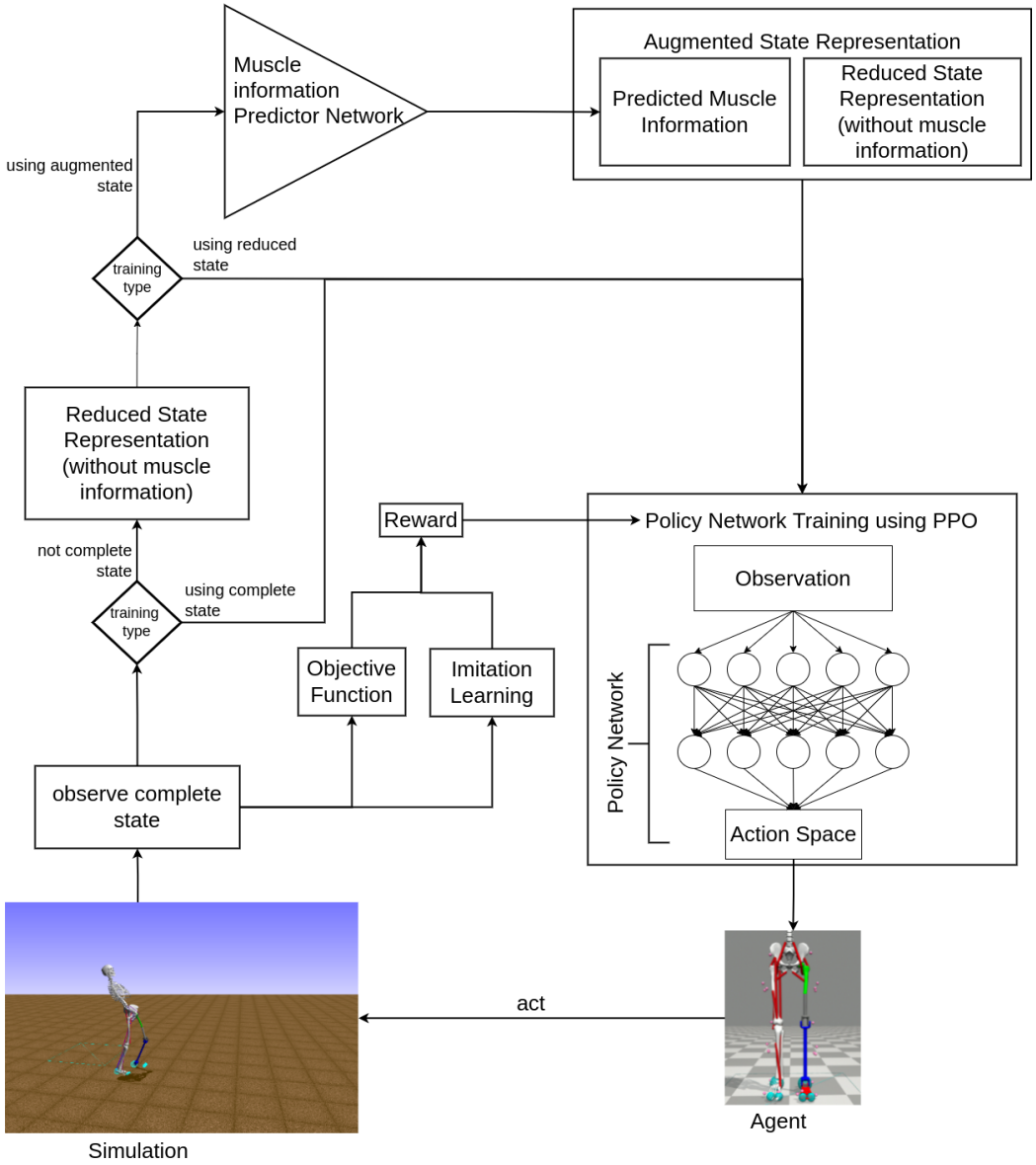



Figure 3.4: Overview of the DRL Framework for learning a with a reduced state observation.

information using the reduced state data. The force, length, and velocity values of the data are normalized to avoid having highly differing values. The force value of the muscle is divided by the maximum isometric force of that muscle while the length is divided by the optimal length of the muscle (see Table A.3).

Model selection in itself is an optimization procedure that could lead to an increase in variance. To counter this the collected 2 million time-steps of data were split into 90% training and 10% testing (Figure 3.5). Model (hyperparameter) selection was

done by selecting the neural network model with the best performance using 5-fold cross-validation on the training data. Every combination of hyperparameter values in Table 3.4 was used in configuring the model. Models with different numbers of hidden layers, hidden layer size, activation function, batch normalization, and dropout settings were trained using Adam optimization and a mean absolute error loss function. Early stopping was used for the regularization of the model. This was done by monitoring the validation loss (MAE of the validation data) and stopping training after 20 epochs of no improvement.

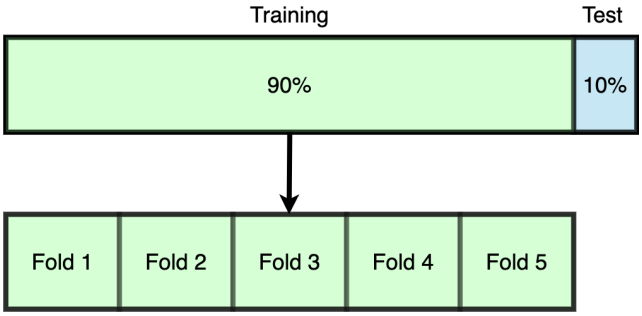



Figure 3.5: Data split for training muscle information. The top row represents the entirety of the collected data (2 million time steps). The bottom row represents the percentage used for model selection with cross-validation. Green blocks represent training data and blue represents data used for testing.

### 3.5.2 Training the Learning Agent

To train the transfemoral amputee agent using proximal policy optimization, methods of optimization had to be decided. The computational complexity of this task does not allow for a hyperparameter sweep like in the prediction of the muscle information (subsection 3.5.1). Values such as a suitable number of iterations for the PPO algorithm, complexity of the policy and value network, and the number of prediction categories for the policy had to be determined. This section describes the method for selecting suitable hyperparameter values in order to train the transfemoral amputee model for the walking task.

A successful episode of the simulation has a duration of 10.5 seconds. This corresponds to 1050 timesteps due to the imitation data being sampled at 10ms. The transfemoral amputee model is trained using this imitation data and a new episode is started if the number of timesteps exceeds the imitation data size. A new episode is also started if the model's pelvis' vertical position drops below 0.6 meters. This signifies that the model has fallen. The stopping criterion for finishing the deep reinforcement learning task is not the number of episodes or the total steps but the total number of iterations. An adequate number of iterations is obtained by training

a model with 2 prediction categories and 228 hidden units for ample iterations to solve the task. This is done using the complete observation state.

Also, different numbers of hidden units in the hidden layer of the policy and value networks were tested. 100, 228, and 312 hidden units were empirically tested and the hidden layer size of the best performing model was used for further experiments. Using the obtained adequate number of iterations and the best performing hidden layer size for the networks, a suitable number for the number of prediction categories is then determined. Table 3.5 gives an overview of the hyperparameters tested for training.

Table 3.5: Summary of hyperparameters tested for the locomotion task. The hidden layer size is the same for both the policy and value networks. The discrete categories are the number of prediction categories.

| **Parameter** | **Value(s)** | | | | | | |
|---|---|---|---|---|---|---|---|
| iterations | $\leq 2700$ | | | | | | |
| hidden layer size | 100 | | 228 | | | 312 | |
| discrete categories | 2 | 3 | 5 | 7 | 9 | 15 | 21 |

After parameter selection, models are then trained using the augmented observation (reduced state + predicted muscle information), and only the reduced observation (observation without muscle information). This is done for different numbers of prediction categories and the performance of the models is compared using different metrics (see subsection 3.6.2).

## 3.6 Performance Criteria

### 3.6.1 Muscle Information Prediction

The performance of the muscle information prediction network is utilized in selecting a suitable network model. The performance of the models is judged by the mean absolute error metric of the model during the supervised prediction task. The MAE for the predicted forces, lengths, and velocities of the models is calculated and the model with the lowest MAE on the cross-validation training task is selected. The MAE for the force, length, and velocity of the 15 individual muscles of the agent are also observed to evaluate the performance. However, these are not used in model parameter selection.

### 3.6.2 Agent's Locomotion

The performance of the agent is evaluated in two ways. The agent's realized reward serves as a performance metric. To further evaluate the performance of the agent, the symmetry of the gait is analyzed using different measures.

To evaluate the agent's performance using the obtained reward, 50 episodes of simulations are performed for each model and the average episodic reward is computed. This serves as a good measure of how well the model is performing because it is the defined problem that the model has to solve by optimizing the network parameters in order to maximize this value.

The other method of judging the performance of the agent used in this project is by observing the symmetry of the gait. To do this, 50 gait cycles are performed by the agent, and the average gait cycle is computed. This is done by computing the average hip, knee, and ankle flexion through the gait cycle. Symmetry scores are then awarded for the similarity of the average left and right (hip, knee, and ankle) joint flexion through the gait cycle. The symmetry of the average joint flexion and the corresponding average imitation data is also computed as an indication of the model's similarity to the provided imitation data. Three measures of symmetry are used in this study. They are discussed below.

#### Root Mean Square Error (RMSE)

The RMSE is a commonly used metric for error. It is defined by Equation 3.9. This metric has the advantage of being in the unit as the observed variable. However, it comes at a cost of not being normalized. This means variables that do not have a high range such as the ankle flexion will potentially have lower RMSE than variables of the agent with a higher range of motion, such as the hip and knee flexion.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(x_{\mathrm{ob},t} - x_{\mathrm{im},t})^2}{T}} \tag{3.9}$$

#### Symmetry Angle

Symmetry angle is an approach introduced by Zifchock et al. (2008), which quantifies gait symmetry/asymmetry. Equation 3.10 shows the function for the symmetry angle given two values; $X_a$ and $X_u$.

$$SA(X_a, X_u) = \frac{(45 - arctan(X_a/X_u)) \cdot 100\%}{90} \tag{3.10}$$

The symmetry angle function takes a pair of observations $X_a$,$X_u$ at discrete indexes of the gait cycle. and computes a symmetry value the mean of these collected

symmetry angles is used as the symmetry angle of both observations. An SA of 0% indicates perfect symmetry, while 100% indicates that both observations are equal and opposite in magnitude.

### Trend Symmetry

Crenshaw & Richards (2006) proposed and utilized the trend symmetry (TS) as a method to evaluate the joint angle symmetry using two time series of joint angle data. The TS values range from 0 to 1 and a TS value of 0 indicates perfect symmetry. This measure of symmetry is important to this study because the other discrete methods (RMSE & SA) have a limitation in computing the symmetry of waveforms. This limitation is a result of their neglect of the temporal information in the gait waveforms.

Trend symmetry measure utilizes the eigenvectors to compare time-normalized gait cycle data. In order to do this, each waveform is first translated by subtracting its mean value from every value in the waveform for the pair of waveforms $X, Y$ (Equation 3.11).

$$\begin{Bmatrix} X_{T_i} \\ Y_{T_i} \end{Bmatrix} = \begin{Bmatrix} X_i \\ Y_i \end{Bmatrix} - \begin{Bmatrix} X_m \\ Y_m \end{Bmatrix} \tag{3.11}$$

The translated points $X_T, Y_T$ from the pair of waveforms are entered into a matrix $M$, where each pair of points is a row. This matrix is then multiplied by its transpose ($M^T \cdot M$), and a square matrix S is obtained. The eigenvectors are derived from this square matrix. Each row of $M$ is then rotated by the angle θ formed between the eigenvector and the X-axis so that the points in M now lie around the X-axis. $X_{R_i}$ and $Y_{R_i}$ represent the rotated elements of one data point from the pair of waveforms.

$$\begin{Bmatrix} X_{R_i} \\ Y_{R_i} \end{Bmatrix} = \begin{Bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{Bmatrix} - \begin{Bmatrix} X_{T_i} \\ Y_{T_i} \end{Bmatrix} \tag{3.12}$$

The variability of the rotated points is then calculated along the X and Y axes. The Y-axis variability is the variability *about* the eigenvector while the X-axis variability is the variability *along* the eigenvector for the matrix $M$. The trend symmetry (TS) is obtained by calculating the ratio of the variability *about* the eigenvector to the variability *along* the eigenvector. The principal eigenvector that is extracted from the matrix describes the orientation of the distribution of points in matrix M such that the variability of M is maximized along the eigenvector.

This measure of trend symmetry which uses the principal eigenvector to analyze the variance of the distribution of the points formed by the pair of waveforms to be compared is beneficial because it provides a measure of symmetry that is not affected by

the difference in magnitude between the two waveforms. This is not possible with the discrete symmetry measure formerly introduced. In addition to being uninfluenced by the range of the observation pair, the trend symmetry measure also has the added benefit of utilizing the entirety of the waveform for its calculation. Unlike the other methods which calculate symmetry by computing the average symmetry of corresponding single values of the observation, the trend symmetry measure computes the symmetry of two waveforms using the entirety of the waveforms.

# Chapter 4

# Results

In this chapter, the results of the experiments described in chapter 3 for training a transfemoral amputee agent to have a human-like gait using proximal policy optimization are presented. The first section presents the results of hyperparameter tuning for the number of iterations, the complexity of the policy and value networks, and the number of discrete prediction categories. In the second section, the results of muscle information prediction are provided. The third section contains the results of the models trained using the complete, augmented, and reduced observation states. The remaining sections offer further results relating to the gait realized from the models.

