APPLICATIONS OF MACHINE LEARNING IN CONTROLLING OF ROBOTICS MEDICAL DEVICES

HOANG NGOC SON

(B.Eng. (Hons), NUS)

A THESIS SUBMITTED FOR THE DEGREE OF

MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2020

Supervisors:

Associate Professor Chui Chee Kong, NUS

Doctor Chua Chin Heng Matthew, NUS-ISS

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

HOANG NGOC SON

(DATE)

ACKNOWLEDGEMENTS

I would like to thank Prof. Chui and Dr. Chua for the opportunity to work on these meaningful and challenging projects. The autonomy given allowed me to explore and experiment with many ideas and directions. I would also like to express my gratitude to my friends and family whose supports have inspired me to complete this thesis.

This research is supported in part by the Singapore Ministry of Health's National Medical Research Council under its Enabling Innovation Grant, Grant No: NMRC/EIG06/2017

TABLE OF CONTENTS

SUMMARY

The advent of machine learning algorithms and the computational muscles required to execute them efficiently have ushered a new era of machine intelligence. This in turn revolutionizes how difficult fields such as machine vision, natural language processing and autonomous decision making are approached. For instance, in the medical sector, deep convolutional neural networks possess tremendous potential in boosting the efficacy and efficiency of radiology. With such applications producing promising results while still in their testing phases, it is worthy to explore more applications of machine learning in the medical setting. As such, this thesis attempts to apply learning algorithms to control medical devices. Specifically, two case studies are presented. The first concerns with the control of a wearable gait rehabilitation device for Parkinson's Disease patients while the second one deals with joints controlling of a laparoscopic robot arm in navigation tasks. In the first case study, a gait phase classification model is created using Long Short-Term Memory neural network architecture to identify the appropriate gait stimulation window. In the second, the control problem is framed as a deep reinforcement learning (RL) problem. A learning algorithm is then developed to train an RL agent to perform the desired tasks. The theories, implementations and experiment results of the two case studies in this thesis are presented with the purpose of demonstrating the feasibility of machine learning applications in the medical sector.

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1. Machine Learning Emergent

Necessity has always been said to be the mother of invention. Such notion implies that innovations are born from a specific need that the existing technology cannot satisfy. Throughout human history, there is a constant need to achieve the highest degree of efficiency. This need has driven numerous brilliant minds to produce ground-breaking and history-changing inventions. From the introduction of the round wheel to the first operational transistor, each innovation has left their marks by either helping human completing specific tasks easier or completely overhaul how such tasks are performed.

As time moves towards a more modern era, it is realized that the highest efficiency is often achieved by reducing human involvement or in extreme cases, removing human factor entirely. That is the concept of automation. With automation heavily adopted at the start of the 20th century, automation agents, in the forms of hardware and software, have raised the productivity of human to unprecedented heights. However, the traditional automated systems are rigid and constant [1]. They require the careful planning and design of humans before they are applied and even then, they can only be utilized under certain conditions. This limitation of conventional automation systems has been somewhat alleviated through advancements made in hardware. General purpose machineries have been introduced that can perform multitude of different tasks without significant modifications of structure or mechanism. A prime example of this is a robot arm with six degree-of-freedom where the end-effector can be easily changed. However, the controlling software that operates such machines is restricted by a set of instructions that has been pre-defined. This again creates a need for a truly automated system where said system can adapt and readjust itself to respond to different situations. In other words, the desired automation system is an automation system that can learn. And again, an important innovation has been born out of necessity. This time, it is machine learning.

Machine learning is by definition is the study of computer algorithms that improve itself through experience [2]. In a machine learning algorithm, a mathematical model that approximates an arbitrary process is constructed using a set of available “experiences”, called training data, with matching outcomes. The model is then used to produce the outcomes of new unseen inputs. It is the ability to generate appropriate responses to inputs without being explicitly programmed that makes machine learning algorithms desirable. The appeal of machine learning algorithms is not only hypothetical. In the recent past, actual systems have employed machine learning techniques in various problems where explicit instructions would have been difficult or outright infeasible.

In 1805, Least Squares analysis, one of the earliest and simplest algorithms that exhibits the learning capability is established [3]. To date, it is still one of the most popular method for regression analysis. In the difficult problem of facial recognition, various machine learning algorithms such as support vector machine and random forest have been applied to great success. Textual analysis is also an area where machine learning shines with Naïve Bayes algorithms being used for spam filtering and Latent Dirichlet Allocation in document topics discovery.

So far, the mentioned methods are some of what are commonly regarded as “traditional” algorithms. In 2012, a new term, “deep learning”, was coined following the introduction of AlexNet [4] – an eight-layers neural networks that opened a new frontier in machine learning. Deep learning brings about models with even greater learning capability, so great that they have the ability to achieve super-human performance. Indeed, in the 2015 ImageNet Large Scale Visual Recognition Challenge, an image-based objects classification competition, ResNet [5] surpassed the human benchmark performance. Similarly, reinforcement learning (another subset of machine learning) algorithm Deep Q-Network which employed deep neural network at its core approaches professional players performance in complex video games [6].

1.2. Applications of Machine Learning in Medical Sector

Since 2012, machine learning and its subset, deep learning, have received utmost attention from the research community as can be seen in the exponential increase of number of Scopus articles throughout the years in Figure 1.

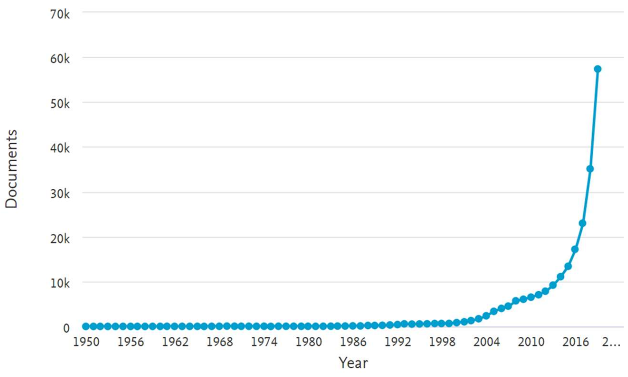

Figure 1: Number of research documents throughout the years

Academia is not the only area that is intrigued by the potential of machine learning. Various industries, both in private and public sector, have found important use cases for learning algorithms as well. In the finance sector, clustering techniques and supervised algorithms such as decision trees have been employed in managing financial risks, including credit risks, market risks and operational risks. Data-driven approaches are also being utilized in the public sector through the uses of chatbots and social media sentiments analysis. Among the various affected sectors, medicine is one of the more critical and stringent fields that requires a more cautious pace in adopting machine learning algorithms. Even so, since late 20th century, supervised algorithms such as Bayesian model and decision trees have been applied to make intelligent systems in healthcare [7], [8]. However, such applications are limited and relied heavily on domain expertise of physicians. As such, most medical advancements made nowadays are observed to have been enabled by deep learning.

While the concepts and the mathematical foundations for deep learning models have been laid out since the 1950s [9], [10], it is until recently when the hardware capabilities and the volume of training data required by such computationally intensive approaches are obtainable that deep neural networks are deemed practical and put to use. However, although deep learning approaches have not been deployed in medicine for a long time, early achievements obtained from employing deep learning models simply cannot be ignored.

