Reinforcement Learning Of A Feedforward Controller With Soft Actor-Critic For A Reaching Task

by

Venkata Praneeth Srungarapu

B.Tech, Electronics and Communications Engineering, KL University, 2015

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Electrical and Computer Engineering)

The University of British Columbia
(Vancouver)

September 2021

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Reinforcement Learning Of A Feedforward Controller With Soft Actor-Critic For A Reaching Task

submitted by **Venkata Praneeth Srungarapu** in partial fulfillment of the requirements for the degree of **MASTER OF APPLIED SCIENCE** in **Electrical and Computer Engineering**.

Examining Committee:

Sidney Fels, Electrical and Computer Engineering Department, UBC
Supervisor

Bryan Gick, Department of Linguistics, UBC
Supervisory Committee Member

Abstract

Learning to control is a complicated process, yet humans seamlessly control various complex movements. Motor theory suggests that humans start motor learning by learning to act in a feedforward manner. However, it is still unclear how humans learn feedforward control strategies. We hypothesize that this mechanism is governed by the criterion of success (reinforcement) or failure (penalty) of the task. Taking this for inspiration, we investigate how we can learn a feedforward controller utilizing reinforcement learning. Additionally, we investigate how the factors such as the difficulty of the task and noise present in the motor system are related to human motor control.

Hence, a one-dimensional muscle-based biomechanical model is built to create a reaching task setup. The model contains an actuator controlled by the antagonistic and agonistic muscle pair and a goal or target to reach. Then, an end-to-end reinforcement-learning-based feedforward controller is learned to estimate control signals while taking the difficulty levels of a reaching task and noise levels into account. To design the learning-based controller, we adapted the model-free RL algorithm "Soft Actor-Critic".

As a result, during training, we observed that the SAC-based feedforward controller has learned to prepare co-activation to reach a target in the kinematic space using a minimum number of controller predictions. Moreover, we found that the controller has learned to estimate high-amplitude muscle activations as a way to adapt to the noise levels in the motor system. Finally, we conducted information analysis similar to Fitts' analysis to determine how the difficulty of the task and noise affected the controller. The effect of the task's difficulty and the noise in the system is determined by finding the relationship between the number of controller

predictions, task difficulty, and the amount of noise. Our analysis demonstrates that the number of controller predictions increases exponentially with the increase in the difficulty of the task with the amount of noise kept constant. A linear relationship exists between the number of controller predictions and the amount of noise with ID kept constant. Additionally, we found that the effect of target width is more dominant than the distance, which confirms Welford's observation.

Lay Summary

With practice, humans learn to generate complex movements. Consider the darts game scenario where the player needs to learn to generate a pre-planned movement before throwing the dart. This pre-planning is called feedforward control. Based on the scores (rewards) associated with throwing darts, a player can improve his feedforward control strategy that helps him win the game. It is called reinforcement learning, and it helps humans learn to calculate feedforward movements. Hence, we investigate how machines can learn feedforward control movements based on reinforcement. Additionally, we investigate the factors that affect the feedforward control strategy in the learning process. To achieve this, we built a reinforcement learning-based controller deployed onto a muscle-based reaching task where the difficulty and the amount of noise are varied. As a result, we successfully taught the system to compute the necessary feedforward movements to complete the target reaching task.

Preface

This thesis is an original intellectual product of the author, Venkata Praneeth Srungarapu. All the design, implementation, and testing for this thesis were carried out at the Human Communications Technologies (HCT) Laboratory in the Electrical and Computer Engineering department at the University of British Columbia (UBC) (Vancouver campus).

Following are the publications resulting from the work described in this thesis.

Peer Reviewed Journal Publications

[J1] A. H. Abdi, B. Sagl, **V. P. Srungarapu,** I. Stavness, E. Prisman, P. Abolmaesumi, and S. Fels. Characterizing motor control of mastication with soft actor-critic. Frontiers in Human Neuroscience, 14:188, 2020.

Peer Reviewed Conference Publications

[C1] P. Saha, **P. Srungarapu**, and S. Fels, "Towards automatic speech identification from vocal tract shape dynamics in real-time MRI", Proc. Interspeech 2018, pp. 1249–1253, 2018.

Peer-reviewed Conference Abstracts and Presentations

[A1] **V. P. Srungarapu**, P. Saha and S. Fels, "Speed-Accuracy Trade-off In Speech Production", ISSP 2020.

[A2] P. Saha, D. R. Mohapatra, **V. P. Srungarapu**, and S. Fels, "Sound stream: Towards vocal sound synthesis via dual-handed simultaneous control of articulatory parameters", The Journal of the Acoustical Society of America, vol. 144, no. 3, pp. 1907–1907, 2018.

[A3] P. Saha, D. R. Mohapatra, **S. Praneeth**, and S. Fels, "Sound-stream II: Towards real-time gesture-controlled articulatory sound synthesis", Canadian Acoustics, vol. 46, no. 4, pp.58–59, 2018.

Author's Contribution

In [J1], I aided in building the SAC model for 3D mandibular model and running the experiments based on comments from Amir Abdi, Dr. Fels. I also helped in proof-reading and editing the manuscript. S. Fels and Ian stavness acted in supervisory role.

In [C1], my main contribution included the implementation of deep learning techniques and running experiments. Saha aided in application of deep learning techniques to vocal tract dynamic MRI sequences and preparing the manuscript. Dr. Fels acted in supervisory role.

In [A1], I led and contributed most of the work related to the reinforcement learning based implementation of muscle excitation estimation. Saha helped in the experimental planning, interpretation of results and manuscript editing. Dr. Fels supervised this research.

In [A2] and [A3], all the authors contributed equally towards the conceptualization of ideas and manuscript writing. My main contribution was the development of a hardware interface using Arduino to control tongue muscle excitations. Srungarapu helped to extract an inventory of cross-sectional areas from the ArtiSynth tongue model and coupling it to the JASS audio synthesis engine.

Thesis Code

We used the ArtiSynth-RL software to implement the Soft Actor-Critic algorithm for learning a feedforward controller that can accomplish reaching task using a biomechanical system and the custom code for this research project can be made available upon a request in GitHub:
https://github.com/amir-abdi/artisynth-rl-jaw/tree/thesis-results

Table of Contents

List of Tables

List of Figures

List of Abbreviations

DeepRL: Deep Reinforcement Learning
EMG: Electromyography
ID: Index of Difficulty
iEMG: intramuscular Electromyography
MT: Movement Time
1D: One Dimensional
PID: Proportional-Integral-Derivative
RL: Reinforcement Learning
SAC: Soft Actor Critic
SAT: Speed Accuracy Tradeoff
sEMG: surface Electromyography
TP: Throughput

Acknowledgments

My growth as a researcher is mainly attributed to my supervisor, Prof. Sidney Fels, and I would like to express my sincere gratitude to him. This thesis would not be possible without his guidance and mind-bending ideas. Many thanks to him for molding me into a proper research engineer. I have learned invaluable lessons on conducting research; it has been my pleasure working with you and learning from you. I will never forget your saying "*If you know the answer to a research question, it is not research*".

Besides my supervisor, I have had the pleasure of working with several outstanding researchers throughout this degree, the foremost being Dr. Antonio Sanchez and Dr. Amir Abdi. I will always be indebted to them for the support, help, and knowledge I received during our interactions. My sincere thanks also go to Professor Bryan Gick for his invaluable suggestions.

The great memories I take of this degree are due, in part, to my colleagues at the Human Communication Technologies (HCT) Lab includes Pramit Saha, Debasish Roy Mohapatra, and Bharan Pourahmadi, and the Brain2Speech group includes Yadong Liu. I thank them for the great conversation and even better company; they made every day at the lab truly fun.

I am fortunate enough to have some truly amazing friends such as Naresh Maroju and Dinesh Kuncham in my life who have given me incredible support throughout my program. Finally, my parents, without whom I would not have been here. Thank you for providing me this opportunity to pursue a master's degree.

Dedication

To the master of infinite universes, Lord Narayana, for looking over me and protecting me.

om namo bhagavate vasudevaya
Hari hi om tatsat

Chapter 1

Introduction

Human motor control is a complicated process involving the brain, limbs, muscles and interacting with the external world. Learning to control movements is an incredibly challenging problem. In this work, we deal with the problem of motor learning. The first step in learning is to predict pre-programmed motor commands [48] that generate rapid movements. The strategy of pre-programming the motor command is generally known as feedforward (open-loop or ballistic) control, and it is crucial in the motor learning mechanism. A simple example where the importance of a feedforward control strategy can be explained is the darts game as shown in Figure 1.1. In a darts game, the player needs to throw the dart and hit the board by preparing a movement in advance. With practice, players can learn to be skillful at calculating the pre-programmed motor commands that help the player win the game. However, it is not obvious how we learn the feedforward control strategy.

One of the plausible approaches to solve the aforementioned problem is to use reinforcement learning (RL) theory as it plays a potential role in learning motor control strategies [21]. RL theory suggests that various motor control strategies are learned by mapping the actions to the consequent rewards and penalties [23]. So, in the darts game analogy, the player can learn the best strategy by mapping actions to the score he receives. A high score acts as the reinforcement, whereas a low score acts as a penalty. The player uses the score as reinforcement to optimize his actions.

Figure 1.1: Darts board game illustration. A classic example of a feedforward control strategy where the scoring of the game acts as the reinforcement.

Based on mapping actions to consequent rewards and penalties, humans optimize or reinforce their motor control strategy. Similar to darts game, humans have to learn to control various motor tasks such as articulatory tasks, target reaching tasks, and many more, and it is possible to learn solely based on a reinforcement signal that informs whether the task is a success or not (low-information) [6, 23]. Therefore, in this work, we investigate this question by enabling machines to learn to control movements in a feedforward mode based on low-information reinforce-

ment learning. Recent advances in reinforcement learning have shown promising performance in learning complex motor skills [52]. Taking advantage of these advancements, we aim to learn a controller that acts in feedforward mode by utilizing an advanced reinforcement learning technique called Soft Actor-Critic.

1.1 Motivation: Learning to speak using reinforcement

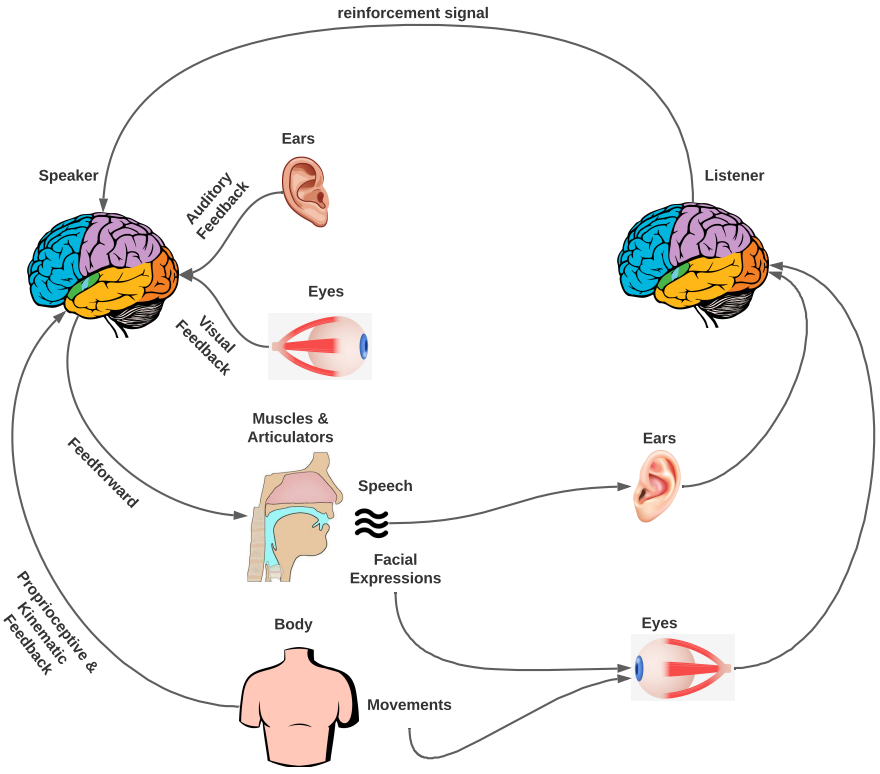

Figure 1.2: Dynamics of the communication chain system between two persons, the speaker and the listener. The controller (brain) of the speaker produces muscle activations that moves the articulators producing speech and the listener processes the information to provide a response signal whether he/she understood or not.

Speech production is fundamentally a motor control process involving articu-

latory movements brought about by a well-co-ordinated synergy of muscle excitations. In the context of neural control of speech production, the main question we have about speech production is how humans learn to perform motor control of speaking [55]. It is unclear what kind of learning mechanism humans use to produce speech. We hypothesize that learning to produce speech is driven by reinforcement learning and it can be better explained by looking at the Brain-To-Speech (B2S) mechanism. Figure 1.2 describes the dynamics of the Brain-To-Speech mechanism. In the B2S framework, a speaker communicates the thoughts originated from the brain (controller) in the form of speech. The listener receives the speech information, processes it, and provides a response or acknowledgment of whether the message was understood or not. This response or acknowledgment acts as a reinforcement signal for the speaker to continue the conversation. If the listener does not understand, then the speaker adjusts his motor control strategy based on the reinforcement [101]. Though it is impoverished, it is the driving factor of learning speech as the listener would never provide feedback that the speaker's articulators are not in the correct position. Hence, the primary source of learning available in this mechanism is reinforcement. Moreover, the controller can utilize all other sources of feedback to improve the reinforcement signal, i.e., the listener can understand the speaker better.

Further, this reinforcement signal can also inform how hard the speech movement task is. When the listener understands the speaker, it is analogous to an easier task. So, the better the listener understands the speaker, the less precise speaker has to be on their targets. If the listener can not understand the speaker, the speaker readjusts his speech motor control strategy resulting in careful and precise articulation. So, reinforcement signal coming from the listener drives the learning of speech motor control strategies. Hence, we are interested to understand how people learn to control speech utilizing a reinforcement signal. However, speech is highly complicated, involving multiple pathways connecting the brain to articulators, and it is incredibly challenging to learn speech motor control. As speech motor control is fundamentally a motor control problem, we simplify the problem space by looking at how people learn to control movements in a reaching task paradigm. Though speaking and reaching are different tasks, they share the same sensory goals [39]. Moreover, articulatory movements involve reaching various kinematic

targets in the articulatory space. Therefore, learning a reaching task based on the low-information reinforcement signal is a critical step and eventually helps us to learn articulatory tasks. Therefore, in this work, as a first step, we learn a reaching task using a reinforcement learning strategy.

1.2 Strategies for learning to control movements

To learn motor control, it is important to look at how humans learned to control movements which can be helpful in providing insights into building learning-based models. Humans, during the initial phase of learning, bootstrap the learning process by execution of slow movements using sensory feedback (closed-loop) as pre-programmed ballistic movements are not learned yet [48]. With practice, humans learn to predict a pre-programmed motor command that makes the hands or actuators cover a greater amount of the movement [48]. Once this is learned, humans can pre-program required movements in a feedforward (open-loop) mode given the task and world information which allows humans to solve motor control tasks in a short period of time. However, it is important to note that it is not always possible to solve complex motor tasks with pure feedforward control. In some cases, feedback control is helpful, particularly when there is a high demand for accuracy. When the feedforward command brings the hand near to the target object, a small corrective movement is generated using feedback (closed-loop) mode, thereby successfully reaching high accurate targets.

In general, there are mainly three types of learning mechanisms possible [10, 41, 64, 72]. One strategy is to learn pure feedforward control to generate rapid and ballistic movements. Another is to use pure feedback control to generate slow and corrective movements. The third strategy is a hybrid approach that combines both feedforward and feedback strategies and switches smoothly between feedforward and feedback. Learning a pure feedforward control mechanism results in ballistic controllers, whereas purely feedback control results in corrective or servo strategies such as proportional-integrative-derivative (PID) controllers. Focusing on learning corrective controllers can not produce relatively fast movements as the feedback-based controllers are inherently time-intensive [37]. On the other hand, learning a pure feedforward controller can generate relatively rapid movements. Then the

same control strategy can be used as a proxy for generating corrective movements. This avoids the need for learning both the ballistic and corrective control strategy while achieving the hybrid nature of the movement control. This way, the controller generates both ballistic and corrective movements.

It is important to note that we consider a simplified hybrid control strategy in this work. Generally, the hybrid control strategy allows humans to switch quickly between feedforward and feedback strategies [45]. Since modeling the switching ability is challenging, we do not consider the switching ability. Rather, the controller waits for the actuator to attain equilibrium which helps to assess the result of the previous ballistic prediction. Hence, the controller used in this work can also be referred to as a feedforward equilibrium controller. Another crucial aspect is minimizing the number of ballistic predictions (n). As the n increases, the number of interactions with the real world or environment also increases, which can cost more resources and time. Therefore, we learn a feedforward equilibrium controller where we optimize the controller based on the reinforcement signal that puts pressure to learn to predict motor commands using a minimum number of predictions that helps the actuator to reach the target in the real world.

1.3 Factors influencing motor learning

In the previous section, we discussed that humans learned a hybrid or dual model of control. Moreover, the hybrid model allows humans to switch between the feedforward and feedback control depending on the context [45]. This idea is further supported by Woodworth's and Fitts' study [28, 98]. Now, the obvious question is, what are the parameters that govern the type of control to be used? Fitts explained this phenomenon that the difficulty of the task could force humans to use both feedforward and feedback control. In a separate study, Van Beers et al. illustrated that humans learn motor control by taking the properties of noise into account [92]. Hence, it can be understood that the difficulty of the task and the noise present in the motor (neuro-motor noise) system can affect the learning strategies. Hence, we additionally investigate how the difficulty of the task and neuro-motor noise can affect motor learning.

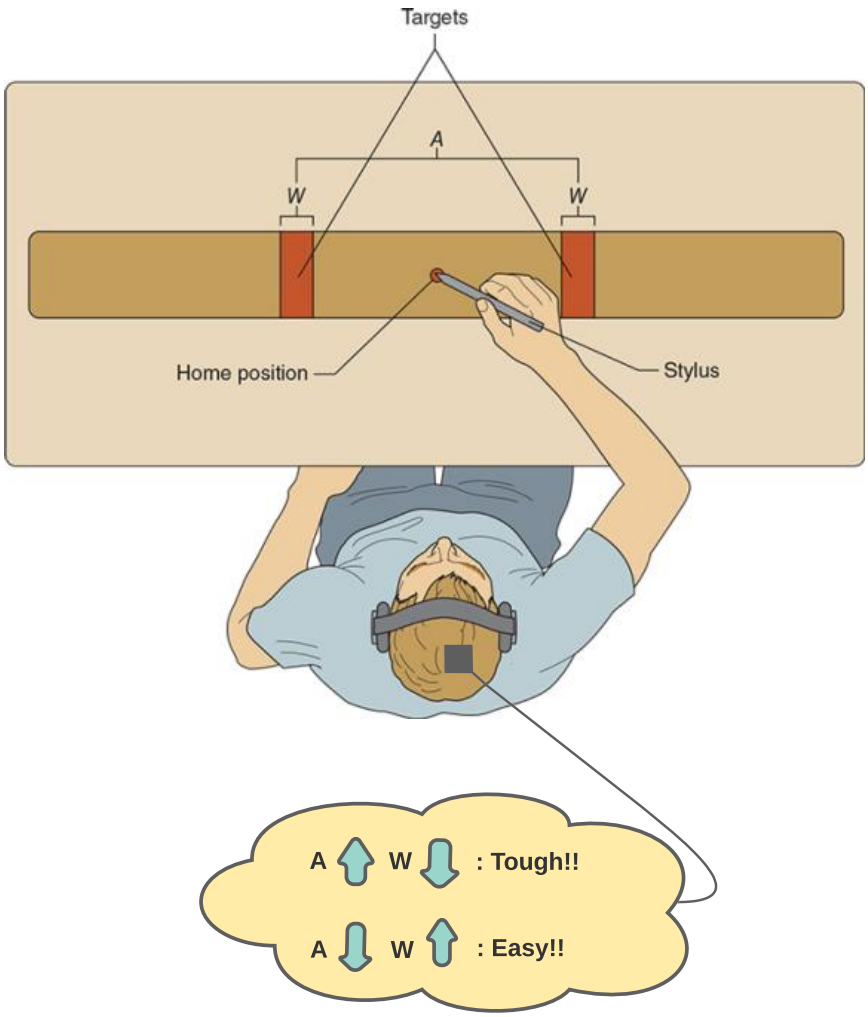

Figure 1.3: Human operator performing a reciprocal tapping task by sensing the task parameters distance between the targets A and width W. Depending on A & W, the control strategy is decided. This figure is adapted from [76].

1.3.1 Difficulty of the task

Fitts' proposed that the difficulty of the task drives the humans to decide on the appropriate control strategy and provided a metric to measure the difficulty of the task (ID) as given in Equation 1.1 using a reciprocal tapping task (see Figure 1.3). It suggests that a single ballistic prediction in feedforward control mode would be sufficient to reach bigger and near targets (easy). As the demand for accuracy (far and smaller targets) increases, humans would require to generate more corrective movements using feedback control [28, 43, 98]. As a result, the time taken to reach the target (movement time) increases as the difficulty of the task increases. This way, the humans demonstrate a different type of control strategy depending on the difficulty of the task.

$$ID = \log_2\left(\frac{2A}{W}\right) \tag{1.1}$$

Similarly, when a controller is learned and the task's difficulty is varied, the number of controller's predictions is expected to increase as it is observed in humans. Therefore, like Fitts' task, we vary the tasks difficulty and utilize the Fitts' ID as a way to measure the difficulty of the reaching task as given in Equation 1.1. During the training phase of the controller, we vary the task parameters such as the target distance and width from trial to trial to vary the task's difficulty. Once the algorithm has learned a feedforward control strategy to reach targets by minimizing the number of predictions, we must observe an increase in the number of controller predictions in the case of small targets (required accuracy is high).

1.3.2 Neuro-motor noise

Another factor that plays a crucial role in motor learning is the noise present in the neuro-motor pathways. Hence, we hypothesize that the noise in the motor system can alter the task's difficulty similar to the amplitude of movements and target width [77]. Moreover, motor control theories have proposed that the brain is a stochastically optimized controller [33]. It means that the brain takes the statistics of the noise into account while it executes movements. Due to these reasons, it is reasonable to learn a controller in the presence of noise to investigate its effect on the movements.

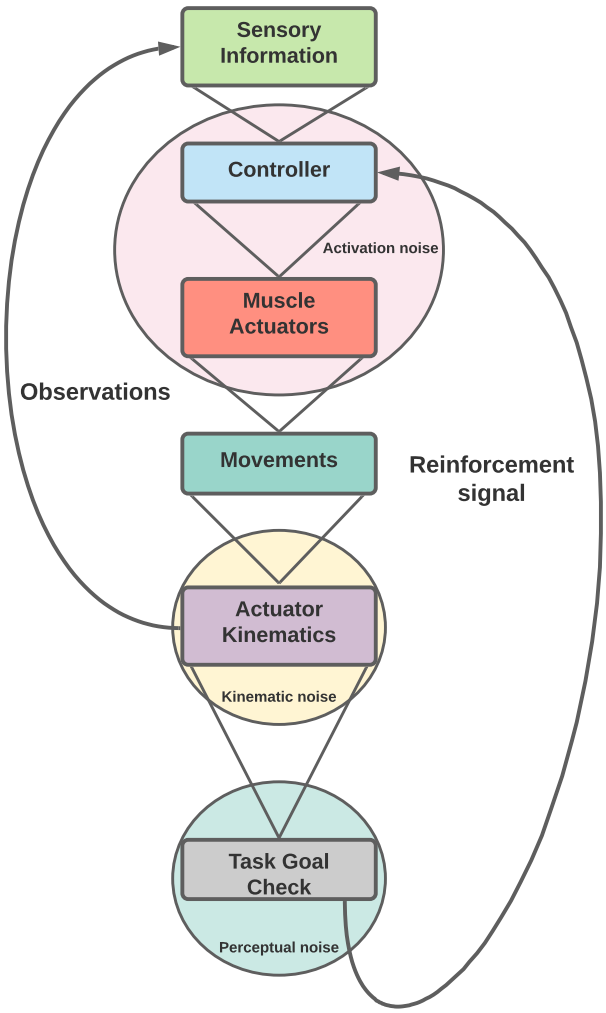

Figure 1.4: Typical neuro-motor control process describing various stage involved in the human motor system and various noise source present at different stages in the motor process.

There can be multiple sources of noise in the human motor system [73]. Essentially, noise can enter the system at all stages shown in Figure 1.4 and the system needs to learn to deal with noise. So, we model specific features of the noise, such as the controller noise and the activation noise in this work. Schmidt et al.

proposed that the output of the human muscular system is contaminated by noise which affects movements [77] in a similar way the distance and the target width effects. Therefore, we train our controller in the presence of a particular noise called muscle activation noise. Additionally, we model another type of noise called the controller noise as there can be noise in the motor commands of the controller [73] (refer section 3.2.3 in chapter 3 for more details on the noise used in this work). Overall, it can be established that the difficulty of the task modifies the behavior or the control strategy of the learned controller, and it would help us get a deeper understanding of what the controller has learned. Hence, we investigate the behavior of the learned ballistic controller to determine whether the controller has learned to tradeoff the number of controller predictions when the difficulty of the task and muscle activation noise levels are high.

To summarize the controller's strategy, we can take the help of the golf game analogy. In golf, first, the player prepares a calculated (pre-programmed) drive (prediction) to hit the ball while considering the wind and other factors. Then the player waits for the ball to stop at a point in the field (waiting for equilibrium). If the initial drive did not put the ball in the hole, the player re-assess the situation and calculates a new prediction if necessary provided the new position of the ball in the field. This continues till the ball is in the hole. However, the player wins if he takes a minimum number of turns or chances. Similarly, our controller prepares muscle activations, driving the actuator towards the target. The controller waits for the actuator to attain equilibrium. If the noise levels (wind in golf analogy) and the difficulty of the task are considerably high, then the actuator attains equilibrium at a different position from that of the target. Then, the controller re-assess the state of the world and calculates another prediction. However, as the number of predictions increases, the cumulative reward decreases, which puts pressure on the agent to learn using a minimum number of predictions.

1.4 Deep Reinforcement Learning

Till now, we discussed the details and methods involved in learning motor control and established a connection between reinforcement learning and motor control. Then, we proposed that reinforcement learning drives the learning in motor

control and speech motor control. This statement is further supported by various experimental studies in the neuroscience domain. These studies suggest that each learning paradigm is neurally correlated to distinct regimes of the human brain. For example, the cerebellum is responsible for supervised learning, the basal ganglia for reinforcement learning, and the cerebral cortex for unsupervised learning [21, 22, 40]. Basal ganglia are a group of subcortical nuclei responsible for action-selection, movement execution, and control, motor behaviors, and play a significant role in reward, reinforcement, and habit development process [18, 21, 71, 99]. Moreover, recent neurophysiological studies demonstrate that basal ganglia coupled with cortical integrators forming a cortico-basal ganglia circuit are responsible for learning motor control and its principles such as speed-accuracy tradeoff (SAT). Mainly, four theories emerged to explain how each component of the brain modulates the SAT mechanism [12]. One of the four theories is the "Synaptic" theory which was introduced for tasks in which participants need to find the right balance between speed and accuracy that optimizes reward rate [30]. The synaptic theory was motivated by reinforcement–learning strategies that suggest that corticostriatal synapses are modified during learning processes aiming at reward maximization [23]. Most of the model-free RL algorithms objective is to determine the actions that result in the maximum reward. Hence, utilizing an algorithm from the family of model-free RL strategies is a reasonable choice for learning a feedforward or ballistic controller.

1.5 Research Questions

To the best of our knowledge, there is no method available in the literature that explains how humans learn feedforward control strategy based on reinforcement while taking the muscle activation noise and difficulty levels of the task into account. Hence, we raise the following research questions with the objective of learning a feedforward controller under different difficulty levels and muscle activation noise environment configurations utilizing the DeepRL algorithm Soft Actor-Critic:

- **What are the adaptations required for a Soft Actor-Critic (reinforcement learning) algorithm to learn a feedforward controller?**

Once the SAC algorithm has learned the feedforward control strategy, then we move on to investigate the controller's performance by varying factors such as task difficulty and noise levels. Therefore, we raise another question:

- **What is the relationship between the index of difficulty and the number of ballistic predictions made by the learned Soft Actor-Critic based feedforward controller in the reaching task?**

Solving these questions can provide us insights into motor control and learning process that occur in humans while leading to solutions for the above research questions.