## 4.1 Hyperparameter search

### 4.1.1 Experiment for number of iterations

The first important hyperparameter to select in the course of training a machine learning model is a suitable amount of learning iterations to perform. Figure 4.1 shows the total reward and timesteps for each episode during training. It also shows the average reward and number of timesteps for an episode for each iteration.

It can be observed that there is a steep increase in the average reward obtained in the first 500 iterations of the DRL task (Figure 4.1c). This is also reflected in the steady increase in the total rewards obtained in the first $\sim 39,000$ episodes of the simulation (Figure 4.1a). This steep increase is accompanied by a similar increase in the number of timesteps performed by the model (Figure 4.1b,4.1c). After this number of performed iterations and episodes, there is no improvement in the total timesteps. This is expected as the maximum number of timesteps is bounded by the size of the imitation data (1050).

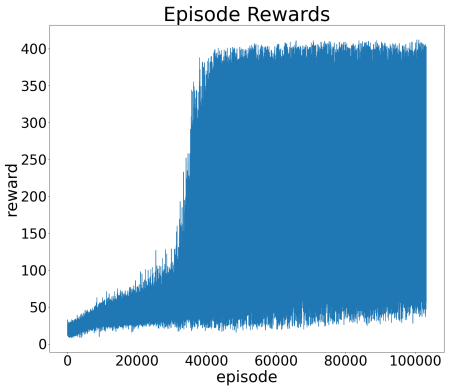
There is noticeable variance in the episodic total reward and timesteps (Figure 4.1a,4.1b) after the steep learning phase. Reward values ranging from $\sim 20$ to 400 can be observed and total timesteps values $\sim 20$ to 1050 can also be noticed. This is due to exploration by the learning algorithm even after finding a solution that completes the learning task. The stochastic exploratory actions try to find an even better solution but as can be observed in Figure 4.1c, a better solution cannot be reliably found as there is no increasing trend in the learning curve for the realized reward per iteration even after $2,500$ iteration. On the contrary, there is a slightly decreasing trend in both the average rewards and timesteps per iteration after the first 700 iterations. This is likely due to the learning algorithm trying to further maximize its immediate rewards at a detriment to the robustness of the model, hence showing early signs of overfitting to the imitation data. For this reason, 700 is an adequate number of iterations to perform in training the agent using PPO and this number of iterations is used in further experiments.

### 4.1.2 Experiment for number of hidden units

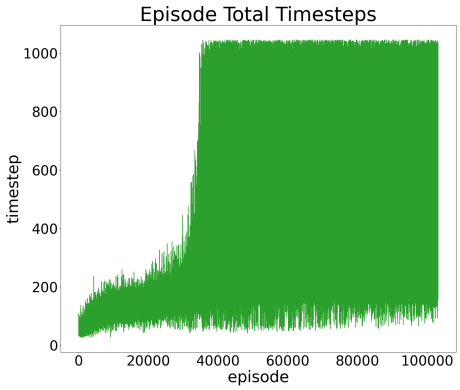
To obtain an adequate complexity for the policy and value networks of the PPO algorithm, several policies were trained on the DRL task using different numbers of hidden units. Table 4.1 shows the results of the different hidden unit sizes. The policy network was trained for 700 iterations of PPO and the rewards for 50 episodes were averaged. A policy and value network with 228 hidden units resulted in the highest average reward, performing better than the more complex architecture of 312 hidden units. 312 hidden units have been used in previous research (De Vree & Carloni, 2021) on healthy and transfemoral prosthesis agents with impressive results. That research, however, used a significantly larger observation space; hence there is an incentive to train with a smaller policy and value neural network size. This suspicion is indeed rewarded with better performance (163.5) on the less complex, 228 hidden units network in comparison to the 312 hidden units network which had an average reward of 158.3. The 100 hidden units network had the lowest performance on the task with an average reward of 156.2 over 50 episodes.

Table 4.1: Average episode reward and training duration for the different number of hidden units in the policy and value networks for training with PPO. The episode rewards are averaged over 50 episodes.

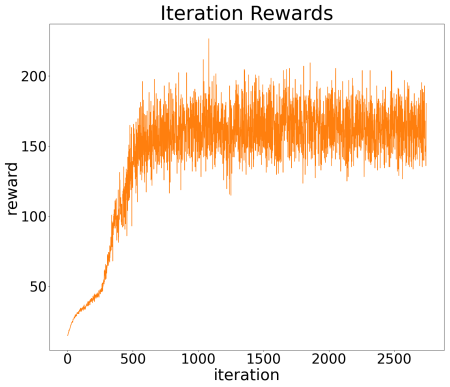
| **Number of Hidden Units** | **Average episode reward** | **Training duration (hours)** |
|---|---|---|
| 100 | 156.2 | 51.9 |
| 228 | 163.5 | 52.2 |
| 312 | 158.3 | 53.3 |



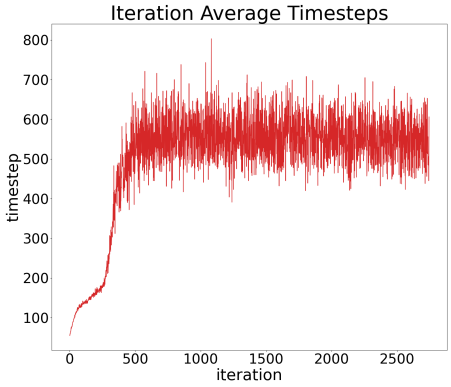

(a) episodic reward




(b) timesteps per episode




(c) average episode reward per iteration




(d) average episode timesteps per iteration

Figure 4.1: rewards and timesteps per episode and iteration. The top row shows the rewards and duration per episode. The bottom row is the average episodic reward and average episode duration during each training iteration.

### 4.1.3 Experiment for number of prediction categories

Table 4.2 shows the average reward per episode for different numbers of prediction categories. This was done by training for 700 iterations of PPO and averaging the episode rewards for 50 episodes. It can be observed that a policy network with 5 prediction categories had the best performance on the task. It resulted in an average reward of 198.7. The next highest average rewards are for the 3 and 7 prediction categories models, with average episode rewards of 194.3 and 187.0, respectively. Other models have a significantly lower reward yield with less than 170 average episode reward. A model with 5 prediction categories was used for subsequent experiments since it produced the best performance on the hyperparameter selection

task.

Table 4.2: Average episode reward and training duration for the different number of prediction categories for training with PPO. The episode rewards are averaged over 50 episodes after 700 iterations of PPO.

| **Number of prediction categories** | **Average episode reward** | **Training duration (hours)** |
|---|---|---|
| 2 | 163.5 | 52.2 |
| 3 | 194.3 | 52.4 |
| 5 | 198.7 | 54.9 |
| 7 | 187.0 | 55.3 |
| 9 | 169.9 | 56.8 |
| 15 | 153.1 | 58.9 |
| 21 | 162.8 | 63.9 |

## 4.2 Muscle Information Prediction

As described in subsection 3.5.2, 5-fold cross-validation was used for hyperparameter selection with respect to the hyperparameter values described in Table 3.4. The five best-performing models out of the 75 models trained on the supervised learning task are listed in Table 4.3. The performance of the models are judged by observing the cross-validation loss - mean absolute error. It was observed that a fully-connected model with 3 hidden layers, 256 hidden units, batch normalization, and drop-out had the best performance on the task. The model had a mean absolute error of 0.11 and a visualization of this model's architecture can be seen in Figure 4.2.

Table 4.3: Hyperparameter values and cross-validation mean absolute error for five best performing models on the muscle information prediction task.

| **Hyperparameter values** | | | | | **cross-validation loss** |
|---|---|---|---|---|---|
| hidden layers | hidden units | batch norm. | drop out | activation | |
| 3 | 256 | ✓ | ✓ | ReLU | 0.110 |
| 3 | 512 | ✓ | × | ReLU | 0.112 |
| 2 | 512 | ✓ | ✓ | ReLU | 0.112 |
| 3 | 512 | × | × | ReLU | 0.114 |
| 2 | 256 | ✓ | ✓ | ReLU | 0.114 |

The described best-performing muscle information prediction model is then trained using the entirety of the training (cross-validation) data as shown in Figure 3.5,


Figure 4.2: Architecture of the best performing muscle information prediction network. Green blocks represent fully-connected layers with the respective number of units displayed. Yellow blocks represent batch normalization, and blue blocks represent drop-out regularization.

using 90% of the entire data for training and 10% for testing. This resulted in a model trained with a validation mean absolute error of 0.109. A plot of the mean absolute error on training and validation data, during the course of training, can be seen in Figure 4.3. This model had an average MAE of 0.082 on the prediction of the normalized muscle forces for the 15 muscles. It also had a 0.018 MAE for the prediction of the normalized length of the muscles and a 0.228m/s MAE for the prediction of the muscles' velocities. A breakdown of the MAE for the individual muscles can be seen in Table 4.4.

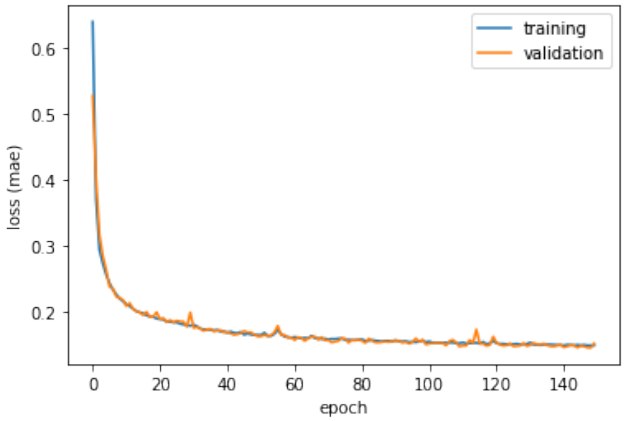



Figure 4.3: Training for the best-performing model for the muscle information prediction task. The model is a fully-connected feed-forward network with 3 hidden layers, 256 hidden units, batch normalization, and drop-out.

## 4.3 Comparison of Results for Models Trained with Different Observation Types

This section compares the emerging rewards for training models using the complete, augmented, and reduced observation spaces. Models are trained for 700 iterations

Table 4.4: Breakdown of muscle information prediction for each muscle.

| **Muscle** | **Prediction Loss (MAE)** | | |
|---|---|---|---|
| | Force (normalized) | Length (normalized) | Velocity (m/s) |
| Hip Abductor | 0.111 | 0.019 | 0.166 |
| Hip Adductor | 0.099 | 0.032 | 0.445 |
| Iliopsoas L | 0.127 | 0.012 | 0.390 |
| Glut max R | 0.081 | 0.037 | 0.488 |
| Hamstring R | 0.006 | 0.006 | 0.009 |
| Rectus Femoris R | 0.161 | 0.022 | 0.296 |
| Vasti R | 0.074 | 0.007 | 0.142 |
| Bifmesh R | 0.082 | 0.014 | 0.369 |
| Gastrocnemius R | 0.007 | 0.007 | 0.012 |
| Soleus R | 0.006 | 0.007 | 0.011 |
| Tib ant R | 0.074 | 0.014 | 0.346 |
| Hip abductor L | 0.120 | 0.018 | 0.523 |
| Hip adductor L | 0.143 | 0.013 | 0.199 |
| Glut max L | 0.006 | 0.008 | 0.009 |
| Iliopsoas L | 0.005 | 0.006 | 0.010 |
| **Average** | 0.082 | 0.018 | 0.228 |

of PPO, using different numbers of prediction categories, and the average reward over 50 episodes is recorded. Table 4.5 contains the described average episodic reward.

It can be observed that models trained for all observation types have the lowest performance whilst utilizing 2 discrete prediction categories. Training the complete observation produces the most episodic reward on average when using a model with 5 prediction categories. This emerges with an average reward of 198.7 per episode. Using an observation state that is augmented with muscle information prediction of the feed-forward neural network also has its highest average episodic reward while using a model with 5 prediction categories. This generates an average episodic reward of 190.9. Models trained using the reduced observation space, unlike the other models discussed, generated the most average reward per episode when trained with 3 prediction categories.

Table 4.5: Average episodic reward for the different number of prediction categories and observation space types. The rewards are averaged over 50 episodes of the agent's locomotion and using a model trained for 700 iterations.

| **Number of prediction categories** | **Average episode reward** | | |
|---|---|---|---|
| | complete observation | augmented observation | reduced observation (without muscle information) |
| 2 | 163.5 | 166.7 | 142.1 |
| 3 | 194.3 | 185.2 | 161.2 |
| 5 | 198.7 | 190.9 | 160.5 |

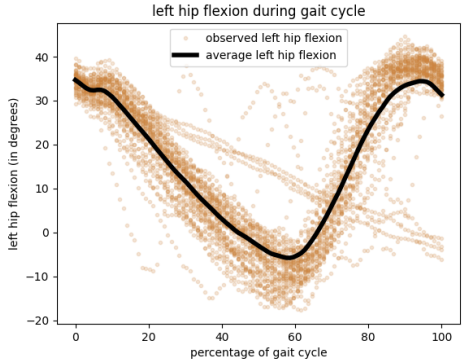
## 4.4 Realized Gait

To investigate the symmetry of the emerging gait pattern using the best-performing DRL model for each observation type (as described in section 4.3), the flexion of different joints during the gait cycle are analyzed. This section reviews the hip, knee, and ankle flexion for instantaneous gait cycles, as well as the average progression of the flexion values during the gait cycle for the reduced and augmented states.