Naturally, radiology, the medical discipline that uses images for diagnosis and diseases treatment, is one of the foremost use cases of deep learning. Convolutional neural network (CNN), the neural network family that the famous AlexNet belongs to, boasts the ability to automatically learns complex image features to detect, localize and classify objects of interest efficiently and proficiently, making them a great tool in radiology. Already, deep and complex network structure developed by Google has been producing impressive results in detecting breasts cancer [11]. Furthermore, CNNs operate on the pixel-level, allowing them to detect even the smallest matching pattern in images, resulting in less false negative when diagnosing [12].

Excitingly, tremendous improvements have also been made when applying deep learning models to the process of drug discovery. Most recently, a chemical compound named DSP-1181, which was created using data-driven models, is starting phase 1 clinical trial. The process of discovering such compound is noted to be less than 12 months whereas the global industry-wide average is from four to six years [13]. In addition, researchers have incorporated deep learning approaches in studying drug interactions [13], [14], compound properties [15]–[17] and new chemical structures generation [18], [19].

With the machine learning-fueled transformation of medicine still in its initial phase, there are numerous aspects of the sector that would greatly benefit from a data-driven methodology. This thesis presents the applications machine learning in two sub-areas of medicine. The first application leans towards an autonomous and remote healthcare settings with the development of a wearable device equipped with a machine learning model as the controlling mechanism. The device aims to improve Parkinson’s Disease patients gait disorders by coordinating their gait rhythm. The second deals with controlling a medical apparatus in clinical procedures. Specifically, the method of using machine learning algorithm to control the movement of a laparoscopy robot arm is described.

The rest of the thesis is organized as follows. Review of relevance existing materials is conducted in Chapter II. Following that, Chapter III details the methodology of the two case studies of the thesis. Chapter IV presents the experimentation results obtained and the discussion these results. Finally, Chapter V concludes the thesis and provide possible improvement over the materials presented in this thesis for future works.

CHAPTER 2: LITERATURE REVIEW

This chapter presents the findings of a literature survey to review existing relevant materials and works done by researchers. Since this thesis focuses on two case studies to be described in the later chapters, the scope of the review is also limited to literature that are related to these two case studies.

2.1. Gait Disorders in Parkinson's Disease Patients

Parkinson's Disease (PD) is a progressive neurodegenerative disorder which causes unintended and uncontrollable movements of human body. Patients diagnosed with PD are observed to have damaged or dead brain cells in area which produces dopamine – a chemical needed to generate smooth, willful motions [20]. While the precise causes for PD are currently unknown, its symptoms are studied intensively and have been divided into four primary classes [20]:

- Tremor - shaking that has a characteristic rhythmic back and forth motion,
- Rigidity - muscle stiffness or a resistance to movement, where muscles remain constantly tense and contracted,
- Bradykinesia - slowing of spontaneous and automatic movement that can make it difficult to perform simple tasks or rapidly perform routine movements,
- Postural instability - impaired balance and changes in posture that can increase the risk of falls.

Most cases of PD start to exhibit symptoms at the age of 60, however, early onset of PD have been recorded to be before the age of 50 as well [21]. It is estimated that there are 7 million people with PD worldwide and this number is projected to be 9 million by 2030 [22].

Due to the movement-impairing nature of PD, gait dysfunction is almost always observed in patients. Gait exhibited in PD patients, also called Parkinsonian gait [23], has the characteristics of small shuffling steps and the general slowness of movements [23]. These characteristics can be related to the decreased displacement and velocity in the initial swing phase along with the general stiffness of movements [24]. Through the gait analysis of PD patients, the stiffness seen is caused by the pelvis and thorax rotating together en bloc which differs from the normal reciprocal pattern [25]. In more severe cases, total loss of movements or hypokinesia can occurs. Such episode is termed "freezing-of-gait" or FOG [26], [27].

Freezing of gait (FOG) occurs when patients cannot move forward despite the intention to do so. This symptom in PD patients is frequently the cause of falling. FOG is common and debilitating especially in later stages of PD[28] due to its serious implications on patients physical as well as mental health [29], [30].

2.2. Gait Rehabilitation for Parkinson's Disease Patients

While there is no definitive treatment for PD, given the severe and adverse impacts it has over patients well-being, medical approaches to alleviate aspects of PD, mainly the symptoms, are attempted with varying degrees of success [20], [31]. One of the aspect of PD that is being actively targeted is the gait dysfunctionality [31], [32].

The first and often the go-to course of treatment for PD is medications. Dopaminergic therapy, or simply dopamine therapy, is the most prevalent and widely considered the most effective course of treatments against symptoms of PD [20]. Dopamine therapy aims to replenish the concentration of dopamine in PD patients. This is commonly done using a chemical compound known as Levodopa or L-DOPA [33], a precursor to the neurotransmitter dopamine. The reason why Levodopa is used instead of dopamine lies in the ability of Levodopa to cross the blood-brain barrier, a semi-permeable border which selectively allows certain compounds to enter the central nervous system in human where pure dopamine cannot cross [34], [35]. In practice, Carbidopa [36] is usually prescribed along with Levodopa to inhibit the premature metabolism of Levodopa to dopamine until the earlier has reached the brain, thus increases the effectiveness of dopaminergic therapy [20], [33]. Dopamine therapy has been proven to be effective in improving motor skills, impulsivity and decision making in PD patients [20],

[31], [33]. However, it is not a miracle drug. While Levodopa helps reduce bradykinesia and rigidity, reduction in rigidity is marginal [20]. Furthermore, dopamine overdose in certain part of the brain in dopaminergic therapy could lead to increased impulsive behaviors [37].

Another approach considered by doctors when patients are not responsive to medications is deep brain stimulation (DBS) [38]. Unlike dopamine therapy, deep brain stimulation is an invasive medical procedure that requires open surgery. The purpose of surgery is to place a small medical device called the neuro-stimulator, often referred to as the brain pacemaker, which sends electrical impulses through implanted electrodes to specific parts of the brain [39]. The electrodes which deliver the impulses are also inserted to specific parts of the brain, normally either the globus pallidus internus or the subthalamic nucleus for PD patients [39], [40], during the surgery as well. The neuro-stimulator can then be programmed to optimize symptom suppression and control side effects of DBS [41]. Though the exact working mechanism of DBS is not clear, this direction of treatment has been found and confirmed to be very positive [20], [42], [43]. Generally, patients with DBS are observed to have 30-60% improvement in motor score evaluation [20], [44]. Despite such results, DBS is not without flaws. DBS has a therapeutic window [45], meaning that it is only effective in certain phases of PD. Complications involving DBS have also been known to include surgical complications (hemorrhage and infection), electrodes migration/misplaced electrodes and curiously, swimming impairment [46], [47]. In addition, DBS does not entirely stop the development of PD and some symptoms may regress overtime [20], [48].