1.6 Contributions

The vital contributions of this work are the adaptations introduced to the SAC architecture to learn a ballistic equilibrium controller successfully. Each adaptation is a contributive factor in learning a ballistic equilibrium controller:

C1: Terminal reward-based ballistic equilibrium controller in a noisy world: We employed the scalar terminal reward structure to address the first research question and introduced actuator and controller noise configurations into the 1D point-to-point biomechanical model. This reward helped the SAC to adapt to the random noise configurations by learning ballistic equilibrium controller behavior in a reaching task setup. In a no-noise condition, we observed that the controller reached the target with only a single prediction irrespective of the ID. On the other hand, with the noise present in the system, the controller ascertained that boosting the magnitude levels of muscle activations can negate the effect of noise to reach equilibrium. It led to learning optimal behavior of a ballistic equilibrium controller, which is a crucial component of the the human motor control framework.

C2: Exponential nature of ID vs. controller predictions curve:
We conducted information analysis to determine the relationship between the ID and the number of controller predictions. We found that the number of controller predictions increased as the ID increased for both the activation and controller

noise configurations. However, the relationship between the number of controller predictions and the Index of Difficulty was found to be exponential. Moreover, we found that the effect of width is more dominant than the effect of distance, which corroborates Welford's observation. Additionally, it also suggests that the ratio of ID and number of predictions is not constant but exponentially increasing with the increase in ID in the reaching task.

Overall, all these adaptations unified together resulted in learning a ballistic equilibrium controller that acted in feedforward mode with SAC architecture and helped us determine the controller behavior under muscle activation noise configuration.

1.7 Summary

To summarize, first, we discussed the motivation of this thesis which arises from the question of how humans learn to control speech? Then, we simplified the problem space from a speech task to a simple control task such as the reaching task. Further, we hypothesized and provided the rationale that reinforcement learning is a possible approach to learning movement control tasks. Additionally, we discussed various learning strategies for motor control and provided a rationale for looking at feedforward control as the first step of learning and the reasoning behind choosing the reinforcement learning approach. Finally, we formulated research questions and explained the contributions that came from this research.

1.8 Thesis Outline

The first chapter provides a detailed introduction, the motivation behind this research, the research questions that drive this work, and the contributions. Chapter two provides a fundamental understanding of the theory of motor control and the related work. Chapter three details the specifics of the biomechanical system and the RL-architecture utilized to learn the feedforward controller. The fourth chapter discusses the results and an in-detail analysis of the learned model. Also, the learning performance of the algorithm was depicted under various experimental conditions. The final chapter details the conclusion, limitations, possible future directions that can take this research further, and the concluding remarks.

Chapter 2

Background and Related Work

The main research objective of this work is to learn a ballistic equilibrium controller that estimates muscle activations required to reach variable width and distant targets present in the kinematic space under muscle activation noise configuration. To achieve this, we use a muscle-based reaching task setup where a ballistic controller is learned to compute muscle activations required to reach targets. Fundamentally, learning a controller to generate movements via computing muscle activations is a classic computational motor control problem. Therefore, this chapter first discusses background details of computational motor control and the tools to deal with a motor control problem. Then, we end this chapter by discussing some of the existing methods to solve motor control problems.

2.1 Computational Motor Control

Motor control theory deals with building computational models that explain various motor control processes that occur in humans [45, 46]. Generally, this process involves various stages as shown in Figure 2.1. The first step involves perceiving the world's information via various sensory organs such as visual, audio to determine the objective or goal of the task. Then, the brain transforms the collected sensory information into neural signals to estimate a pre-programmed motor command. This motor command is usually muscle activations or joint activations, which eventually actuates hands, limbs, or body to complete the goal of the task.

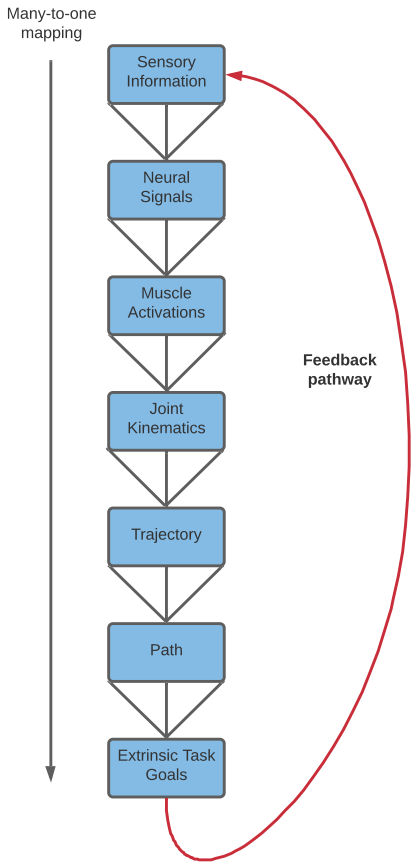

Figure 2.1: Multiple stages involved in the motor control system. This image is adapted from [45].

Additionally, there is a feedback pathway that helps in generating corrective movements. Typically, the feedback is in the form of an error signal using which the motor learning occurs [47]. Similarly, we built a computational model for our muscle-based ballistic equilibrium controller with some adjustments.

In our computational model, the sensory information includes the position of the target in the kinematic space, width of the target. Given the sensory information, the deep neural network (brain or controller) prepares muscle activations. These muscle activations actuate the body or limbs to generate kinematics or move-

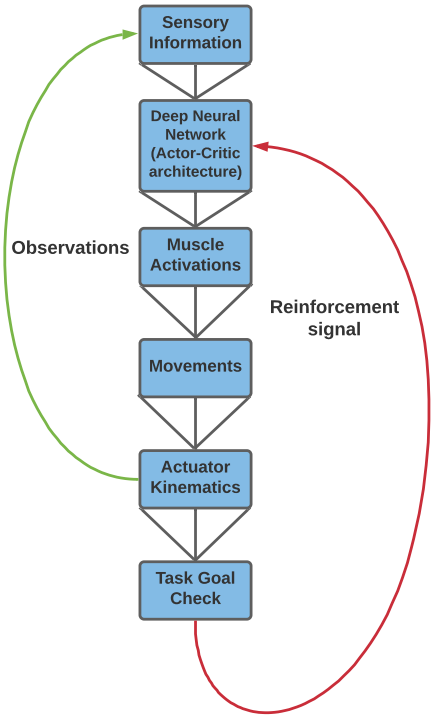

Figure 2.2: Motor planning steps involved in learning the ballistic equilibrium controller. This image is adapted from [45].

ments. A significant difference is that our computational model gets the feedback in the form of reinforcement as illustrated in 2.2. It is argued that reinforcement is the most evident and evaluative feedback for humans in learning motor control [7]. Hence, we use reinforcement to learn a feedforward controller, which is the main research question in this work.

2.1.1 Estimating Muscle Activations

One of the significant aspects in motor control is to estimate muscle activations [8, 9, 14, 16, 20, 89]. Likewise, our computational model for learning a feedforward controller also involves the step of computing muscle activations. There are mainly three categories of methods that are available in the literature to compute the muscle activations.

One, muscle activations can be measured directly using a clinical procedure called electromyography (EMG) [38]. There are two modalities known as intramuscular electromyography (iEMG) and surface electromyography (sEMG) to detect and measure the electric activity produced from the muscle fibers. iEMG requires the insertion of thin electrode needles into the participants' skin to record the muscles' electrical activity [38]. It is an invasive procedure and is known to cause discomfort for the participants. sEMG measures the electric activity by planting the surface electrodes on the skin, and it is the most feasible method [38]. However, these procedures can be unreliable due to multiple reasons. These methods are highly susceptible to noise caused by mechanical artifacts and interference of other muscle activities [25]. Thereby, it introduces the need to eliminate the noise from the data collected. Further, it is impossible to record the electrical activity of the muscles located very deep as the muscles are hard to access.

Second, traditional optimal control methods can also be utilized to solve motor control problem that deals with computing muscle excitations [45, 90]. Many methods emerged based on feedforward and feedback control to estimate muscle activations. Kawato et al. proposed a pure feedforward control strategy that solves a reaching task by computing muscle activations to generate movements in a robotic arm (approximated model of human arm) [49]. Since pure-feedforward can not accommodate visual feedback, the author observed degradation in accuracy. On the other hand, pure-feedback control such as Proportional, derivative, and integral (PID) controllers without the feedforward control is not sufficient for generating relatively fast movements [37]. Hence, many works utilized a hybrid approach that combines both feedforward and feedback strategies to estimate muscle excitations [10, 37] and successfully learned to control musculoskeletal systems. Though these works deal with learning motor control, we believe that a vital aspect observed in humans, i.e., reinforcement, is missing. However, we consider reinforcement learning and use the reinforcement signal to drive the learning process to control movements in our work.

Besides optimal control methods, computational biomechanics can also be used to solve muscle excitations. Computational biomechanics is considered to a limited extent as one of the few possibilities to understand and estimate muscular activation patterns of humans [24]. Calculating muscle activations given the

kinematics is known as the inverse dynamics problem in biomechanical modeling. A handful of studies have been conducted to solve the inverse problem via computational biomechanical solutions [84, 88]. The inverse dynamics problem can be considered an optimization problem for which numerical solvers were utilized to solve. These solvers have either a static or dynamic viewpoint to the optimization problem. In the static optimization method, the objective is to solve an objective function for estimating the muscle activations at each timestep, which drives the agent through the desired trajectory [66]. It has been a widely accepted approach for estimating muscle activations due to its simplicity and low computational costs [68, 80, 88]. Dynamic optimization methods' objective is to optimize an integral cost function by considering muscle forces, joint torques, and other performance measurements, as time-dependent variables [4, 5]. These methods require relatively higher computational power compared to the static optimization methods. More importantly, any inverse dynamics solution inherently relies on the availability of motion trajectories as inputs; however, kinematics are not easy to obtain from human and animal subjects and are susceptible to sensor noise. Even though these methods help solve the inverse problem, they solve a well-formulated optimization function to compute muscle excitations and forces rather than learning to solve it. However, our research problem needs a learning mechanism that can estimate muscle activations. As we established in the chapter 1, reinforcement learning is the driving factor in learning the motor control process. That is why we adopted the RL strategy to learn to compute muscle activations.

2.1.2 Biomechanical Simulation

Biomechanics simulation is a powerful tool in simulating the dynamics of various biological processes [85]. Biomechanical simulation can be used to carefully control the dynamics and environment to study how learning mechanisms respond to changes. Additionally, it can help determine the muscles' mechanical properties, such as variation of the tension with length and velocity, muscle activations that are difficult to measure. More importantly, biomechanically driven simulated frameworks can also offer a rich framework to automate the learning processes of motor tasks [1, 52, 91], which allowed researchers to conduct various studies re-

lating to motor control. Hence, in this work, we take advantage of biomechanical simulation by designing a muscle model that can assist us in performing complex investigations in the muscle space and kinematic space.

2.2 Motor Control Laws

Many studies were conducted to explain the motor control and learning processes in humans [27, 98]. First, a study was conducted by Woodworth that explained how humans generate aiming movements. Followed by Woodworth's study, Fitts has laid the foundation in this domain with his pioneering work. Later, many improvements to Fitts' study have been proposed, such as Welford's law [95]. All these works formulated a metric for the difficulty of the task called the index of difficulty (ID), which affects the movement time in humans.

2.2.1 Woodworth's Law

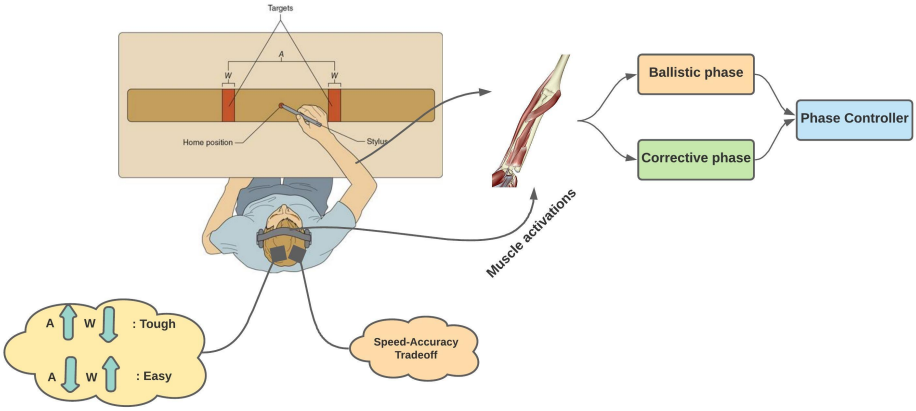

Figure 2.3: Human operator producing ballistic and corrective movements with a phase controller mechanism.

The first substantial study relating to determining the relationship between speed and accuracy was conducted by R.S. Woodworth. In 1899, Woodworth conducted a series of experiments to interpret the processes underlying the control of goal-directed movements [98]. While encompassing many aspects of neuro-motor

control, his work also focused on identifying the parameters responsible for the relation between speed and accuracy of goal-directed arm movements. Woodworth's experiments required the participants to make reciprocal (back & forth) horizontal sliding movements with a pencil over the paper's surface. He proposed that goal-directed movements mainly have two types of phases. The movement starts with an initial impulse phase followed by a current control phase. The initial impulse phase results in ballistic movements, and the current control phase results in corrective movements to correct any spatial errors. This behavior can be interpreted differently. There is no requirement of corrective movements for the easier tasks, whereas, in the case of difficult tasks, corrective movements are essential.

Moreover, humans can transition from the ballistic phase to the corrective phase while performing targeted movements. Once the ballistic trajectory is underway, the current control phase continuously estimates spatial errors to correct using visual feedback to reach a target. When the initial movement results in overshooting or undershooting the target, the brain instantaneously switches to corrective mode to reach the target. This is the reason behind the smooth and continuous nature of human movements. However, it is possible to model and learn to reach the target with only the ballistic phase. Once the agent learns an estimate of ballistic trajectory, the agent can continue updating it until it reaches the target. This approach is simpler in terms of modeling. Therefore, we only learn a feedforward controller that can solve reaching tasks.

2.2.2 Fitts' Law

After Woodworth's law, another groundbreaking motor control called Fitts' law was introduced to study SAT in humans. Fitts designed a serial tapping task where a human operator is supposed to tap a handheld stylus alternately between two target plates as rapidly as possible for a predetermined duration. The two targets are rectangular and oriented as shown in Figure 2.4. As part of his experimental paradigm, he presented various target configurations to the human operator by changing the target amplitude and width. He proposed that varying the target amplitude and target width can make the task easy or difficult. Based on this idea, he formulated a difficulty metric known as Index of Difficulty (ID) that informs how

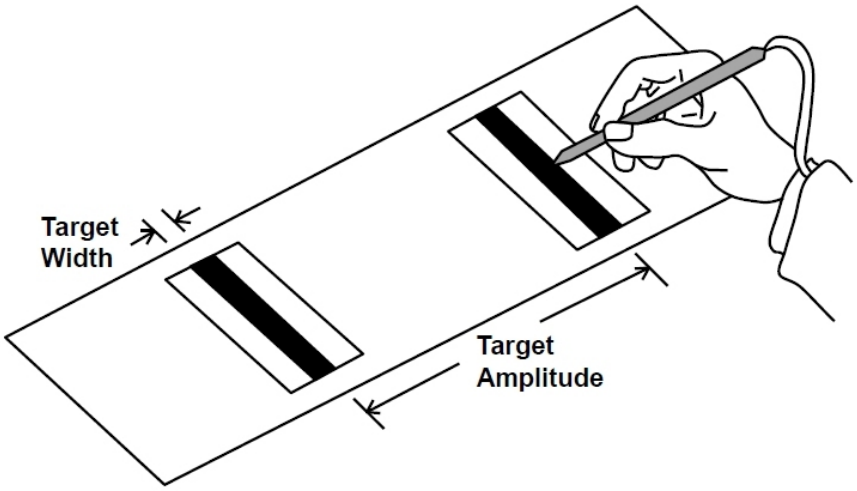

Figure 2.4: Fitts' original paradigm: serial tapping task.

difficult a task can be. The ID of the task is given by the formula 2.1, where ID is the index of difficulty, A is the target amplitude, W is the width of the target. The units of the ID are measured in bits.

$$ID = \log_2\left(\frac{2A}{W}\right) \tag{2.1}$$

Fitts collected the data from various users by presenting tasks with various difficulty levels and recorded their performance in terms of movement time (measured in seconds). Using the collected data, he performed linear regression analysis to determine the relationship between the movement time and the ID as shown in figure 2.5. The regression analysis determined that movement time has a linear relationship with the ID as given by the equation 2.2, where MT is the movement time, a is the intercept of the regression line, and b is the slope of the regression line.

$$MT = a + b * ID = a + b\log_2\left(\frac{2A}{W}\right) \tag{2.2}$$

Moreover, he interpreted that the slope of the "MT-ID" curve indicates the in-

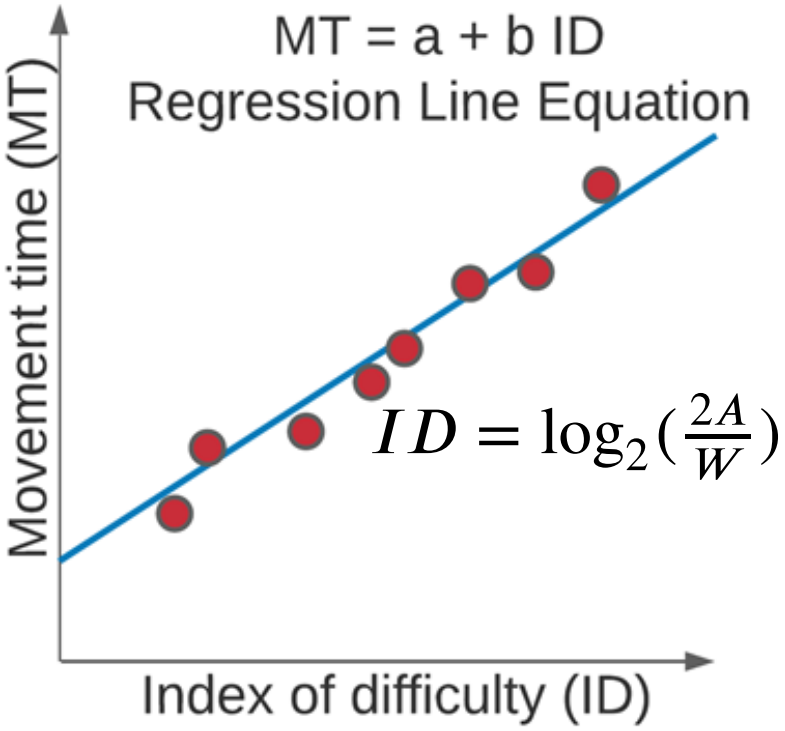

Figure 2.5: Variation of movement time with index of difficulty.

formation rate or throughput (measured in bits/sec) required to complete the task as shown in the equation 2.3, where TP is the throughput, MT is the mean movement time, and ID is the index of difficulty. Also, he observed that the TP remained more or less constant, which means as the ID changes, the MT negates this change, keeping the throughput constant. Further, it can also be interpreted that TP does not depend on the movement amplitude (A) and target width (W), which constitutes ID.

$$TP = \frac{ID}{MT} \tag{2.3}$$

Taking inspiration from Fitts' analysis, we also vary the ID by modifying the distance and target width in the training phase. Additionally, we also introduced muscle activation noise in the training phase, which modifies the task's ID.

2.2.3 Refinements To Fitts' ID

Many refinements were introduced to improve Fitts' law as there are caveats present in the Fitts' ID formulation. The main caveat is that the fitts' ID is not leading to satisfactory curve-fitting of the MT-ID data [94]. To improve the curve-fitting, minor adjustments were introduced separately by Welford and Mackenzie to the mathematical formulation of Fitts' ID. Welford's formulation is given by the equation 2.4.

$$ID = \log_2\left(\frac{A}{W} + 0.5\right) \tag{2.4}$$

Welford found that the slight adjustment of the ID formulation resulted in improvements in regression-line fit compared to Fitts' ID. Moreover, Welford's ID formulation provides improved rationale behind the index of difficulty. When A becomes zero, i.e., the user is on the target, ID becomes infinite using the fitts' index of difficulty, which suggests that the task is incredibly difficult. However, if the user is on the target already, it means the target is acquired, and at that instant, the difficulty of the task is almost zero. Welford's law added an offset value of 0.5 that improved reasoning behind the index of difficulty. He also made a slight adjustment to the movement time equation as shown in the equation 2.5. He proposed that the distance and the width have separate effects on the movement time, which is the reason he separated the distance and width term in the MT formulation in [95].

$$MT = a + b_1\ \log_2(A) + b_2\ \log_2(\frac{1}{W}) \tag{2.5}$$

Taking the inspiration from Shannon's information capacity theorem, Mackenzie increased the target amplitude by a factor of W. Now, the ID becomes as given by the equation 2.6

$$ID = \log_2\left(\frac{A+W}{W}\right) = \log_2\left(\frac{A}{W}+1\right) \tag{2.6}$$

This slight adjustment gave better curve fitting than Fitts' and Welford's ID, demonstrating the linear nature of the relationship between the index of difficulty and the movement time. In addition, Mackenzie's formulation of ID provides a better explanation of the edge cases. For example, when the human operator has already reached the target and need not perform any movements, the ratio $\frac{A}{W} \to 0$ and the $ID \to 0$ as $\log_2(1)$ is zero.

2.3 Two-Phase control model

Previously mentioned control laws have characterized the humans' approach of learning the motor control with the human in the loop [27, 28, 94]. However, there is a fascinating question: how can we create an end-to-end learning model that emulates the human motor learning process? The two-phase control model models the motor learning process to some extent using a cascaded neural network model using robotic arm reaching task.

This method is similar to the descriptive model of Woodworth as it also has two components. As the computational model of the two-phase control model shown in Figure 2.6, the mechanism of this method can be explained as follows: First, task specification is where the objective or the goal of the task is specified. It provides the parameter values specifying the movement time, target position, and target size (i.e., required accuracy). These values are fed to the feedforward controller (the cascade network) as inputs along with the visual information of the target, which prepares a ballistic movement for the specified duration. The specified duration is dependent on the visual information of the target location, i.e., width and distance. This way, the network can easily understand that distance and width are two independent parameters.

At the end of the ballistic phase, control is passed to the postural controller. The postural controller is responsible for generating corrective movements after the ballistic movement. Corrective sub-movements arise as physical oscillations caused by the mechanical properties of the musculoskeletal system. Oscillations are not observed when the position of the hand at the end of the ballistic phase matches

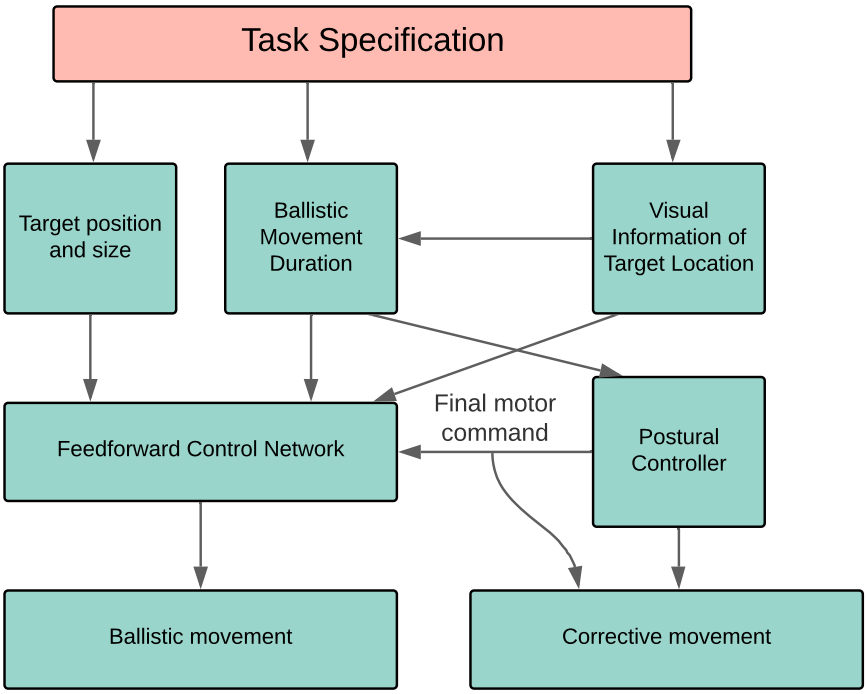

Figure 2.6: Two-phase control model for generating arm movements. This figure is adapted from [37].

with the stationary posture commanded by the postural controller. Oscillations are observed if the two positions differ or same, but the final velocity is non-zero (the hand is not resting). The postural controller specifies the values for the oscillations to converge. This method described why and how the passive oscillation due to the viscoelastic properties of the musculoskeletal system makes movements more accurate under conditions of visual feedback. The authors found that their cascade model generated a planning time-accuracy tradeoff and a quasi-power-law type of speed-accuracy tradeoff while performing hand movements.

This is the only approach available in the literature that uses neural networks to learn to generate aimed arm movements via learning both the feedforward and feedback control. One noticeable difference is that the generation of corrective

movements in this work is different from our method. We recalculate the feedforward command, which acts as a proxy for corrective movements, whereas this method uses a separate feedback controller to generate corrective movements. The major difference is that we use the reinforcement approach, whereas this approach is based on supervised learning.

2.4 Previous works in DeepRL for motor control

With the recent advances in deep learning and its integration into the RL framework, outstanding solutions for control and decision-making problems were introduced [62] Further, DeepRL has achieved considerable success in continuous action spaces [32], extending its possibilities to learn complex motor control problems. Recent studies involved learning to predict joint angles and angular velocities. These studies primarily focused on arm [31, 44] and gait control [53, 69]. Few works have been carried out in the muscle-driven RL-based motion synthesis [13, 42] as well. Besides, solving motor control tasks via predicting muscle activations [2] and joint activations [15] has also been achieved using model-free RL methods. Moreover, works like [1] and [53] helped the researchers to use model-free DeepRL solutions in order to study and understand complex biomechanical environments. These efforts gained more traction after the "learning to run" challenge of NeurIPS 2017, where multiple policy gradient-based controllers were learned to generate gait patterns [53]. Surprisingly, all the solutions that successfully solved this challenge are model-free RL algorithms, which demonstrate the capability of these methods.

However, there are a few subtle differences from these works to our work. First, despite the fact that neuro-motor noise plays a key role in motor learning [92], these works did not consider noise. However, the human motor system has abundant noise and learned motor control considering the properties of noise [92]. Another difference is that we vary the target width along with the distance between the targets because many behavioral studies demonstrate that the width of the target also plays an important role in motor learning and behavior [27, 94]. Therefore, we train our Soft Actor-Critic to learn the feedforward controller under muscle activation noise configuration with varying difficulty levels of the task.

2.5 Summary

In summary, this chapter gives the required background and overview of the related works in the motor control research domain and a detailed introduction to Deep Reinforcement Learning. First, we presented the background details of the computational motor control frameworks, significance, and challenges involved in predicting muscle activations. Second, some of the descriptive control laws that model and interpret the speed-accuracy trade-off principles are discussed. Finally, introduction to reinforcement learning, Deep Reinforcement Learning, building blocks of the core algorithm in our proposed methodology were elucidated.

Chapter 3

SAC-Based Architecture To Learn Ballistic Equilibrium Controller For A Muscle-Based Reaching Task

This chapter discusses the details of the SAC-based architecture designed to learn a feedforward controller using a muscle-based reaching task under muscle activation and controller noise environment configurations. To achieve this, we first design a biomechanically driven muscle model that can be used to create a motor control task, namely a reaching task that inherently has the muscle activation noise. Then, we deploy a deep reinforcement learning algorithm known as Soft Actor-Critic that can be trained to learn the reaching task via reinforcement under activation noise configuration. For designing a reaching task experiment, we considered a point-to-point muscle-based biomechanical model as shown in Figures 3.1, 3.2, 3.7.

3.1 Motivation behind the point-to-point model

This section discusses the motivation behind the muscle model that is used to design the reaching motor task and the choice of the algorithm to learn a feedforward controller.