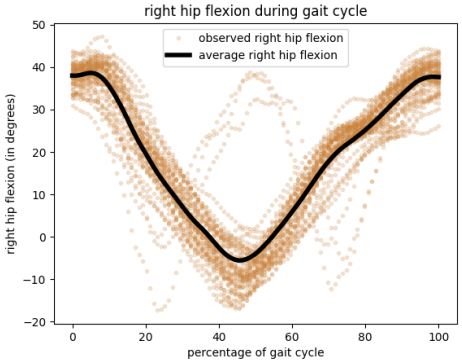
### 4.4.1 Complete observation

The realized gait for training the best performing model which consists of 228 hidden units, and 5 prediction categories and was trained for 700 iterations, as described in the hyperparameter search can be seen in Figure A.4. After 40 iterations of training, the policy is unable to coordinate the actions of the model in a way that generates a gait pattern. It is still unable to generate a gait after 100 iterations but it learns to move a leg forward but falls afterward. The agent is still unable to move forward after 200 iterations. After 400 iterations of training, the model learns a policy that generates a human-like gait. However, at this stage of training, the model is still quite unstable and falls often, without completing its task. At 700 iterations, the model learns a more robust manner of human-like gait and completes the task.

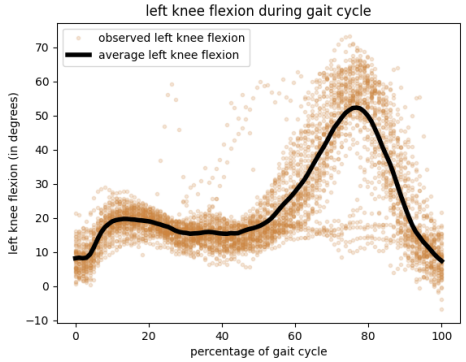
Figure 4.4 shows the instantaneous and average hip, knee, and ankle flexion for the left and right legs of the agent during the gait cycle. In Figure 4.4a and Figure 4.4b, we observed high symmetry between the left and right hip during locomotion. There is little variance in the emerging pattern for the instantaneous gait cycles. This means that the agent exhibits a natural walking pattern without much deviation during the task. The same can also be observed in Figure 4.4c and Figure 4.4d for the left and right knee flexion, respectively. Only a few deviations are observed with a peak at about 20% to 40% of the gait cycle. There is very high



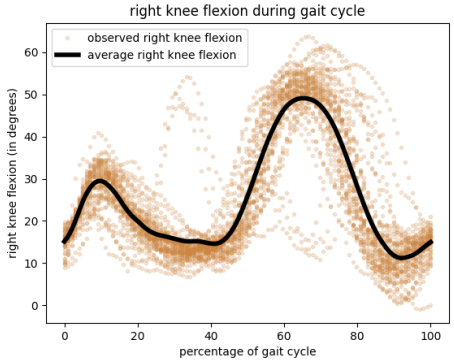

(a) left hip flexion



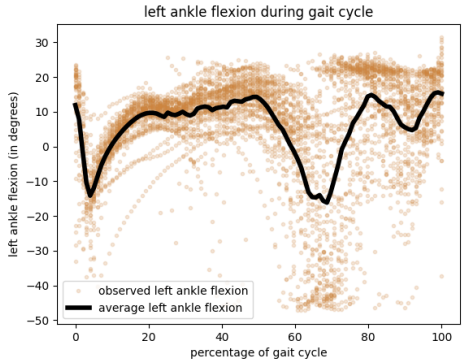

(b) right hip flexion



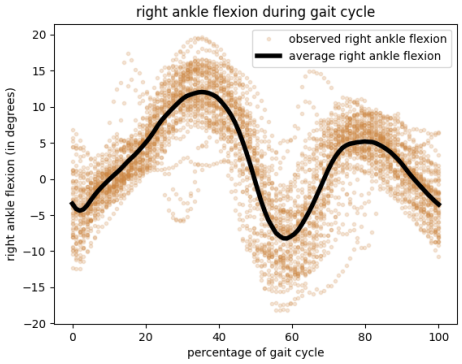

(c) left (prosthesis) knee flexion




(d) right (healthy) knee flexion




(e) left (prosthesis) ankle plantar flexion




(f) right (healthy) ankle plantar flexion

Figure 4.4: Left and right hip, knee, and ankle angles during the gait cycle using the complete state observation. The policy network has 228 hidden units, and 5 prediction categories and was trained for 700 iterations of PPO.

symmetry between the healthy and the prosthesis knee during the gait cycle, resulting in a natural walking pattern. There is, however, much less symmetry between

the ankle flexion of the healthy and the prosthetic leg. The healthy leg generates a normal ankle flexion angle during the gait cycle. The ankle flexion shows a slightly similar average flexion during the gait cycle; with an initial rise and peak at $\sim 30\%$ of the gait cycle, followed by a dip at $\sim 65\%$ of the gait cycle. The symmetry of the ankle angles is much lower than the other joints. There is also very great variation in the ankle flexion during different gait cycles for the prosthesis leg. The symmetry is further analyzed in section 4.5.

### 4.4.2 Reduced observation

The hip, knee, and ankle flexion for both legs of the agent during 50 gait cycles is shown in Figure 4.5. For the left (prosthetic) leg, the hip and knee flexion during a gait cycle has a noticeable pattern (Figure 4.5a, Figure 4.5c). There is much higher variance in the gait of the agent when compared to the hip flexion for the model trained with the complete observation state. However, there is a trend nonetheless.

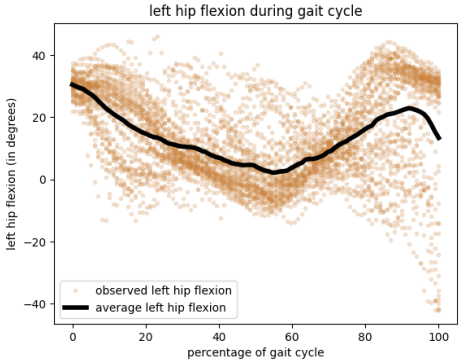
The left (prosthesis) ankle flexion of the agent during gait cycle (Figure 4.5e) shows no clear trend. This is however, similar to the results obtained for the model trained with the complete observation state (Figure 4.4e). The PPO model does not learn a policy that streamlines the actions performed by the agent's ankle to fit a particular gait pattern. Rather, it learns a policy robust enough to perform locomotion but this comes at a cost of human-like movement for that joint and decreased similarity to the imitation data. Similarly, the right (healthy) ankle of the agent has high variation in its flexion during the gait cycle (Figure 4.5f). This is, however, very different from the clear pattern observed when using the model trained with the complete observation state (Figure 4.4f).

In general, there is much more consistency in the gait pattern for the prosthetic leg than for the healthy leg while training with the reduced observation state. This suggests a dependence on the missing muscle information in order to generate a better gait pattern.

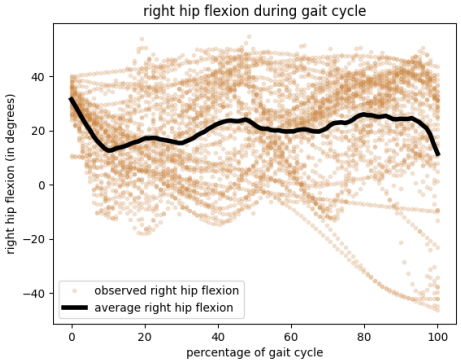
### 4.4.3 Augmented observation

Figure 4.6 shows the progression of the flexion of the hip, knee, and ankle joints of the agent for 50 gait cycles. The results show a clear trend in the left and right hip and knee flexion during the gait. The results also show little variation in the values of the hip and knee angle during the gait cycles in comparison to the gait obtained from the reduced observation state model. However, there is more variation compared to the gait for the model trained with the complete muscle information.

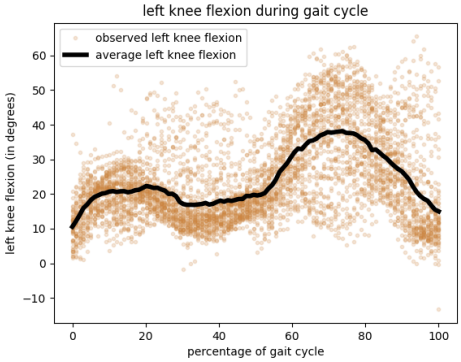
The right (healthy) ankle plantar flexion (Figure 4.6f) reveals a pattern for the gait cycles with a few observable deviations from the mean. The left (prosthetic) ankle, however, does not form as straightforward, a gait pattern. Similar to the results



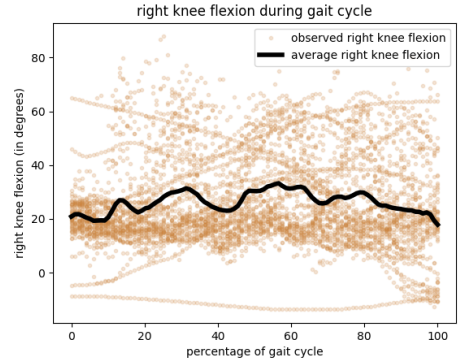

(a) left hip flexion



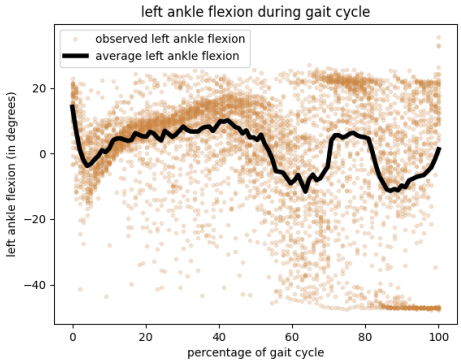

(b) right hip flexion



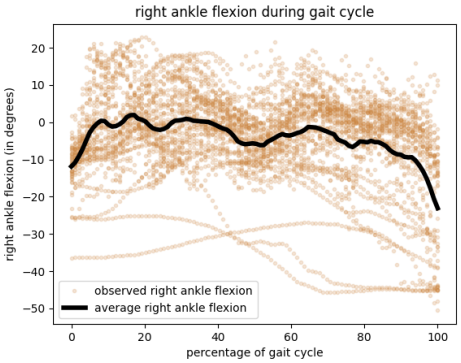

(c) left (prosthesis) knee flexion




(d) right (healthy) knee flexion



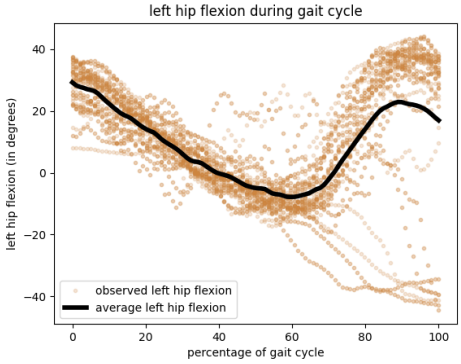

(e) left (prosthesis) ankle plantar flexion



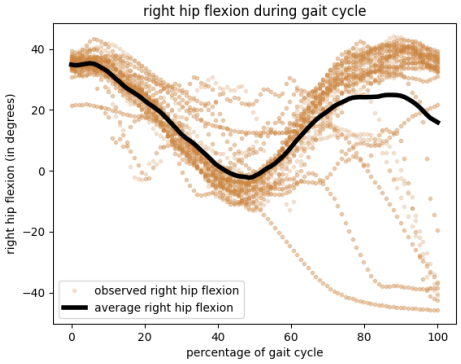

(f) right (healthy) ankle plantar flexion

Figure 4.5: left and right hip, knee, and ankle angles during the gait cycle using the reduced state observation (no muscle information). The policy network has 228 hidden units, and 5 prediction categories and was trained for 700 iterations of PPO.

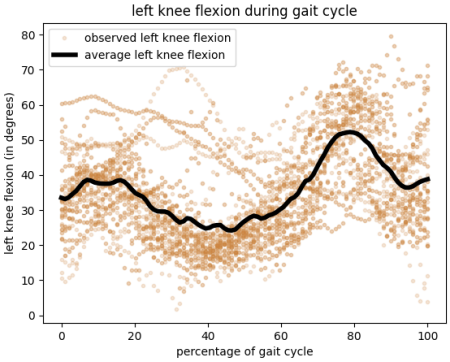
obtained for the left ankle flexion of the model trained with the complete state (Fig-



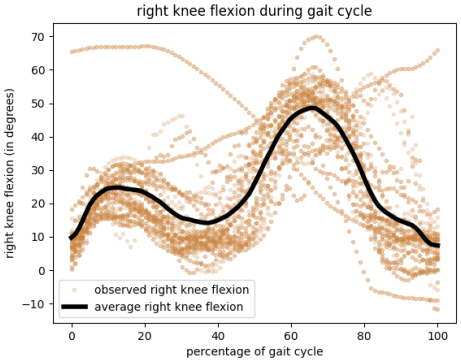

(a) left hip flexion



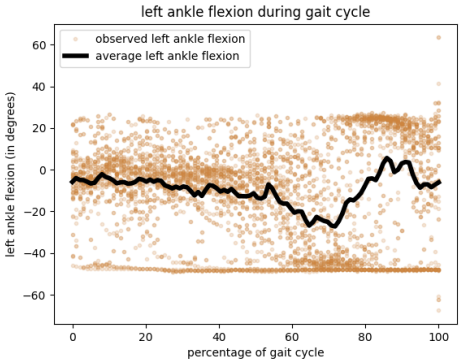

(b) right hip flexion



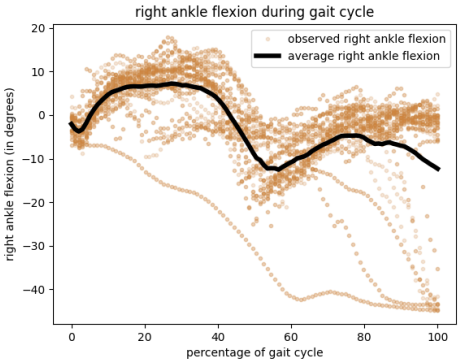

(c) left (prosthesis) knee flexion




(d) right (healthy) knee flexion




(e) left (prosthesis) ankle plantar flexion




(f) right (healthy) ankle plantar flexion

Figure 4.6: left and right hip, knee, and ankle angles during the gait cycle using the augmented state observation (muscle information predicted by feed-forward neural network). The policy network has 228 hidden units, and 5 prediction categories and was trained for 700 iterations of PPO.

ure 4.4e), the emerging pattern contains high variance. There is, however, still a similar trend for both ankles with the minimum flexion occurring at $\sim 60\%$ to $80\%$ of the gait cycle.