Aside from direct methods aiming to tackle the diseases itself, other non-invasive methods aiming to improve patients daily living and quality of life. Such non-pharmacological approaches include physiotherapy, occupational therapy, speech and language therapy and nutritional methods [49]. Among those, physiotherapy is often deployed to combat patients gait dysfunction due to PD progression [32], [50]. Physiotherapy for gait rehabilitation in PD comprises of physical training activities (treadmill walking, tai chi, therapy exercise) and assistive cueing by visual, auditory and tactility cues [32], [51]. Generally, patients participating in physical activities and trainings have improved gait function intrinsically [50], [52] and are observed to have reduced freezing-of-gait (FOG) episodes [53]. Reducing and preventing FOG episodes is especially important in preventing fallings of PD patients. Normally, gait cues are given by physiotherapist in the form of audio and visual stimulations in a therapy session [54], [55]. However, the sessional nature of physiotherapy and the shortage of qualified physiotherapists are attributes that limit physiotherapy effectiveness to short-term [53]. As such, with the advances in medical technology, automatic cueing systems have been developed that would allow for a more continuous assistance for PD patients [56].

On using visual cues, portable devices that use light as an indicator have been proposed. Visual cues usually work by projecting a line on the path of the patient with the aim of prompting where to step. The simplest realization of this idea is a type of walking stick or cane equipped with lighting called the laser-cane [57]. More modern but bigger instruments with lighting capability are walkers and stabilizers[58]. It is found that both families of equipment offer marginal benefit to PD patients during "on" state where medication is effective and little to no effects during "off" state where patients do not respond to medications [58], [59]. Auditory cues are also sometimes added along with visual cues as well.

In a more modern approach, programmable devices that are small in form factor and are wearable have been introduced to be the new generation of gait assistive devices for PD patients. Walkasins, developed by RxFunction, aims to assist patients with balancing difficulties [60]. The device employs pressure sensors placed in insoles to measures changes in foot pressure. Based on the readings of pressure sensors, the on-board micro-controller then actuates the vibrating motors, which provides instant tactile cues to users. The cues, according to developers, could be interpreted as a new balance sense that reduces the risks of patients falling down. Also measuring foot pressure, LEAFS [61] is a device that help patients training for symmetric gait by providing audible cues when symmetric gait is achieved. A recently proposed device with the name of "Laser Shoes" [62] combines the traditional line-generating laser cues with the wearable computer paradigm in assistive devices design. The laser visual cue is activated during the heel strike by a switch under the insole of the stepping foot. The developers believe this feature provides users with visual stimulation tuned to their own gait rhythm. Clinical trial of Laser Shoes reported a significant reduction in FOG episodes in both “on” (45.9%) and “off” (37.7%) state and 56.6% shorter frozen time [62]. Similar to Laser shoes, another wearable device with onboard computer have also been shown to reduce FOG duration in clinical trials [63]. However, different from Laser Shoes, this device utilizes vibratory stimulation activated by analyzing tri-axial acceleration information of users’ legs.

In all the above mentioned modern assistive devices, one common feature, and a requirement at the same time, is the ability to collect and analyze walking data of users, whether they are vertical ground reaction forces or leg accelerations. These types of data that depends on time and having data points that are related in time are called time-series. Following this paradigm of wearable applications that utilizes time-series data to provide stimulation for PD patients, the device to be proposed in section 3.1 is designed to be a smart wearable assistive device delivering continuous sensory stimulation with a time-series data-driven activation mechanism for PD patients.

2.3. Time-series Classification using Recurrent Neural Network

Neural networks in the form of feed-forward network such as multi-layer perceptron and radial basis function network have demonstrated astounding learning ability and have impeccable performances on numerous classification and regression tasks. However, such high capability of feedforward neural networks are not sustained when applying in time-series settings [64]. The drawback exists perhaps because feedforward neural networks are not designed for tasks where input data is correlated in the time axis. Furthermore, ordinary feedforward neural networks cannot operate on inputs with arbitrary length.

Naturally, with the important of time-series as one of the most common type of data encountered, there is a need to introduce a new type of neural network to harness the information of time-series. And so, designed especially to tackle time-series data, recurrent neural network [65] is developed.

Formally, recurrent neural network (RNN) is defined as a class of artificial neural networks where the neurons of one layer do not only connect to neurons in the previous and subsequent layers but also to others in the same layer. These same layer connections are arranged to form a directed graph to represent a temporal sequence. This feature is what allows RNN to exhibit temporal dynamic behaviors. The classic structure of a RNN is shown in Figure 2.

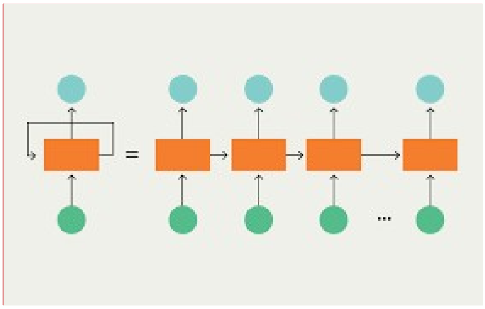

Figure 2: Recurrent Neural Network architecture (compact and unrolled form) *[66]*

In unrolled form, it can be seen that a RNN neuron accepts not only the training data as input but also the output from the previous neuron. Mathematically, operations inside the RNN can be summarized in two equations (2.3.1) and (2.3.2). The context of the first time-step is usually randomly initialized.

$$a^t = g_1(W_{aa}a^{t-1} + W_{ax}x^t + b_a) \tag{2.3.1}$$

$$y^t = g_2\left(W_{ay}a^t + b_y\right) \tag{2.3.2}$$

where:

a^t is the context vector at time step t,

$W_{i,j}$ are the weight matrices,

b_i are the bias terms.

Since their inception, RNNs have been deployed for various sequence modelling tasks such as speech and handwriting recognition to various degree of successes [67]–[70]. However, the original RNN architecture is known to have one critical flaw. Typical RNN is trained using the backpropagation through time (BPTT) algorithm – a variant of the backpropagation algorithm for normal MLP. As such, due to the repeated gradient multiplications in BPTT from the chain rule, accumulated gradients would either quickly explode when intermediate gradients are larger than 1 (exploding gradient) or quickly diminish if the intermediate gradients are smaller than 1 (vanishing gradient) [71]. Such a serious flaw hinders applications of RNN to time-series with long-term dependencies. With the exploding gradient problem, gradient clipping is often the solution, however, the vanishing gradient problem remains an open issue.

There have been several proposed solutions to address the vanishing gradient problem. Among those, one particular elegant solution that modifies the RNN architecture called Long Short-Term Memory (LSTM) [71] has received many praises and has been applied widely for time-series problem. The architecture of a LSTM unit is shown in Figure 3. The features that allow LSTM to remain numerically stable even in deep network are the introduction of the state unit and the addition operation between the state unit and the output of the LSTM cell. With these two features, gradients can flow in a more direct way between LSTM cells while delivering meaningful learning signals through the addition operation.

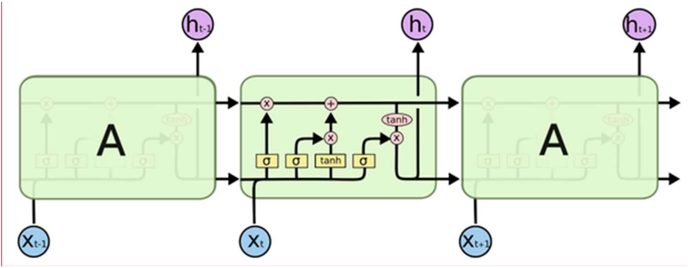

Figure 3: LSTM unit architecture [72]

Applications that utilized LSTM are numerous, among which are difficult problems such as text-to-speech [73, p.], speech recognition [74], question-answering [75] and image captioning [76]. These are all sequence modelling type of applications that benefitted significantly using LSTM. Solutions for numerical time-series problems such as ECG monitoring [77], fraud detection [78] and stock price prediction [79] also enjoyed great successes. In this thesis, LSTM neural network is applied to classify a time-series data gathered using two tri-axial accelerometers with the purpose of using the classification to control the actuation of a wearable gait stimulation device. The setting of this application is similar to the system described in [80], which belongs to the same project as this thesis.