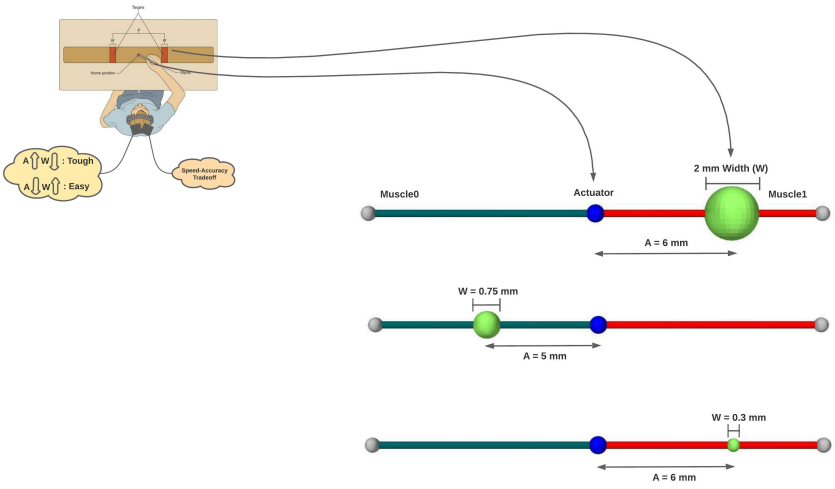

Figure 3.1: Antagonistic-agonistic muscle based biomechanical model that can generate 1D movements that are similar to the movements in the Fitts' task.

3.1.1 Why point-to-point model?

Fitts formulated a well-designed experimental paradigm where a human operator can make horizontal movements to reach the targets. Additionally, he provided a way to calculate the task's difficulty depending on the amplitude of the horizontal movements and the target width. Since our research questions also include determining the behavior of the learned controller, we designed a linear muscle-based reaching task that is similar to Fitts' task. The muscle-based model captures the Fitts' task's intricacies well while keeping the task not so complicated, such as human pantograph arm models or other higher degree-of-freedom models. Though our muscle model is simple, it still preserves the complexities of the reaching task problem. Because to learn to reach targets, the learning algorithm needs to determine the right amount of net excitation to co-activate the muscles to reach an equilibrium state. There exists an infinite number of solutions to this problem (degree of freedom problem) [33]. Choosing an appropriate co-activation from an infinite number of solutions is challenging. On top of this, we also dynamically

vary the difficulty of the task and muscle activation noise levels making the system stochastic. Hence, it makes the reaching task even more challenging for the SAC algorithm to learn.

3.1.2 Why SAC?

As mentioned earlier, we take the approach of reinforcement to solve our research questions. In reinforcement learning, there exist different types of learning algorithms. The one that is in our best interest is the family of model-free RL algorithms because their objective is to determine the optimal policy through maximizing the expected return (see Appendix, [3]), which is also observed in humans [23]. So, the model-free learning algorithms are a suitable choice, and it is possible to learn the task with any algorithm from the model-free family. For example, Abdi et al. solved the reaching task problem using the normalized advantage function-based Deep Q-network (DQN) algorithm [32]. In this work, we chose the soft actor-critic algorithm for its sample efficiency and robustness to the hyper-parameter tuning [35].

3.2 Experimental Paradigm

Now, we explain the experimental paradigm used in this work by discussing the research objective, experimental hypothesis, independent variables in the experiment, noise configurations used in the experiments, and apparatus used to execute experiments.

3.2.1 Research objective

The objective of this experiment is two folded as discussed previously. First, learning a feedforward controller using SAC (reinforcement algorithm) under activation noise configuration, and second, analyzing the behavior by looking at what the algorithm has learned of the learned controller. Keeping these research objectives in mind, we formulated this experimental framework to control the tasks' difficulty and noise levels.

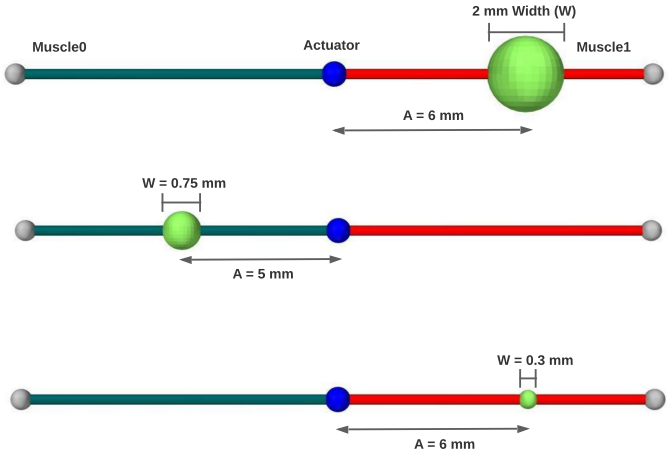

Figure 3.2: Experimental setup used for investigating the effect of ID and noise on the motor control.

3.2.2 Experimental hypothesis

In a reaching task under various noise sources, the RL controller must act in purely feedforward or ballistic mode to determine the desired co-activation to reach the target by minimizing the number of feedforward or ballistic predictions based on reinforcement signal. Then the agent should take a minimum number of ballistic predictions to reach the targets with low ID and uncertainty (easy) and increase the number of predictions as the ID and the noise in the system increases.

3.2.3 Independent variables

The variables modified in our experiments are the distance between the actuator and target (A), target width (W), and the amount of uncertainty or activation noise (σ) present in the system. These parameters alter the task's difficulty allowing the agent to get exposed to targets with various difficulty levels during the training.

A & W:

Fitts has suggested that the distance between the targets and the target width defines the task's difficulty. Similarly, in our reaching task, the distance between the actuator and the target and the target width as shown in Figure 3.2 governs the difficulty of the task. Thus, we use Fitts' ID to calculate the difficulty of the task in our experimental analysis.

Motor noise

Apart from the distance and the target width parameters, the noise present in the human motor apparatus (motor noise) also affects the movements [77]. Many motor control theories suggest that the brain is a stochastic control system [33] that has learned to generate movements by considering the properties of the noise. Noise can exist at various levels in the motor system. For example, there exists a random noise in the muscular model [77] and the motor commands or outputs of the controller [73]. As it is difficult to consider every noise source, we chose two kinds of noise sources in this work. One is the activation noise, and the second one is the controller noise.

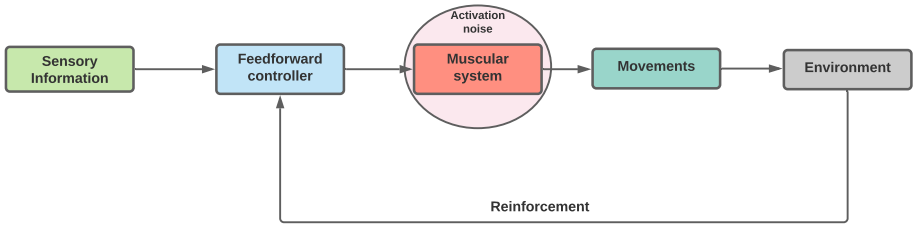

Figure 3.3: Simplified representation of a generic motor control system with activation noise highlighted.

Activation noise: Activation noise is the noise present in the muscular system as shown in Figure 3.3. This type of noise can be modeled by continuously co-activating the muscles with random activations at every simulation step. It is independent of the controller noise, and it can be interpreted as the tremors observed

in motor diseases such as Parkinson's disease. People affected with motor control disorders can experience tremors or continuous shaking in their hand movements, making it difficult to have steady hands. Similarly, the actuator continuously oscillates due to the activation noise, hindering the actuator from getting to the equilibrium point. Therefore, if the activation noise is high in the system and the target is far and small, the actuator can never get to equilibrium leading to the failure of the task.

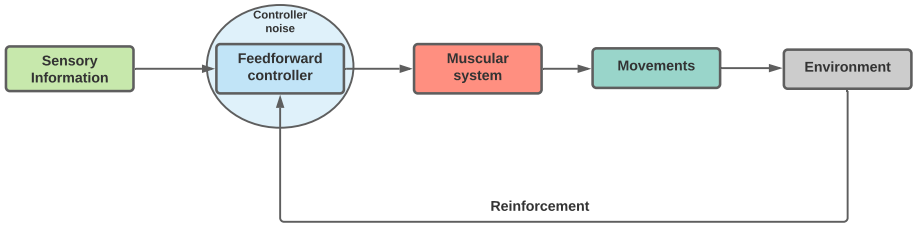

Figure 3.4: Simplified representation of a generic motor control system with controller noise highlighted.

Controller noise: The controller noise configuration is defined as noise present in the motor commands (muscle activations) predicted by the controller as shown in 3.4. So, when the controller predicts the co-activation, the prediction gets contaminated by the noise. The controller noise can be interpreted as the noise that comes from the gap between what the brain or controller predicts and the actual signal that goes to the muscles. If the noise levels are high and the target width is far and small, the actuator might never reach the target leading to the failure of the task. This way, the difficulty of the task is modified by varying the level of uncertainty in the system. Since the controller noise is random, the controller gets exposed to various levels of noise ranging from low to high during training, which allows the agent to understand and consider the statistics of the noise while estimating co-activation. In this work, a Gaussian random process is used to generate random muscle activations which are added to the controller's prediction.

Effect of motor noise: When there is no noise in the system, the muscle system becomes a deterministic system. The deterministic (noise-free) system can be solved analytically without needing to leverage powerful DeepRL algorithms. However, with noise, the system becomes stochastic, and it is not obvious how to solve the problem analytically, yet the effect of noise on learning can be understood to some extent provided the amount of uncertainty. However, as our object is to learn to predict co-activation, we take the approach of RL. The controller's fundamental task is to determine the co-activation required to reach the target in this learning problem. Assuming that there are no activation constraints such as excitation regularization (i.e., low magnitude activations) embedded in the system, the controller can choose co-activation with magnitude ranging from zero (low) to one (high). But, low co-activation solutions are more susceptible to noise. When the noise is added to the predicted low-magnitude co-activation, the total magnitude of the co-activation changes to a higher degree, resulting in more displacement than the controller predicted. Hence, the likelihood of missing the target increases. Then, the agent gets penalized for missing the target. This imparts pressure on the agent to increase the magnitude levels of the co-activation as a way to negate the effect of noise in the system.

Overall, similar to Fitts' analysis, we vary the independent parameters: distance to the target, target width in our experiments. So, in our experiments, the task's difficulty varies depending on the values of A & W, as Fitts suggested. Additionally, we consider another independent parameter in our experiments, termed muscle activation noise (σ), because we hypothesize that the muscle activation noise also contributes to the task's difficulty. It means near and bigger targets with low activation noise in the system are the easiest. Farther and smaller targets with high activation noise in the system are the most difficult to reach. So, during training, we vary A, W, and σ parameters from episode to episode as the reinforcement learning algorithms are trained in an episodic fashion [87] (see Appendix for more details on the episode). This way, the agent gets exposed to various difficulty levels, which creates pressure to explore various strategies to reach the target.

3.2.4 Working mechanism of experiment

So far, we have discussed the working mechanism of each component of the architecture. Now, we discuss how all these components work together as a whole unit in an experiment.

During the initial phase of an experiment, the actuator rests at origin as no activations are coming from the controller. The controller (actor-network) starts exploring the task space by predicting random activations to start the learning process. For every prediction, corresponding observation (action, actuator position A_p, target position) will be collected and stored in the buffer. Meanwhile, the perception process checks the success criteria and assigns a scalar reward or penalty to every prediction made by the controller. Depending on the type of muscle activation noise, the noise gets added into the system.

At the end of each episode, the activations are set to zero, so; the actuator starts from the origin at the beginning of the episode. Starting the actuator always from the origin might give the impression of introducing a learning bias of reaching the target only from the point of origin, i.e., it can not perform the reciprocal task. However, there would not be such bias in the system as the agent learns to predict the muscle activations provided any position of the actuator in the kinematics space within a certain radius. This radius is a parameter to generate targets within the prescribed range in the kinematic space during training. For example, if the radius is set to 6 *mm*, the targets generated that are generated randomly will be within the range -6 *mm* (negative sign indicates left direction) to 6 *mm* (right direction). The following video showcases the controller performing the reciprocal task with the controller noise configuration during the testing phase. However, the controller was trained with the zero-excitation setting (forcing the controller to set to origin after every episode). The controller is provided with a target at 4 *mm*, switching alternatively from right to left, and the target width is configured to 0.3 *mm*.

3.2.5 Apparatus

The tools used for conducting our experiments include a platform-agnostic software service called ArtiSynth-RL, a biomechanical modeling toolkit, ArtiSynth [57], and a python-based deep learning library called “PyTorch” to implement SAC al-

gorithm.

Point-To-Point Model

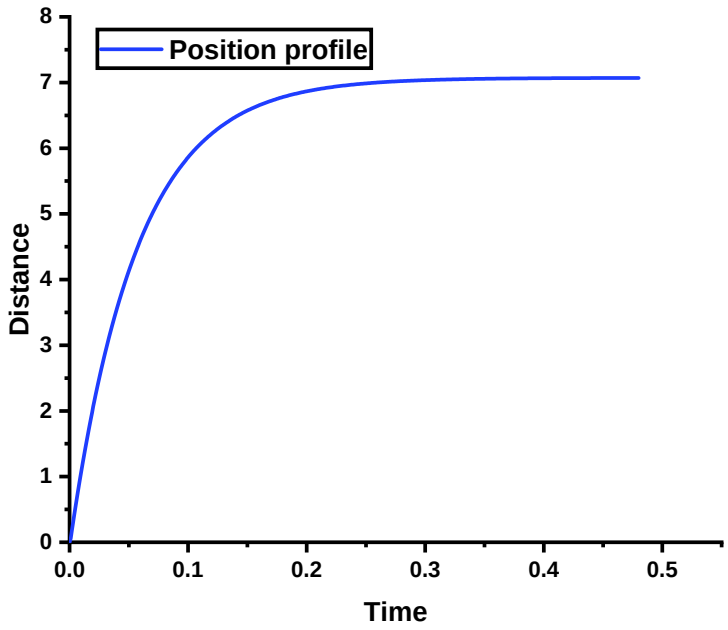

Figure 3.5: Position profile of the critically damped of out point model given a 0.5 net excitation covering a distance of 7 *mm*.

Applying co-activation to the muscles moves the actuator or mass object (blue spherical object in Figure 3.7) to reach an equilibrium point in the kinematic space. One muscle can exert opposing force to the other muscle, holding the actuator at a specific point. Both ends of the muscles are fixed (grounded), whereas the other ends are joined by a controller (object) as shown in Figure 3.7. The actuator gets pulled when there is an external force or activation applied. Depending on the net muscle excitation applied to the muscles, the actuator moves away or towards the target. If the second muscle (Muscle1) activation is higher than the first muscle (Muscle0), the actuator gets pulled to the right as shown in Figure 3.7. Controller

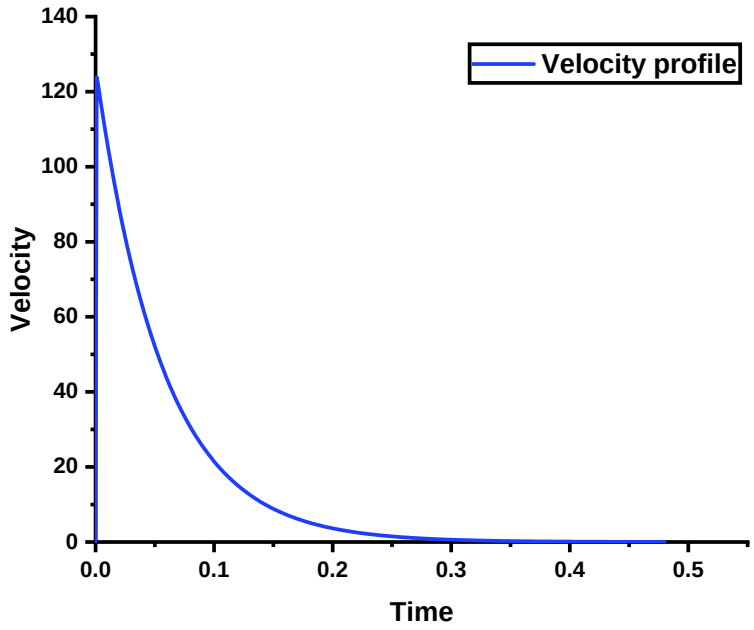

Figure 3.6: Velocity profile of the critically damped of our point model given a 0.5 net excitation.

gets pulled to the left if the first muscle activation is higher than the second muscle. A simple analogy that best explains the point model's dynamics would be the tug of war game where the two teams pull the rope from both ends. The team that exerts more force can bring the other team over the boundary line, leading to the game's winning. Similarly, the actuator gets pulled in the direction of the high net force. The point model is composed of linear muscles, i.e., the muscles in our model have a linear relationship between the length and tension, as well as linear damping as shown in Equation A.3. On the other hand, there are muscle models that have a non-linear relationship between length and force [11, 61, 67]. Non-linear muscle mechanics make the motor control problem at hand more challenging. So, we utilize linear muscles to make the muscle mechanics simpler and the motor task not too complicated. It is important to note that this simplification does not make

the learning challenge easier. The system still has to determine a unique solution among an infinite solution space. Additionally, it has to learn to counteract the noise that hinders the actuator from reaching the target.

First, we discuss the details of the point-to-point model. The point model constitutes an antagonist and agonist muscle pair as shown in Figure 3.7. Muscles utilized in our model are designed to have a linear relationship between length and tension as given in the equation A.3.

$$F(x,\dot{x}) = \left(af_{max} + f_{passive}f_{max}\right)x' + d\dot{x} \tag{3.1}$$

where x' is defined as $x' = \frac{x - x_{opt}}{x_{max} - x_{opt}}$, x_{opt} is the optimum length of the muscle, x_{max} is the maximum length of the muscle, x is the displacement of the muscle, a is the activation vector, f_{max} is the maximum force, $f_{passive}$ is the proportion of f_{max} to apply as passive tension, $\dot{x}$ is the velocity, and d is the damping coefficient.

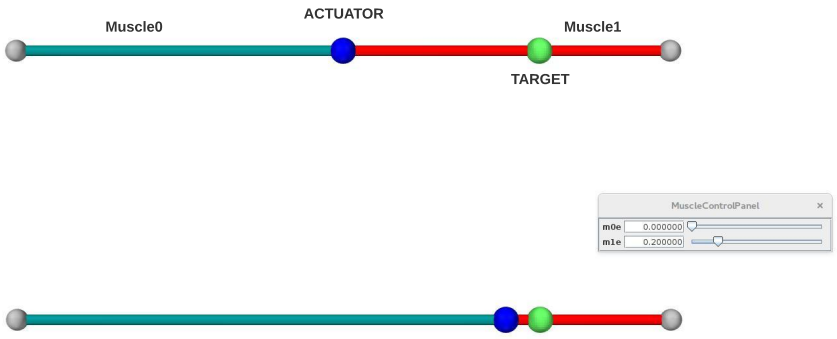

Figure 3.7: Two-muscle model setup.

Given the activations, the point model can generate the kinematics using the forward dynamics model. The mechanical and muscle properties of the system are given in the tables 3.1, 3.2 respectively and are kept constant within an experiment. In other words, mechanical parameters will not be changed from episode to episode

Table 3.1: Mechanical properties of the point-to-point model

Damping coefficient: d (Ns/mm)	Passive fraction	Mass of the controller (grams)
0.001	0.1	1

Table 3.2: Muscle properties of the Point-To-Point model

Muscle length (mm)	Maximum muscle length (mm)	Rest length (mm)
10	20	10

within an experiment, and these have the same effect on all the generated activations. An important point to note is that the system's damping is set so that the system is critically damped. As a result, there will not be any oscillations observed in the system on the application of activations as shown in Figure 3.5 and 3.6. It can be observed that, upon the net activation of 0.5, the actuator has generated a displacement of 7 *mm* without any oscillations. Similarly, net activation of 0.5 has produced a velocity profile without any oscillations given in Figure 3.6.

Effect of mechanical parameters Other than the ID, activation, and controller noise, some factors can affect the movement of the actuator. Since we utilize a biomechanical system where the system is composed of linear muscles, mechanical parameters of the system can affect the movement behavior of the system. Most of the biomechanical system's functionality is governed by mechanical parameters such as damping in the muscles, spring constant, damping coefficient, and the actuator's mass, and so on. The parameter in our best interest is muscle damping, as it is one of the critical factors that can play a major role in effecting movements. Moreover, muscle damping determines the damping coefficient, which governs the system behavior. Depending on the damping coefficient, the system can be characterized as critically damped, under-damped, and over-damped. Among the three cases, the under-damped case can be useful to generate high-speed movements to reach a target. Because there is a time instance where the actuator hits the target before the commencement of the oscillations. However, due to the oscillations actuator can not reach the equilibrium as the system is unstable. But these oscillations can be stopped by an external component such as a postural controller whose

task is to stop the actuator from oscillating [37]. In this work, for simplicity, we consider only a critically damped system case at the moment.

ArtiSynth-RL platform

In order to create a DeepRL framework that bridges the biomechanical platform and the DeepRL, a plugin based on Representative state transfer (REST) API was developed. It facilitates the deployment of reinforcement learning-based motor control strategies while interacting with biomechanical models in real-time. It is a scalable server-client interface that enables many instances of ArtiSynth to run in parallel for synchronous/asynchronous RL training. This plugin is interface-friendly to any deep learning library that can be written in the Python programming language. REST calls (i.e., GET, POST) were utilized to perform specific utility commands such as reset the task, get action and state-space sizes, etc. (for more details, see [1]). This work uses this platform to deploy the SAC algorithm unto the point-to-point model to develop a feedforward control strategy.

3.3 Overview of the SAC-based architecture for learning feedforward controller

In this section, we discuss the overall architecture and the building blocks of the architecture shown in Figure 3.8. Then finally we explain how we learned a feed-forward controller with the SAC algorithm.

As shown in Figure 3.8, the architecture consists of three major units or blocks: An SAC-based neural network that is responsible for generating motor commands, a muscular system (point-to-point muscle model) that takes input as motor commands, and a simplified perception process that deals with feedback and reward. We discuss each of these blocks individually to explain their tasks in the learning or training process.

At any instance, single muscle activation is communicated to the point model. The activations will cause the actuator to move from one position to another and eventually brings the actuator to a resting position. The scalar reward signal drives the learning of the controller. The observations include the current position and velocity of the actuator, target position, target width, and these are fed into the policy

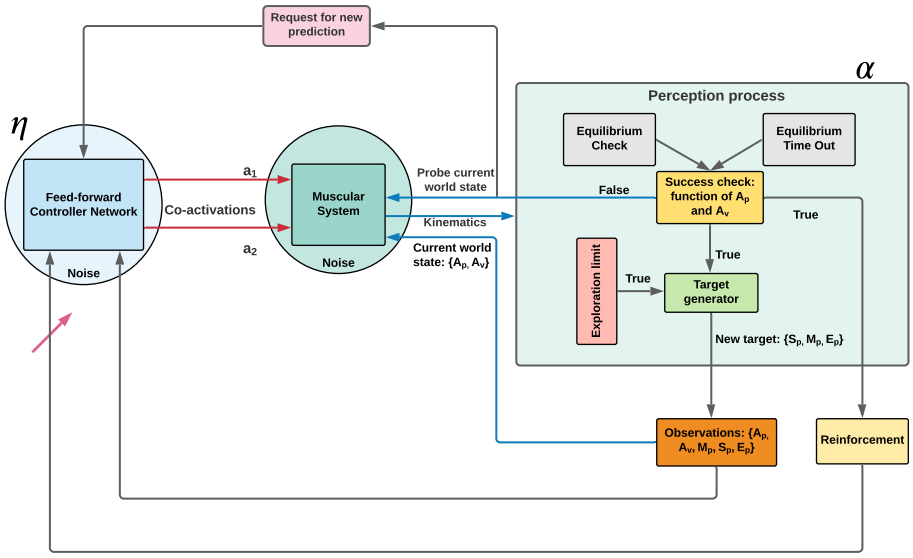

Figure 3.8: Architecture of the RL framework. Feed forward controller network prepares a muscle activation values which are mixed with gaussian noise and represented by a_1, a_2. Point-To-Point model is the biomechanical system as shown in Figure 3.7. Upon activations, the point-to-point model generate kinematics (position and velocity of the actuator). Observation vector include actuator position, starting point of the target, middle point of the target, ending point of the target, actuator velocity are denoted by A_p, S_p, M_p, E_p, A_v respectively. α and η are the execution or running rates of the perception and controller processes respectively.

network and Q-network. This whole process is repeated until an optimal policy is learned. Now, let us look more deeply into each component of the architecture.

3.3.1 Feedforward controller

The main task of the feedforward controller (roughly translated as the brain) is to learn to prepare a motor command or muscle activations required for the actuator to reach the task under the activation noise configurations given the position of the actuator and the target, velocity of the actuator and the target size. The input and the outputs of the controller are represented in a block diagram format (see Figure

3.9).

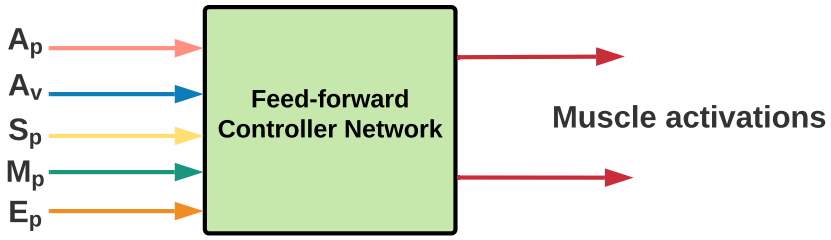

Figure 3.9: Schematic representation of inputs and outputs of the feed-forward controller network. A_p is the actuator's current position. A_v is the actuator's velocity. S_p, M_p, and E_p are the starting point, middle point, and ending point of the target.

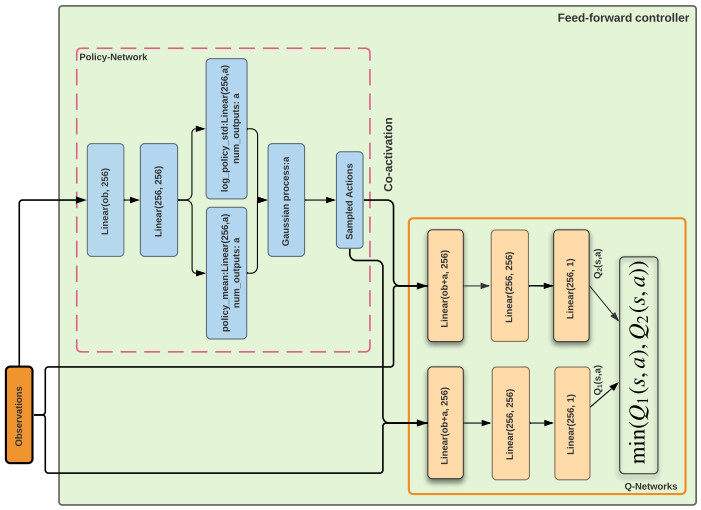

Figure 3.10: Detailed representation of the feed-forward controller network. Observations include A_p, A_v, S_p, M_p, and E_p. Policy network prepares the co-activation given the observations. Q-network predicts the Q-function given the observations and actions or predictions of the policy network.

In SAC, the feedforward controller consists of a policy network and two Q-networks. Policy network prepares actions (muscle activations or co-activation) given the observations. The other two networks are unified together called Q-network as it learns an optimal Q-value function that informs the policy network whether the activations resulted in maximum expected return or not. The policy network is commonly referred to as the actor, and the two Q-networks combined together are called the critic. Multi-layer perceptron network architectures are used as policy and Q-networks.

The policy network generates muscle activations that are required to generate kinematics in the biomechanical system. The resulted changes of the world are acquired as observations and used as inputs for both the policy and Q networks. These observations include the current position of the actuator, actuator velocity denoted by A_p and A_v as shown in Figure 3.8. Another vital input of the controller is the target width represented by the starting, middle point, and ending point of the target denoted by S_p, M_p, and E_p respectively. The actor-network prepares co-activation using these inputs, and as a result, the actuator moves. The actuator eventually reaches an equilibrium state as the perception process waits till the actuator gets to equilibrium. Hence, our controller can be called a feedforward equilibrium controller or ballistic equilibrium controller. The new observations are collected and stored in a memory buffer and used as data samples to train the policy and Q-networks. More details about SAC and how it is used to learn to control movements are discussed in the section 3.4.