## 4.5 Symmetry of Models

Two quantities are observed in order to evaluate the quality of the realized gait. Firstly, the symmetry of the left (prosthetic) and right (healthy) legs are observed. This is done by computing the average RMSE, the symmetry angle, and the trend symmetry for the observed mean hip, knee and ankle flexion during the gait cycle. Similarly, the similarity of the listed flexion quantities during the gait cycle is compared to that of the imitation data.

### 4.5.1 Left - right leg symmetry

Table 4.6 shows the RMSE, symmetry angle and trend symmetry for left and right - hip, knee, and ankle flexion for the imitation data and models trained with complete, reduced, and augmented observation state. The imitation data has a hip flexion trend symmetry of 0.01. In comparison, the model trained with the complete observation produced a hip flexion trend symmetry of 0.04. This is the most symmetrical model (as described by the trend symmetry) for the hip and ankle flexion.

The model trained using the augmented observation state has the best performance of all the trained models in terms of the knee flexion trend symmetry with a TS score of 0.05. It has slightly worse performance than the complete observation state model for the hip and ankle flexion trend symmetry. The reduced observation state model produces the worst symmetry scores of the models. The observed hip, knee, and ankle trend symmetry of the left and right legs were; 0.46, 0.45, and 0.37, respectively. These are significantly worse than the left-right trend symmetry of the imitation data, and the other observation state models. The ankle flexion trend symmetry during the gait cycle shows a significantly worse symmetry score than the hip and knee TS score for the complete and augmented observation space models.

### 4.5.2 Model - imitation data symmetry

Similar to the results presented in the previous section, the models trained with the complete and augmented observation states produced better symmetry with the imitation data gait cycle than the model trained using the reduced observation state. Table 4.7 shows the results of the hip, knee, and ankle flexion symmetry for the average gait cycle, and the imitation data. It shows good hip flexion symmetry (TS) in both legs for the complete and augmented observation space models. The reduced observation model also has a high level of symmetry in the left leg (TS = 0.04). However, there is much worse trend symmetry with the imitation data for

Table 4.6: Symmetry of the joints in the left and right leg during gait.

| **Attribute** | **Symmetry Measure** | **Model** | | | |
|---|---|---|---|---|---|
| | | imitation data | complete observation | reduced observation | augmented observation |
| Hip Flexion | RMSE | 2.45 | 6.08 | 10.56 | 9.70 |
| | SA | 12.64 | 25.89 | 18.11 | 25.46 |
| | TS | 0.01 | 0.04 | 0.46 | 0.07 |
| Knee Flexion | RMSE | 0.46 | 10.92 | 9.94 | 15.11 |
| | SA | 0.67 | 10.03 | 10.94 | 16.36 |
| | TS | 0.01 | 0.06 | 0.45 | 0.05 |
| Ankle Flexion | RMSE | 0.31 | 8.38 | 10.40 | 8.30 |
| | SA | 2.02 | 44.13 | 78.35 | 69.90 |
| | TS | 0.02 | 0.19 | 0.37 | 0.20 |

the reduced state model in the right leg (TS = 0.25). This can also be observed in the knee and ankle flexion trend symmetry of the reduced observation state model. The left leg shows high symmetry with the imitation data but the right (healthy) leg shows a much worse trend symmetry with the imitation data.

The complete observation state model generates a gait with a better hip and ankle symmetry than the augmented observation state model. However, the model trained with the augmented observation state has a better ankle flexion trend symmetry with the imitation data than the complete observation state model for both legs. Both models have relatively high symmetry with the imitation data for the hip and knee flexion during the gait cycle. There is, however, an observable decrease in symmetry with the imitation data, for the flexion of the ankles. The average flexion of the joints during the gait cycle which was used to obtain these results can be seen in Figure 4.7.

## 4.6 Kinetic Analysis

Additional analysis was done on the model trained with the augmented observation state. This includes investigating the stiffness of the prosthetic knee and ankle joints during gait, as well as observing the force of the agent's muscles during the phases of the gait cycle. A gait cycle is made up of 8 phases: initial contact, loading response, mid stance, terminal stance, pre-swing, initial swing, mid swing, and terminal swing. The first five of the mentioned phases are known as the stance phase of the gait and the latter three listed phases are the swing phase of the gait.

Figure 4.9 shows the normalized knee and ankle actuator torques during a gait cycle

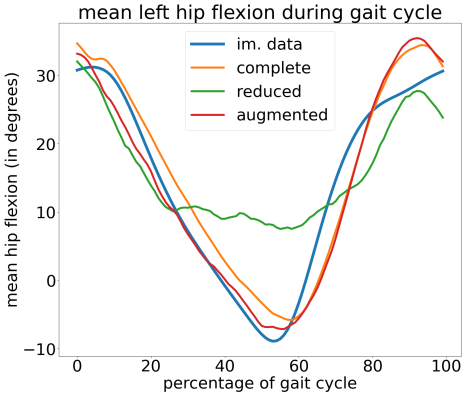
Table 4.7: Symmetry measures for models (trained with different observation spaces) and the imitation data. The average left and right, hip, knee, and ankle flexion of models are compared to the imitation data.

| **Leg** | **Attribute** | **Error Measure** | **Model** | | |
|---|---|---|---|---|---|
| | | | complete observation | reduced observation | augmented observation |
| Left | Hip Flexion | RMSE | 4.02 | 8.06 | 3.87 |
| | | SA | 19.75 | 49.31 | 13.48 |
| | | TS | 0.02 | 0.04 | 0.02 |
| | Knee Flexion | RMSE | 7.44 | 9.45 | 11.99 |
| | | SA | 8.99 | 10.39 | 12.56 |
| | | TS | 0.07 | 0.04 | 0.10 |
| | Ankle Flexion | RMSE | 8.44 | 5.36 | 8.02 |
| | | SA | 57.13 | 42.80 | 62.96 |
| | | TS | 0.16 | 0.07 | 0.12 |
| Right | Hip Flexion | RMSE | 6.48 | 16.70 | 9.54 |
| | | SA | 24.55 | 34.40 | 28.27 |
| | | TS | 0.04 | 0.25 | 0.08 |
| | Knee Flexion | RMSE | 7.46 | 16.69 | 9.42 |
| | | SA | 8.58 | 13.94 | 10.06 |
| | | TS | 0.07 | 0.41 | 0.08 |
| | Ankle Flexion | RMSE | 6.48 | 8.61 | 6.87 |
| | | SA | 44.61 | 55.00 | 36.33 |
| | | TS | 0.20 | 0.33 | 0.17 |

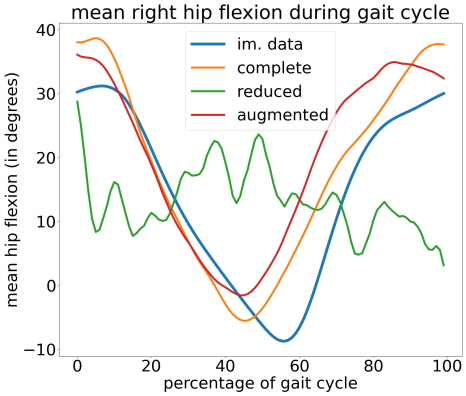
and Figure 4.8 shows the normalized knee and ankle torques corresponding to the angle of the joint during the gait cycle. The torques shown are normalized using the amputee agent's weight. The gait phases are also highlighted to give a better description of the torques. These torque values have been filtered and smoothened. This is because of the sporadic nature of the actuator control used. Due to the lack of constraint on the output of the neural network, there is freedom for the network to result in highly fluctuating activations of the actuators. This results in bursts of force and undulating torques during the gait cycle. To present a clearer depiction of the knee and ankle torques, a moving average filter was applied to the instantaneous normalized torque waveform using a window size of 0.11 seconds. Linear extrapolation was used to fill the truncated parts of the waveform that resulted from the moving average filter. A Savitzky-Golay filter with a polynomial order of 3 was then applied to smoothen the result.

Figure 4.10 shows box plots of the muscle forces during the phases of a gait cycle for the 11 muscles in the right (healthy) leg of the amputee agent. The combination

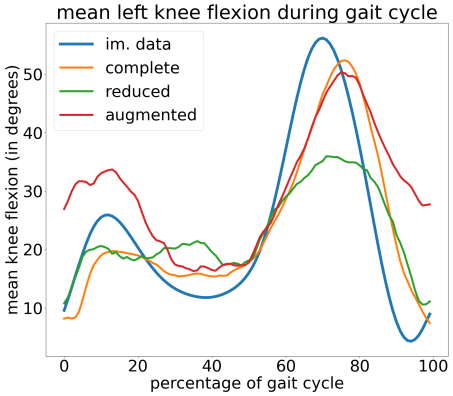
of the actuator stiffness and the muscle forces during the gait informs us of the practicality of the solution outside of the simulation and would be discussed further in the next chapter.



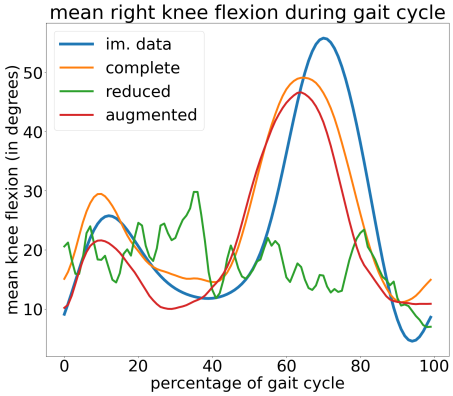

(a) left hip flexion



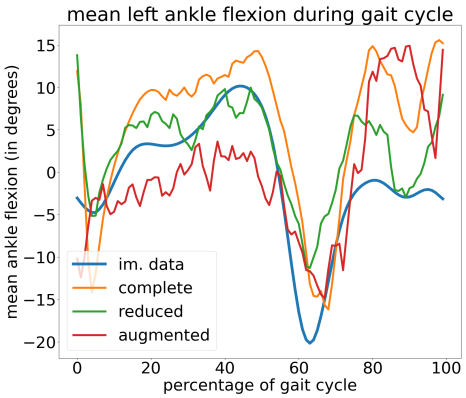

(b) right hip flexion



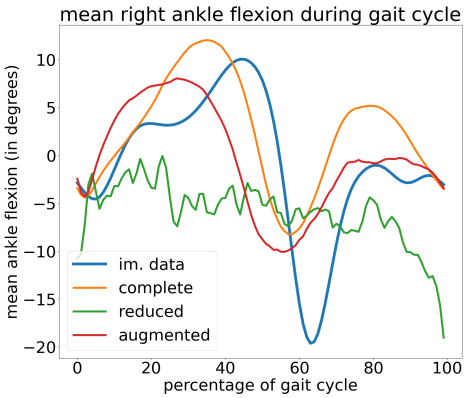

(c) left (prosthesis) knee flexion




(d) right (healthy) knee flexion

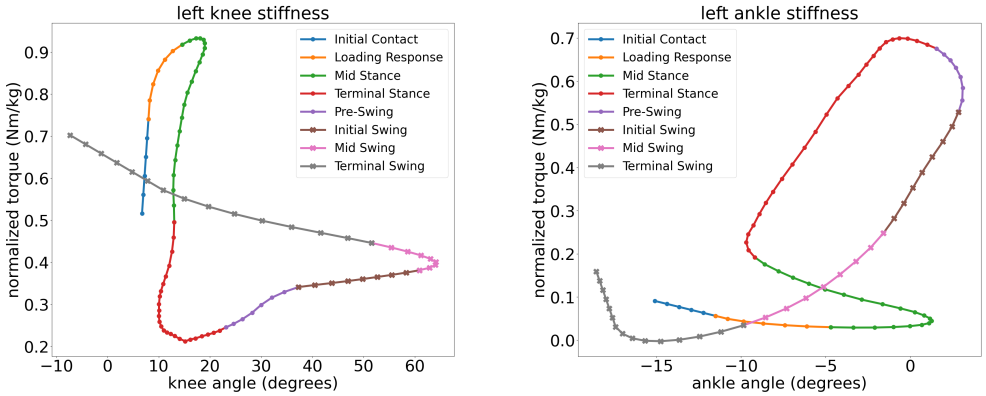



(e) left (prosthesis) ankle plantar flexion



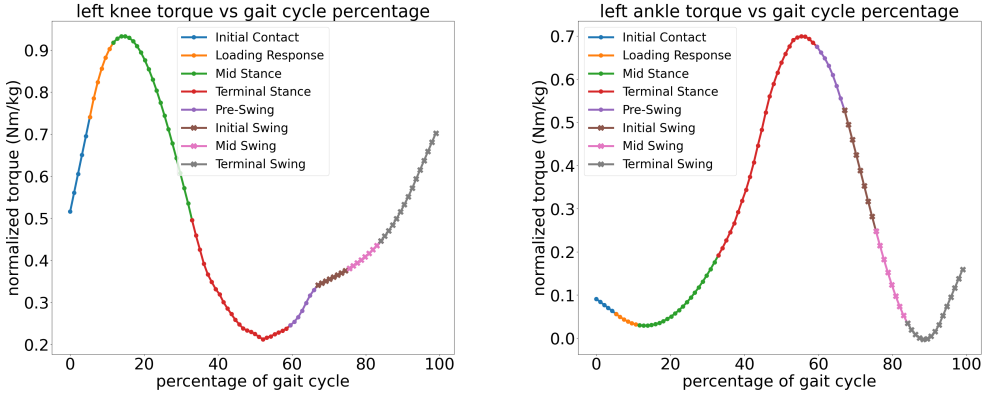

(f) right (healthy) ankle plantar flexion

Figure 4.7: average left and right hip, knee, and ankle angles during the gait cycle using the complete, reduced and augmented observation models, as well as the imitation data.