2.4. Laparoscopic Surgery

Laparoscopy is defined as a type of minimal invasive surgery usually performed in the abdomen and pelvis [81]. The typical characteristic of laparoscopy is small incisions, usually in the dimension of 0.5 to 1.5 cm, used to insert laparoscope arms inside the operating area [82]. With such small incisions, it is required to use a camera to provide doctors and operators with field vision. Regarding the placement of the camera, laparoscope can be divided into two main categories [83]:

- Camera not on the laparoscope: external camera captures images using rod-lens system,
- Camera on the laparoscope: a small digital camera at the end of laparoscope tip.

With the latter, the rod-lens system is not required to deliver vision to operators, hence, the laparoscope tube in this case can be made from flexible and bendable material. However, this type of laparoscopes has limited maneuverability and is not user-friendly. In addition, their imaging quality is inferior to rod-lens rigid systems [84]. In contrast, rigid laparoscopes guarantee high fidelity imaging through a novel optical system enclosed in the laparoscope tube [85], [86]. This optical system replaced the small, hard to manufacture lenses in the traditional laparoscopes with glass rods, hence the name rod-lens. This innovation not only makes laparoscopes easier to manufacture but also delivers significantly better image clarity and brightness [85].

Laparoscopy has been used to diagnose and treat various medical conditions. For diagnosis, laparoscopy can be used to detect conditions such as pelvic inflammatory disease, endometriosis and some types of abdominal cancer [82]. For treatment, laparoscopy is mainly used for removing inflamed appendix, gallbladder and sections of intestine [82]. Before the patient undergoes a laparoscopy surgery, anesthesia is applied so that the patient would be unconscious for the operation. Then, the surgeon cut an initial small incision through which a tube is inserted to inflate the abdomen with inert gas, creating an operating space for the surgery. Also through this tube, a laparoscope is inserted into the just created cavity to relay images of the working area. Depending on the type of laparoscopy, subsequent incisions may be made for the insertions of other surgical instruments. The surgeon can manipulate and maneuver these instruments precisely with the vision provided by the laparoscope. The typical duration for diagnostic laparoscopy ranges from 30 to 60 minutes [82]. However, the operation may take longer if it is for the purpose of treating a condition.

Compared to traditional open surgery for treating the same type of conditions, laparoscopy has several advantages. First, the small incision size of laparoscopy reduces pain while hastens recovery [87], [88]. The reduced pain means that local anesthesia can be applied as opposed to general anesthesia needed for open surgery [89]. While the durations of laparoscopy procedures are longer, patients undergo such treatment are discharged earlier, often within the same day [82]. Second, surgical complications such as hemorrhaging are observed to occur less in laparoscopy [87], [90]. Infection risks to internal organs due to exposure are minimized as well [91].

However, there are also disadvantages to laparoscopy. First, due to the closed working space, operators have limited range of motion compared to open surgeries, resulting in a loss in dexterity [92]. Second, surgeon can only rely digital imaging of the working area. As such, other important information such as depth perception and tactile sensations are loss, making delicate operations like tying sutures difficult [92]. Third, controlling of laparoscopy surgical instruments is not entirely intuitive and requires extensive training to master [93].

A relatively recent development in laparoscopy surgery technology is the incorporation of robots to certain procedures. This is known as “robot-assisted laparoscopy” [82]. In robot-assisted laparoscopy, rather than directly operating on patients, surgeons control robot arms to execute procedures. With the robot arms as proxies, doctors can have increased control over how the procedures proceed. For instance, robot arms translate surgeons’ larger hand movements into smaller motions suitable for closed space, giving surgeons more precision in their control [94], [95]. Furthermore, hand tremor, albeit very small, can be filtered out by sophisticated torque control of robot arms [94]. Evidences supporting robot-assisted laparoscopy suggest that it has lower risk of complications than regular laparoscopy or open surgery. Currently, robot-assisted laparoscopy is used in prostate cancer treatment [96], [97], gynecological surgery [94], thoracoscopic surgery [98] and endometriosis [95], [99] among various medical conditions.

2.5. Control Learning using Reinforcement Learning

Reinforcement learning (RL) is the study of how a software agent learns to make optimal decisions in a certain environment to maximize a quantity known as the cumulative reward. Along with supervised and unsupervised learning, RL is one of the main machine learning paradigms that is actively studied today. Often times, RL is said to be the middle ground between supervised learning and unsupervised learning since while majority of the learning is done by the agents, RL agents still require some form reward as learning signal in order to optimize themselves.

Typically, a RL problem is formulated as a Markov Decision Process (MDP). As such, many of the developed algorithms in RL utilize dynamic programming techniques [100], [101]. A Markov Decision Process is formally defined as a discreet time stochastic model describing a sequence of possible events where the probability of each events depends only on the state obtained and the decision made in the previous event [102]. The MDP provides the framework to model decision making in situation where the outcomes are uncertain and do not solely depend on the decisions. MDP satisfies an important property which states that to make an optimal decision at a given state, only the observations from the current state is required and that decision is as good as the one can be made knowing all the past states [103]. This property is called the Markov Property, following Andrey Markov who first introduced the concept of Markov chain [102].

Mathematically, a MDP can be described with a 4-tuple (S, A, P_A, R_A) where:

- S is the set of all possible states called the state space,
- *A* is the set of all possible actions (assume possible actions in a state is constant for all states),
- $P_A(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that action *a* at state *s* will lead to the outcome of *s'* in the next time step,
- $R_A(s, s')$ is the immediate received upon transitioning from state *s* to state *s'* due to action *a.*

With this setting, the goal of MDP or reinforcement learning is then to devise a policy to produce actions at states such that the expected cumulative discounted reward is maximized. This discounted cumulative reward is defined in equation (2.5.1). The discount factor, positive and smaller than 1, is introduced to ensure that the infinite sum of immediate rewards converges, however, it can also be used as a variance regularization parameter for some algorithms [104].

$$G = E_{s_{t+1} \sim P_{a(s_t, s_{t+1})}} \left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \right] \tag{2.5.1}$$

In addition, there are three more important definitions that are frequently used in RL literature. At this point, it is good to note that the policy that produce the action (of agent) is usually denoted as π.