3.3.2 Perception Process

The main task of this perception process is to continuously apply the co-activation till it gets the new prediction of co-activation. While the activations are getting applied, it simultaneously checks the status of the actuator. At every simulation step, the perception process checks whether the actuator is in equilibrium or not. If the actuator is not in equilibrium, then it waits till the actuator is in its equilibrium (velocity ≈ 0). Once the actuator is in its equilibrium position, the perception checks if the actuator is on the target or not. Since this process continuously runs at the simulation rate, its update rate is higher than the feedforward controller's

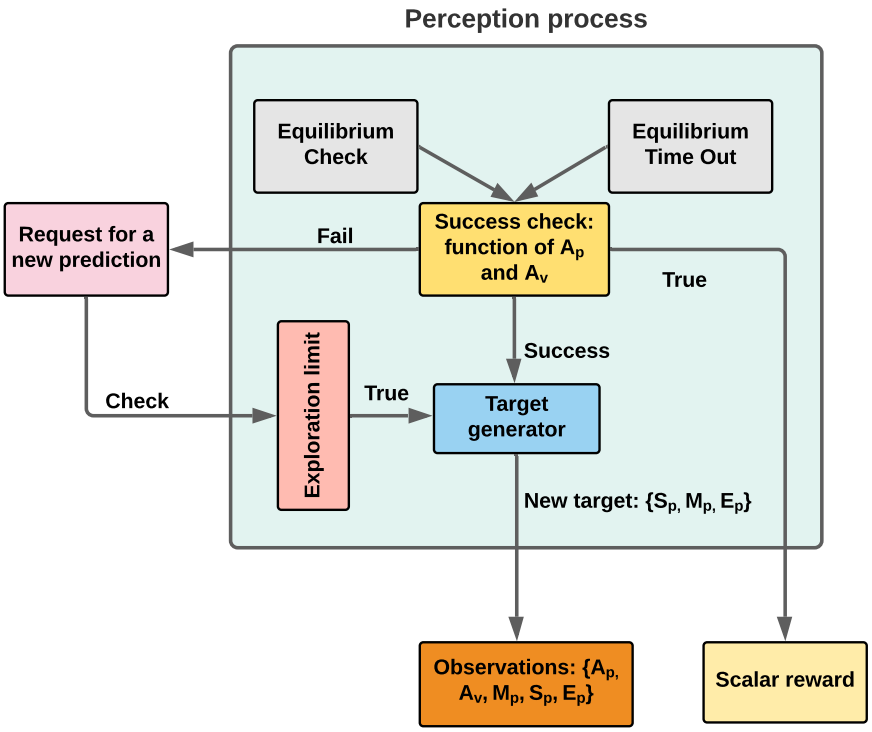

Figure 3.11: Schematic representation of the perception process.

update rate. This idea is inspired by the human's feedforward-perception loop. However, it is important to note that the perception process in humans does not wait for the limbs or actuators to get to equilibrium. Rather, a correction movement is generated as the ballistic movements unfold. Since it is challenging to design this feature, as a first step, we simplified the perception process to wait for the equilibrium. Waiting for equilibrium depends on the magnitude of co-activation. The higher the co-activation force, the lesser the time it takes to reach equilibrium. Therefore, there is a dynamic delay between the controller predictions, unlike the previous works such as [3], and [32].

Since there is noise present in the system, the actuator sometimes may never reach equilibrium during training. So, to avoid this, we maintain a time-out con-

dition for the equilibrium. We employ a fixed delay for the actuator to get to equilibrium. If it fails to reach equilibrium within the tine-out value, the old target configuration (A, W) is replaced by a new target configuration. Then, the perception process sends out a request to the controller asking for the new prediction for the new target configuration. The remaining blocks of the perception process are described in the following sub-sections.

3.3.3 Success criteria

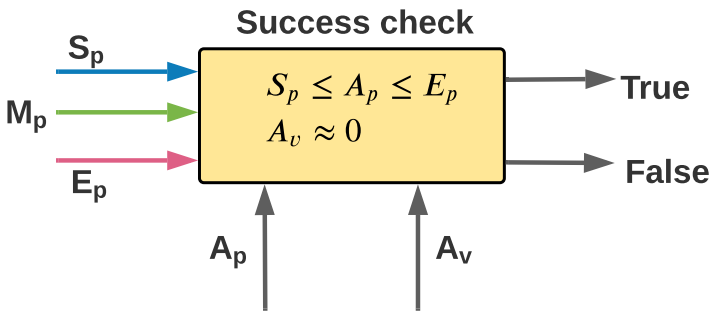

Figure 3.12: Criteria for success in reaching task for the agent.

One of the key elements in the RL framework is the success criteria. It informs the agent whether a certain muscle activation prediction led to success or not. The agent is considered successful when the actuator is at equilibrium, and the target's position is within the prescribed target width. For example, imagine a 1 mm target located at 5 mm.In order to be successful, the agent should predict muscle activations that bring the actuator's position anywhere between 4 mm (starting point) and 6 mm (ending point), and the actuator should be resting (i.e., $A_v \approx 0$). Success criteria are checked continuously after every update. Every action is mapped to a reward or credit, helping the agent distinguish the best policy from the bad one.

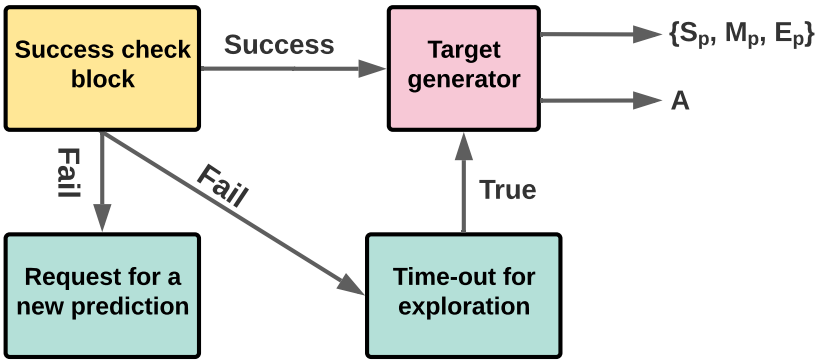

Figure 3.13: Criteria for generating new targets.

3.3.4 Target generator

This block represents the mechanism of generating new targets during the training phase. The target generation mechanism is dependant on two conditions, the success criteria and the exploration limit. A new target is generated if the agent successfully reaches the target, with a new location and required accuracy. If the agent fails to reach the target, then the agent checks the exploration time-out condition. Exploration time-out is a fixed number of predictions that the controller is allowed to do per episode. If the number of controller predictions crosses the prescribed limit and the agent fails to reach the target, a new target is generated. This is a general practice in many DeepRL techniques, which helps the agent not to stuck in bad policies. In our work, the exploration limit is set to 100 predictions per episode. If the controller fails to reach the target in an episode, a new target is generated with a different configuration from the previous target as the target configurations are generated in a random fashion.

3.4 Learning feedforward controller with Soft Actor-Critic

In reinforcement learning, the objective is to determine a domain-specific policy representation π [70], which maximizes the expected return (J). A policy is a

function that maps the agent's action space to the state space. In training, the agent's task is to find an optimal policy (π^*) that results in maximum rewards. The policy can be either a deterministic or probabilistic distribution (stochastic) mapping function.

$$\pi^* = \arg\max_{\pi} J(\pi) \ , \tag{3.2}$$

$$J(\pi) = \mathbb{E}_{a_t \sim \pi(.|s_t)} \big[\sum_t \gamma^t r(s_t, a_t) \big] \ , \tag{3.3}$$

where s_t and a_t is the state and the agent's action at time t respectively, γ is discounting factor for future rewards, and $r(.)$ represents the reward or penalty associate with actions given the states.

In the muscle-based reaching task (see Figure 3.7), the actions are the co-activation that moves the actuator in the kinematic space. So, to reach the target, the agent must learn to estimate the co-activation required. Hence, the agent starts off the learning process by interacting with the environment (muscle-model) by randomly predicting co-activation and gathers respective states and rewards. This stage in training is generally referred to as exploration. Then, the agent exploits the current optimal policy. This process is called exploitation. An RL agent needs to maintain a tradeoff between exploration and exploitation during the training.

We utilize the Soft Actor-critic (SAC) algorithm for learning reaching task which belongs to the family of model-free RL. Unlike other RL algorithms, SAC is considered robust to hyper-parameter tuning, which is a convenient option for our motor control problem. SAC maintains an additional term in the optimal policy equation as given in Equation 3.3 and the final optimal policy equation is given in Equation 3.4. $\mathscr{H}(.)$ is the entropy of the policy given by the Equation 3.5. The entropy term is controlled by the temperature term α that governs the strength of the exploration. Higher the value, the higher the importance of exploration.

$$\pi^* = \arg\max_{\pi} \ \mathbb{E}_{a_t \sim \pi(.|s_t)} \big[\sum_t \gamma^t \big(r(s_t, a_t) + \alpha \mathscr{H}(\pi(.|s_t)) \big) \big] \ , \tag{3.4}$$

$$\mathscr{H}(\pi(.|s_t)) = \mathbb{E}\big[-\log \pi(.|s_t) \big] \ , \tag{3.5}$$

Moreover, the agent is rewarded according to the entropy $\mathscr{H}(.)$ of the policy as a way to avoid learning bad policies.

In our implementation, the SAC algorithm has two main neural networks(function approximators): the policy network (actor) and the action-value function network (critic). The policy and action-value functions are parameterized by ϕ and θ, respectively. In training, at each timestep t, the actor-network prepares a co-activation (a_t) given the current state of the environment s_t. Consequently, the actuator ends up in a new state s_{t+1}. In the muscle-based model, the state comprises the target position, target width, current position of the actuator. During training, the history of interactions with the environment is stored in a memory buffer ($\mathscr{D}$), known as the replay buffer [62]. Each sample in the replay buffer, (s_t, a_t, s_{t+1}, r_t), is an observation tuple that contains current state, executed action, next state, and the reward associated with the transition. The data in the replay buffer is the training data which helps the training of the Q and π networks multiple times [62].

The action-value function represents the expected reward as shown in Equation 3.6. It can also be interpreted as an estimation of the value of an action given a state based on the reward signal. The Q-function is computed recursively with the help of Bellman equations [56] as shown below:

$$Q_\theta(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathscr{D}} \left[V_{\bar{\theta}}(s_{t+1}) \right] , \tag{3.6}$$

$$V_{\bar{\theta}}(s_t) = \mathbb{E}_{a_t \sim \pi_\phi(.|s_t)} \left[Q_{\bar{\theta}}(s_t, a_t) - \alpha \log \pi(a_t|s_t) \right] . \tag{3.7}$$

In the above equations, $V(.)$ is the value function defined as the expected future reward of a state. Because of the entropy term, the value function is changed according to the equation 3.7. In the context of the muscle model, the value function approximately indicates the likelihood of the agent to reach the target and receive high reinforcement. Using the value function, SAC calculates the Q-function as given in Equation 3.6.

As mentioned earlier, the data samples collected in the memory buffer are used for training the policy and Q function networks. The parameters of the Q-network are updated by backpropagating the Q-function error using stochastic gradient descent. The Q-function error is calculated as the mean-squared error between the

predicted Q-values and the ground-truth Q-values. The ground truth Q-values are calculated using the current reward (r_t) and the discounted state-value function ($\gamma V_\theta(s_{t+1})$). Thus, the objective function of the Q-network becomes:

$$J(Q_\theta) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathscr{D},\ a_{t+1}\sim\pi_\phi}\left[\left(Q_\theta(s_t,a_t) - \left[r_t + \gamma\mathbb{E}_{s_{t+1}\sim\mathscr{D}}\left[V_{\bar{\theta}}(s_{t+1})\right]\right]\right)^2\right]. \tag{3.8}$$

It is important to note that Equations 3.7, 3.8 use $\bar{\theta}$ instead of θ to denote the parameters of the networks. This is to showcase the common practice employed in training reinforcement algorithms. Critic network is modeled with two nueral networks with the same architecture but with independent parameters [63]. The secondary network is called the target network, denoted as $Q_{\bar{\theta}}$. It is used to estimate the *assumed* ground-truth value of the next state in Equations 3.7 and 3.8. Maintaining a target critic network has demonstrated in stable and smooth training process [63].

Another feature of the SAC that played a key role in learning the reaching task is the double Q-learning [36, 93]. Two Q-networks are used together to representing a single critic. Since there are two critic networks (main and target), two Q-networks are used for both of them. Minimum of the outputs of two Q-networks is used as an estimate of the Q-values:

$$Q_\theta(s_t,a_t) = \min(Q_{\theta 1}(s_t,a_t), Q_{\theta 2}(s_t,a_t))\ . \tag{3.9}$$

Using double-Q learning has resulted in speeding up the training and improve the performance of the algorithm [34].

Now coming to the policy network, the parameters of π_ϕ are updated to maximize the expected future return as well as the expected entropy. If the Q-function (critic) is assumed to be telling the truth, finding the optimal policy is the same as maximizing $\mathbb{E}_\pi[V_{\bar{\theta}}(s)]$. This can be expanded, based on Equation 3.7, as follows

$$J(\pi_\phi) = \mathbb{E}_{a\sim\pi_\phi, s\sim\mathscr{D}}\left[Q_\theta(s,a) - \alpha\log\pi(a|s)\right]. \tag{3.10}$$

Unlike Q-networks, the policy objective is optimized via stochastic gradient ascent based on the random samples drawn from the replay buffer ($\mathscr{D}$) as the idea is to

maximize the expected return. The training is carried till the objective functions of the actor and critic networks are converged to a saturation point.

3.4.1 Reward Structure

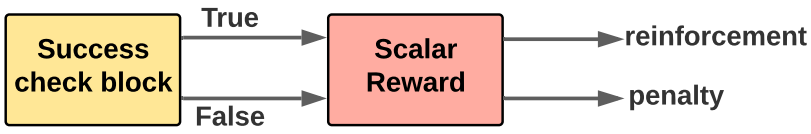

Figure 3.14: Scalar reward structure. Reinforcement on success and penalty otherwise.

In our work, we have a simple and basic reward structure, and a reinforcement is awarded only if the controller is successful in the task. In other words, the agent is penalized for every action prediction except for the prediction that led to successful reaching. This mechanism is analogous to the reinforcement observed in humans [23]. Block diagram representation of the reward structure is shown in Figure 3.14. Reinforcement represents a positive scalar value and the penalty represents a negative scalar value. In our work, we used +5 and -5 to represent the positive and negative reinforcement. This reward structure pressures the system to minimize the number of controller predictions. It is a well established fact that the SAC objective is to maximize the expected return as the first term in Equation 3.4 [34, 35] suggest. With the above reward structure, the case where the expected return would be high is when the controller has taken a single prediction and reached the target. For example, if the controller has taken 10 predictions (i.e., 9 predictions led to failure and 10^{th} prediction is successful), then the expected return would not be a high expected return. As the number of predictions are increasing, the penalties gets accumulated. Consequently, it brings down the expected return value which is not a rewarding behavior. So, the only possible way to get maximum return for the agent is to use minimum number of predictions to reach the target. Since, the networks are trained to maximize the expected return, agent gets pressurized to use

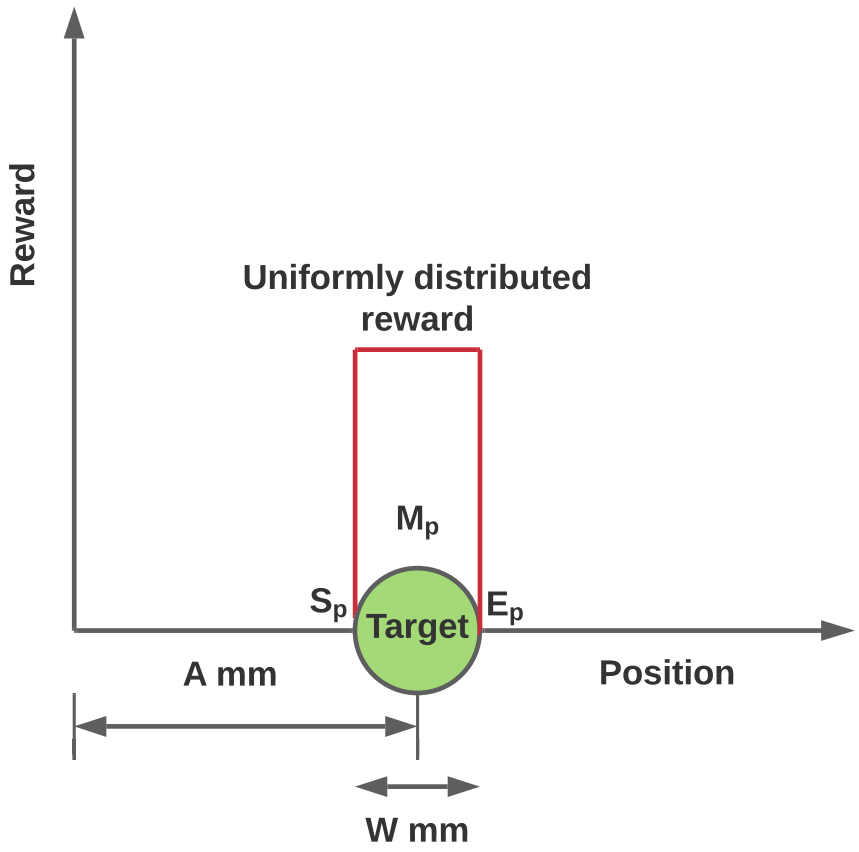

Figure 3.15: Uniformly distributed reward structure.

minimum number of predictions to complete the task.

Additionally, there is another why the above reward structure is more appropriate and reasonable in this work. Scalar reward provides us an uniform distribution across the target width. In other words, the scalar reward is invariable to the target width unlike other complicated reward structures such as euclidean distance available in literature [3, 32].

Another key point to notice is that the noise also partly responsible for SAC to determine the unique co-activation solution. Since the noise increases the likelihood of making the error or missing the target when the controller predicts low-

amplitude co-activation, the agent determined that the boosting the co-activation amplitude can avoid the penalty. As a result, the feedforward controller learns to predict the co-activations that are high in magnitude while keeping the number of predictions to the minimum.

3.5 Training and architectural details

Our architecture utilized two function approximators (neural networks), one for state-action pair approximation and the other for policy optimization parameterized by θ and ϕ, respectively. In all our experiments, the Q-network is a multi-layer perceptron with two hidden layers with Rectifier Linear Unit (ReLU) activations after the first two layers. As shown in Figure 3.10, the size of both the hidden layers was configured to 256. The observation space of the RL agent includes the current position of the controller, goal position, and current muscle activation values. The Q_θ network estimates the action-value function denoted as $Q(s,a)$. A Gaussian policy network is utilized, which outputs the statistical parameters mean μ and standard deviation σ to compute a Gaussian distribution $\mathcal{N}(\mu,\sigma)$. Then, the actions are sampled from the normal distribution using the reparameterization technique [54]. The resultant actions are communicated to the simulation environment ArtiSynth to drive the simulation forward. During the testing phase, the mode of the distribution is considered as the optimal/learned action of the agent.

3.6 Summary

This chapter explains the methodology utilized to learn and investigate ballistic controllers in an elaborate fashion. First, we discussed the motivation underlying the choice of the biomechanical task and the algorithm. Then, we formulated the research objective, experimental hypothesis and elaborated the dynamics of the muscle model. Then we explained the architecture and discussed each component's task in the architecture. Finally, we explained how the proposed SAC architecture learned the feedforward control strategy.

Chapter 4

Feedforward controller and it's Behavior

The primary goal of our DeepRL based architecture is to learn a feedforward controller that predicts the desired co-activation for a target in the kinematic space and investigate the behavior of the learned controller. So, the Soft Actor-Critic algorithm has been utilized to learn a feedforward controller with a scalar reinforcement signal with various task difficulty levels under muscle activation and controller noise configurations. The scalar reinforcement provides feedback on whether the task is successful or not. The adaptations that played a crucial role in learning the ballistic equilibrium controller are 1) Scalar reinforcement, 2) Varying the index of difficulty, 3) Injection of noise in the muscle system and controller. All these adaptions unified together led to the building of the SAC-based controller that acts in a feedforward manner.

To carry out our experimental investigations constructively, we first set up a deterministic system and then solve the reaching task using SAC. We can validate whether the SAC method can solve the deterministic system, which is a relatively more straightforward and well-studied problem, by solving the noise-free system. Additionally, solving a deterministic system can provide a measure of performance and reliability of the SAC method. Then, we move on to solving more complex scenarios where the task's difficulty is modified by varying target width, the distance between the actuator and the target, and the muscle activation noise levels using

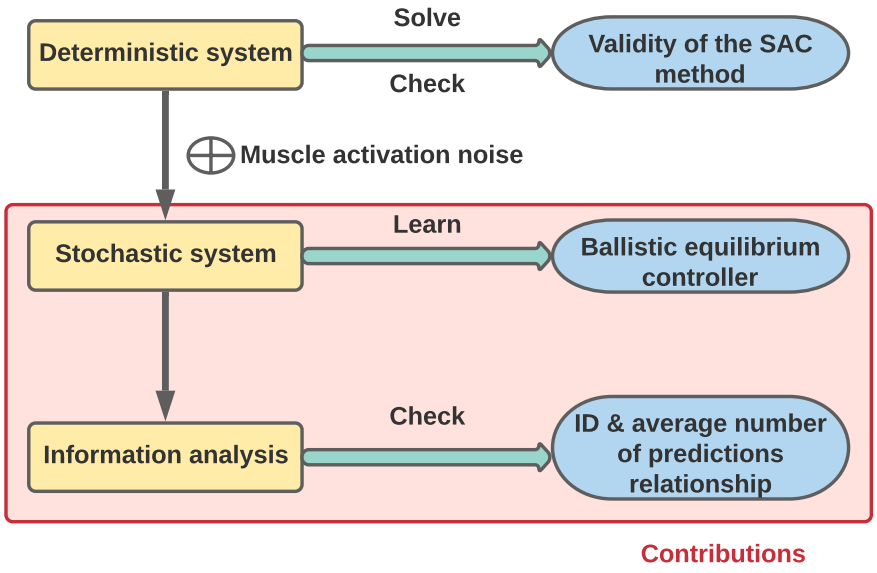

Figure 4.1: This figure represents the workflow of this chapter. First, we discuss the deterministic setting and how the SAC is used to solve the deterministic problem. Then, we move to the more complex scenario of the stochastic system to learn a ballistic or feedforward equilibrium controller. After, information analysis is carried out to evaluate the controller's performance.

SAC.

Solving a noise-free (deterministic) system is a well-studied problem [3, 32] compared to the noisy system (stochastic) case. Therefore, the main contributions of this work arise from solving a stochastic system to learn a feedforward controller and conducting the information analysis to determine whether the controller has learned to tradeoff the number of predictions for difficult tasks as indicated in the figure 4.1. We conducted information analysis (similar to Fitts' analysis) on the learned controller to determine the relationship among the Index of Difficulty, amount of noise, and the average number of controller predictions (analogous to movement time) to check for the tradeoff behavior.

4.1 Reaching task in a deterministic world

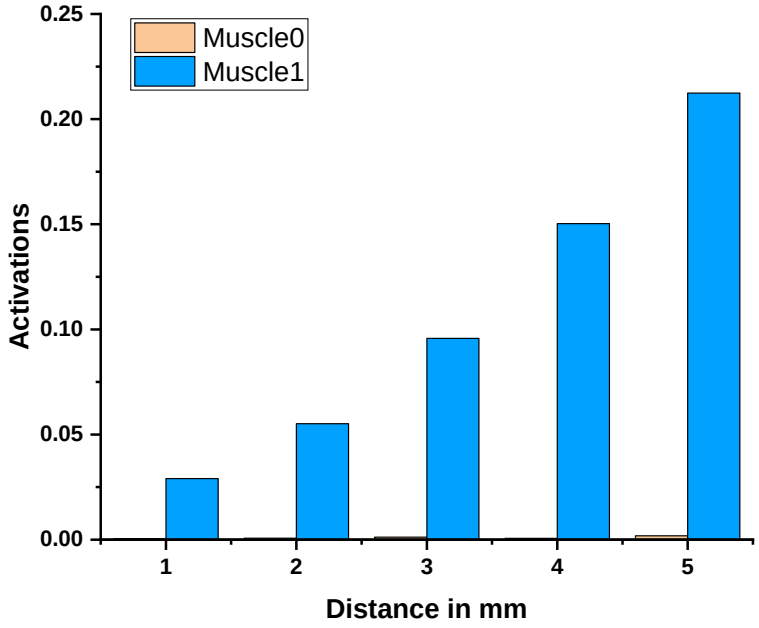

Figure 4.2: Co-activation predicted by the SAC-controller in a deterministic setting. Controller is reinforced if the controller has minimized the euclidean distance with the least amount of activations of both muscles.

Deterministic, by definition, means that the system's behavior is known. There is no randomness involved in determining the output of the system given an input. If there is noise in the system, inherently, the system becomes non-deterministic as the noise is random and the output is varied stochastically. To make our system deterministic, we remove the randomness in the system. The randomness or noise in our system is present in the muscular system (activation noise) and controller (controller noise). So, we remove the activation and controller noise making the system noise-free (deterministic). Additionally, as the system becomes deterministic, the controller must reach a target with a single ballistic prediction. Therefore, the width of the target does not factor into the difficulty of the task. Hence, the

central premise of the problem reduces to figuring out the co-activation required to make certain displacement in the kinematic space. However, there exists an infinite number of co-activation solutions to reach a particular point. So, we impose some boundary conditions on the activations to ensure the system can produce unique co-activations. The boundary condition is the activation regularization which puts pressure on the algorithm to minimize the Euclidean distance between the target and the actuator while using the least amount of activations to reach the target. The reward used for this experiment is given in Equation 4.1.

$$R = \begin{cases} -\log_{10}(d_e) - (MM^T) & T_p \leq C_p \leq T_p + w \\ -5 \end{cases} \tag{4.1}$$

Where M is the activation value predicted by the controller and d_e is the distance error calculated as Euclidean distance error, and w is the static target width used in the experiment. The above reward helps the system learn to reach the target by rewarding the agent when it reaches the target (minimizing Euclidean distance) with the least amount of excitation values (excitation regularization). First-term $-\log_{10}(d_e)$ results in high reward when the d_e comes close to zero. Additionally, the second term $-MM^T$ results in a high reward when the controller predicts the co-activation with low values. So, this imparts pressure on the algorithm to learn to reach the target by minimizing the distance-error via excitation minimization. It is important to note that $\log_{10}$ is added to d_e to normalize the d_e term. Adding log with base 10 will ensure the first term stays between 0 to -2. On the other hand, the lowest and highest excitations possible are 0 and 1, respectively, for both muscles as the muscle excitations are also normalized (0 to 1). Therefore, the maximum and minimum value of the dot product MM^T also gives 0 and 2, respectively. This allows the agent to understand that minimizing distance and using the least magnitude excitation values are equally important. Because minimizing both the distance-error and excitations will result in a maximum reward. When the agent fails to reach the target, the -5 scalar value is awarded as a penalty. In addition, the width in this experiment is kept static across all the episodes. The value for the target width used is 0.3. Using the reward mentioned above, the controller must learn to predict the activation values with the least dot-product value. This can be

achieved by predicting the least amount of activations values for both muscles.

From the graph 4.2 it is evident that the activations for the targets that are present at 1.15 *mm*, 2.15 *mm*, 3.15 *mm*, 4.15 *mm*, and 5.15 *mm* distance away to the right of the actuator are low in magnitude. Moreover, the muscle0 activation values are almost zero. Further, while calculating the dot product, the contribution of muscle0 in the reward gets even lower. This behavior demonstrates that the controller has learned to use the least amount of co-activation. Besides, the overall dot-product becomes less as the muscle1 excitation values are also less in magnitude. As a result, the agent received more reinforcement while successfully reaching the target. So, the controller is working as is expected, and we can conclude that SAC is a reliable algorithm to continue further complex investigations.

4.2 Reaching task with variable A & W in a noisy world

The previously explained deterministic experiment is a reliability test for the SAC algorithm. However, the motor system is usually stochastic in nature [33]. So, the main experiment in this work requires making the current biomechanical system stochastic. We achieve this by inducing noise in the system at various stages. Since the system is stochastic, it is a more complex and challenging problem to solve. Noise sources are present at every stage of the human motor mechanism. As it is not possible to investigate every single source of noise, we mainly concentrate on the muscle activation noise and the controller noise.

4.2.1 Addition of noise

Noise plays a significant role in the human motor system. Fitts proposed that humans can not perform quick movements while performing high-precision movements. Humans tend to make errors. Therefore, an inherent agreement between speed and accuracy is observed and commonly referred to as the speed-accuracy tradeoff. Fitts suggested that one of the responsible factors for this phenomenon is noise in the motor system. Hence, we introduced the noise into the point-to-point biomechanical system, which generates jittering behavior. Noise can be modeled in many ways as there are multiple sources available in the motor system. We mainly investigate the behavior of the controller under the actuator and controller

noise configurations of the world.