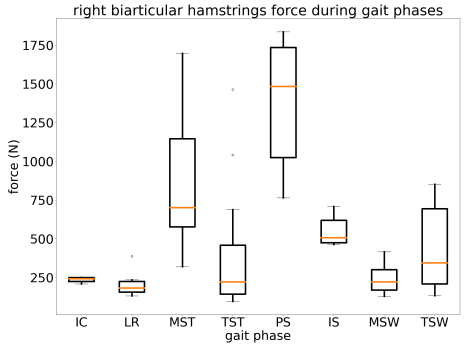

(a) knee (b) ankle

Figure 4.8: Knee and ankle actuator stiffness during gait cycle

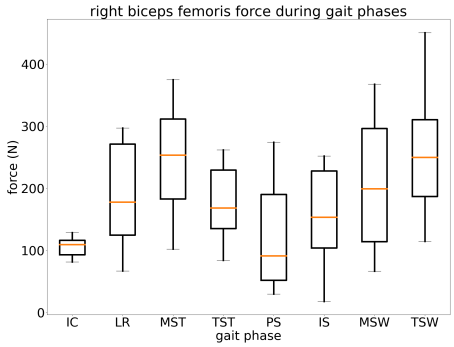



(a) knee (b) ankle

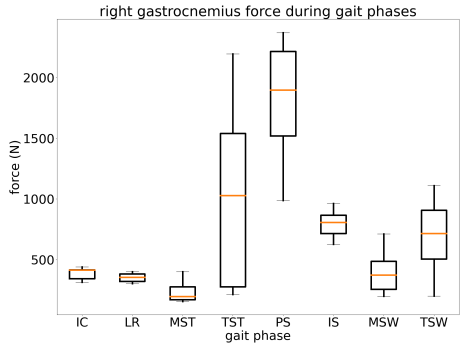
Figure 4.9: Knee and ankle actuator torque during gait cycle



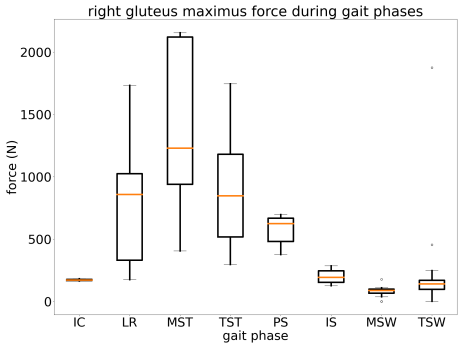

(a) biarticular hamstrings



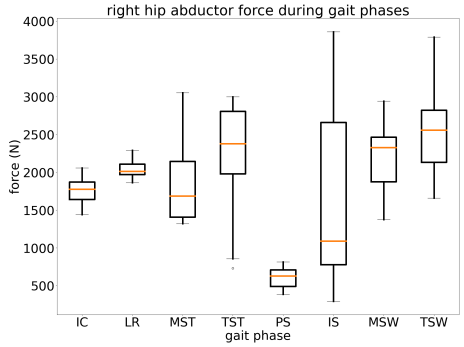

(b) biceps femoris



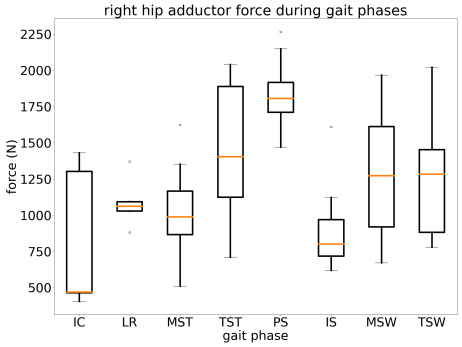

(c) gastrocnemius




(d) gluteus maximus

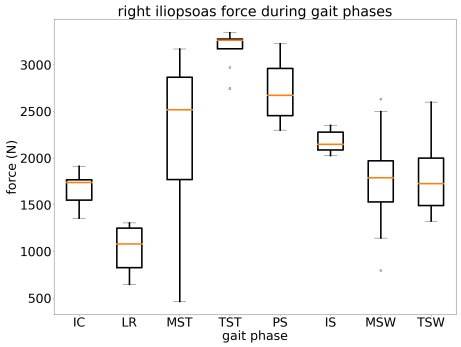



(e) hip abductor

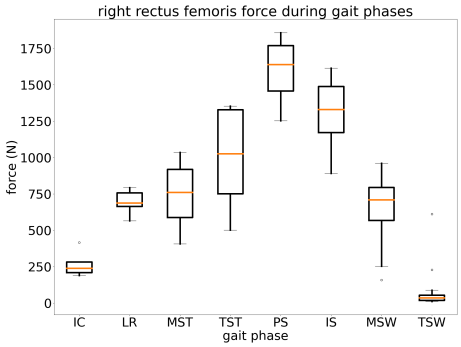



(f) hip adductor

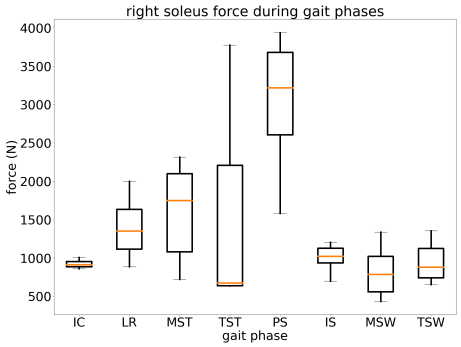
Figure 4.10: Muscle force during phases of gait cycle (*Continued on next page*)



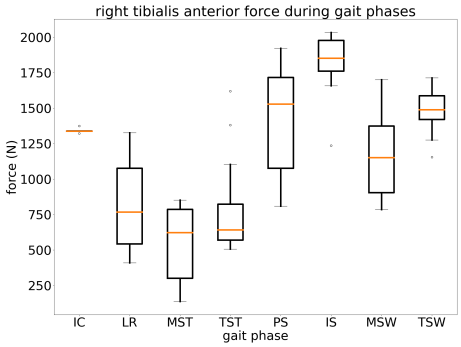

(g) iliopsoas



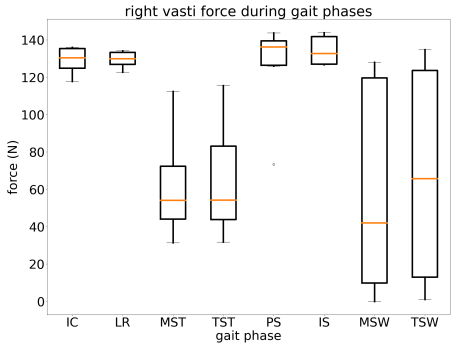

(h) rectus femoris




(i) soleus




(j) tibialis anterior




(k) vasti

Figure 4.10: Muscle force during phases of gait cycle (*continued*)

# Chapter 5

# Discussion and Conclusion

This chapter first discusses the results obtained in this research. It then summarizes the obtained results and their implications and provides answers to the questions which prompted this research. Recommendations for further research within this line of study are provided and finally, concluding remarks are made.

## 5.1 Discussion

### 5.1.1 Hyperparameter selection

It was observed that models trained with a higher number of prediction categories (above 7) generated lower rewards than using 3 - 7 prediction categories. This is likely due to an increase in the action space, requiring more iterations to find a good solution for the DRL task. The 700 iterations chosen from the number of iterations selection were on a 2 prediction categories model which had a much smaller action space. These larger action space models are more difficult to train because of the increased search space for the policy to find a good solution. This means that a larger number of iterations would be required to adequately train them. This would also imply a much larger training duration without a guarantee of improved performance and also reflects on the possible challenges in training the model using a continuous action space.

Using 2 prediction categories also resulted in a lower realized average episode reward. This is likely due to the mechanism of the model. There is less control over the agent's actions. The 15 muscles of the agent are either maximally exciting or maximally contracting at each timestep. The knee and ankle actuators are also maximally actuated in either direction with respect to the action generated by the policy. There is no *do-nothing* state for the action of the agent to represent a good current state. Hence, there is a highly erratic pattern of activation of the agent. This is very

evident in the activations of the actuators.

### 5.1.2 Analysis of model results

The results of the models trained using the complete, reduced, and augmented state observations were presented in different ways. First, the average reward of 50 episodes of the trained model was obtained. To further evaluate the models, the symmetry of both legs, as well as the symmetry with imitation data was calculated.

While training the PPO model using 2, 3, and 5 prediction categories, training with the complete observation state generated the best performance (obtained reward) for the agent. The augmented observation state also yielded adequate performance on the locomotion task. It even outperformed the complete observation state model in one instance (2 prediction categories). There was, however, a drop-off in the performance of the model when using the reduced observation for the agent. This large disparity in rewards generated from training with the reduced state and the other observation states suggests that the muscle information or at least an approximation of it is important in the state representation of the agent in order to adequately perform locomotion with the DRL framework.

Over 50 episodes of the simulation, the hip, knee, and ankle flexion for the right (healthy) leg did not have a clear trend during the gait cycle (Figure 4.5) unlike in the left (prosthetic) leg when trained with the reduced state. There was high variation in the flexion progression for different gait cycles. The lack of coordination of the right (healthy) leg by the reduced state model is further reflected by the lower symmetry with the imitation data of the hip, knee, and ankle for that leg. For this reduced state model, the symmetry for the left leg and the imitation data is much higher than the symmetry of the right leg. This result is expected because there are a lot more muscles in the healthy leg and the reduced state observation does not have information on these values in its description. The learning agent thus learns a robust way of maximizing rewards. This learned policy for the reduced state model is not well-correlated with the imitation data and causes the agent to have a less symmetrical and human-like gait in comparison to the model trained using the complete and augmented observation states.

The stance phase represents about 60% of the gait cycle (Laribi & Zeghloul, 2020). This is true for the imitation data (Figure 4.7f). It is also observable in the left (prosthetic) leg of the model trained with the complete, augmented, and reduced observation states. There is a steep increase in the hip flexion from approximately 60% of the gait cycle for the rest of the gait cycle (the swing phase). This corresponds to the agent's foot leaving the ground, the hips swinging the leg forward, and finally, the contact of the feet and the ground. In the right (healthy) leg, however, the stance phase occurs on average, for only 50% of the gait cycle. This is due to the quicker motion of the prosthesis in comparison to the healthy leg. During the

mid and terminal swing phases, the prosthetic leg is able to swing faster than the healthy leg. This is consistent for both the complete and augmented observation state models and can be due to a number of factors. One reason for this could simply be the lighter mass of the prosthesis in the simulation. The imitation data used to train the model was obtained from able-bodied adults. Transfemoral amputee users of mechanical and microprocessor-controlled prostheses have been observed to have such asymmetry in their gait (Kaufman et al., 2012).

Overall, the variance of the hip, knee, and ankle flexion for the 50 episodes of simulation of the model that is trained with the augmented state is much more similar to that of the model trained using the complete state. The model has a more natural gait pattern when trained by augmenting the missing muscle information with a pre-trained neural network than when trained without the muscle information; again, indicating the significance of the muscle information or an approximation of it, in the observation state in order to generate a healthy gait pattern.

### 5.1.3 Real-world Performance

The proposed method of augmentation of muscle information using a neural network can be used in real-time applications because it is not computationally expensive. To further evaluate the practicality of the utilized methods outside the simulation environment, the muscle force of the agent as well as the actuator stiffness was analyzed during a gait cycle of the simulation using the augmented observation state. This approach is practical because of the gross accuracy of the Hill-type muscle model of Opensim.