First, to quantify the "goodness" of a particular state, the state-value function is introduced. Second, the Q-function or action-value function provide an estimate of how "good" an action performed in a particular state. The last definition is the advantage function which is the difference between action-value function and the state-value function evaluated at the same state. The advantage function gives the idea of how much an action performed in a state would results in a better or worse outcome than the current estimated outcome of that state based on state-value function. The mathematical definitions of the three function are given in equation (2.5.2), (2.5.3) and (2.5.4).

$$Q^{\pi}(s,a) = E_{s_{t+1}\sim P_{a(s_t,s_{t+1})},\ a_t\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R_{a_t}(s_t,a_t,s_{t+1})\middle| s_t = s, a_t = a\right] \tag{2.5.2}$$

$$V^{\pi}(s) = E_{s_{t+1}\sim P_{a(s_t,s_{t+1})},\ a_t\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R_{a_t}(s_t,a_t,s_{t+1})\middle| s_t = s\right] \tag{2.5.3}$$

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{2.5.4}$$

Since the introduction of the RL framework, numerous algorithms and techniques have been introduced with the aim of solving RL problems correctly and efficiently. Starting with the simplest cases where the state space is small, actions are discrete and the transition dynamics of the environment are known, algorithms like policy and value iterations [100], [105] which borrow ideas from dynamic programming are formulated. Such algorithms rely on the Bellman's Principle of Optimality which states that "an optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" [106]. This principle allows for the division of a MDP into smaller sub-problems where each problem represents the optimal decision making in an arbitrary state. Conveniently, the optimality of a certain decision can be quantified using the action-value or the state-value. With such notion, it is clear that the common idea between various algorithms is to first establish the state-value/action-value of each and every states and then make decisions that will lead to the highest returns at each state. The first part of this idea is where major approaches differs from each other.

The first difference in establishing the state-value or the action-value is the way they are estimated. The first approach, termed Monte-Carlo methods [107], [108], calculate these values using reward information collected through interacting with the environment in a complete sequence of state transitions (usually called a trajectory). The calculated values can be considered to be “correct” or unbiased since they are actual results from interacting with the environment. The second approach is time-difference (TD) methods [107], [109]. In TD method, the recursive structure from the definition of state-value and action-value is used to form bootstrap estimates of these values. These estimates are then updated iteratively using rewards gained through interactions with the environment. TD methods are proven to converge under certain conditions [107], [109]. In practice, TD methods are more common and have been used in popular algorithms such as TD(0) [107], SARSA [110] and Q-learning [111]. There also exists a middle ground between these two approaches called eligibility traces which unifies the two approaches under one framework [107]. Algorithms that utilize this method include TD(λ) [107], [112] and SARSA(λ) [107], [113].

The mentioned RL algorithms store the estimated state-values and/or action-values in a table and search this table to choose the best actions for each state. This works fine in environments with reasonably small state and action space. Examples of this kind of environment would be a 2-D bounded maze or a tic-tac-toe board. However, as the size of the state space and the action space increase, the tabular methods fall short as the number of rows approaches infinity. This problem is especially evident for continuous environments. This issue is perhaps the reason why RL never gained much tractions even though it has been introduced quite early.

Recently, with the impressive demonstration of neural network as a universal function approximator, RL algorithms, utilizing such a powerful tool, have become much more successful. Neural network addresses complex environments issue handily by learning to estimate the state-value/action-value without having to memorize past interactions with the environment. The network is trained in TD fashion using bootstrap targets. The most notable instance of neural network-powered RL algorithm is perhaps the deep Q-network algorithm [6] that learned to play complex video games that normal Q-learning would not be able to. Furthermore, with the introduction of backpropagation algorithm [65] and fully differentiable programming [114], a new type of algorithms called policy gradient [115], [116] is formulated. In policy gradient algorithms, the policy is directly represented by a neural network while the RL objective is computed explicitly from this policy representation. The policy parameters are updated using the back-propagated learning signals gained by applying gradient ascent on the RL objective. Naturally, policy gradient methods also require samples acquired by interacting with the environment to perform policy training. The earliest algorithm in the policy gradient family is the REINFORCE algorithm which maximizes the expected returns by directly apply gradient ascent [117].

With neural network as the work horse, RL algorithms have enjoyed much success with some of the most influential algorithms being hybrid between policy gradient and value iteration: Actor-Critic[118], Deep Deterministic Policy Gradient (DDPG) [119], Trust-Region Policy Optimization [120], Proximal Policy Optimization [121] and Soft Actor-Critics [122].

Up to this point, the introduced algorithms are model-free, meaning that they do not concern with the dynamics of the environment. However, model-based algorithms, which use neural networks to learn the environment dynamics, are getting attention recently. Several works [123], [124] have shown that incorporating dynamics learning into algorithms can speed up the learning process.

2.6. Application of Reinforcement Learning in Laparoscopy

With the increasing popularity of robot-assisted laparoscopy and the technological advancements in robotics, it is appealing to apply learning algorithms to automate the control of laparoscopic surgical instruments. Specifically, given the recent successes in control learning of RL agents in complex environments, it is worthy to attempt to achieve the same results in a laparoscopy environment. Already, there have been several interesting works with that address this application.

In [125], a combination of RL algorithm and Probabilistic Roadmap is used to perform path planning for a surgical robot arm. The authors in [125] showed that the in a simulated environment, the agent was able to devise a collision avoidance path in dynamic situations. The RL environment created in this work is a grid environment based on the image of the operating area of the gallbladder.

Another work [126] focuses on the pre-operative phase of laparoscopy surgery. According to the author, preoperative planning for entry positioning and angle of laparoscope is crucial and improves the quality of robot-assisted laparoscopy significantly. Following that, the RL algorithm Deep Deterministic Policy Gradient (DDPG) is applied for automatic positioning of laparoscopic instruments. The authors showed that the end results validated their approach.

Applications of RL algorithms also branch into a more human-robot collaboration direction [127]. In a robot-assisted training approach, the authors in [128] propose a demonstration trajectory generator to produce sample trajectories for trainees and at the same time to provide feedbacks on their attempts. The system is trained using on both trajectories generated from a trained PPO RL agent and domain experts. Human-robot collaboration in the intraoperative phase is also attempted in [129] where a RL agent learns policies for tensioning of orthotropic gauze. In details, the agent derives a tensioning policy to hold an orthotropic gauze in place for a surgeon to cut using surgical scissor.

Following the applications of RL algorithms in the laparoscopic environment, this thesis presents a case study where an agent learns to control the movements of a laparoscope tip by directly controlling the target velocities of its motors.

CHAPTER 3: MATERIAL & METHOD

In this section, two case studies on applying machine learning algorithms to control medical devices are presented. In each study, an overview describing the high-level information is given, followed by the design and implementation details.

3.1. Gait Assistive Wearable Device

The device was developed with a number of hard design constraints. First, the device size and weight must be low enough to be qualified as wearable instrument. Second, as the intended users are PD patients, must of whom are seniors, the device must be easy to wear and operate, requiring minimal external monitoring and maintenance. Third, the device must be safe to handle and poses little to no risks to users. In addition, with the intention of delivering continuous sensory stimulation, the device should be able to withstand long operating duration and should not irritate users when wearing. However, above all else, the device must be working properly in delivering stimulation at the correct interval with the appropriate intensity.

3.1.1. Overview of Architecture

The developed device contains three main nodes: one parent node and two child nodes. The child nodes responsibility is to act as a gateway through which the parent node interacts with physical world. Specifically, a child node collects gait data from user through sensors and transfers these data back to the parent node. Then, the child node receives back instructions from the parent node on whether to activate the stimulation mechanism or not. The parent node acts as the main computing unit for the system, analyzing gait sensor data collected by the child node and deciding the activation of stimulation. The arrangement as well as the interactions between components are aptly shown in Figure 4.