Muscle activation noise: Muscle activation noise or simply activation noise modeled in this work is the noise that is innately present in the muscular system. This noise is always present in the system. Hence, it is applied at every time step of the simulation. Due to these continuous noise perturbations, the actuator is continuously oscillating around the point of origin — the higher the amount of noise, the more the oscillations. A simple analogy to explain the effect of noise is that healthy or normal people exhibit essential or physiological tremor [19]. Similarly, the actuator in the muscle model exhibits a small amplitude oscillating behavior due to the muscle activation noise introduced in our system. This noise can potentially affect the targets demanding high accuracy, especially when the task objective involves reaching equilibrium. This noise is also present even in the absence of movements. The actuator noise is sampled from a gaussian noise distribution with zero mean (μ) and 0.03 standard deviation (σ) as shown in Figure 4.3. As the noise is a Gaussian process, during training, the agent gets more exposed to the noise activation values ranging from -0.03 to 0.03 (see the highlighted portion of 4.3). The likelihood of getting exposed to high activation values is low. In other words, during training, minimum exposure of the high activation values is still available. It is important to note that too much activation noise (increasing the standard deviation of the noise process) can damage the whole training process. Because the actuator can never reach the target and hence there would not be any reinforcement to learn. Therefore, we chose a value of 0.03 sigma value to make the task difficult enough while not making it too difficult to reach the target. This way, the agent gets exposed to the noise ranging from low to high activation values in a controlled fashion, which helps the agent understand the noise process's dynamics.

Controller Noise: The controller noise is the noise that is present within the controller predictions as there can be noise added in the motor commands [73]. Hence, when the controller predicts the muscle activations desired to reach a target in the kinematics space, we add activations sampled from a random Gaussian process with zero mean and 0.06 standard deviation as shown in Figure 4.5. Therefore, the

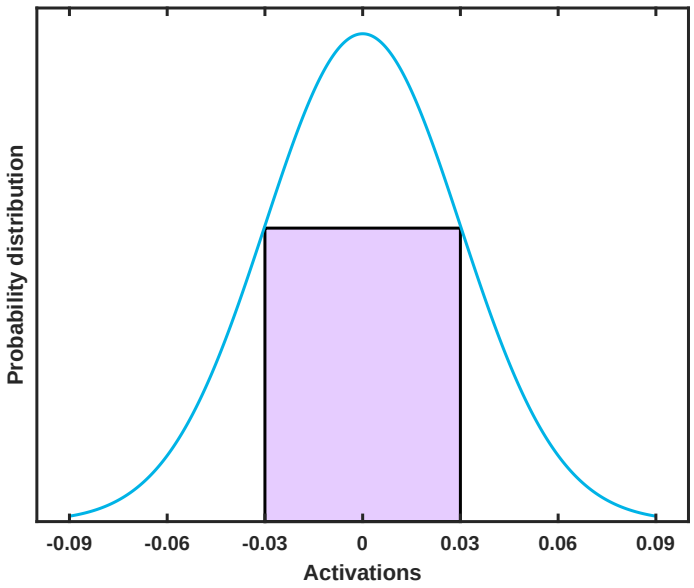

Figure 4.3: Activation noise distribution is present in the system with zero mean μ and 0.03 standard deviation σ. When the activation noise process is sampled, the probability of the resultant activations being within the range of -0.06 to 0.06 is higher.

activations that get applied to the point model controller will be different from the actual prediction of the controller as the noise activation values are added as shown in Figure 4.4. It can be observed that the standard deviation of the controller noise process is higher than that of activation noise. Because the noise activation values sampled from 0.03 standard deviation might not create required disturbances in the controller's prediction. If the noise values are low in magnitude, then the controller can easily adjust to the noise values by boosting the magnitudes of the activations, as the noise is applied only at the time of prediction as opposed to the activation noise. This would make the task a bit easier for the controller. Hence, for this noise configuration, we increased the standard deviation of the noise process to 0.08. Then, the sample activations can increase the probability of missing the target, making it considerably difficult to reach.

Controller noise affects the predictions made by the controller (neural networks) by adding the randomly sampled activation values from the Gaussian noise

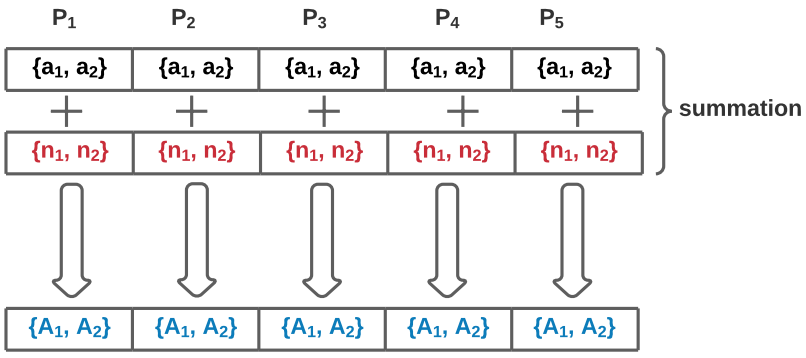

Figure 4.4: Schematic representation of the controller noise working mechanism. a_1 and a_2 represents the activations predicted by the controller for the first muscle and second muscle respectively. n_1 and n_2 represents the noise activations sampled from the gaussian noise distribution for the first muscle and the second muscle respectively. A_1 and A_2 represents the resultant muscle activations after adding the noise activations to the controller predicted activations.

distribution as shown in Figure 4.5. As a result, the actuator achieves equilibrium in a different position.

Target width as uniform noise: In addition to the controller and activation noise, target width can also be interpreted as uniform noise. The fundamental motivation for Fitts' law is the information capacity theorem known as Shannon's theorem. This theorem determines the information rate of a communication channel of Bandwidth B when a signal S is transmitted.

$$C = B \log_2 \left(\frac{S}{N} + 1 \right) \tag{4.2}$$

Fitts' rationale behind the ID formulation is that the human operator that performs a movement over a certain amplitude or distance to acquire a target of a certain width is demonstrating a "rate of information transfer" [26]. The movement amplitudes are like signals, and target tolerances or widths are like noise. In a noiseless sys-

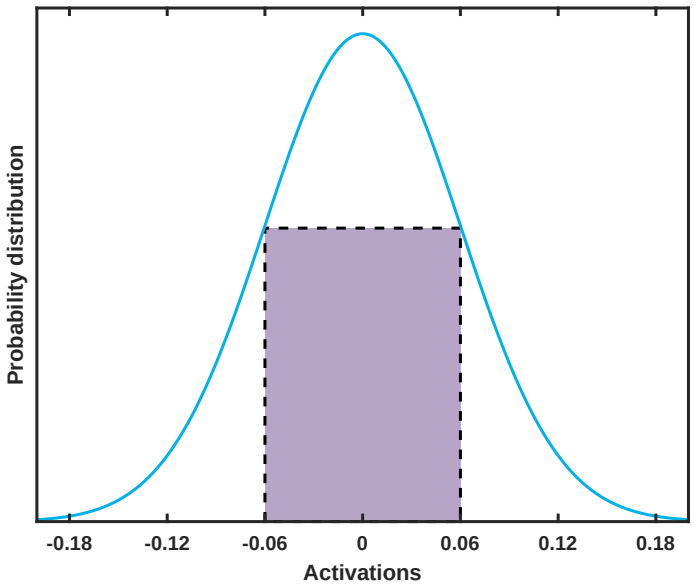

Figure 4.5: Controller noise distribution is present in the system with zero mean μ and 0.06 standard deviation σ. When the noise process is sampled, the probability of the resultant activations being within the range of -0.06 to 0.06 is higher.

tem (i.e., $N = 0$), the Signal-To-Noise (SNR) ratio will become infinity indicating a lossless transmission. Since the width is analogous to noise, in the reaching task, zero-width means that the target is a point that requires an infinite amount of precision, and the human is required to perform a movement that is exactly equal to A. Similarly, when $N \neq 0$ or non-zero width means that the target is an infinitesimally small point with kinematic noise uniformly distributed around the point. So, the human operator can make some errors as long as the end-effector is within the target width. For instance, if the target is at 4 *mm* with a target width of 1 *mm*, the human operator can make an amplitude of 3.5 *mm* and still be considered successful in reaching the target.

$$MT = a + b\ ID = a + b \log_2 \left(\frac{2A}{W} \right) \tag{4.3}$$

It is important to note that Fitts considered the target width modeled as a uni-

form noise distribution, and it is present in the task space. However, there can be multiple sources of noise present inherently in the human motor system. Mackenzie et al. [59] suggested that Fitts' ID can be extended to any source of noise to accommodate the effect of other types of noise in the ID formulation.

4.3 Validation of feedforward controller

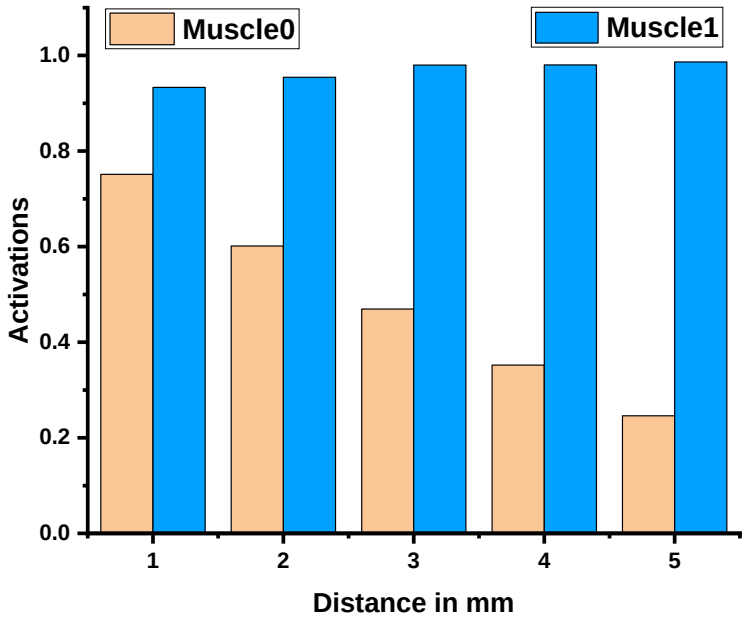

Figure 4.6: Co-activation predicted by the SAC-controller in a stochastic setting with ***activation noise***. Controller is reinforced only on reaching the target.

As the motor control theory suggests, humans first learn to pre-program motor commands, and once they learn, they generate motor commands in an open-loop (feedforward) mode to perform motor tasks [48]. These feedforward commands are enough to complete motor tasks. However, when there is a high need for accuracy, the feedforward is not enough. Based on feedback, corrective movements are

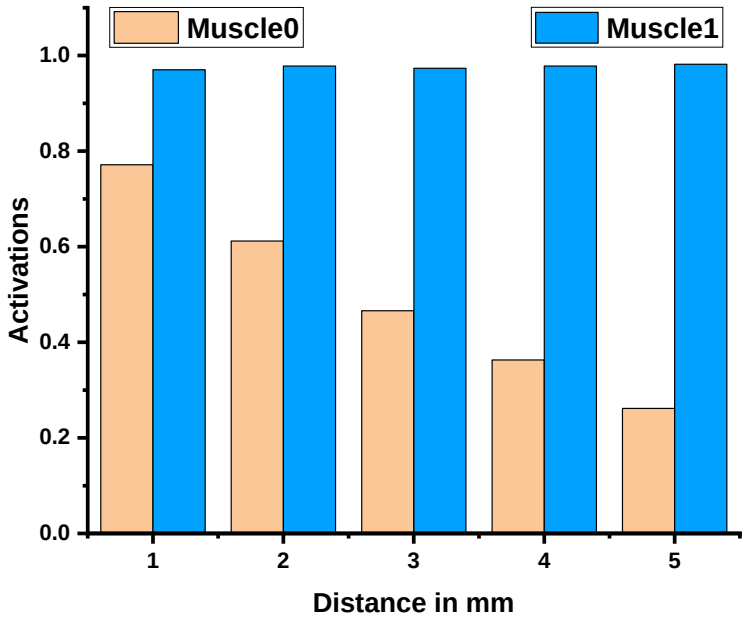

Figure 4.7: Co-activation predicted by the SAC-controller in a stochastic setting with ***controller noise***. Controller is reinforced only on reaching the target.

generated to complete motor tasks. Similarly, in our work, we built a SAC-based controller that learned to prepare co-activation that brings the actuator onto the target in a feedforward manner. We found that the controller has learned to reach the target with a single prediction when there is less emphasis on accuracy (i.e., bigger targets). However, in the case of high-accuracy targets (smaller targets), the controller has invested multiple predictions. The number of predictions is dependent on the amount of noise present in the system. Additionally, as shown in graphs 4.6 and 4.7, the controller's predicted co-activation are high in magnitude for both the muscles due to the noise present in the system. The controller has learned to predict the activations as a way to counteract the noise activations. The noise in the system can increase the likelihood of missing the target, which leads to a penalty. To avoid being penalized, the controller boosted up the activation values so that

even if the noise exists, it can overcome its effect, thereby making sure to reach the target or receive reinforcement. One important point to note here is that the activation values reported in graphs 4.6 and 4.7 are for targets present at distances 1 *mm*, 2 *mm*, 3 *mm*, 4 *mm*, 5 *mm* with 1 *mm* target width. Further, the controller has taken more than one prediction for some of the targets as the difficulty of the task increases as distance increases. So, we took the average of all the predictions and reported them in the graphs.

4.4 Behavior analysis of the feedforward controller

In this section, we explain the behavior of the learned feedforward controller by using information analysis. Information analysis deals with characterizing or evaluating the controller's performance given various difficulty levels of the task under the activation noise and controller noise configurations. It can help us understand what the SAC-based controller has learned and its behavior to reach the target.

4.4.1 Information analysis

Information theory has been an effective tool to appraise human movements [27]. This approach takes the perspective of humans as information processors [81]. Inspiring from the information theory, Fitts extended the information theory concepts to measure the human motor system [58]. He formulated a metric to quantify the difficulty of the task (in terms of bits) and then used the ID to quantify the human motor performance via "throughput" or "information rate" [27]. Since Fitts' study, many works followed the same pathway to appraise the human motor performance, and it has also been applied to evaluate various input devices in the human-computer interaction field [65]. We also follow the path of Fitts and conduct the information analysis to investigate the performance of our learned feedforward or ballistic controller.

One major exception is that Fitts' ID has considered only uniform noise in the form of target width, whereas we also consider other sources of noise in our analysis, i.e., muscle activation and controller noise. It means, along with the distance to the target A, target width W, an additional parameter exists in the form of muscle activation and controller noise, which can modify the task's ID.

Index of Difficulty

According to Fitts, various target configurations (combinations of A & W) have different IDs. From Fitts' law, it is understood that multiple configurations of the target are possible depending on the distance and the target width. Each configuration has a specific ID value assigned to it. As the difficulty of the task increased, the movement time has increased linearly. Due to the noise in the system, it is not viable to calculate the movement time in our work. In the case of actuator noise, the actuator may never reach equilibrium if the amount of noise is significantly high. However, we can utilize the "number of controller predictions" instead of movement time. Therefore, we analyze the controller's behavior by determining the relationship between the ID and the "number of controller predictions."

Average number of controller predictions

Here, the average number of controller predictions is calculated as an alternative measure of movement time instead of calculating movement time. The reasons behind this are explained as follows: First, our controller, for every prediction, waits for the actuator to get to equilibrium. When the muscle activation noise is present in the system, the actuator might struggle to reach an equilibrium point. If the amount of noise is significantly higher and the target width is small and far, the actuator may never reach an equilibrium point, and it would take an infinite amount of time to reach equilibrium. So, in this case, it is not possible to have a proper measure for movement time. So, to avoid this, a time-out mechanism is employed to notify the controller to send out a new prediction if the actuator never reached the equilibrium after a fixed amount of time. Therefore, for convenience, we use the number of times the controller sends the muscle activations predictions to reach the target as an alternative measure for movement time. Moreover, the metric number of controller predictions is proportional to the movement time. If the number of controller predictions is high, it can be interpreted as more movement time.

Target Configurations & Representations:

During training, the agent is presented with various target configurations. The target configuration is characterized by the distance between the target and the

Table 4.1: Data points for 4 mm distance. Each target has the width property and is represented by the starting point S_p, center point C_p, and End point E_p distances.

Distance	**Width**	**Target Representation**
(d mm)	(w mm)	($< S_p - M_p - E_p >$)
4	1	$< 3.50 - 4 - 4.50 >$
	0.9	$< 3.55 - 4 - 4.45 >$
	0.8	$< 3.60 - 4 - 4.40 >$
	0.7	$< 3.65 - 4 - 4.35 >$
	0.6	$< 3.70 - 4 - 4.30 >$
	0.5	$< 3.75 - 4 - 4.25 >$
	0.4	$< 3.80 - 4 - 4.20 >$
	0.3	$< 3.85 - 4 - 4.15 >$
	0.2	$< 3.90 - 4 - 4.10 >$

actuator D and the required accuracy (target width) W. In the training phase, the distance to the target is randomly selected from 0 *mm* to 7 *mm*. Similarly, the required accuracy to be considered successful in reaching task is modified, thereby generating various target configurations. For instance, for a target located at 4 *mm* with a target width of 0.5 *mm*, the ID value according to Fitts' formulation is 4 bits. As the D and W change, the ID value changes. Table 4.2 shows the ID values of various target configurations. ID in the table is calculated according to the Fitts' formulation of ID, which is given as $\log_2(\frac{2D}{W})$.

Each target is represented by the start point S_p, middle point or center point M_p, and the end point E_p. Depending on the target width, the S_p, M_p, and E_p values change. Examples of target representations are shown in table 4.1 for distance 4 *mm*.

Data Acquisition:

Information analysis of the controller is conducted by collecting observations of various target configurations. We selected the target configurations that have non-duplicate ID values out of all the possible target configurations. The selected data encompasses the least ID, highest ID, and moderate ID values. For each target

Table 4.2: Selected data points that encompasses all the ID values ranging from 1 to 6 bits.

Distance	**Width**	**Fitts' ID**
D (mm)	*W (mm)*	*ID (bits)*
1.0	1.0	1.00
1.0	0.6	1.73
1.0	0.4	2.32
2.0	1.0	2.00
2.0	0.6	2.73
2.0	0.5	3.00
3.0	0.6	3.32
3.0	0.2	4.90
4.0	0.5	4.00
4.0	0.3	4.73
4.0	0.2	5.32
5.0	0.8	3.64
5.0	0.5	4.32
5.0	0.3	5.05
5.0	0.2	5.64

configuration shown in table 4.2, we collected the number of predictions n made by the controller to reach the target successfully. Hence, owing to the uncertain nature of random noise introduced in the system, the number of predictions listed in the table is calculated as the average number of predictions taken over 100 repetitions of the same task.

4.4.2 Feedforward equilibrium controller with activation noise in the world

In this experiment, there is no constraint imposed on the co-activation predicted by the controller. Hence, the agent has the freedom to choose activations whose magnitude ranges from low (0) to high (1). The main objective of this experiment is to study the effect of the actuator noise on the performance of the learned ballistic equilibrium controller. Due to actuator noise, reaching equilibrium ($A_v \approx 0$) becomes difficult as the actuator continuously oscillates. More the actuator noise,

the more difficult it is to reach equilibrium. Hence, the resting condition is kept to a small value ($A_v < 0.1$) instead of 0. Another important setting of this experiment is that there are no constraints imposed on the activation levels. The controller can predict any activation values ranging from a minimum value of 0 to a maximum value of 1. Also, the agent is trained with 0 mean (μ) and 0.03 standard deviation (σ). However, during the testing, we collected the observations by varying the noise levels, which will be discussed later in detail.

During training, the agent encountered various targets at different positions (distances) and varying target widths. This feature allows the agent to learn to reach targets with various target widths and come to a resting position while learning to counter the noise.

Linear fitting:

One of Fitts' study observations is that MT has a linear relationship with the logarithmic ratio of A and W. Now, we also performed scatter plot analysis for the data we have collected, which is shown in Figure 4.8

From the graph 4.8, it is evident that as the ID increased, the number of controller predictions also increased, which concurs with the Fitts' observation. But the main question here is: is it linear? From the data it is more like an exponential instead of linear. To confirm this, we performed a linear fitting and the corresponding analysis is shown in graph 4.9. As a result, the resulted linear equation has only 0.73 of R^2 value and Pearson's correlation coefficient (R) of 85%. This means that the relationship is non-linear which deviates from the Fitts' observation.

Exponential relationship of ID & n:

We performed exponential regression analysis and found that the exponentially growing curve (resultant regression curve) best describes the collected data. The resulting regression equation has a general exponential form as shown in equation 4.6 This indicates that the agent is making more predictions to reach the target as the targets are becoming more difficult. Moreover, the agent's predictions are exponentially increasing as the ID is increasing, which is evident in graph 4.10.

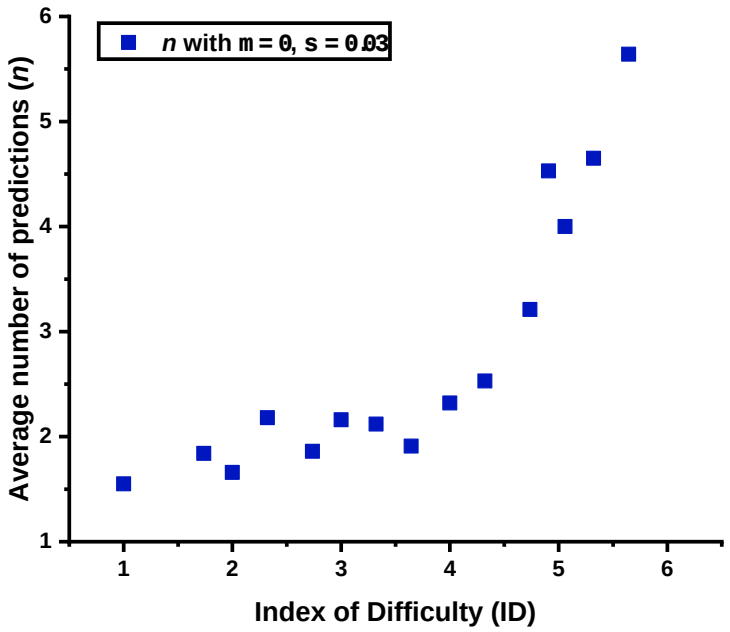

Figure 4.8: This graph represents the scatter plot performed between the number of controller predictions and the ID with actuator noise present in the system. In this case, actuator noise distribution is set to have zero mean μ and 0.03 standard deviation σ.

$$n = a\, e^{\frac{ID}{b}} + n_0 \tag{4.4}$$

Also, we varied the amount of actuator noise levels and collected the corresponding observation data to check how noise affects the number of controller predictions. Interestingly, as the amount of noise increases, the number of controller predictions also increases and vice versa. First, we reduced the amount of noise, i.e., the standard deviation of the actuator noise process is reduced from 0.03 to 0.01, and the corresponding data is shown in graph 4.11. The graph shows that the average number of predictions was comparatively less than the 0.03 standard deviation (σ) data. Moreover, it can be understood that the slope of 0.01 σ case is

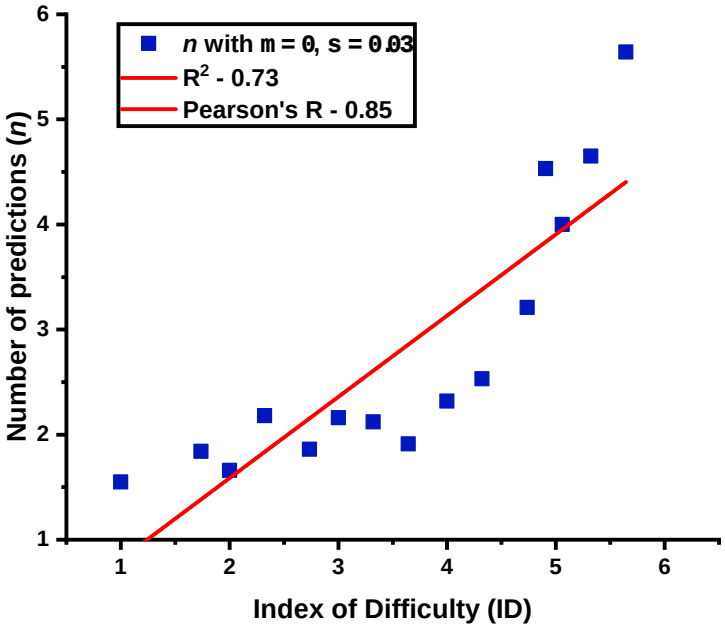

Figure 4.9: This figure represents the relation between the number of controller predictions and the ID under actuator noise configuration. In this case, actuator noise distribution is set to have zero mean μ and 0.03 standard deviation σ. Linear regression is performed to determine the relationship.

much less than that of 0.03 σ. In addition, it is evident that there is a breaking-point at ID = 4, which is when the number of controller predictions starting increasing exponentially. This observation also holds in the case of 0.03 σ.

Now, we increased the amount of noise level by increasing the σ value from 0.03 to 0.05. We should observe that the number of controller predictions should increase compared to that 0.01 σ and 0.03 σ. In other words, the slope of the ID-n curve should be high when the amount of noise is high. The data with 0.05 σ shows that it matches our hypothesis.

Another observation is that the slope of the ID-n curve is also exponential since the derivative of the exponential is also exponential. However, the inverse of the

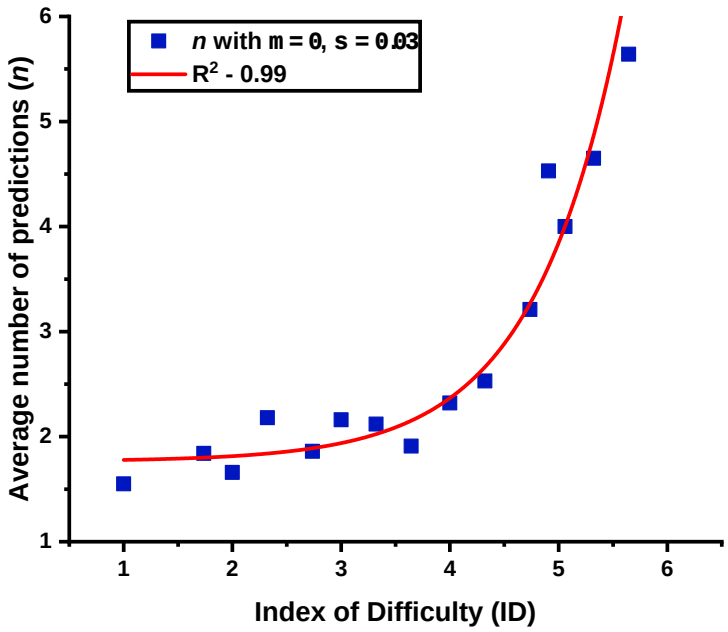

Figure 4.10: This figure represents the relation between the number of controller predictions and the ID under actuator noise configuration. In this case, actuator noise distribution is set to have zero mean μ and 0.03 standard deviation σ. Exponential regression is performed to determine the relationship.

slope is not to be treated as throughput as the number of controller predictions is dependant on two variables.