#### Actuator stiffness

Understanding the stiffness behavior of joints could help in further understanding the locomotion of an agent (Günther & Blickhan, 2002). The investigation of the knee and ankle stiffness of the simulated prosthesis is important for improving the prosthesis and the device design. The resulting normalized torques in Figure 4.8 are consistent with results obtained from human studies. Kern et al. (2019) found a maximum normalized ankle torque of less than $1.5 Nm/kg$ for 20 participants for a walking task with an added mass of up to 30% of the participant's mass. It is also consistent with the results of Günther & Blickhan (2002) where the maximum normalized knee and ankle torques were $2.02 Nm/kg$ and $3.53 Nm/kg$, respectively for a running task. In this study, the normalized maximum observed normalized knee and ankle torque during the gait of the agent is $0.93 Nm/kg$ and $0.70 Nm/kg$, respectively. The value of these joint torques lies within a reasonable magnitude for the knee and ankle torque during a gait. It is, however, unconventional for the observed maximum knee torque to be greater than the ankle torque. This is because the ankle should act as a lever for propulsion (Bojsen-Møller, 1979) and is normally the largest contributor to positive work (Farris & Sawicki, 2011). Hence, this result

also shows a relatively low stiffness in the prosthetic ankle joint. This could lead to more work done by the knee actuator and hip of the prosthesis user, in order to maintain a normal gait pattern. This can further be improved by methods such as reward shaping to encourage more work being done by the ankle actuator of the prosthesis.

### Muscle force

The realized muscle force of the agent during a gait cycle of the simulation is close to the real-world activation of muscles for different phases of the gait cycle.

During the initial contact (IC) phase of the gait cycle, the tibialis anterior muscle generates a high amount of force Figure 4.10j. This is expected because the tibialis anterior muscle's primary function is ankle flexion and extension and in the IC phase of the gait cycle, there is weight acceptance by the ankle. Similarly, there is an increase in the muscle force in the pre-swing (PS) phase, between the terminal stance (TST) and the initial swing (IS) phases. This gait phase is a transition between the stance and swing phase of the gait cycle and the foot is pushed and lifted off the ground. There is an increase in force for the muscles responsible for ankle flexion and extension (tibialis anterior and soleus) in this gait phase (Figure 4.10j & Figure 4.10i).

At the terminal stance phase, the iliopsoas muscle (Figure 4.10g) which contributes to the flexion of the hip joint records its maximum force with a median of over $3000N$. Similarly, the rectus femoris muscle (Figure 4.10h) also contributes to the flexion of the hip and increases its force in the TS and PS gait phases, in preparation for the swing phase of the gait cycle. Conversely, the generated force by the gluteus maximum (Figure 4.10d) reduces in the TST and PS phases and decreases further during the swing phase of the gait cycle. The gluteus maximum's primary function is hip extension. This is why to achieve locomotion this muscle's activation decreases during the swing phase of the gait, in order to achieve the necessary flexion in the hip for the swinging motion of the leg. This is also observed in the biarticular hamstrings (Figure 4.10b) which contribute to the flexing of the knee joint. For this muscle, much lower force values are observed in the swing phase of the gait.

In the early stages of the swing phase, there is a need for the knee's flexion to increase in order to have sufficient clearance from the ground to swing the leg forward. The biarticular hamstrings (Figure 4.10b) also contribute to the flexion of the knee and it can be observed that there is a significant increase in the force generated by the muscle during the PS phase. This is however not replicated in the biceps femoris which also contributes to the flexion of the knee. A reason for this is likely that the DRL agent learns a policy that mostly utilizes the biarticular hamstrings for the flexion of the knee but not the biceps femoris. Reward shaping can also be used to reduce the over-reliance on specific muscles such as the biarticular hamstrings by

incorporating some biomechanical information in the calculation of rewards.

## 5.2 Summary of Thesis

In this project, an architecture for training a biomechanical agent without access to muscle information was developed. This approach makes use of an artificial neural network in order to predict the agent's muscle information; the force, length, and velocity of the 15 muscles of the transfemoral amputee agent. DRL models optimized with the proximal policy optimization algorithm were trained using the complete observation state - which contains kinematic information and the force, length, and velocity of the agent's muscles; the augmented state - with kinematic data and the muscle information predicted by a pre-trained artificial neural network; and a reduced state - which had the agent's kinematic data but no muscle information included.

Firstly, a suitable artificial neural network architecture was chosen for the prediction task. This model was trained in a supervised learning scenario. The metric for selecting the model was the minimum MAE on the predicted force, length, and velocity values for the 15 muscles of the agent using 5-fold cross-validation. A deep neural network with 3 hidden layers produced the lowest cross-validation MAE loss with a value of 0.11. This network architecture had an MAE of 0.082N, 0.18m, and $0.228\text{ms}^{-1}$ on the muscle force, length, and velocity predictions respectively.

To train the PPO model, a fully-connected policy network with 228 hidden units was utilized. A similar neural network architecture was also used for the value function with the difference being the number of units at the output layer - the value function produces a single value hence, a single unit is used as the output of the network. 700 iterations of PPO resulted in a reward of 194.3 while training with the complete observation space. The augmented (kinematic data + predicted muscle information) and reduced (kinematic data without muscle information) observation states had 190.9 and 161.2 resulting rewards respectively. These results indicate that training the PPO model using the complete observation state of the model results in a higher obtained reward. The PPO model trained using the augmented muscle information also performs well; with a reward close to using the complete observation state. Conversely, the model trained using the reduced state which does not contain any muscle information performed significantly worse than the two other models as expressed by their rewards. It was however surprising that the reward obtained in an instance of training with the augmented observation state model outperformed the model trained with the complete observation. This slight increase in performance further supports the notion that the approximate prediction of the muscle information is sufficient to implement an intelligent control of the amputee agent.

Further analysis carried out on the realized gait produced results consistent with the obtained rewards of the models. The reduced state observation model showed a low level of symmetry between the right and the left leg during gait. This was shown to be mostly due to the lack of coordination in the right limb (which had 11 muscles). The complete and augmented state models, however, displayed a high level of left-right leg symmetry, as well as symmetry with the imitation data.

## 5.3 Answers to Research Questions

- **Can a transfemoral amputee model be trained to have a regular gait using Deep Reinforcement Learning without muscle information?**

  The empirical results from this study show that a transfemoral amputee agent that is trained without muscle information can perform the locomotion task. This is evident in the completion of the locomotion task. However, the results also indicate a significant deviation from the gait pattern of the imitation data; particularly in the healthy leg. This is supported by the higher asymmetry in the gait of a model trained without the muscle information in the state representation, and lower obtained reward for three of such models. This suggests that although such a model can be trained to perform locomotion, the resulting gait pattern is likely not going to be as intended.

- **Does muscle information approximation yield features that are useful for learning locomotion?**

  Training with a model that had an approximation of the muscle force, length, and velocity information of the agent generated a significantly larger episodic reward than the reduced state model in 3 out of 3 simulations. These empirical results suggest that the approximated muscle features play an important role in learning locomotion for such a transfemoral amputee.

- **What are the benefits (if any) of utilizing the augmentation of muscle information in producing a gait?**

  The results from this study strongly suggest that training the DRL algorithm by augmenting the muscle information data, leads to a better performance than using the reduced state representation. This is especially noticeable in the accuracy of the realized gait to the target gait (imitation data). Training with muscle information augmentation also resulted in a more symmetric gait pattern. This is for the most part, due to more accurate gait movement in the healthy leg of the agent which contains a much higher number of muscles. Having access to these muscles (or an approximation of it) has therefore shown to be of benefit to the training process.

## 5.4 Recommendations for Future Research

- **Reward shaping**

  Several results of this project have suggested the benefit of adding supplementary rewards. The erratic pattern of actuator activations produces bursts of force that would not be desired in a real prosthesis. More research can be done using reward shaping to reduce the high variability of the actuator torque. Similarly, future research within this line of study can also incorporate reward shaping based on human knowledge of muscle activations during gait, in a bid to achieve even more realistic locomotion of the agent.

- **Incorporate second-order information**

  In this project, features such as (rectilinear and translational) position and velocities were included in the observation to train a muscle information predictor network in a supervised approach. Second-order acceleration values were not included in this observation space. Having access to only first-order information, the network could be limited in its prediction capacity for the force of the muscles which is based on acceleration and even in the prediction of the muscle velocities. Further research within this line of study can investigate the benefits (if any) of using second-order information in muscle information approximation, and in extension, how that affects gait generation.

- **Utilizing continuous action space**

  Results from this study revealed a diminishing pattern of rewards when the number of prediction categories of the policy network was increased. More research is recommended on the relationship between this increasing action space and the performance of a model trained with a reduced (or augmented) observation space. Ultimately this relationship can be further observed by optimizing a policy network that has a continuous action space.

- **Other locomotion tasks**

  This research and the conclusions therein were done by an agent and imitation subject, walking on flat terrain. While far from being a trivial optimization task, this research has raised more questions on the applicability of the proposed augmented model or the reduced model on other locomotion tasks such as ramp or stairs ascent and descent.

## 5.5 Conclusion

The purpose of this research was to observe if a transfemoral amputee model can be trained to walk using a state-of-the-art Deep Reinforcement Learning algorithm

while using a reduced number of state observers. Based on the results obtained, it can be concluded that disallowing access to the muscle information of the agent yields a significant decline in the human-like locomotion ability of the trained agent. Augmenting the reduced state with predicted muscle information from a pre-trained neural network proved sufficient to improve the reward and gait symmetry of the agent. Using this technique, the performance of the agent was comparable to that of using the complete observation (i.e. including muscle information) of the agent. This suggests the importance of the muscle information for the purpose of realizing a human-like gait for the trained model. Despite the impressive results of training with the augmentation process, it is worth exploring ways to further improve the gait and the practicality of the agent.

# References


Anand, A. S., Zhao, G., Roth, H., & Seyfarth, A. (2019). A deep reinforcement learning based approach towards generating human walking behavior with a neuromuscular model. In *2019 ieee-ras 19th international conference on humanoid robots (humanoids)* (p. 537-543). doi: 10.1109/Humanoids43949.2019.9035034

Ananthakrishnan, A., Kanakiva, V., Ved, D., & Sharma, G. (2018). Automated gait generation for simulated bodies using deep reinforcement learning. In *2018 second international conference on inventive communication and computational technologies (icicct)* (p. 90-95). doi: 10.1109/ICICCT.2018.8473310

Arslan, Y. Z., Karabulut, D., Ortes, F., & Popovic, M. B. (2019). Exoskeletons, exomusculatures, exosuits: Dynamic modeling and simulation. *Biomechatronics*.

Becker, S. (1991, 01). Unsupervised learning procedures for neural networks. *Int. J. Neural Syst.*, *2*, 17-33. doi: 10.1142/S0129065791000030

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics)*. Berlin, Heidelberg: Springer-Verlag.

Bojsen-Møller, F. (1979, August). Calcaneocuboid joint and stability of the longitudinal arch of the foot at high and low gear push off. *Journal of anatomy*, *129*(Pt 1), 165—176. Retrieved from https://europepmc.org/articles/PMC1233091

Camargo, J., Ramanathan, A., Flanagan, W., & Young, A. (2021). A comprehensive, open-source dataset of lower limb biomechanics in multiple conditions of stairs, ramps, and level-ground ambulation and transitions. *Journal of Biomechanics*, *119*, 110320. Retrieved from https://www.sciencedirect.com/science/article/pii/S0021929021001007 doi: https://doi.org/10.1016/j.jbiomech.2021.110320

Castro, J., Mantas, C., & Benıtez, J. (2000). Neural networks with a continuous squashing function in the output are universal approximators. *Neural Networks*, *13*(6), 561-563. Retrieved from https://www.sciencedirect.com/

science/article/pii/S0893608000000319 doi: https://doi.org/10.1016/S0893-6080(00)00031-9

Chapelle, O., Scholkopf, B., & Zien, A. (2010). *Semi-supervised learning*. MIT Press.

Chebotar, Y., Hausman, K., Kroemer, O., Sukhatme, G., & Schaal, S. (2017). *Regrasping using tactile perception and supervised policy learning.* Retrieved from https://aaai.org/ocs/index.php/SSS/SSS17/paper/view/15236

Chebotar, Y., Hausman, K., Zhang, M., Sukhatme, G., Schaal, S., & Levine, S. (2017, 03). Combining model-based and model-free updates for trajectory-centric reinforcement learning.