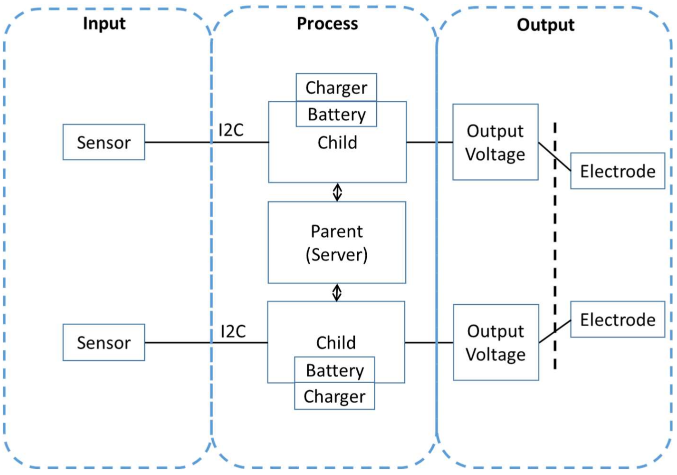

Figure 4: Overview architecture

Figure 4 also illustrates the three phases in one operation cycle of the device: input phase, processing phase and output phase. Naturally, several design decisions have to be made during the development process. These decisions are application-dependent and often not clear-cut. Thus, the option with relative highest perceived benefits with reference to the hard and soft constraint is chosen.

First, the type of gait data that would be used to monitor the user gait phase need to be specified. There are two sources of data available for a leg-wearable application: the vertical ground reaction force and the acceleration. Both of these information have been intensively studied and used in numerous researches on gait phases monitoring and recognition [130]–[132]. However, the use of VGRF in this application is unattractive due to the nature of force-sensitive resistor (FSR) – the dominant type of sensor to measure VGRF. To measure VGRF with FSR, it is necessary to place the sensor such that it is in contact with both user's foot and the surface being stepped on. This can only be achieved by either requiring user to wear a special, FSR-integrated footwear (for example socks or insoles) which would require addition effort from users and would limit their fashion choice or anchoring the FSR at the heel of user which may cause inaccurate data due to misplacement and again, requires users to spend more effort in wearing. On the other hand, accelerometers, in the form of

Inertia Measurement Units (IMU), are generally smaller and do not have strict placement requirements, making them substantially more viable than FSR.

The second design decision encountered is selecting the type of output stimulation. Conventional cues using audio or visual means have been used to assist PD patients predominantly for their simplicity and perceived effectiveness. However, several works have shown that their benefits are at best moderate [55], [133]. Recently, proprioceptive sensory such as vibration has been explored as a potential type of cue for PD patients. Some works [134]–[136] receive results suggesting that proprioceptive stimulation is a promising alternative and maybe even better than attentional stimulations like visual and audio. Thus, proprioceptive stimulation is deemed more direct and more suitable for the device cueing mechanism. One noteworthy sensory cue research direction is the usage of muscle vibration to prompt users [137]. Interestingly, through the study of muscle vibration effects on PD patients, sensory deficits are hypothesized to be related to FOG mechanism [134]. Based on this hypothesis, transcutaneous electrical nerve stimulation (TENS), a kind of electrical stimulation applied widely with therapeutic purposes, is chosen as the method to deliver stimulation [138], [139].

In term of physical structure and packaging, in this application, it is more ideal to have the number of separate entities or components limited to two. However, as presented in Figure 4, the developed device has three separate entities. This is not without reason. In the actual development process, the two-entities approach had been attempted. The overview architecture of this attempt is shown in Figure 5.

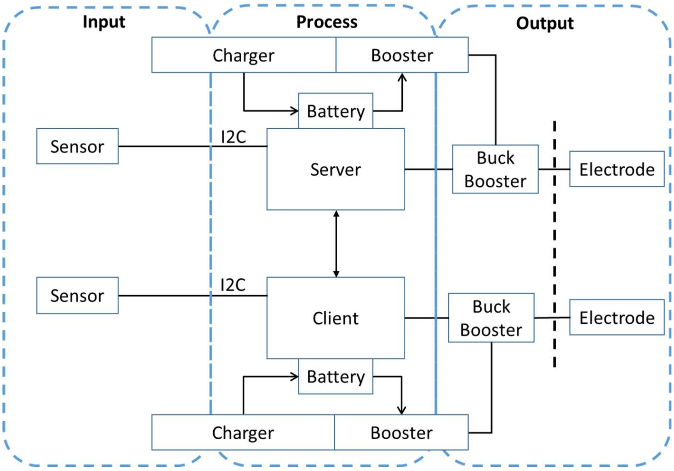

Figure 5: Alternative architecture

In this architecture, the node attached to user leg must be able to collect gait data, analyze it (by running through a neural network model to be described in the next section) and produce the stimulation. Hence, the required computational power in each node is increased substantially, mainly due to the data processing task, resulting in demand for better onboard processors. Unfortunately, with available resources, a better processor come with the cost of bigger and heavier batteries. Thus, with the eventual dimension of 13 x 7 x 3.5 cm and the weight of more than 200 grams for each node in full packaging, the architecture is not suitable to be deployed. In contrast, by moving all processing tasks to another separate node, the leg-contacting nodes size and weight can be significantly reduced to 5.5 x 5.5 x 2 cm and less than 100 grams. The main processing node, without needing to have peripherals to conduct input/output operations, is small and light enough to be carried conveniently. Therefore, the three-entities architecture is greenlighted.

3.1.2. Model Construction

At the core of the application is the data model from which the device receives the signal to activate the output stimulation. Thus, a proper problem formulation must be constructed in order to identify the appropriate solution model. The current scenario can be summarized as follows.

In this application, a system or model is required to decide when to activate the cueing system to deliver electrical stimulation to users' leg through the analysis of input gait data.

From the above, it is evident that one of the most critical part in this application is the timing of stimulation. Precisely identifying when to deliver cues is essential in maximizing the effectiveness of the device and hence should be done carefully. In order to do so, understanding of human gait is needed. Human gait is an episodic, rhythmic locomotion to propel human body forward and can be described to have six phases:

1. Heel Strike
2. Foot Flat
3. Mid-Stance
4. Heel-Off
5. Toe-Off
6. Mid-Swing

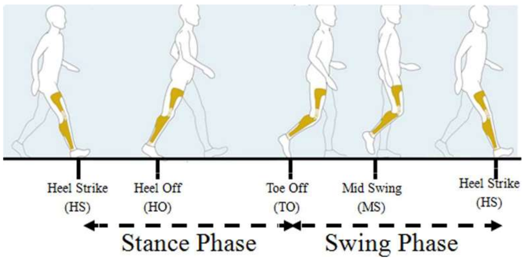

Figure 6: Human gait phases [140]

The gait phases listed occur on one leg and is illustrated in Figure 6. Among the six phases, heel strike is considered the ideal window during which should stimulation be activated. The reason for this selection is that heel strike phase of one leg marks the start of the swing phase of the other leg. Since the goal of cueing is to prompt the user to move their feet and continue the gait cycle, this window seems to be ideal to deliver stimulation. Thus, the sequence of activations for the device can be constructed and shown in Figure 7. At this point, the desired working mechanism of the model has been specified: analyze the input gait data to determine which node should be activated through the detection of heel strike. In the form of a classification problem, each input data point is classified into one of two classes where the first class (class 0) contains all the data points collected during the phases which node 1 should be activated and the second class (class 1) contains data points for which node 2 should be activated.