Discussion: The main observation from the data is that the relationship between the number of controller predictions, an equivalent metric for movement time, and the index of difficulty is exponential. Fitts found it to be linear. However, Fitts' analysis has only two main dimensions affecting the movement of a human operator, i.e., distance to the target and the target width modeled as a uniform noise. In comparison, we have an extra dimension that affects the agent's movements, i.e., muscle activation noise, more specifically actuator noise, in this experiment. In

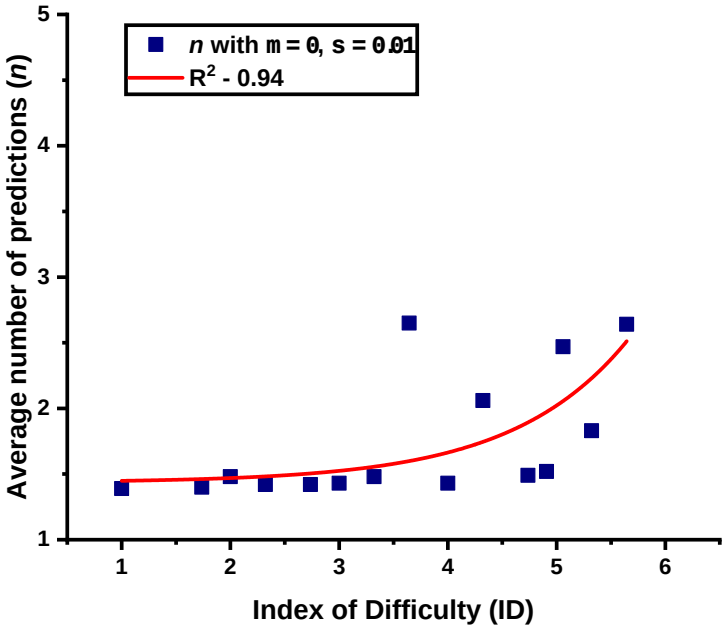

Figure 4.11: This figure represents the relation between the number of controller predictions and the ID under actuator noise configuration. In this case, actuator noise distribution is set to have zero mean μ and 0.01 standard deviation σ. Exponential regression is performed to determine the relationship.

other words, the difficulty of the target is a function of distance to the target, target width, and the amount of noise. For example, if a target is near and bigger, but the amount of noise is significantly higher than the actuator may never reach the target and equilibrium, then the task is indeed difficult. For a task to be easier, the target must be closer, bigger, and have lower noise levels. Similarly, for a task to be difficult, the target should be farther, smaller, and have higher noise levels. In addition, the data shows that there is a breaking point at ID equal to 4 bits. For the targets with an ID greater than 4 bits, the number of controller predictions started exponentially increasing with the ID. Also, as the noise level increased, the slope of the exponential increment increased. Now, if we look at the target configurations

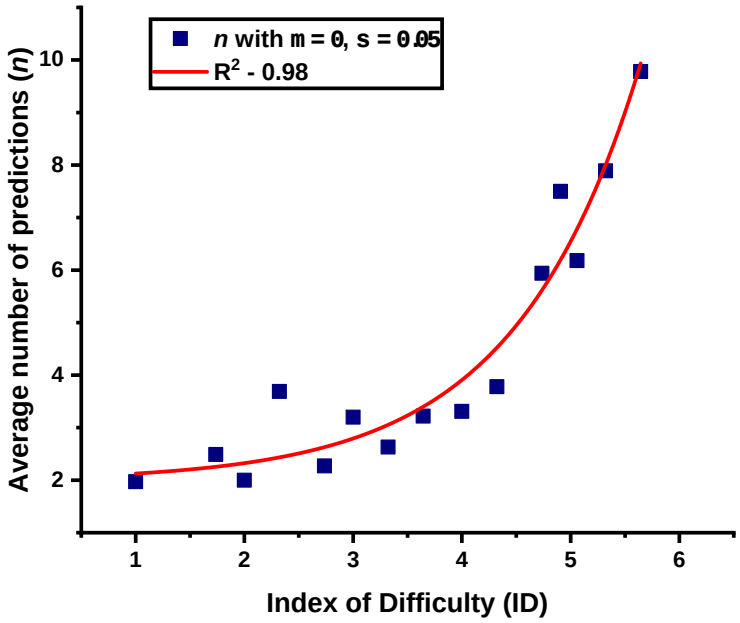

Figure 4.12: This figure illustrates the relation between the number of controller predictions and the ID under actuator noise configuration. In this case, actuator noise distribution is set to have zero mean μ and 0.05 standard deviation σ. Exponential regression is performed to determine the relationship.

with ID value more than 4 bits given in table 4.2, all those target configurations have one aspect in common. All the targets are farther and smaller. It means that when the target width is smaller, and there is actuator noise present, the controller is investing more predictions as it is not easy to reach. This eventually leads to an outlier case where the controller can never reach the target meaning that the controller would take an infinite number of predictions. So, we can understand that the resulted ID-n curve is an exponentially growing curve tending to infinity. That is why the ID and the number of controller predictions have an exponential relation rather than a linear relation.

Table 4.3: Coefficients of the exponential regression equations with varying actuator noise levels.

Noise level	Coefficient 1	Coefficient 2	Base value
σ	a	b	n_0
0.01	0.0053 ± 0.02	1.06 ± 0.91	1.43 ± 0.23
0.03	0.0054 ± 0.004	0.80 ± 0.16	1.76 ± 0.10
0.05	0.06 ± 0.08	1.15 ± 0.33	1.98 ± 0.40

Linear relationship of σ & n:

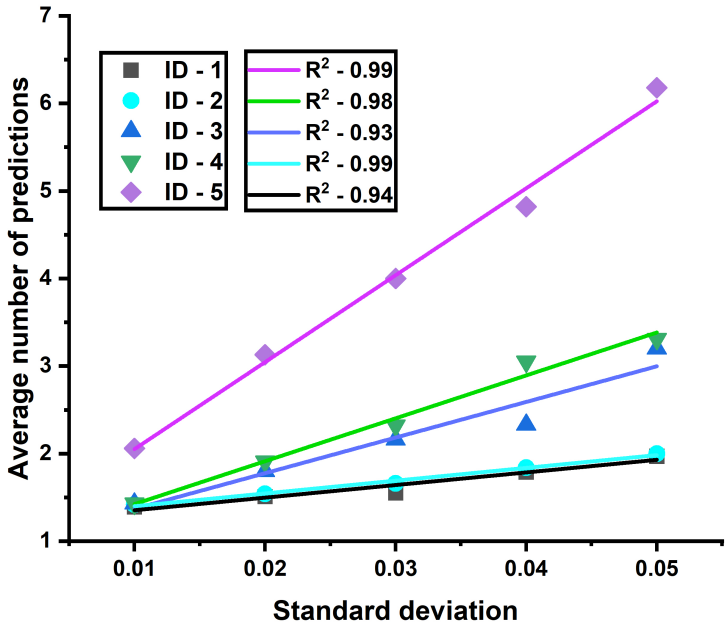

Figure 4.13: This graph indicates the trend of average number of controller predictions with the variation of actuator noise levels with ID kept constant. The actuator noise levels are represented by the standard deviation of the actuator noise process.

Now we determine the relationship between the average number of controller

Table 4.4: Slope and intercept of the linear regression equations shown in the graph 4.13 for each ID.

Index of Difficulty	**Slope**	**Intercept**
ID	*a*	*b*
1.0	14.40 ± 1.95	1.21 ± 0.06
2.0	14.60 ± 0.75	1.25 ± 0.02
3.0	40.70 ± 6.15	0.96 ± 0.20
4.0	49.00 ± 3.50	0.93 ± 0.11
5.0	99.30 ± 5.08	1.05 ± 0.16

predictions and the amount of actuator noise factor σ. We plotted the average number of controller predictions and varying the amount of noise by keeping the ID fixed. We found that the average number of predictions is linearly related to the amount of noise for a specific ID target, as shown in the equation. Where n is the number of controller predictions, a & b are the slope & intercept, respectively. σ is the standard deviation of the noise process. From the graph, it can be observed that there is a trend in the slope; as the amount of noise increased, the average number of controller predictions increased. Also, the slope of the equations is high in high ID targets, which indicates the n value will become high.

$$n = a\,\sigma + b \tag{4.5}$$

Discussion: From the graph 4.13 it is evident that the slope of the $ID-5$ curve is higher than the $ID-4$ despite the fact that the ID difference is only 1 bit. However, having the same ID difference of 1 bit, the difference between the slope $ID-3$ and $ID-4$ curves is not high as observed in $ID-4$ & $ID-5$. Similarly, the same can be applied to the curves $ID-1$ & $ID-2$. According to Fitts, there should not be a big difference in slope as there is only a 1-bit difference in the ID. The main reason behind this can be understood by looking at the target configurations with the index of difficulties 1, 2, 3, 4, 5 in the table 4.2. The targets used for $ID-4$ & $ID-5$ are medium and small, i.e., the target widths are 0.5 *mm* 0.3 *mm*, respectively. Similarly, the targets used for $ID-2$ & $ID-3$ are big and medium, i.e., the target

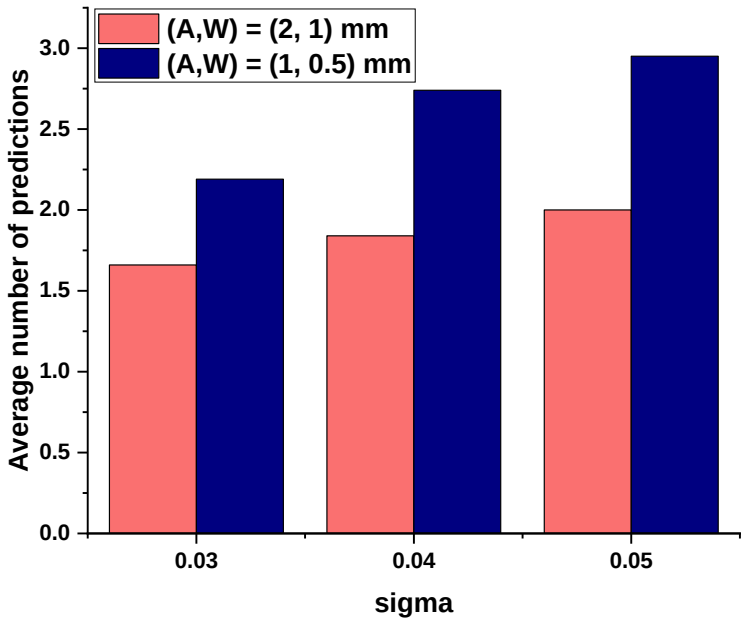

Figure 4.14: This graph illustrates the comparison of average number of controller predictions for the same ID with different activation noise levels. The noise levels are represented as the sigma or standard deviation of the actuator noise process.

widths are 1 *mm* and 0.5 *mm*, respectively. Hence, the high difference in the slopes of $ID-4$ & $ID-5$ and $ID-2$ & $ID-3$ curves is observed. It can be interpreted that the effect of width is more than the effect of distance with activation noise in the system. It means that our observation data complies with Welford's observation that the distance and the target width have different effects on the movements [94].

To investigate this observation further, we collected the average number of predictions invested by the controller for the target configurations with the same ID. According to Fitts, a target located at 2 *mm* with a width of 1 *mm* has the same ID as the target located at 1 *mm* with 0.5 *mm* width. However, we observed that the controller invested more predictions in the case of 0.5 *mm* target located at 1 *mm* when compared to 1 *mm* target located at 2 *mm* as shown in graph 4.14. As the

noise increased, the $(1, 0.5)$ target seemed difficult to reach compared to $(2, 1)$ target. It demonstrates that the target width and the distance have different effects on the controller, and target width is more influential than the distance.

4.4.3 Feedforward equilibrium controller with controller noise in the world

The main objective of this experiment is to study the effect of the controller noise on the performance of the learned ballistic equilibrium controller. Due to controller noise, reaching equilibrium ($A_v \approx 0$) becomes difficult as the controller continuously oscillates. More the controller noise, the more difficult it is to reach equilibrium. Hence, the resting condition is kept to a small value ($A_v < 0.01$) instead of 0. Another important setting of this experiment is that there are no constraints imposed on the activation levels. The controller can predict any activation values ranging from a minimum value of 0 to a maximum value of 1. Also, the agent is trained with 0 mean (μ) and 0.06 standard deviation (σ).

During training, the agent encountered various targets at different positions (distances) and varying target widths. So, it allows the agent to learn to reach targets with various target widths and come to a resting position while learning to counter the noise.

Linear fitting:

One of the observations from Fitts' study is that MT has a linear relationship with the logarithmic ratio of A and W. Now, we also performed scatter plot analysis for the data we have collected, which is shown in Figure 4.15

The graph shows that as the ID increased, the number of controller predictions also increased, which concurs with the Fitts' observation. But, is it linearly increasing? From the data it is more like an exponential instead of linear. Hence, we performed a linear fitting and the corresponding analysis is shown in graph 4.16. Consequently, the resulted regression line has 0.85 of R^2 value and Pearson's correlation coefficient (R) of 92%. In general, anything more than 0.9 of R^2 value is considered that the regression equation represents the data better. Therefore, the relationship can be considered as non-linear which deviates from the Fitts' obser-

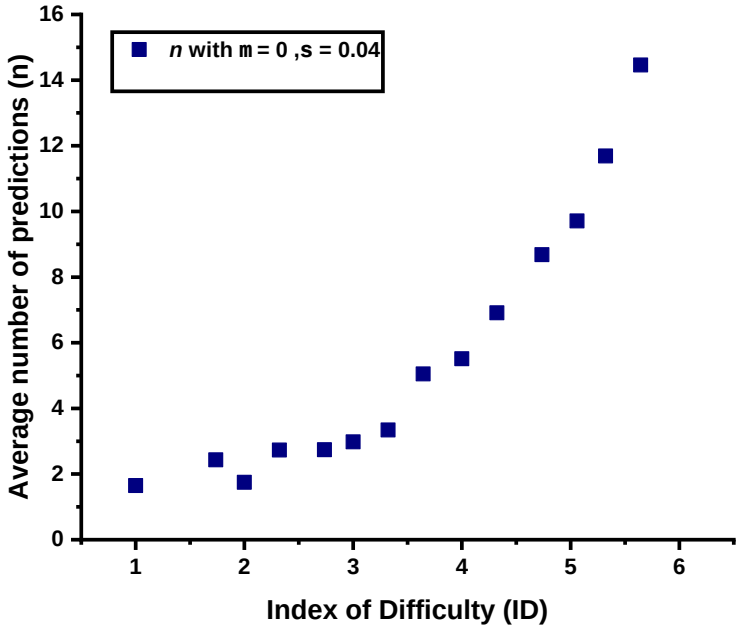

Figure 4.15: This graph highlights how the number of controller predictions vary with ID. In this case, controller noise distribution is set to have zero mean μ and 0.04 standard deviation σ. Linear fitting of the observation data. ID vs. number of predictions made by the controller.

vation. Though the controller noise data resulted in better linear fitting than the actuator noise, the R^2 value of 0.85 is still not a strong value to suggest a linear relation between ID and n.

Exponential relationship of ID & n:

Like in the case of activation noise experiment, exponential regression analysis is performed and found that the resultant exponential regression curve best describes the collected data. The resultant regression equation has a general exponential form as shown in equation 4.6 This indicates that the agent is making more predictions to reach the target as the targets are becoming more difficult. Moreover, the agent's predictions are exponentially increasing as the ID is increasing, which is evident in

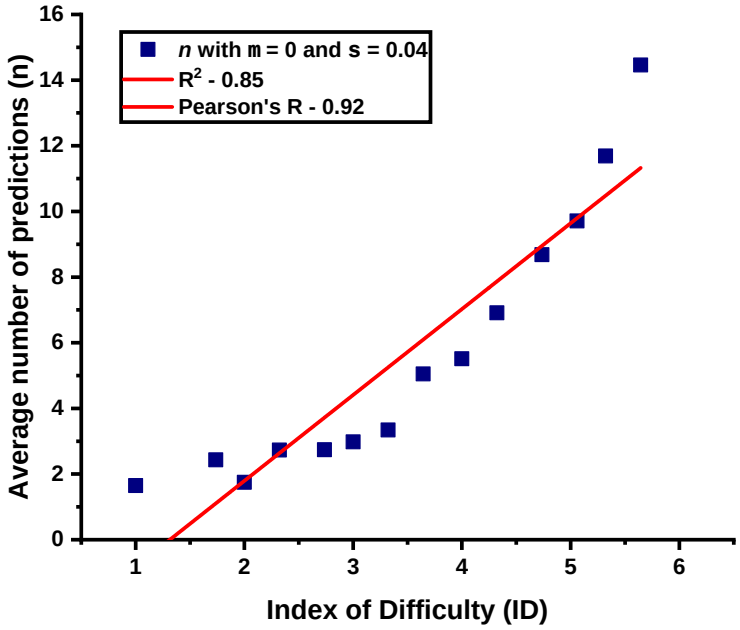

Figure 4.16: This graph shows the variation of controller predictions with the variation of ID under the controller noise configuration. In this case, controller noise distribution is set to have zero mean μ and 0.04 standard deviation σ. Linear fitting is performed on the observation data.

graphs 4.17, 4.18, 4.19.

$$n = a\,e^{\frac{ID}{b}} + n_0 \tag{4.6}$$

Also, we varied the amount of controller noise levels and collected the corresponding observation data to check how noise affects the number of controller predictions. Interestingly, as the amount of noise increases, the number of controller predictions also increases and vice versa. First, we reduced the amount of noise, i.e., the standard deviation of the controller noise process is reduced from 0.06 to 0.05, and the corresponding data is shown in graph 4.17. The graph shows that the average number of predictions was comparatively less than the 0.06 stan-

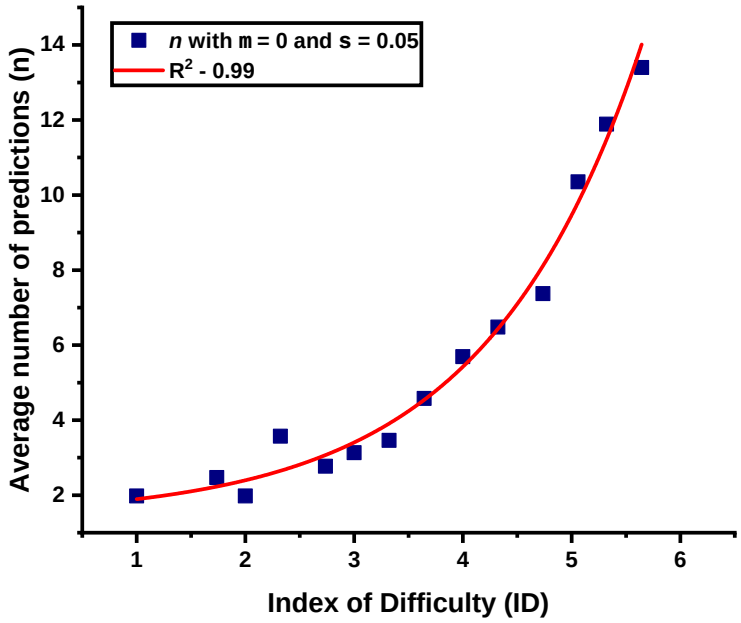

Figure 4.17: This graph demonstrates the exponential trend of controller predictions with the variation of ID under the controller noise configuration. In this case, controller noise distribution is set to have zero mean μ and 0.05 standard deviation σ. Exponential curve fitting is performed on the observation data. Relation between ID and the number of predictions made by the controller is determined.

dard deviation (σ) data. Moreover, it can be understood that the slope of 0.01 σ case is much less than that of 0.03 σ. As observed in the activation noise case, there is a breaking-point at ID = 4 when the number of controller predictions increases exponentially. It holds true in the case of 0.06 σ.

Now, we increased the amount of noise level by increasing the σ value from 0.06 to 0.08. We should observe that the number of controller predictions should increase compared to that 0.05 σ and 0.06 σ. In other words, the slope of the ID-n curve should be high when the amount of noise is high. The data with 0.08 σ shows that it matches our hypothesis.

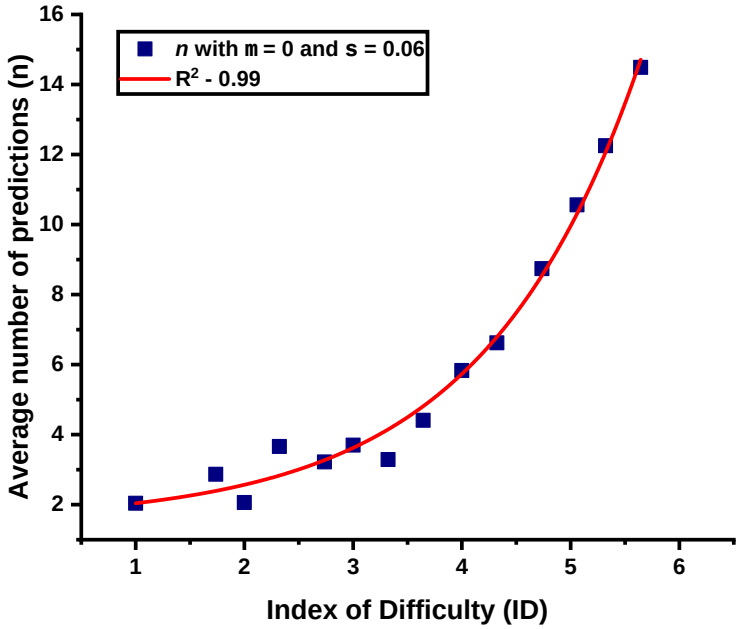

Figure 4.18: This graph demonstrates the exponential trend of controller predictions with the variation of ID under the controller noise configuration. In this case, controller noise distribution is set to have zero mean μ and 0.06 standard deviation σ. Exponential curve fitting is performed on the observation data. Relation between ID and the number of predictions made by the controller is determined.

Another observation is that the slope of the ID-n curve is also exponential since the derivative of the exponential is also exponential. However, it is important to note that the inverse of the slope is not to be treated as throughput as the number of controller predictions is dependant on two variables.

Linear relationship of σ & n:

Similar to the analysis we conducted in the activation noise case, we investigate how the number of controller predictions is affected by the controller noise levels in the system. So, we plotted the average number of controller predictions against

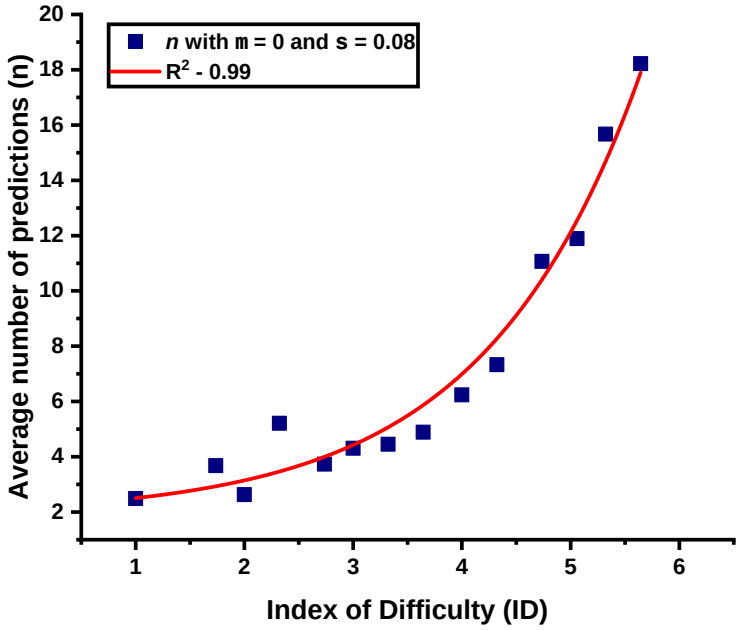

Figure 4.19: This graph demonstrates the exponential trend of controller predictions with the variation of ID under the controller noise configuration. In this case, controller noise distribution is set to have zero mean μ and 0.08 standard deviation σ. Exponential curve fitting is performed on the observation data. Relation between ID and the number of predictions made by the controller is determined.

various noise levels by keeping the ID fixed. Graphs for the ID values 1, 2, 3, 4, 5 bits are shown in graph 4.20. It can be observed that there is a linear trend in each ID graph. Also, the number of predictions increased as the ID and noise levels are increased. However, the increase in the number of predictions with the noise levels is not more like we observed in the activation noise case. This is because the impact of activation noise is relatively stronger on the number of predictions. Therefore, it is difficult for the agent to reach equilibrium in the case of activation noise. So, as the activation noise level increased, it resulted in a high increase in the predictions. Whereas in the controller noise case, achieving equilibrium is not

Table 4.5: Coefficients of the exponential regression equations with varying controller noise levels.

Noise level	**Coefficient 1**	**Coefficient 2**	**Base value**
σ	a	b	n_0
0.05	0.25 ± 0.15	1.44 ± 0.24	1.39 ± 0.42
0.06	0.26 ± 0.16	1.44 ± 0.25	1.51 ± 0.47
0.08	0.31 ± 0.25	1.44 ± 0.32	1.86 ± 0.70

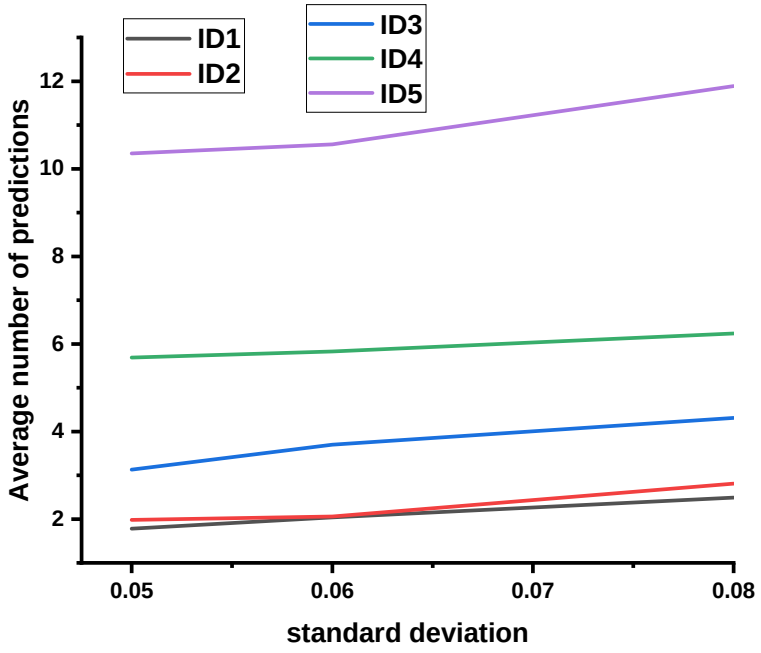

Figure 4.20: This graph indicates the relationship between the average number of controller predictions and the amount of controller noise. The controller noise levels are represented by the standard deviation of the controller noise process. ID1 represents 1 bit of index of difficulty, ID2 represents 2 bits of index of difficulty, and so on.

as complex as in the activation noise case. In other words, it is easy to negate the effect of controller noise compared to activation noise. Hence, the increase in the

average number of predictions is not high as in the activation noise case.

4.5 Summary

This chapter showed how the system had learned a feedforward control strategy while A, W, and noise parameters were varied in the experiment. Within that, we performed a sanity check with the deterministic system setting, i.e., checking the SAC performance on the noise-free system. Then, we deployed the SAC algorithm on the noisy system where there is activation and controller noise and dynamic target width. After, we tested some edge cases in the noisy world to help us understand that the system is doing what it is supposed to do intuitively. These test cases were also used to describe the effect of factors such as the number of predictions, ID, and the amount of noise (both activation and controller) on the learned controller. We demonstrated that there exists an exponential relation between the index of difficulty and the number of controller predictions with noise levels kept static. Besides, we showed that there exists a linear relation between the amount of noise and the number of controller predictions for targets with various IDs. This observation is found to be true for both the noise configurations.

Chapter 5

Conclusions & Future Directions

5.1 Conclusions

The main conclusions that we take from this study are that the relationship between the ID and the number of controller predictions (n) is exponential due to the extra parameter muscle activation noise and noise-n relationship is linear. Further, we conclude that the noise in the system plays a vital role in motor control as it contributes towards the difficulty of the task, thereby affecting the number of controller predictions. Finally, our study reveals that the effect of width is more dominant than the distance, which coincides with the findings of various previous studies [51, 60, 82, 96]. In summary, all these findings raised some fundamental research questions and opened new research pathways.

5.2 Summary of Contributions

As a first step, we solved a crucial aspect of human motor control, i.e., learning a feedforward controller in a noisy world. This learning is achieved by deploying a DeepRL algorithm known as Soft Actor-Critic onto a muscle-based reaching motor task. Then, a custom reward function based on a terminal scalar reinforcement signal is employed, which helped the algorithm to learn a feedforward control strategy. In a noise-free world, the controller has learned to reach a target with a minimum number of controller predictions. On the other hand, when the controller

is presented with varying task difficulty and noise levels in a noisy world, the system has learned to boost the magnitude levels of the co-activation. We observed an increase in the number of controller predictions in the case of small and far targets for a particular noise configuration. Similarly, when the noise level is increased, the number of controller predictions is also found to be increased for various ID tasks. Further, we conducted information analysis to describe the nature of the relationship between the number of predictions, the Index of difficulty, and the amount of noise (activation and controller). From our analysis, we found that the number of controller predictions exponentially increased as the difficulty of the task increased. Additionally, we observed that, given a task with various IDs, the number of controller predictions increased linearly with the increase in both activation and controller noise. Besides, we found that the effect of width is more dominant than the distance, which explains the non-uniform change in the controller predictions when the noise and ID change.

5.3 Limitations

Our current learning architecture has few limitations. First, we discuss the limitations of our architecture from the perspective of the motor control theory and then talk about the drawbacks in the context of the reinforcement algorithm.

- **Linear muscles:** As described in chapter 3, we use muscles that have a linear relationship between length, tension, and damping coefficient. Hence, the results and the analysis are conducted using linear muscles. However, using non-linear muscle models such as Hill and Blemker can introduce more complexities and constraints in the actuator movements, thereby influencing the controller's performance to some extent and provide biologically accurate observations.

- **Difference in the testing and training tasks:** The noise process used in the training process has fixed statistical parameters. For example, in the training of activation noise experiment, the mean (μ) and standard deviation (σ) of the noise process are set to 0 and 0.03, respectively. However, during testing, the observation data was also collected for the cases

$(\mu, \sigma) = \{(0, 0.01),\ (0, 0.05)\}$ to capture the noise effect on the controller's prediction. However, during the training with $(0, 0.03)$ noise parameters, the agent gets more exposed to noise values that are within the range of -0.03 to 0.03. When tested with $(0, 0.05)$, the agent might not have been exposed enough to adapt to the higher noise cases. As a result, the testing becomes a slightly unfair.