Chebotar, Y., Kalakrishnan, M., Yahya, A., Li, A., Schaal, S., & Levine, S. (2017). Path integral guided policy search. In *2017 ieee international conference on robotics and automation (icra)* (p. 3381-3388). doi: 10.1109/ICRA.2017.7989384

Chow, C., & Jacobson, D. (1971). Studies of human locomotion via optimal programming. *Mathematical Biosciences*, *10*(3), 239-306. Retrieved from https://www.sciencedirect.com/science/article/pii/0025556471900629 doi: https://doi.org/10.1016/0025-5564(71)90062-9

Crenshaw, S. J., & Richards, J. G. (2006). A method for analyzing joint symmetry and normalcy, with an application to analyzing gait. *Gait Posture*, *24*(4), 515-521. Retrieved from https://www.sciencedirect.com/science/article/pii/S0966636205002602 doi: https://doi.org/10.1016/j.gaitpost.2005.12.002

Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(1), 30-42. doi: 10.1109/TASL.2011.2134090

Davy, D., & Audu, M. (1987). A dynamic optimization technique for predicting muscle forces in the swing phase of gait. *Journal of Biomechanics*, *20*(2), 187-201. Retrieved from https://www.sciencedirect.com/science/article/pii/0021929087903101 doi: https://doi.org/10.1016/0021-9290(87)90310-1

de Boer, S. (2021). *Deep reinforcement learning for physics-based musculoskeletal model of a transfemoral amputee with a prosthesis walking on uneven terrain.* University of Groningen FSE Student Theses. Retrieved from http://fse.studenttheses.ub.rug.nl/id/eprint/25498

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* (Unpublished doctoral dissertation). USA. (AAI7609381)

Delp, S., Anderson, F., Arnold, A., Loan, P., Habib, A., John, C., ... Thelen, D. (2007, 12). Opensim: Open-source software to create and analyze dynamic simulations of movement. *Biomedical Engineering, IEEE Transactions on*, *54*, 1940 - 1950. doi: 10.1109/TBME.2007.901024

De Vree, L., & Carloni, R. (2021). Deep reinforcement learning for physics-based musculoskeletal simulations of healthy subjects and transfemoral prostheses' users during normal walking. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *29*, 607-618. doi: 10.1109/TNSRE.2021.3063015

Duan, Y., Chen, X., Houthooft, R., Schulman, J., & Abbeel, P. (2016). *Benchmarking deep reinforcement learning for continuous control.* arXiv. Retrieved from https://arxiv.org/abs/1604.06778 doi: 10.48550/ARXIV.1604.06778

El Bouchefry, K., & de Souza, R. S. (2020). Chapter 12 - learning in big data: Introduction to machine learning. In P. Škoda & F. Adam (Eds.), *Knowledge discovery in big data from astronomy and earth observation* (p. 225-249). Elsevier.

Farris, D. J., & Sawicki, G. S. (2011). The mechanics and energetics of human walking and running: a joint level perspective. *Journal of The Royal Society Interface*, *9*, 110 - 118.

Feger, M. A., Donovan, L., Hart, J. M., & Hertel, J. (2015, 04). Lower Extremity Muscle Activation in Patients With or Without Chronic Ankle Instability During Walking. *Journal of Athletic Training*, *50*(4), 350-357. Retrieved from https://doi.org/10.4085/1062-6050-50.2.06 doi: 10.4085/1062-6050-50.2.06

Friedman, J. H. (1994). An overview of predictive learning and function approximation. In V. Cherkassky, J. H. Friedman, & H. Wechsler (Eds.), *From statistics to neural networks* (pp. 1–61). Berlin, Heidelberg: Springer Berlin Heidelberg.

Günther, M., & Blickhan, R. (2002). Joint stiffness of the ankle and the knee in running. *Journal of Biomechanics*, *35*(11), 1459-1474. Retrieved from https://www.sciencedirect.com/science/article/pii/S0021929002001835 doi: https://doi.org/10.1016/S0021-9290(02)00183-5

Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., & Levine, S. (2018). Composable deep reinforcement learning for robotic manipulation. In *2018 ieee international conference on robotics and automation (icra)* (p. 6244-6251). doi: 10.1109/ICRA.2018.8460756

Haykin, S. (2009). *Neural networks and learning machines* (No. v. 10). Prentice Hall.

Hebb, D. O. (1949). *The organization of behavior: a neuropsychological theory*. Science editions.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., ... Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82-97. doi: 10.1109/MSP.2012.2205597

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359-366. Retrieved from https://www.sciencedirect.com/science/article/pii/0893608089900208 doi: https://doi.org/10.1016/0893-6080(89)90020-8

Huang, S., & Ferris, D. P. (2011). Muscle activation patterns during walking from transtibial amputees recorded within the residual limb-prosthetic interface. *Journal of NeuroEngineering and Rehabilitation*, *9*, 55 - 55.

Hussein, A., Gaber, M. M., Elyan, E., & Jayne, C. (2017, apr). Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, *50*(2). Retrieved from https://doi.org/10.1145/3054912 doi: 10.1145/3054912

Kakade, S., & Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *In proc. 19th international conference on machine learning* (pp. 267–274).

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., ... Levine, S. (2018, 29–31 Oct). Scalable deep reinforcement learning for vision-based robotic manipulation. In A. Billard, A. Dragan, J. Peters, & J. Morimoto (Eds.), *Proceedings of the 2nd conference on robot learning* (Vol. 87, pp. 651–673). PMLR. Retrieved from https://proceedings.mlr.press/v87/kalashnikov18a.html

Kaufman, K., Frittoli, S., & Frigo, C. (2012, 01). Gait asymmetry of transfemoral amputees using mechanical and microprocessor-controlled prosthetic knees. *Clinical biomechanics (Bristol, Avon)*, *27*, 460-5. doi: 10.1016/j.clinbiomech.2011.11.011

Kern, A. M., Papachatzis, N., Patterson, J. M., Bruening, D. A., & Takahashi, K. Z. (2019). Ankle and midtarsal joint quasi-stiffness during walking with added mass. *PeerJ*, *7*.

Kidziński, Ł., Mohanty, S. P., Ong, C., Hicks, J., Francis, S., Levine, S., ... Delp, S. (2018). Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. In S. Escalera & M. Weimer (Eds.), *Nips 2017 competition book.* Springer: Springer.

Koopman, B., Grootenboer, H. J., & de Jongh, H. J. (1995). An inverse dynamics model for the analysis, reconstruction and prediction of bipedal walking. *Journal of Biomechanics*, *28*(11), 1369-1376. Retrieved from https://www.sciencedirect.com/science/article/pii/0021929094001857 doi: https://doi.org/10.1016/0021-9290(94)00185-7

Laribi, M. A., & Zeghloul, S. (2020). Chapter 4 - human lower limb operation tracking via motion capture systems. In M. Ceccarelli & G. Carbone (Eds.), *Design and operation of human locomotion systems* (p. 83-107). Academic Press. Retrieved from https://www.sciencedirect.com/science/article/pii/B9780128156599000044 doi: https://doi.org/10.1016/B978-0-12-815659-9.00004-4

Leshno, M., Lin, V. Y., Pinkus, A., & Schocken, S. (1993). Multilayer feedforward networks with a non-polynomial activation function can approximate any function. *New York University Stern School of Business Research Paper Series*.

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, *17*(39), 1–40. Retrieved from http://jmlr.org/papers/v17/15-522.html

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., & Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, *37*(4-5), 421-436. Retrieved from https://doi.org/10.1177/0278364917710318 doi: 10.1177/0278364917710318

Levine, S., Wagener, N., & Abbeel, P. (2015). Learning contact-rich manipulation skills with guided policy search. In *2015 ieee international conference on robotics and automation (icra)* (p. 156-163). doi: 10.1109/ICRA.2015.7138994

Li, M., Wen, Y., Gao, X., Si, J., & Huang, H. (2021). Toward expedited impedance tuning of a robotic prosthesis for personalized gait assistance by reinforcement learning control. *IEEE Transactions on Robotics*, 1–10. Retrieved from http://dx.doi.org/10.1109/TRO.2021.3078317 doi: 10.1109/tro.2021.3078317

Liu, L., & Hodgins, J. (2017, 06). Learning to schedule control fragments for physics-based characters using deep q-learning. *ACM Transactions on Graphics*, *36*, 1. doi: 10.1145/3072959.3083723

Marshall, R., Wood, G., & Jennings, L. (1989). Performance objectives in human movement: A review and application to the stance phase of normal walking. *Human Movement Science*, *8*(6), 571-594. Retrieved from https://www.sciencedirect.com/science/article/pii/0167945789900043 doi: https://doi.org/10.1016/0167-9457(89)90004-3

McCulloch, W. S., & Pitts, W. (1943, Dec 01). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115-133. Retrieved from https://doi.org/10.1007/BF02478259 doi: 10.1007/BF02478259

Melo, L. C., & Maximo, M. R. O. A. (2019). *Learning humanoid robot running skills through proximal policy optimization.*

Mhaskar, H., & Micchelli, C. A. (1992). Approximation by superposition of sigmoidal and radial basis functions. *Advances in Applied Mathematics*, *13*(3), 350-373. Retrieved from https://www.sciencedirect.com/science/article/pii/019688589290016P doi: https://doi.org/10.1016/0196-8858(92)90016-P

Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-Hill.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., ... Kavukcuoglu, K. (2016). *Asynchronous methods for deep reinforcement learning.*

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. Retrieved from https://arxiv.org/abs/1312.5602 doi: 10.48550/ARXIV.1312.5602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., ... Hassabis, D. (2015, 02). Human-level control through deep reinforcement learning. *Nature*, *518*, 529-33. doi: 10.1038/nature14236

OpenSim. (2012). Opensim user's guide [Computer software manual]. Retrieved from https://simtk-confluence.stanford.edu:8443/display/OpenSim/User%27s+Guide

Park, H., Yu, R., Lee, Y., Lee, K., & Lee, J. (2020, 07). *Understanding the stability of deep control policies for biped locomotion.*

Peng, Z., Luo, R., Huang, R., Hu, J., Shi, K., Cheng, H., & Ghosh, B. K. (2020). Data-driven reinforcement learning for walking assistance control of a lower limb exoskeleton with hemiplegic patients. In *2020 ieee international conference on robotics and automation (icra)* (p. 9065-9071). doi: 10.1109/ICRA40945.2020.9197229

Raveendranathan, V. (ongoing). Simplified transfemoral amputee model for deep reinforcement learning. *Internal Research*.

Raveendranathan, V., & Carloni, R. (2020). Musculoskeletal model of an osseointegrated transfemoral amputee in opensim. In *2020 8th ieee ras/embs international*

*conference for biomedical robotics and biomechatronics (biorob)* (p. 1196-1201). doi: 10.1109/BioRob49111.2020.9224422

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65 6*, 386-408.

Rosenblatt, F. (1962). *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Spartan Books. Retrieved from https://books.google.nl/books?id=7FhRAAAAMAAJ

Salwan, D., & Kant, S. (2020, 04). Ddpg vs ppo in prosthetics. *Journal of Critical Reviews*, *7*, 2020. doi: 10.31838/jcr.07.09.482

Sartoretti, G., Paivine, W., Shi, Y., Wu, Y., & Choset, H. (2019, Oct). Distributed learning of decentralized control policies for articulated mobile robots. *IEEE Transactions on Robotics*, *35*(5), 1109–1122. Retrieved from http://dx.doi.org/10.1109/TRO.2019.2922493 doi: 10.1109/tro.2019.2922493

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015). *Trust region policy optimization.* arXiv. Retrieved from https://arxiv.org/abs/1502.05477 doi: 10.48550/ARXIV.1502.05477

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal policy optimization algorithms.*

Sonoda, S., & Murata, N. (2017, sep). Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, *43*(2), 233–268.

Sutton, R., & Barto, A. (2018). *Reinforcement learning, second edition: An introduction*. MIT Press.

Sutton, R., Barto, R., Barto, A., Barto, C., Bach, F., & Press, M. (1998). *Reinforcement learning: An introduction*. MIT Press. Retrieved from https://books.google.nl/books?id=CAFR6IBF4xYC

Thelen, D. (2003, 03). Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults. *Journal of biomechanical engineering*, *125*, 70-7. doi: 10.1115/1.1531112

Tu, X., Li, M., Liu, M., Si, J., He, & Huang. (2020). *A data-driven reinforcement learning solution framework for optimal and adaptive personalization of a hip exoskeleton.*

Wen, Y., Li, M., Si, J., & Huang, H. (2020). Wearer-prosthesis interaction for symmetrical gait: A study enabled by reinforcement learning prosthesis control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *28*, 904-913.