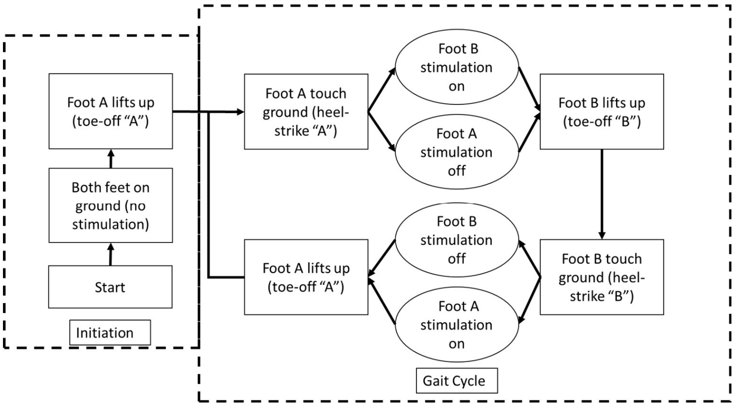
Foot A lifts up (toe-off "A")
Both feet on ground (no stimulation)
Start
Initiation
Foot A touch ground (heel-strike "A")
Foot B stimulation on
Foot A stimulation off
Foot B lifts up (toe-off "B")
Foot A lifts up (toe-off "A")
Foot B stimulation off
Foot A stimulation on
Foot B touch ground (heel-strike "B")
Gait Cycle

Figure 7: Stimulation activation cycle

Regarding to input data, since each node on user's legs contains one tri-axial accelerometer, leg acceleration information in three axes *(x, y* and *z)* can be captured. The node continuously captures acceleration data to form a sequential stream of data, or a time-series, where each data point has 3 features correspond to three axes. The two time-series, one from each leg, are then merged feature-wise to form a single time-series having six features. In addition, two synthetic features, the magnitude of total acceleration vector from two legs, are computed and added in as well. By adding in these two synthetic features, it is found that the performance of the model is increased. Thus, the final input time-series is then obtained with a total number of eight features.

To map the input data to the desired output, a specific type of recurrent neural network named Long Short-Term Memory is constructed. The model architecture consists of two layers: a LSTM layer and a fully connected layer arranged in a sequential manner. This architecture is summarized in Figure 8.

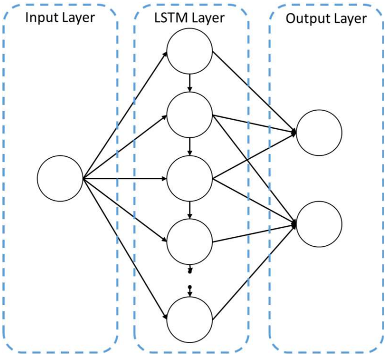

Figure 8: LSTM neural network architecture

The fully connected layer in this case is also the output layer and has the output dimension of two, same as the number of classes. In essence, the output vector contains the score reflecting how likely a data point is classified into a particular class by the network. The raw output score is in the form of floating point number and can even be negative. This score is then scaled and transformed into probability estimation using the softmax function shown in equation 3.1.1. Thus, for a given data point, the class for which that point has the higher score or probability of belonging to is chosen.

$$s_j = \frac{e^{o_j}}{\sum_{k=1}^{C} e^{o_k}}, \qquad for\ j = 1,2,\dots,C \tag{3.1.1}$$

$where\ o_j\ is\ the\ raw\ ouput\ score\ for\ class\ j.$

Regarding the input data, for normal feedforward neural network, the input is a *d*-dimension vector where *d* is the number of features each data point has. However, RNNs, and LSTMs specifically, are designed specifically for sequential data and process said data in multiple time steps. Thus, the input data point to be fed to the model in this case is actually a block of eight-feature data points arranged temporally. Concisely, an input data point is a (*t-by-d*) matrix where *t* is the number of time steps to be included and *d* is the number of features of one data point. At this point, a hyper-parameter to be decided, the number of time steps *t*, have been encountered.

Hyper-parameters are crucial in building neural networks. Understood to be any pre-defined network parameters that is not changed during training, hyper-parameters form the shape the model (units, layers) and its behaviors (learning rate, momentum). In a way, the abundance of hyper-parameters in the its design represents the adaptability and the expressive power of neural networks. Since hyper-parameters effects on the learning and behavior of neural networks are not clearly understood, hyper-parameters tuning is usually required to find the optimal network architecture and performance. However, as the hyper-parameters search space is immense and most hyper-parameters recommendations are based on empirical results, the most optimal values are not always achievable. In designing the network architecture, the hyper-parameters that needs deciding are the number of LSTM layers and the number of units in each layer. Hyper-parameters are also needed in training the network. For training phase, learning rate and input batch-size are the most important hyper-parameters that directly affecting the training efficiency of the network. Overall, hyper-parameters for the developed neural network is summarized in Table I.

Table I: Hyper-parameter values

Hyper-parameters	**Value**
Time-step size	40
Number of LSTM units	54
Learning rate	0.001
Input batch	40

Visually inspecting the input data, the number of LSTM layer is set to one first to gauge the capability of the network. Later on, it is found that using only one layer is sufficient for this problem. The number of units is harder to choose as there are no concrete guidelines in such area. Thus, a grid search strategy where the number of units starts at ten and is slowly incremented until a good model performance is observed. By doing so, it is found that 54 is a good choice for number of units. Figure 9 shows the performance of the model with different number of hidden units.

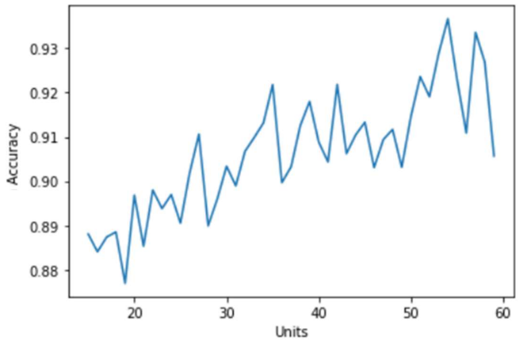

Figure 9: Validation accuracy with different number of units

Input step size or the look-back window is empirically chosen to be 40 due to the finding that a heel strike occurred in the span of 30 to 37 data points. To support the training procedure, batches of training data is supplied to the network. This method called mini-batch training has demonstrated to bring a number of benefits such as better convergence rate, lower risk of encountering local minima and numerical stability. Empirical results [141, p.] recommend that the number of samples per batch is kept low at 32 to 64. Thus, 40 is chosen in this application. Learning rate is kept at the default of 0.001 but would be decreased if training stagnates. With this, the LSTM neural network architecture is finalized. Training and performance details of the model are covered in Section IV.

3.1.3. Deployment System

After having obtained the data model to control the activation of the stimulation mechanism, the next step is to actually build the physical system that reflect the architecture devised in Figure 4.