- **Sample efficiency of model-free RL algorithms:** All the model-free RL methods suffer from the sample inefficiency problem [34, 100]. There are multiple reasons for the sample inefficiency of these algorithms [35]. The SAC algorithm used in this work is known for its sample efficiency [35] compared to other algorithms in the domain. However, it still suffers from the sample-inefficiency problem when compared to the model-based methods. As a result, the algorithm has to interact with the environment to collect millions of training samples which could increase the overall training time by a great extent. Therefore, proper measures must be taken to resolve the sample inefficiency problem.

5.4 Future Directions

The future directions from this work arise from the limitations and the simplifications of the current architecture. For instance, as described in chapter **??**, we overlooked the ability to transition from feedforward to feedback smoothly in our RL architecture. Similarly, there are multiple future directions which are discussed as follows:

- **Lack of next state prediction:** Feedforward control with predictive state estimation given the current state (for example, limb's position and velocity) is a common practice in modeling the human motor systems [73]. It allows the system to calculate an error signal which can be used as a form of feedback even before the sensory feedback is available [97]. At present, our feedforward network does not predict the next state given the current state; instead, it waits for the actuator to get to equilibrium. This would introduce latency

and other problems in the system, which could be avoided by enabling the feedforward network to predict the next state.

- **Simplified feedback design:** At the moment, our architecture has learned feedforward control. However, it is essential to note that our controller can still correct the error using reinforcement. Nevertheless, the corrections made by our controller are different from the way of corrective strategy in humans. Therefore, it can be established that the feedback strategy in the current architecture is a simplified version of the feedback that is available in the human motor system. In motor theory, there are mainly two types of feedback: intrinsic and extrinsic [7]. While the intrinsic is said to be reinforcement, extrinsic feedback is the feedback from a supervisor or teacher [75]. The extrinsic feedback can be used to switch to the feedback control before the sensory feedback is available [97] and to improve the intrinsic feedback, reinforcement [7]. However, our architecture is based on intrinsic (reinforcement) feedback rather than extrinsic. Therefore, our architecture can not account for the extrinsic feedback. So, it is reasonable to include supervisory feedback as well in the architecture.

- **Prediction of next state using feedforward:** Computational models for motor control include the estimation of the next state [47] in the design. Enabling the ability to predict the next state offers some advantages. First, this prediction can be used to create a form of feedback that can be used to correct, faster than sensory feedback. Second, this can be used as a deciding factor for the controller to switch to feedback control. Once the switching mechanism is modeled, the controller can exhibit speed-accuracy tradeoff (SAT), which is observed in humans [50].

- **Speed Accuracy Tradeoff (SAT):** The main advantage of predicting the next state, as explained previously, is that it would allow the system to exhibit the speed accuracy-tradeoff (SAT) mechanism [50]. Because the prediction of the next state can be used to notify the system when the controller's pre-programmed motor command estimates are going off track (for example, overshooting the target). Then, the system can decide to switch to feedback

control to generate corrections to reach the target. However, it is not possible if the system has to wait for the actuator's equilibrium every time. Additionally, it might speed up the learning process as it avoids the delay induced due to the equilibrium. Overall, exhibiting SAT signifies that the controller operates like humans and can build human-level intelligent agents.

- **Inability to generate multi-control strategy:** One limitation in our architecture is that model-free RL algorithms can not learn multiple control strategies at the same time. For example, simultaneously learning both feedforward and feedback controls can not be achieved. Therefore, it poses a challenge to enable the controller to generate human-like corrective movements. However, to make our architecture emulate the human motor behavior, it is imperative that the controller must learn both feedforward and feedback control strategies [45].

- **Index of Difficulty:** Our analysis of the feedforward controller's performance has revealed that the noise present in the motor system impacts the difficulty of the task. It brings the question: what is the correct formulation for the index of difficulty (ID) that models the effect of motor noise. Hence, this opens up an opportunity to conduct studies on human subjects to investigate the effect of motor noise and deduce the ID formulation that is a function of distance (A), target width, and the amount of uncertainty in the system.

- **Motor program:** Another future direction is predicting a motor program (sequence of motor commands) rather than a single motor command. The motor program is responsible for the generation of movements in humans [50]. Enabling the controller to predict a sequence of co-activation at every time step would speed up the learning process. Because, in the typical RL-framework, to execute n motor commands, the controller interacts n number of times with the environment. If the controller can roll out a n length motor program at a time instance t, the whole program can be communicated to the environment and applied in one shot. So, the number of environmental interactions can be reduced, speeding up the learning process.

5.5 Concluding Remarks

Overall, this work solved the learning of the feedforward control problem with some simplifications while highlighting many challenges in this research domain. These challenges must be addressed as solving these can create learning models that respect the motor principles (for example, speed accuracy-tradeoff) observed in humans. Additionally, this thesis exposed various limitations of the current architecture and its impediments to building learning models that emulate human behavior. Solving these roadblocks can help researchers accomplish further advancements in creating end-to-end learning agents that can yield human-level performance in motor tasks.

Bibliography

[1] A. H. Abdi, M. Malakoutian, T. Oxland, and S. Fels. Reinforcement learning for high-dimensional continuous control in biomechanics: An intro to artisynth-rl. In *Deep Reinforcement Learning Workshop, 33rd Conference on Neural Information Processing Systems NeurIPS*, 2019. URL https://arxiv.org/abs/1910.13859. → pages 18, 26, 40

[2] A. H. Abdi, P. Saha, V. P. Srungarapu, and S. Fels. Muscle excitation estimation in biomechanical simulation using NAF reinforcement learning. In *Computational Biomechanics for Medicine*, pages 133–141. Springer International Publishing, Aug. 2019. doi:10.1007/978-3-030-15923-8_11. URL https://doi.org/10.1007/978-3-030-15923-8_11. → page 26

[3] A. H. Abdi, B. Sagl, V. P. Srungarapu, I. Stavness, E. Prisman, P. Abolmaesumi, and S. Fels. Characterizing motor control of mastication with soft actor-critic. *Frontiers in Human Neuroscience*, 14:188, 2020. → pages 30, 44, 51, 54, 104

[4] F. C. Anderson and M. G. Pandy. A dynamic optimization solution for vertical jumping in three dimensions. *Computer Methods in Biomechanics and Biomedical Engineering*, 2(3):201–231, Jan. 1999. doi:10.1080/10255849908907988. URL https://doi.org/10.1080/10255849908907988. → page 18

[5] F. C. Anderson and M. G. Pandy. Dynamic optimization of human walking. *Journal of Biomechanical Engineering*, 123(5):381–390, May 2001. doi:10.1115/1.1392310. URL https://doi.org/10.1115/1.1392310. → page 18

[6] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5055–5065, 2017. → page 2

[7] A. G. Barto. Reinforcement learning. In *Neural systems for control*, pages 7–30. Elsevier, 1997. → pages 16, 88

[8] E. Bizzi and V. C. Cheung. The neural origin of muscle synergies. *Frontiers in computational neuroscience*, 7:51, 2013. → page 16

[9] E. Bizzi, V. C. Cheung, A. d'Avella, P. Saltiel, and M. Tresch. Combining modules for movement. *Brain research reviews*, 57(1):125–133, 2008. → page 16

[10] D. Blana, R. F. Kirsch, and E. K. Chadwick. Combined feedforward and feedback control of a redundant, nonlinear, dynamic musculoskeletal system. *Medical & Biological Engineering & Computing*, 47(5):533–542, Apr. 2009. doi:10.1007/s11517-009-0479-3. URL https://doi.org/10.1007/s11517-009-0479-3. → pages 5, 17

[11] S. S. Blemker and S. L. Delp. Three-dimensional representation of complex muscle architectures and geometries. *Annals of biomedical engineering*, 33(5):661–673, 2005. → page 37

[12] R. Bogacz, E.-J. Wagenmakers, B. U. Forstmann, and S. Nieuwenhuis. The neural basis of the speed–accuracy tradeoff. *Trends in neurosciences*, 33 (1):10–16, 2010. → page 11

[13] A. S. Broad. Generating muscle driven arm movements using reinforcement learning. 2011. → page 26

[14] V. C. Cheung, A. d'Avella, M. C. Tresch, and E. Bizzi. Central and sensory contributions to the activation and organization of muscle synergies during natural motor behaviors. *Journal of Neuroscience*, 25(27):6419–6434, 2005. → page 16

[15] A. Clegg, W. Yu, J. Tan, C. K. Liu, and G. Turk. Learning to dress. *ACM Transactions on Graphics*, 37(6):1–10, Dec. 2018. doi:10.1145/3272127.3275048. URL https://doi.org/10.1145/3272127.3275048. → page 26

[16] A. d'Avella, P. Saltiel, and E. Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature neuroscience*, 6(3): 300–308, 2003. → page 16

[17] T. Degris, M. White, and R. S. Sutton. Off-policy actor-critic. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 179–186, 2012. → page 118

[18] M. R. DeLong and T. Wichmann. Circuits and circuit disorders of the basal ganglia. *Archives of neurology*, 64(1):20–24, 2007. → page 11

[19] G. Deuschl, P. Krack, M. Lauk, and J. Timmer. Clinical neurophysiology of tremor. *Journal of clinical neurophysiology*, 13(2):110–121, 1996. → page 58

[20] N. Dominici, Y. P. Ivanenko, G. Cappellini, A. d'Avella, V. Mondì, M. Cicchese, A. Fabiano, T. Silei, A. Di Paolo, C. Giannini, et al. Locomotor primitives in newborn babies and their development. *Science*, 334(6058):997–999, 2011. → page 16

[21] K. Doya. What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural networks*, 12(7-8):961–974, 1999. → pages 1, 11

[22] K. Doya. Complementary roles of basal ganglia and cerebellum in learning and motor control. *Current opinion in neurobiology*, 10(6):732–739, 2000. → page 11

[23] K. Doya. Reinforcement learning: Computational theory and biological mechanisms. *HFSP journal*, 1(1):30, 2007. → pages 1, 2, 11, 30, 50

[24] A. Erdemir, S. McLean, W. Herzog, and A. J. van den Bogert. Model-based estimation of muscle forces exerted during movements. *Clinical Biomechanics*, 22(2):131–154, 2007. ISSN 02680033. doi:10.1016/j.clinbiomech.2006.09.005. → page 17

[25] D. Farina, R. Merletti, B. Indino, and T. Graven-Nielsen. Surface emg crosstalk evaluated from experimental recordings and simulated signals. *Methods of information in medicine*, 43(01):30–35, 2004. → page 17

[26] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954. doi:10.1037/h0055392. URL https://doi.org/10.1037/h0055392. → page 60

[27] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 47(6):381, 1954. → pages 19, 24, 26, 64

[28] P. M. Fitts. Cognitive aspects of information processing: Iii. set for speed versus accuracy. *Journal of experimental psychology*, 71(6):849, 1966. → pages 6, 8, 24

[29] M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rywHCPkAW. → page 115

[30] J. I. Gold and M. N. Shadlen. Banburismus and the brain: decoding the relationship between sensory stimuli, decisions, and reward. *Neuron*, 36 (2):299–308, 2002. → page 11

[31] V. Golkhou, M. Parnianpour, and C. Lucas. Neuromuscular control of the point to point and oscillatory movements of a sagittal arm with the actor–critic reinforcement learning method. *Computer Methods in Biomechanics and Biomedical Engineering*, 8(2):103–113, Apr. 2005. doi:10.1080/10255840500167952. URL https://doi.org/10.1080/10255840500167952. → page 26

[32] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838. PMLR, 2016. → pages 26, 30, 44, 51, 54, 104, 115

[33] E. Guigon, P. Baraduc, and M. Desmurget. Optimality, stochasticity, and variability in motor behavior. *Journal of computational neuroscience*, 24 (1):57–68, 2008. → pages 8, 29, 32, 57

[34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/haarnoja18b.html. → pages 49, 50, 87

[35] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. URL http://arxiv.org/abs/1812.05905. → pages 30, 50, 87

[36] H. V. Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010. → page 49

[37] M. Hirayama, M. Kawato, and M. I. Jordan. The cascade neural network model and a speed-accuracy trade-off of arm movement. *Journal of Motor Behavior*, 25(3):162–174, Sept. 1993. doi:10.1080/00222895.1993.9942047. URL https://doi.org/10.1080/00222895.1993.9942047. → pages xv, 5, 17, 25, 40

[38] A. Holobar, D. Farina, M. Gazzoni, R. Merletti, and D. Zazula. Estimating motor unit discharge patterns from high-density surface electromyogram. *Clinical Neurophysiology*, 120(3):551–562, Mar. 2009. doi:10.1016/j.clinph.2008.10.160. URL https://doi.org/10.1016/j.clinph.2008.10.160. → page 17

[39] J. F. Houde. Sensorimotor adaptation in speech production. *Science*, 279 (5354):1213–1216, Feb. 1998. doi:10.1126/science.279.5354.1213. URL https://doi.org/10.1126/science.279.5354.1213. → page 4

[40] J. C. Houk and S. P. Wise. Distributed modular architectures linking basal ganglia, cerebellum, and cerebral cortex: their role in planning and controlling action. *Cerebral cortex*, 5(2):95–110, 1995. → page 11

[41] K. Iqbal and A. Roy. Stabilizing PID controllers for a single-link biomechanical model with position, velocity, and force feedback. *Journal of Biomechanical Engineering*, 126(6):838–843, Dec. 2004. doi:10.1115/1.1824134. URL https://doi.org/10.1115/1.1824134. → page 5

[42] J. Izawa, T. Kondo, and K. Ito. Biological arm motion through reinforcement learning. *Biological cybernetics*, 91(1):10–22, 2004. → page 26

[43] K. M. Jagodnik and A. J. van den Bogert. Optimization and evaluation of a proportional derivative controller for planar arm movement. *Journal of Biomechanics*, 43(6):1086–1091, Apr. 2010. doi:10.1016/j.jbiomech.2009.12.017. URL https://doi.org/10.1016/j.jbiomech.2009.12.017. → page 8

[44] K. M. Jagodnik, P. S. Thomas, A. J. van den Bogert, M. S. Branicky, and R. F. Kirsch. Human-like rewards to train a reinforcement learning controller for planar arm movement. *IEEE Transactions on Human-Machine Systems*, 46(5):723–733, Oct. 2016. doi:10.1109/thms.2016.2558630. URL https://doi.org/10.1109/thms.2016.2558630. → page 26

[45] M. I. Jordan and D. M. Wolpert. Computational motor control, 1999. → pages xiv, 6, 14, 15, 16, 17, 89

[46] M. Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9(6):718–727, Dec. 1999. doi:10.1016/s0959-4388(99)00028-8. URL https://doi.org/10.1016/s0959-4388(99)00028-8. → page 14

[47] M. Kawato. Internal models for motor control and trajectory planning. *Current opinion in neurobiology*, 9(6):718–727, 1999. → pages 15, 88

[48] M. Kawato, K. Furukawa, and R. Suzuki. A hierarchical neural-network model for control and learning of voluntary movement. *Biological cybernetics*, 57(3):169–185, 1987. → pages 1, 5, 62

[49] M. Kawato, Y. Maeda, Y. Uno, and R. Suzuki. Trajectory formation of arm movement by cascade neural network model based on minimum torque-change criterion. *Biological cybernetics*, 62(4):275–288, 1990. → page 17

[50] S. W. Keele. Movement control in skilled motor performance. *Psychological bulletin*, 70(6p1):387, 1968. → pages 88, 89

[51] S. W. Keele. *Attention and human performance*. Goodyear Publishing Company, 1973. → page 85

[52] Ł. Kidziński, S. P. Mohanty, C. F. Ong, J. L. Hicks, S. F. Carroll, S. Levine, M. Salathé, and S. L. Delp. Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 101–120. Springer, 2018. → pages 3, 18

[53] Ł. Kidziński, S. P. Mohanty, C. F. Ong, Z. Huang, S. Zhou, A. Pechenko, A. Stelmaszczyk, P. Jarosik, M. Pavlov, S. Kolesnikov, S. Plis, Z. Chen, Z. Zhang, J. Chen, J. Shi, Z. Zheng, C. Yuan, Z. Lin, H. Michalewski, P. Milos, B. Osinski, A. Melnik, M. Schilling, H. Ritter, S. F. Carroll, J. Hicks, S. Levine, M. Salathé, and S. Delp. Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. In S. Escalera and M. Weimer, editors, *The NIPS '17 Competition: Building Intelligent Systems*, pages 121–153, Cham, 2018. Springer International Publishing. ISBN 978-3-319-94042-7. → page 26

[54] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114. → page 52

[55] P. K. Kuhl. Early language acquisition: cracking the speech code. *Nature reviews neuroscience*, 5(11):831–843, 2004. → page 4

[56] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003. → page 48

[57] J. E. Lloyd, I. Stavness, and S. Fels. Artisynth: A fast interactive biomechanical modeling toolkit combining multibody and finite element simulation. In *Soft tissue biomechanical modeling for computer assisted surgery*, pages 355–394. Springer, 2012. → page 35

[58] I. S. MacKenzie. A note on the information-theoretic basis for fitts' law. *Journal of motor behavior*, 21(3):323–330, 1989. → page 64

[59] I. S. MacKenzie. Fitts' law as a performance model in human-computer interaction. 1993. → page 62

[60] D. E. Meyer, R. A. Abrams, S. Kornblum, C. E. Wright, and J. Keith Smith. Optimality in human motor performance: ideal control of rapid aimed movements. *Psychological review*, 95(3):340, 1988. → page 85

[61] M. Millard, T. Uchida, A. Seth, and S. L. Delp. Flexing computational muscle: modeling and simulation of musculotendon dynamics. *Journal of biomechanical engineering*, 135(2), 2013. → page 37

[62] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. URL http://arxiv.org/abs/1312.5602. → pages 26, 48

[63] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. doi:10.1038/nature14236. URL https://doi.org/10.1038/nature14236. → pages 49, 114, 115

[64] K. D. Mombaur, R. W. Longman, H. G. Bock, and J. P. Schlöder. Open-loop stable running. *Robotica*, 23(1):21–33, 2005. → page 5

[65] K. L. Norman and J. Kirakowski, editors. *The Wiley Handbook of Human Computer Interaction*. John Wiley & Sons, Ltd, Feb. 2018. doi:10.1002/9781118976005. URL https://doi.org/10.1002/9781118976005. → page 64

[66] E. Otten. Inverse and forward dynamics: models of multi-body systems. *Philos Trans R Soc Lond B Biol Sci*, 358(1437):1493–1500, Sep 2003. ISSN 0962-8436. doi:10.1098/rstb.2003.1354. URL https://www.ncbi.nlm.nih.gov/pubmed/14561340. 14561340[pmid]. → page 18

[67] C. Peck, G. Langenbach, and A. Hannam. Dynamic simulation of muscle and articular properties during human wide jaw opening. *Archives of Oral Biology*, 45(11):963–982, 2000. → page 37

[68] D. R. Pedersen, R. A. Brand, and D. T. Davy. Pelvic muscle and acetabular contact forces during gait. *Journal of Biomechanics*, 30(9):959–965, Sept. 1997. doi:10.1016/s0021-9290(97)00041-9. URL https://doi.org/10.1016/s0021-9290(97)00041-9. → page 18

[69] X. B. Peng, G. Berseth, K. Yin, and M. V. D. Panne. DeepLoco. *ACM Transactions on Graphics*, 36(4):1–13, July 2017. doi:10.1145/3072959.3073602. URL https://doi.org/10.1145/3072959.3073602. → page 26

[70] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, May 2008. doi:10.1016/j.neunet.2008.02.003. URL https://doi.org/10.1016/j.neunet.2008.02.003. → page 46

[71] P. Redgrave, T. J. Prescott, and K. Gurney. The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience*, 89(4):1009–1023, 1999. → page 11

[72] S. Schaal and C. G. Atkeson. Open loop stable control strategies for robot juggling. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 913–918. IEEE, 1993. → page 5

[73] S. Schaal and N. Schweighofer. Computational motor control in humans and robots. *Current opinion in neurobiology*, 15(6):675–682, 2005. → pages 9, 10, 32, 58, 87

[74] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *ICLR (Poster)*, 2016. → page 115

[75] R. SCHMIDT. Motor control and learning. *Human Kinetics*, 1999. → page 88

[76] R. Schmidt and T. Lee. *Motor learning and performance 6th edition with web study guide-loose-leaf edition: From principles to application*. Human Kinetics Publishers, 2019. → pages xiv, 7

[77] R. A. Schmidt, H. N. Zelaznik, and J. S. Frank. Sources of inaccuracy in rapid movement. In *Information Processing in Motor Control and Learning*, pages 183–203. Elsevier, 1978. doi:10.1016/b978-0-12-665960-3.50014-1. URL https://doi.org/10.1016/b978-0-12-665960-3.50014-1. → pages 8, 10, 32

[78] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel. Trust region policy optimization. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1889–1897. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045319. → page 120

[79] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL http://arxiv.org/abs/1707.06347. → page 120

[80] A. Seireg and R. Arvikar. The prediction of muscular load sharing and joint forces in the lower extremities during walking. *Journal of Biomechanics*, 8 (2):89–102, Mar. 1975. doi:10.1016/0021-9290(75)90089-5. URL https://doi.org/10.1016/0021-9290(75)90089-5. → page 18

[81] S. C. Seow. Information theoretic models of hci: A comparison of the hick-hyman law and fitts' law. *Human-computer interaction*, 20(3): 315–352, 2005. → page 64

[82] M. R. Sheridan. A reappraisal of fitts' law. *Journal of Motor Behavior*, 11 (3):179–188, 1979. → page 85

[83] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014. → page 118

[84] I. Stavness, J. E. Lloyd, and S. Fels. Automatic prediction of tongue muscle activations using a finite element model. *Journal of biomechanics*, 45(16):2841–2848, 2012. → page 18

[85] I. K. Stavness. *Byte your tongue: A computational model of human mandibular-lingual biomechanics for biomedical applications*. PhD thesis, University of British Columbia, 2010. → page 18

[86] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction, 2011. → page 116

[87] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. → page 34

[88] D. G. Thelen, F. C. Anderson, and S. L. Delp. Generating dynamic simulations of movement using computed muscle control. *Journal of Biomechanics*, 36(3):321–328, Mar. 2003. doi:10.1016/s0021-9290(02)00432-3. URL https://doi.org/10.1016/s0021-9290(02)00432-3. → page 18

[89] L. H. Ting and J. M. Macpherson. A limited set of muscle synergies for force control during a postural task. *Journal of neurophysiology*, 93(1): 609–613, 2005. → page 16

[90] E. Todorov and M. I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature neuroscience*, 5(11):1226–1235, 2002. → page 17

[91] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. → page 18

[92] R. J. van Beers. Motor learning is optimally tuned to the properties of motor noise. *Neuron*, 63(3):406–417, Aug. 2009. doi:10.1016/j.neuron.2009.06.025. URL https://doi.org/10.1016/j.neuron.2009.06.025. → pages 6, 26

[93] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016. → pages 49, 115

[94] A. T. Welford. The measurement of sensory-motor performance: Survey and reappraisal of twelve years'progress. *Ergonomics*, 3(3):189–230, 1960. → pages 23, 24, 26, 76

[95] A. T. Welford. Fundamentals of skill. 1968. → pages 19, 23

[96] A. T. Welford, A. H. Norris, and N. Shock. Speed and accuracy of movement and their changes with age. *Acta psychologica*, 30:3–15, 1969. → page 85

[97] D. M. Wolpert, Z. Ghahramani, and M. I. Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995. → pages 87, 88

[98] R. S. Woodworth. Accuracy of voluntary movement. *The Psychological Review: Monograph Supplements*, 3(3):i, 1899. → pages 6, 8, 19

[99] H. H. Yin and B. J. Knowlton. The role of the basal ganglia in habit formation. *Nature Reviews Neuroscience*, 7:464 EP –, Jun 2006. URL https://doi.org/10.1038/nrn1919. Review Article. → page 11

[100] Y. Yu. Towards sample efficient reinforcement learning. In *IJCAI*, pages 5739–5743, 2018. → page 87

[101] H. Zhang, H. Yu, and W. Xu. Listen, interact and talk: Learning to speak via interaction. *arXiv preprint arXiv:1705.09906*, 2017. → page 4

Appendix A

Supporting Materials

A.1 2-Muscle mechanical system

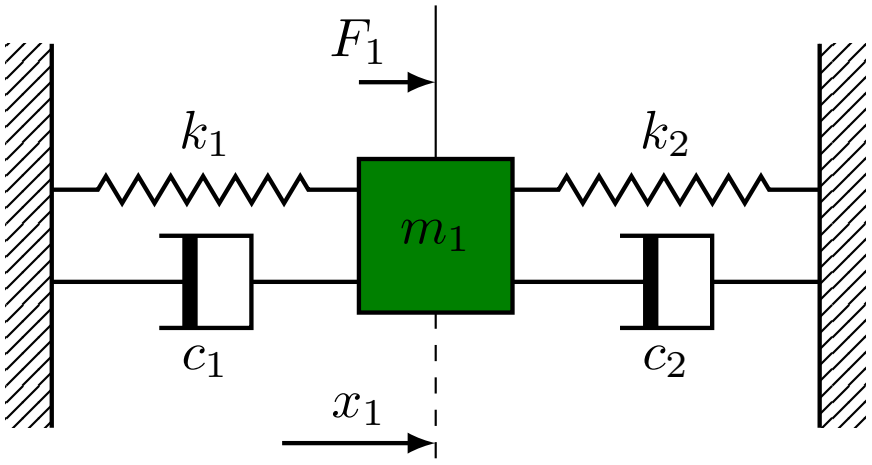

Figure A.1: Point-To-Point model represented as a two spring mass damper model. An object with mass M is coupled with two identical springs pulling the object by activating the springs.

The muscle model used in our work is essentially a spring-mass-damper system. As shown in Figure A.1, the muscles are represented as spring with dampers

joined with a mass object. Let us assume that F is an external force which is an active component of the muscles. Assuming the external force is equal to zero, i,e., $F = 0$, then the equilibrium equation is:

$$F = m\ddot{x} + (d_1 + d_2)\dot{x} + (k_1 + k_2)x = m\ddot{x} + (D)\dot{x} + (K)x = 0 \tag{A.1}$$

where D and K are the effective damping and stiffness coefficient of the springs respectively. Let us define ζ is damping ratio that determines the stability of the system which is defined as $\frac{D}{2\sqrt{KM}}$ and natural frequency as the $\omega_n = \sqrt{\frac{K}{M}}$. The equation A.1 is a homogeneous ode equation that can be solved to get complementary function solution Y_{CF}. Depending up on the damping ratio ζ, the behavior of the system depends. If $\zeta = 1$, then the system is critically damped, $\zeta < 1$ under-damped, and $\zeta > 1$ over-damped. Setting up the system to be critically damped is a common practice in control theory. The solution to the equation A.1 is as follows:

$$x(t) = G_1 e^{S_1 t} + G_2 e^{S_2 t} \tag{A.2}$$

where $S_{1,2} = (-\zeta \pm \sqrt{\zeta^2 - 1})\omega_n$

Now, the equation for the cases where external force is not zero, the equation can be written with the help of linear axial muscle equation described in ArtiSynth []. The linear muscle equation is shown in A.3

$$F(x, \dot{x}) = \big(a f_{max} + f_{passive} f_{max}\big)x' + d\dot{x} \tag{A.3}$$

where x' is defined as $x' = \frac{x - x_{opt}}{x_{max} - x_{opt}}$, f_{max} is the maximum force, $f_{passive}$ is the proportion of f_{max} to apply as passive tension, and $a(t)$ are the muscle activations.

Equation A.3 has an active component, passive and friction component. For writing the equation for external force, the active component $a f_{max} x'$ is utilized. Then the net external force becomes $f_2^{active} - f_1^{active}$. Now, the final equation can be written as follows:

$$f_2^{active} - f_1^{active} = f_{max}\big(a_2 x_2' - a_1 x_1'\big) \tag{A.4}$$

$$f_{max}\left(a_2 x_2' - a_1 x_1'\right) = m\ddot{x} + (D)\dot{x} + (K)x \tag{A.5}$$

The equation is in the form of non-homogeneous ode and it can be solve to get the particular solution Y_{PS} via techniques to solve non-homogeneous ode. Then the complete solution becomes $Y_{GS} = Y_{CF} + Y_{PS}$.

This problem of solving for the kinematics given activations is referred to as forward problem and the solution can be obtained by solving the above second-order differential equation. However, the inverse problem is relatively complicated and there has been some research done on this problem.