Wen, Y., Si, J., Brandt, A., Gao, X., & Huang, H. H. (2020). Online reinforcement learning control for the personalization of a robotic knee prosthesis. *IEEE Transactions on Cybernetics*, *50*(6), 2346-2356. doi: 10.1109/TCYB.2019.2890974

Wen, Y., Si, J., Gao, X., Huang, S., & Huang, H. H. (2017). A new powered lower limb prosthesis control framework based on adaptive dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(9), 2215-2220. doi: 10.1109/TNNLS.2016.2584559

Williams, R. J. (1992, may). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, *8*(3–4), 229–256. Retrieved from https://doi.org/10.1007/BF00992696 doi: 10.1007/BF00992696

Wu, Y., Mansimov, E., Liao, S., Grosse, R., & Ba, J. (2017). *Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation.* arXiv. Retrieved from https://arxiv.org/abs/1708.05144 doi: 10.48550/ARXIV.1708.05144

Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. Retrieved from https://arxiv.org/abs/1004.4170 doi: 10.48550/ARXIV.1004.4170

Zifchock, R. A., Davis, I., Higginson, J., & Royer, T. (2008). The symmetry angle: A novel, robust method of quantifying asymmetry. *Gait Posture*, *27*(4), 622-627. Retrieved from https://www.sciencedirect.com/science/article/pii/S0966636207002111 doi: https://doi.org/10.1016/j.gaitpost.2007.08.006

# Appendix A

## A.1 Observation Space

Table A.1: Complete observation space for the transfemoral amputee agent. [1-46] is the reduced observation space. [47-91] are the muscle information that are predicted using a feed-forward network.

| index | Agent's body part | feature | measurement unit |
|---|---|---|---|
| 1 | pelvis | position (x) | m |
| 2 | pelvis | position (y) [height] | m |
| 3 | pelvis | position (z) | m |
| 4 | pelvis | velocity (x) | m/s |
| 5 | pelvis | velocity (y) | m/s |
| 6 | pelvis | velocity (z) | m/s |
| 7 | pelvis | list | radians |
| 8 | pelvis | rotation | radians |
| 9 | pelvis | tilt | radians |
| 10 | pelvis | list velocity | rad/s |
| 11 | pelvis | rotation velocity | rad/s |
| 12 | pelvis | tilt velocity | rad/s |
| 13 | right leg | ground reaction force (x) | N |
| 14 | right leg | ground reaction force (y) | N |
| 15 | right leg | ground reaction force (z) | N |
| 16 | right hip | abduction | rad |
| 17 | right hip | flexion | rad |
| 18 | right knee | flexion | rad |
| 19 | right ankle | flexion | rad |
| 20 | right hip | abduction velocity | rad/s |
| 21 | right hip | flexion velocity | rad/s |

*Continued on next page*

| index | Agent's body part | feature | measurement unit |
|---|---|---|---|
| 22 | right knee | flexion velocity | rad/s |
| 23 | right ankle | flexion velocity | rad/s |
| 24 | left leg | ground reaction force (x) | N |
| 25 | left leg | ground reaction force (y) | N |
| 26 | left leg | ground reaction force (z) | N |
| 27 | left hip | abduction | rad |
| 28 | left hip | flexion | rad |
| 29 | left knee | flexion | rad |
| 30 | left ankle | flexion | rad |
| 31 | left hip | abduction velocity | rad/s |
| 32 | left hip | flexion velocity | rad/s |
| 33 | left knee | flexion velocity | rad/s |
| 34 | left ankle | flexion velocity | rad/s |
| 35 | left knee actuator | force | N |
| 36 | left knee actuator | velocity | rad/s |
| 37 | left knee actuator | control | $\mathbb{R}$ |
| 38 | left knee actuator | power | W |
| 39 | left knee actuator | activation | $\mathbb{R}$ |
| 40 | left knee actuator | actuation | $\mathbb{R}$ |
| 41 | left ankle actuator | force | N |
| 42 | left ankle actuator | velocity | rad/s |
| 43 | left ankle actuator | control | $\mathbb{R}$ |
| 44 | left ankle actuator | power | W |
| 45 | left ankle actuator | activation | $\mathbb{R}$ |
| 46 | left ankle actuator | actuation | $\mathbb{R}$ |
| 47 | right hip abductor | fiber force | N |
| 48 | right hip abductor | fiber length | m |
| 49 | right hip abductor | fiber velocity | m/s |
| 50 | right hip adductor | fiber force | N |
| 51 | right hip adductor | fiber length | m |
| 52 | right hip adductor | fiber velocity | m/s |
| 53 | right iliopsoas | fiber force | N |
| 54 | right iliopsoas | fiber length | m |
| 55 | right iliopsoas | fiber velocity | m/s |
| 56 | right gluteus maximus | fiber force | N |
| 57 | right gluteus maximus | fiber length | m |
| 58 | right gluteus maximus | fiber velocity | m/s |
| 59 | right biarticular hamstrings | fiber force | N |
| 60 | right biarticular hamstrings | fiber length | m |
| 61 | right biarticular hamstrings | fiber velocity | m/s |

*Continued on next page*

| index | Agent's body part | feature | measurement unit |
|---|---|---|---|
| 62 | right rectus femoris | fiber force | N |
| 63 | right rectus femoris | fiber length | m |
| 64 | right rectus femoris | fiber velocity | m/s |
| 65 | right vasti | fiber force | N |
| 66 | right vasti | fiber length | m |
| 67 | right vasti | fiber velocity | m/s |
| 68 | right biceps femoris | fiber force | N |
| 69 | right biceps femoris | fiber length | m |
| 70 | right biceps femoris | fiber velocity | m/s |
| 71 | right gastrocnemius | fiber force | N |
| 72 | right gastrocnemius | fiber length | m |
| 73 | right gastrocnemius | fiber velocity | m/s |
| 74 | right soleus | fiber force | N |
| 75 | right soleus | fiber length | m |
| 76 | right soleus | fiber velocity | m/s |
| 77 | right tibialis anterior | fiber force | N |
| 78 | right tibialis anterior | fiber length | m |
| 79 | right tibialis anterior | fiber velocity | m/s |
| 80 | left hip abductor | fiber force | N |
| 81 | left hip abductor | fiber length | m |
| 82 | left hip abductor | fiber velocity | m/s |
| 83 | left hip adductor | fiber force | N |
| 84 | left hip adductor | fiber length | m |
| 85 | left hip adductor | fiber velocity | m/s |
| 86 | left gluteus maximus | fiber force | N |
| 87 | left gluteus maximus | fiber length | m |
| 88 | left gluteus maximus | fiber velocity | m/s |
| 89 | left iliopsoas | fiber force | N |
| 90 | left iliopsoas | fiber length | m |
| 91 | left iliopsoas | fiber velocity | m/s |

## A.2 Muslce information prediction

Table A.2: Results of muscle information prediction network on the validation data.

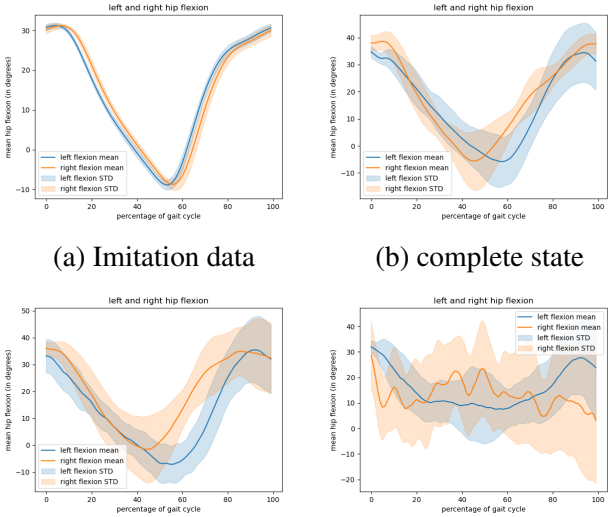
| Muscle | Force | | | | Length | | | | Velocity | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAE | min | max | sigma | MAE | min | max | sigma | MAE | min | max | sigma |
| Hip Abductor | 0.111 | 0.003 | 1.144 | 0.1792 | 0.019 | 0.555 | 1.036 | 0.070 | 0.166 | -4.459 | 2.91 | 0.929 |
| Hip Adductor | 0.099 | 0.006 | 1.055 | 0.196 | 0.032 | 0.314 | 1.049 | 0.078 | 0.445 | -5.744 | 4.396 | 0.819 |
| Iliopsoas L | 0.127 | 0.032 | 1.280 | 0.362 | 0.012 | 0.913 | 1.327 | 0.090 | 0.390 | -2.615 | 6.344 | 1.116 |
| Glut max R | 0.081 | 0.001 | 0.997 | 0.144 | 0.037 | 0.673 | 1.188 | 0.104 | 0.488 | -5.899 | 1.417 | 1.227 |
| Hamstring R | 0.006 | 0.008 | 0.938 | 0.151 | 0.006 | 0.330 | 1.208 | 0.190 | 0.009 | -8.139 | 17.257 | 0.9407 |
| Rectus Femoris R | 0.161 | 0.000 | 1.050 | 0.176 | 0.022 | 0.452 | 1.486 | 0.215 | 0.296 | -6.655 | 5.872 | 1.481 |
| Vasti R | 0.074 | 0.000 | 0.840 | 0.129 | 0.007 | 0.645 | 1.357 | 0.166 | 0.142 | -9.790 | 1.961 | 2.182 |
| Bifmesh R | 0.082 | 0.000 | 1.291 | 0.354 | 0.014 | 0.086 | 1.287 | 0.087 | 0.369 | -40.385 | 23.197 | 1.217 |
| Gastrocnemius R | 0.007 | 0.000 | 0.697 | 0.107 | 0.007 | 0.308 | 1.307 | 0.202 | 0.012 | -12.403 | 8.019 | 1.947 |
| Soleus R | 0.006 | 0.000 | 0.527 | 0.069 | 0.007 | 0.426 | 1.349 | 0.189 | 0.011 | -13.630 | 2.920 | 2.341 |
| Tib ant R | 0.074 | 0.032 | 1.228 | 0.299 | 0.014 | 0.570 | 1.336 | 0.127 | 0.346 | -4.290 | 18.604 | 1.783 |
| Hip abductor L | 0.120 | 0.000 | 1.188 | 0.234 | 0.018 | 0.558 | 1.211 | 0.076 | 0.523 | -8.502 | 7.609 | 1.386 |
| Hip adductor L | 0.143 | 0.000 | 1.132 | 0.222 | 0.013 | 0.377 | 1.136 | 0.091 | 0.199 | -10.067 | 8.731 | 1.217 |
| Glut max L | 0.006 | 0.000 | 1.204 | 0.243 | 0.008 | 0.669 | 1.384 | 0.108 | 0.009 | -11.815 | 17.786 | 1.876 |
| Iliopsoas L | 0.005 | 0.000 | 1.330 | 0.331 | 0.006 | 0.603 | 1.339 | 0.096 | 0.010 | -10.190 | 10.376 | 1.706 |
| **Average** | 0.082 | 0.000 | 1.330 | 0.252 | 0.018 | 0.308 | 1.486 | 0.191 | 0.228 | -40.385 | 23.197 | 1.884 |

## A.3 Maximum isotropic force and optimal length of muscles

Table A.3: Maximum isotropic force and optimal fiber length of 11 muscles of the transfemoral amputee agent.

| Muscle | Maximum isotropic force (N) | Optimal length (m) |
|---|---|---|
| hip abductor | 4460.29 | 0.0845 |
| hip adductor | 3931.8 | 0.087 |
| iliopsoas | 2697.34 | 0.117 |
| gluteus maximus | 3337.58 | 0.157 |
| biarticular hamstrings | 4105.46 | 0.069 |
| rectus femoris | 2191.74 | 0.076 |
| vasti | 9593.95 | 0.099 |
| biceps femoris | 557.11 | 0.11 |
| gastrocnemius | 4690.57 | 0.051 |
| soleus | 7924.99 | 0.044 |
| tibialis anterior | 2116.81 | 0.068 |

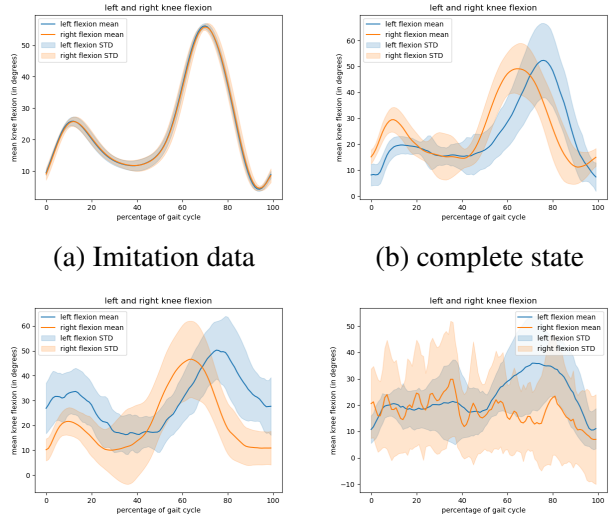
## A.4 Symmetry of left and right joints




(a) Imitation data (b) complete state

(c) augmented state (d) reduced state

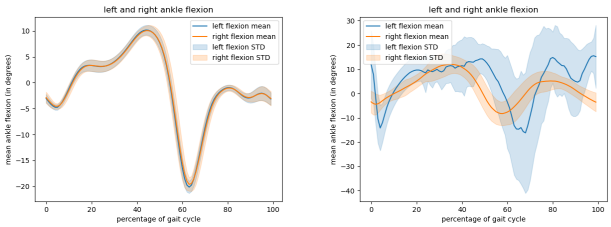
Figure A.1: Mean and standard deviation of hip flexion during gait using different observation state representations.




(a) Imitation data (b) complete state

(c) augmented state (d) reduced state

Figure A.2: Mean and standard deviation of knee flexion during gait using different observation state representations.

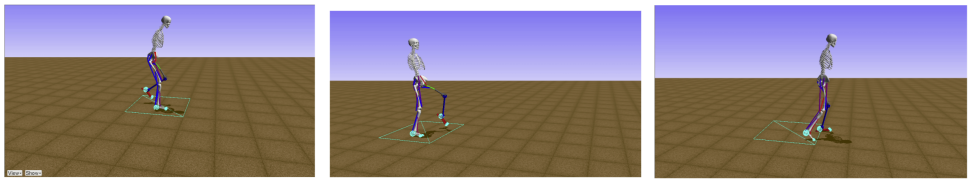



(a) Imitation data (b) complete state

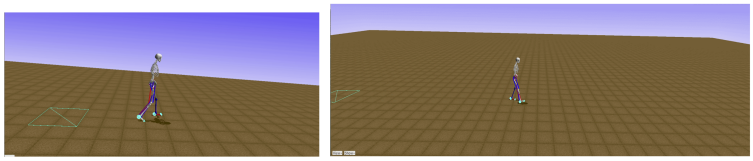
(c) augmented state (d) reduced state

Figure A.3: Mean and standard deviation of ankle flexion during gait using different observation state representations.

## A.5 Visualization of Simulation



(a) 40 iterations (b) 100 iterations (c) 200 iterations



(d) 400 iterations (e) 700 iterations

Figure A.4: Snapshots of transfemoral amputee model during training process. Model trained using complete observation state, 5 prediction categories, and 228 hidden layer size.