A.1.1 Equilibrium-point Analysis

In this work, fundamentally we deal with the required co-activation values to reach a particular target present at a particular point in the kinematic space. This problem is generally referred to as inverse problem [3, 32]. It is still a burning and very crucial research problem that needs to be solved. We take a DeepRL approach to solve this problem. However, there should be some baseline to compare to the DeepRL based approach. For this purpose, we conducted a grid search with our muscle model where multiple samples of activation-displacement data has been collected. This data collection involves generating the various combinations of co-activation values. All these combinations are applied in a serial fashion and the corresponding displacement values are collected. At $t = 0$, the muscle model starts activating the actuator and the activations stay active till it reaches the equilibrium.Like this, we collected 10,000 samples to form a database. From this database, the low magnitude co-activations were filtered out and the relationship between the co-activation and the actuator displacement is shown in Figure A.2. As a result, we found that the relationship between the activation and the distance travelled by the controller is quadratic in nature as shown in the Figure A.2. This process is also called mapping the activations to kinematics.

It is important to note that the inverse mapping is many-to-one problem. In other words, there can be many possible co-activation values leading to the same position in the kinematic space. Here, we assumed that the targets are present on the right side of the controller. As this a many-to-one problem, imposing certain

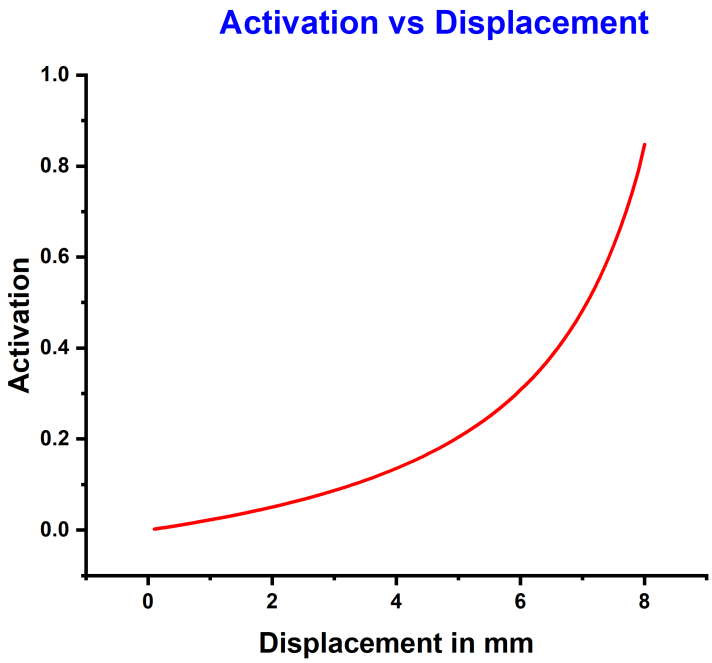

Figure A.2: The behavior of the system to reach a particular point in the kinematic space with the least activations condition.

constraints on the system can result into unique solution. So, we imposed the least activations constraint. Therefore, the left muscle activation is expected to be minimum (i.e., 0.0) and the right muscle is activated to reach a particular point. The y-axis shown in the graph A.2 represents the net activation required or the minimum activation for the right muscle.

A.2 Deep Reinforcement Learning

A.2.1 Introduction to Reinforcement Learning

Reinforcement Learning (RL) is one of the machine learning strategies in the Artificial Intelligence (AI) research domain alongside supervised learning and unsupervised learning. RL involves building a computational approach to learning

from interaction with the environment. This learning from interaction deals with mapping the situations (states) to maximize a reward signal. This framework is often referred to as Learning by Reinforcement or Reinforcement Learning. This learning technique is somewhere in between supervised and unsupervised learning. RL is different because the agent collects the data samples from the environment depending on the reward signal; unlike in the supervised approach, the agent has access to the annotated data by an external expert. Moreover, it is distinct from unsupervised learning because the agent determines actions that return the maximum reward instead of finding the hidden representations of the available unlabeled data (offline data collection). Data required for training is collected during the training process (online data collection). Hence, there is a necessity to trade-off between exploration and exploitation, which is absent in other learning strategies. We discuss how this problem can be solved in the later sections. In summary, the RL system is essentially a closed-loop control system with a reward signal acting as feedback where the agent maps the states to actions through maximizing the reward signal.

A.2.2 RL Framework

Generally, in any RL setup, the primary components are the agent and the environment. At each step t, the agent executes action a_t in the environment and receives the state s_t and a reward signal which is a function of the action executed and the resultant state. Fig. 2.1 shows a diagrammatic representation of this framework.

A.2.3 Deep Reinforcement Learning

Deep neural networks coupled with Reinforcement Learning gives rise to a promising domain called Deep Reinforcement Learning. Deep neural networks are generally utilized to approximate non-linear functions such as Value-function and State-action function (Q-function). Hence, in this domain, deep neural networks are sometimes referred to as function approximators. Deep neural networks have shown promising performance in function approximating that helped reinforcement learning to solve various tasks in various domains. Notably, DeepRL architectures have solved various motor control tasks such as gait analysis, muscle

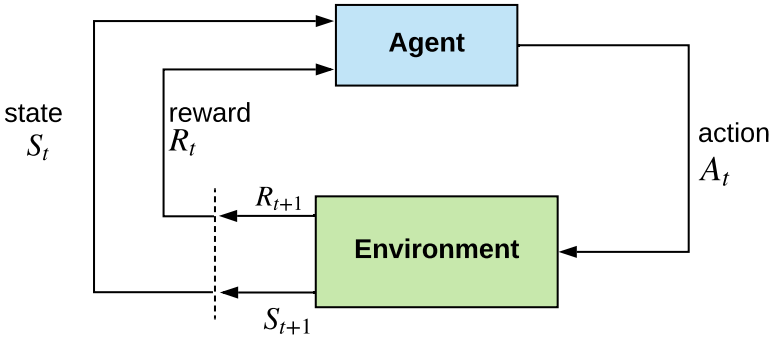

Figure A.3: A typical RL framework

activation estimation, and locomotion problems.

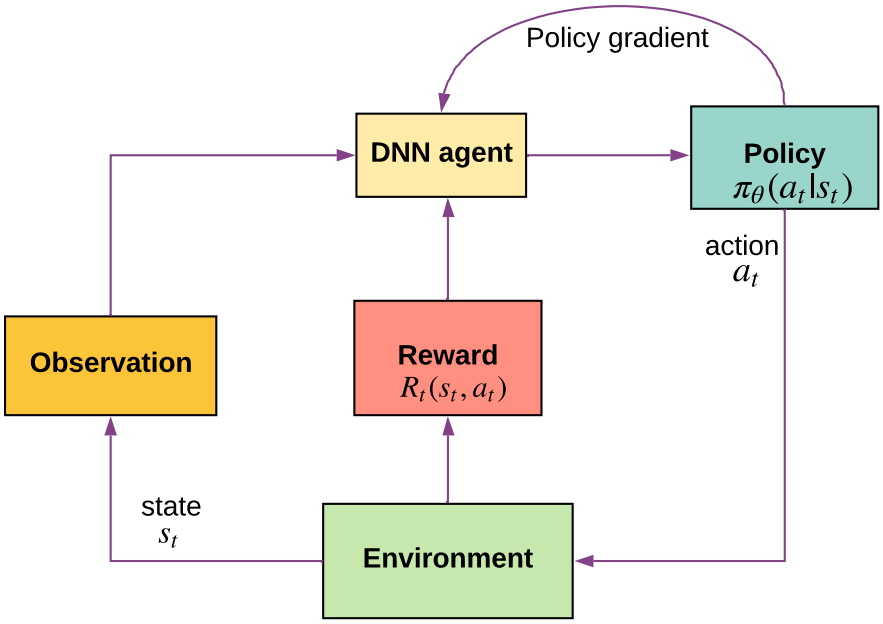

Figure A.4: A typical DeepRL framework

A.2.4 Components of DeepRL

Agent

An agent is a program that interacts with the environment by applying specific actions, collecting observations, and receiving rewards for the actions performed. In the DeepRL framework, the agent is a deep neural network that can map observations to either a value of the state-action pair or explicitly to actions (policy) parameterized by the weights.

Episode

The main theme of RL is built on maximizing the reward the agent receives during an episode. An episode is a sequence of states, actions, and rewards that comes between an initial-state and a terminal-state. The terminal state is when the agent succeeds in the task. The training of DeepRL occurs in an episodic fashion.

Policy

The policy is defined as the agent's behavior given a state s. In DeepRL notation, a policy is denoted by π_θ, which means a policy π parameterized by the neural network weights θ. It can also be seen as a mapping function that can map the state s to the agent's action a. Mainly, the policy is categorized into two types:

- Deterministic; denoted by $\mu_\theta(s) = a$.
- Stochastic; denoted by $\pi_\theta(a|s) = \mathbb{P}_\pi[A = a|S = s]$.

Value Function

The value function is a metric to measure how beneficial a state or an action is by predicting estimated return. Return is a total sum of discounted rewards going forward. The formulation of return G_t starting from t is given by equation 2.1

$$G_t = R_{t+1} + \gamma R_{t+2} + = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{A.6}$$

Now, the state-value function of a state is the expected return at a time instance t and a state s as denoted in equation 2.2

$$V_\pi(s) = \mathbf{E}_\pi[G_t|S_t = s] \tag{A.7}$$

Similarly, Q-value (action-value) function is given by equation 2.3

$$Q_\pi(s,a) = \mathbf{E}_\pi[G_t|S_t = s, A_t = a] \tag{A.8}$$

Moreover, the state-value function can be recovered from the action-value function by utilizing the policy (probability distribution over actions) as shown in equation 2.4:

$$V_\pi(s) = \sum_{a \in A} Q_\pi(s,a)\pi_\theta(a|s) \tag{A.9}$$

Also, we can define an advantage function (A-value) as shown in equation 2.5:

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s) \tag{A.10}$$

Using the advantage function has its perks which will be discussed in the later sections.

Since the objective is to find the optimal value and optimal policy that results in maximum return, using the above equations, we can define the optimal value and policy as follows:

$$\begin{aligned} V_*(s) &= \max_\pi V_\pi(s) \\ Q_*(s,a) &= \max_\pi Q_\pi(s,a) \end{aligned} \tag{A.11}$$

$$\begin{aligned} \pi_* &= \operatorname*{argmax}_\pi V_\pi(s) \\ \pi_* &= \operatorname*{argmax}_\pi Q_\pi(s,a) \end{aligned} \tag{A.12}$$

Reward

The reward is the central aspect of any model-free RL framework. It is a scalar value that can be obtained densely or sparsely. The idea of the reward is to provide feedback on the agent's recent activity. Moreover, the reward is local, i.e., it reflects the agent's success due to the latest action predictions. So, an agent always looking to reinforce by performing actions that result in larger rewards may face a problem in the future and end up learning a bad policy. Therefore, the agent's objective should be to attain the maximum accumulated reward over its sequence of actions.

Observations

Apart from reward, observations of the environment are other sources of information for the agent to learn. Important to note here is that there is a subtle difference between the state and observation. The state of an environment could include all the information about all possible future states' transition probabilities, given the current state. On the other hand, observation is a less informative version of the state which provides some information to the agent around the future states, given the current state.

A.2.5 Taxonomy of Deep Reinforcement Learning Algorithms

Model-Free vs. Model-Based RL

One of the main aspects in classifying the algorithm type is; whether the agent has prior knowledge of the system or the environment dynamics (model). Algorithms that take advantage of the model are called model-based. Algorithms where the model is unknown and the agent has to learn solely based on the experience are called Model-Free methods. Choosing between these two main classes is a bit challenging task. Most of the time, the system dynamics are not known to the agent. Hence, this led to the extensive development and popularity of the model-free methods, although they are sample inefficient. As discussed in the Chapter 1, the neural regime that is responsible for all the motor control processes including speed-accuracy tradeoff is basal ganglia where these processes occur by maximizing the reward. The objective of model-based RL is to optimize a cost

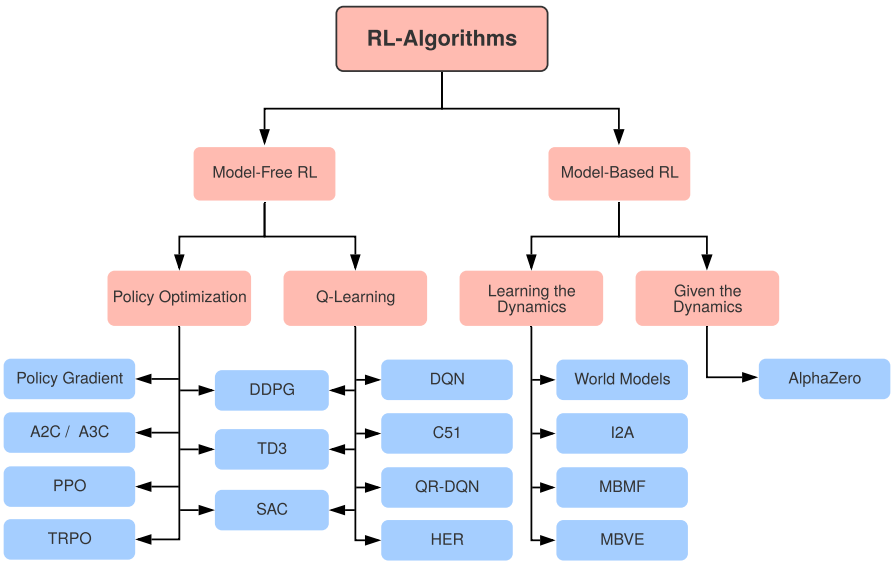

Figure A.5: Taxonomy of RL Algorithms

function rather than a reward function. Model-Free RL methods' objective is to determine the optimal action that results in maximum reward. Hence, model-free RL is more suitable and close to the motor control process occur in humans. Further, model-free methods are simple compared to model-based as model-based require kinematic-error to train, whereas model-free methods require a simple yet powerful terminal reinforcement. Therefore, in this work, to address our challenging problem, we opted to take leverage of model-free methods due to their simplicity. A simple classification of RL algorithms was depicted in Fig 2.2.

A.3 Algorithms in Model-Free RL

Algorithms in Model-Free RL are of mainly two types: 1. Value-based learning 2. policy-based learning In this section, we discuss some of the important algorithms based on policy-based and value-based learning.

Q-learning Methods

Bellman Equations: Bellman equations lay the foundation for value-based learning. He proved that the bellman equations provide the optimal value function. To understand this, consider a simple Markov decision process (MDP) as shown in Figure A.6 and decompose the value function into their recursive form:

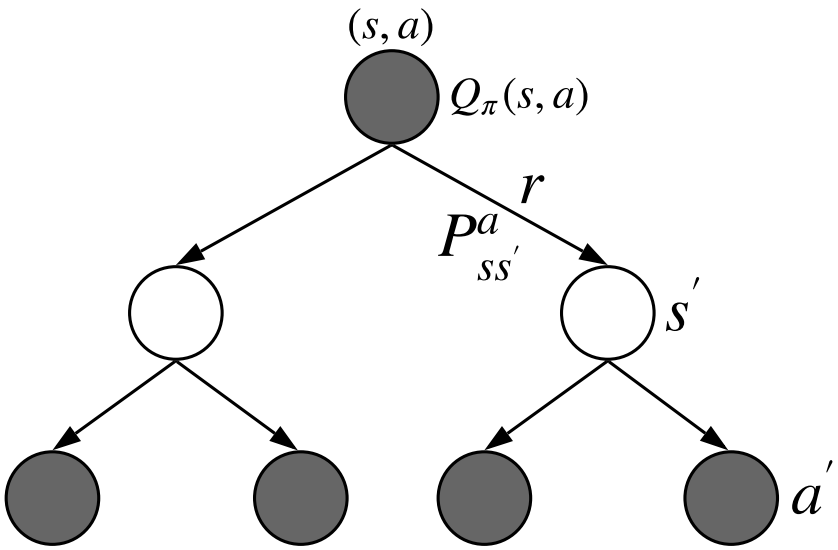

Figure A.6: A stochastic MDP

$$V(s) = \mathbf{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s] \tag{A.13}$$

$$Q(s,a) = \mathbf{E}[R_{t+1} + \gamma \mathbf{E}_{\mathbf{a}\sim\pi} Q(s_{t+1}, a) | S_t = s, A_t = a] \tag{A.14}$$

Furthermore, extending the V and Q by following the policy π gives us the following equations.

$$V_\pi(s) = \sum_{a \in A} \pi(a|s) \Big(R(s,a) + \gamma \sum_{s' \in S} P^a_{ss'} V_\pi(s') \Big) \tag{A.15}$$

$$Q_\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} P^a_{ss'} \sum_{a' \in A} \pi(a'|s') Q_\pi(s',a') \tag{A.16}$$

Now, from the bellman optimal equation theorem, the optimum state-value and action-value are given by the following equations, which looks very similar to equations 2.11 and 2.12:

$$V_*(s) = \max_{a \in A} \Big(R(s,a) + \gamma \sum_{s' \in S} P^a_{ss'} V_*(s') \Big) \tag{A.17}$$

$$Q_\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} P^a_{ss'} \max_{a' \in A} Q_*(s',a') \tag{A.18}$$

One important thing is, transition probabilities are not always known. So we can not directly take advantage of the bellman equations. However, these equations provide fundamental background for many of the algorithms, which can be seen in the later sections.

Q-Learning: Q-learning is an off-policy RL algorithm aiming to find a policy that maximizes the future return by optimizing the Q-function. It is off-policy because this algorithm selects actions with the best Q-values rather than depending explicitly on the current policy. In an algorithm, the agent can exploit or explore the environment. Exploiting is to take the actions with maximum Q-value from a look-up table as the reference. Exploring is to apply random actions which allow the agent to explore. Both exploitation and exploration are essential for the agent to perform well, and it is crucial to have an appropriate balance between these two processes. The parameter epsilon ε is maintained to have a trade-off between exploration and exploitation. Depending upon the epsilon parameter, either action with maximum Q-value or random actions is chosen. The algorithm of Q-learning is four-fold:

- At time step t, start from state S_t with an empty Q table, choose the action A_t according to the epsilon parameter.

- By applying action A_t to environment, obtain the next state S_{t+1}, reward R_{t+1}.
- Update the Q function using the following update rule:

$$Q(S_t, A_t) \longleftarrow (1-\alpha)Q(S_t, A_t) + \alpha\Big(R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, a)\Big) \tag{A.19}$$

- Increment the time step and repeat from step 1 until convergence.

Even though, Q algorithm has shown promising results for the environments with low-dimensional state-action space. It is not feasible to store Q-values for all state-action pairs, mainly when the state and action space is high-dimensional. This drawback has been addressed with the Deep Q-learning algorithm.

Deep Q-Learning: In Q-learning, one of the primary drawbacks is unable to memorize optimal Q-values for all the state-action pairs in the case of large state-action spaces. This problem can be solved by learning to approximate a mapping function that can map state-action pairs to the optimal Q-values. General practice is to deploy a deep neural network with parameter θ for the approximation of Q-value function, which is denoted by $Q(s, a; \theta)$. However, in practice, When Q-learning is coupled with non-linear function approximators (neural networks), the algorithm suffers from severe instability and divergence problems during training. These problems are addressed and solved in Deep Q-Network paper [63] which improvised and stabilized the training performance to a great extent by introducing two techniques, "Experience Replay" and "Periodic Update of Target Network."

Experience Replay: The primary reason for divergence during training is the correlation among the observation sequences. This issue can be solved by introducing an experience replay buffer and drawing samples randomly. Every observation tuple (S_t, A_t, R_t, S_{t+1}) in an episode at a time step e_t is stored in a memory buffer D. Sampling the replay buffer data makes the data independent and identically distributed (IID) and maintains smoothness in the data distribution. Moreover, maintaining IID data satisfies the fundamental requirement of the stochastic gradient optimization technique. It helps in stabilizing the learning process of the neural networks.

Target Q-Network: The main reason for the instability during training is related to the correlation between samples, which creates an issue for the Bellman equation estimation. It provides us the value of $Q(s,a)$ via $Q(s',a')$ which is referred to as bootstrapping. However, both states s and s' are very similar since they have only one step between them, and it is incredibly hard for the neural networks to distinguish between them. Therefore, updating our network parameters to make our predicted $Q(s,a)$ closer to the desired value, the $Q(s',a')$ value can be altered implicitly. It is a concern due to the change of $Q(s',a')$ in the Bellman equation, and as a result, this can bring much instability in the training process. The solution is to maintain a target network, where we clone the Q-network and use it for the $Q(s',a')$ value in the Bellman equation. This network is periodically updated with the main Q-network every C step, bringing stability to the training process. The algorithm given in terms of pseudo-code can be found in [63].

Overall, there are many variants of DQN [29, 74, 93] to improve the algorithm's performance. However, one main point here to note is Deep Q-Networks can only be applied to discrete space. It can not be applied to continuous space environments. Although DQN was successfully extended to continuous spaces [32], many powerful algorithms based on the policy gradient approach were introduced to deal with continuous spaces.

Policy Gradient Methods

In Q-learning, the current policy of the agent is dictated by the value functions. However, there is an alternative where we can directly optimize the agent's policy rather than determining optimal value-functions. The main aim of any reinforcement learning strategy is to find an optimal policy of an agent that maximizes rewards. These methods' objective is to model and optimize the policy. The policy is parameterized by the neural network's weights/parameters θ that can be updated to learn the optimal course of actions (policy). The objective function in terms of policy that can be utilized to optimize θ is denoted as follows:

$$J(\theta) = \sum_{s \in S} d^{\pi}(s) V^{\pi}(s) = \sum_{s \in S} d^{\pi}(s) \sum_{a \in A} \pi_{\theta}(a|s) Q^{\pi}(s,a) \tag{A.20}$$

where $d^{\pi}(s)$ is the stationary distribution of Markov chain for π_θ (policy distribution parameterized by θ) As the name suggests, applying the gradient to equation 2.15 and utilizing gradient ascent, finding the best set of parameters for the policy distribution provides a maximum return. However, it is not easy to calculate the gradient $\nabla_\theta J(\theta)$ as it is dependant on both the actions and the model of the environment, which is not generally known. Fortunately, this problem is solved by policy gradient theorem, which simplifies the computation of the gradient $\nabla_\theta J(\theta)$. This theorem-proof is detailed in [86] and the following equation gives the gradient:

$$\nabla_\theta J(\theta) = \mathbf{E}_\pi[Q^\pi(s,a)\nabla_\theta \ln \pi_\theta(a|s)] \tag{A.21}$$

The policy gradient theorem is fundamental to many policy gradient algorithms. Many algorithms introduced minor changes in the above policy gradient equation since the vanilla version results in a high variance in gradient updates. This issue is discussed more in the following sections.

Reinforce: Reinforce is a simple technique that takes the advantage of expected return. Consider the policy gradient equation and rearrange in terms of expected return:

$$\nabla_\theta J(\theta) = \mathbf{E}_\pi[G_t \nabla_\theta \ln \pi_\theta(a|s)] \tag{A.22}$$

In reinforce algorithm, the major point to understand is to calculate the expected return G_t from the collected trajectories. Furthermore, this can be utilized to update our policy gradient.

- Initialize the policy $\pi(a|s)$ with random parameters θ.
- Obtain and store transition tuples (S_t, A_t, R_t, S'_t) using policy π_θ.
- For every step t in the trajectory, estimate expected return G_t.
- Update policy parameters using the following equation:

$$\theta \longleftarrow \theta + \alpha\gamma^t G_t \nabla_\theta \ln \pi_\theta(a|s) \tag{A.23}$$

- Repeat from step 2 until convergence.

One of the main setbacks of the algorithm is the high variance in the gradient updates. It is mainly due to the Q-value estimate. Therefore, to reduce the variance of gradient estimation, advantage estimate $A(s,a) = Q(s,a) - V(s)$. The same trick has been used extensively utilized in more advanced algorithms as well.

Actor Critic Architecture: Actor-Critic architecture is a core component in many popular model-free algorithms, which improved the vanilla policy gradient method's performance. From the vanilla policy gradient, it is evident that the main component is the policy model and the scaling component value function. If a Q-value estimate is used, it results in a high variance of the gradient estimates, which causes divergence issues. So, an advantage estimate was utilized to reduce the variance in reinforce method. Learning to estimate the value-function can help a lot as it helps in reducing the variance in the gradient updates. Therefore, in Actor-Critic architecture, we maintain a neural network called "*Critic*" that can provide estimates of the value function. Actor-network updates the parameters θ and returning the probability distribution of actions in the direction suggested by the critic-network. Vanilla actor-critic and reinforce collect data points according to the target policy, which should be optimized. However, combining some of the techniques used in Off-policy methods have proven to be useful. First, maintaining an experience replay buffer allows the agent to use past samples, which increases the sample efficiency. Also, the agent can have better exploration since the data collection is followed by behavioral policy rather than target policy. It adds a new component to the on-policy gradient equation making the objective function into an off-policy gradient equation, as shown below.

$$J(\theta) = \sum_{s \in S} d^{\beta}(s) \sum_{a \in A} Q^{\pi}(s,a) \pi_{\theta}(a|s) = \mathbf{E}_{s \sim d^{\beta}} \Big[\sum_{a \in A} Q^{\pi}(s,a) \pi_{\theta}(a|s) \Big] \tag{A.24}$$

Now, rearranging the gradient equation is given below:

$$\nabla_{\theta} J(\theta) = \mathbf{E}_{\beta} \Big[\frac{\pi_{\theta}(a|s)}{\beta(a|s)} Q^{\pi}(s,a) \nabla_{\theta} \ln \pi_{\theta}(a|s) \Big] \tag{A.25}$$

where $d^{\beta}(s)$ is the stationary distribution of the behavior policy β and $\frac{\pi_\theta(a|s)}{\beta(a|s)}$ is commonly referred to as importance weight. Since it is hard to compute the gradient of Q-value, it can be ignored without much damage to algorithm performance and it was proved in [17].

Q^{π} is the action-value function estimated with respect to the target policy π

There are various representations of the gradient, which helps to reduce the variance in the gradient estimates. However, the actor-critic architecture is the fundamental component to many advanced algorithms such as Deep Deterministic Policy Gradient (DDPG), Soft Actor-Critic (SAC).

Deep Deterministic Policy Gradients: It is crucial to discuss Deterministic Policy gradients (DPG) to understand Deep Deterministic Policy Gradients (DDPG). In the previous methods so far, we modeled our policy as a probability distribution over actions A given the current state. This way of modeling the policy is called stochastic. Another way is to model our policy as deterministic, where our policy directly outputs the desired action values. It is usually denoted by $\mu(s)$. Now, the objective function in terms of the deterministic policy can be written as:

$$J(\theta) = \int_S \rho^{\mu}(s) Q(s, \mu_\theta(s)) ds \tag{A.26}$$

Where $\rho^{\mu}(s)$ is the probability of the agent's visitation to the next state given the current state and $Q(s, \mu_\theta(s)$ is the Q-value given a deterministic policy.
The gradient for the above equation can be calculated as below using the derivative product rule:

$$\nabla_\theta J(\theta) = \mathbf{E}_{s\sim\rho^{\mu}}[\nabla_a Q(s,a) \nabla_\theta \mu_\theta(s)|_{a=\mu_\theta(s)}] \tag{A.27}$$

The above gradient is called deterministic policy gradient (DPG) since the policy is deterministic. This deterministic policy can be viewed as a particular case of the stochastic policy. In DPG paper [83], it was proved that deterministic policy is a particular case of stochastic. However, there is a caveat here that is important for the agent's performance. Since the policy is deterministic, it is hard for the agent to explore unless there is enough noise coupled with the environment. Therefore,

general practice is to add noise process to the policy that enables the agent to do enough exploration, i.e., $\mu(s) + \varepsilon\mathcal{N}$ where ε is the scaling factor for noise and $\mathcal{N}$ is the Ornstein-Uhlenbeck noise process. DDPG is a mode-free off-policy algorithm based on Actor-Critic architecture, which combines the advantages of both DQN and DPG. Below is the algorithm from the paper DDPG.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.

Initialize target network Q' and μ' with weights $\theta^{Q'} \longleftarrow \theta^Q$, $\theta^{\mu'} \longleftarrow \theta^\mu$. Initialize replay buffer R

for *episode = 1, M* **do**

Initialize a random process $\mathcal{N}$ for action exploration. Receive initial observation state s_1

for *t = 1, T* **do**

Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise. Execute action a_t and observe reward r_t and observe new state s_{t+1}.

Store transition (s_t, a_t, r_t, s_{t+1}) in R. Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) in R. Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$ Update critic by minimizing the loss:

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$$

Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i, a=\mu(s)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

Update the target networks:

$$\theta^{Q'} \longleftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \longleftarrow \tau\theta^Q + (1-\tau)\theta^{\mu'}$$

end

end

There are also many DeepRL algorithms [78, 79] based on policy optimization techniques that can solve the control problem. However, the methodology we used in our work is not very critical to understand these methods.