Following the proposed architecture, there are three nodes which correspond to three separate packages. Among them, there are two two-ways communication channels between each child node and the parent node. The communication channels are in the model of client-server network connection where the server is the parent node and the children are the clients. The connections are implemented as Wi-Fi connections with Transmission Control Protocol (TCP) as the internet protocol. The connections serve to transfer gait data collected from child node to parent node and the instruction from parent node to child node. The parent node is responsible for providing the Wi-Fi access point through which the child nodes can use to connect to the parent node. Thus, all onboard microcontrollers and micro-computer must have Wi-Fi capability.

Aside from the main controller, each node also has various other electrical components to fulfill all required functionalities such as battery, battery charger and boosters. The entire components lists for parent node and child node are shown in Table II and Table III respectively.

Table II: Parent node components

#	Component
1	ASUS Tinker Board S
2	4000mAh Li-Ion battery
3	Tactile button switch (SPST)
4	Adafruit PowerBoost 1000 Charger

Table III: Child node components

#	Component
1	Adafruit FeatherHUZZAH ESP8266
2	Pololu MinIMU-9 v5
3	1200mAh Li-Ion Battery
4	SgBotic SPDT Slide Switch
5	Adafruit MiniBoost 5V, 100mA Charge Pump
6	XL6009 DC-DC Buck Booster 5V->40V

The device housing is designed according to the electrical components required. Overall, the dimensions of each node of the device is kept as low as possible while keeping the structural integrity uncompromised. Specifically, for child nodes, extra attention is paid to minimizing the depth so as to make the node less bulky and obstructive to users. As all parts are to be 3D printed, they are also designed to have as few over-hanging details as possible. Figure 10 and Figure 11 illustrate the 3D model constructed for parent node and child node.

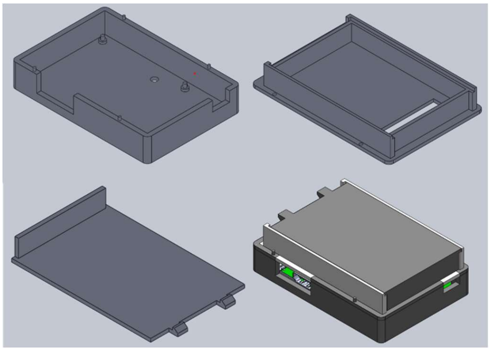

Figure 10: 3D model of parent node

The parent node is designed to operate in tandem with the child nodes as it will wait indefinitely for the child connections. Thus, it should be kept alongside user to maintain the connection with child nodes. Conveniently, the casing of the parent node is small enough to be kept in pockets or handbags.

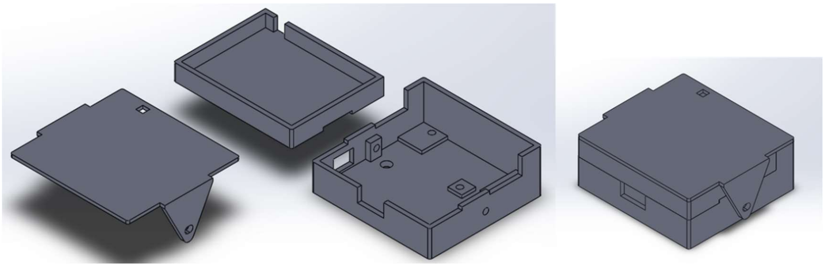

Figure 11: 3D model of child node

After fabrication, the outside dimensions for parent node are 13 x 7 x 3.5 cm and for child node are 5.5 x 5.5 x 2 cm with the wall thickness of 2.5mm and 1.5mm respectively.

The TENS system incorporated in the child node consists of a boost converter and a conductive gel-pad. The boost converter, placed inside the housing, receives a 5V input current and outputs a current with adjustable boosted voltage ranging from 5 to 40V. This voltage is used as the electrical stimulation to user and is regulated to operate in pulses with the frequency of 50Hz by the node controller. The output voltage is then connected to the conductive gel-pad through two snap buttons on the back of the housing. While the gel-pad is adhesive, Velcro straps are added onto it to reinforce the contact and to station the entire child node on user's leg. Figure 12 (adapted from [80]) shows the placement of the device on user's leg and the orientation of the IMU inside the housing.

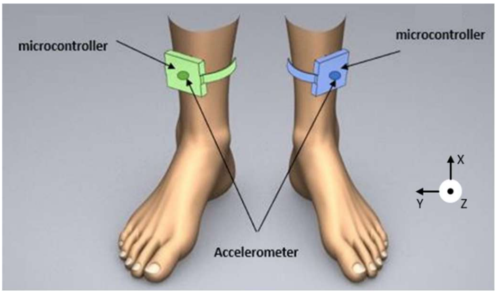

Figure 12: Device position and orientation of IMU

This position is believed to be the most non-obstructive to users' activities as well as users' clothing. In the case that it is user's desire to change the position of the child node, the child node can be moved to other location relatively easy as the Velcro straps are adjustable. However, further calibrations are needed to either transform the IMU orientation to the original orientation or to retrain the model entirely. Figure 13 shows the complete child node along with the gel-pad and the parent node. There is a small hole on the top of the casing that is aligned to the potentiometer which can be used to adjust the output voltage. This knob can be turned using a small flat-blade screwdriver inserted through the small hole though it is not recommended for users to do so on their own as multi-meter is required to measure the output voltage.

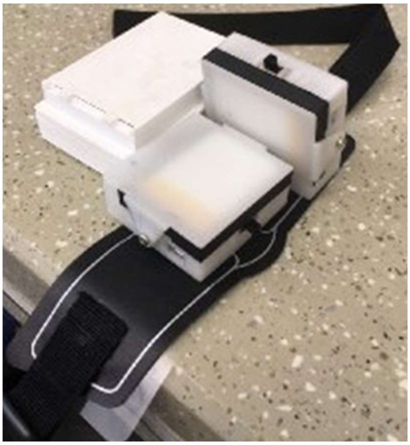

Figure 13: Completed device

3.2. Training Reinforcement Learning Agent in Laparoscopic Surgery Simulation

3.2.1. V-REP Laparoscopic Environment

The Virtual Robot Experimentation Platform or V-REP in short is a 3D robot simulation software developed by Coppelia Robotics. The software, available for free in education edition, features advanced physics engines that allow for complex dynamics calculations and real-world physics simulations. On its own, V-REP is user-friendly with intuitive interfaces while also boasts highly customizable simulator and simulation using built-in Lua and C programming interfaces. Users can also interact with V-REP environment and control V-REP behavior using remote programming interfaces in Python, MATLAB and Octave. The ability to simulate precise mechanics of complex models at lightning speed and native support for Python API makes V-REP an understandably attractive choice for developing and testing reinforcement learning algorithms.

For the purpose of training an agent to control a laparoscope robot arm, a virtual testing environment was developed in V-REP. By default, a virtual environment in V-REP is an open space with a size-adjustable flat board which serves as the ground for robot models to stand on. Models or objects that are present in a V-REP environment will have their kinematics properties such as position, orientation, velocity tracked and if enabled, dynamics properties such as torque, force experienced are monitored as well. Figure 14 illustrates a default V-REP environment. The laparoscopy virtual environment is a default V-REP environment with a 3D model laparoscope imported. Since the laparoscope model is highly complex, it is better to create this in a specialized software like SolidWorks and the export the model in the STL file format supported by V-REP. The imported laparoscope 3D model in V-REP is shown in Figure 15.

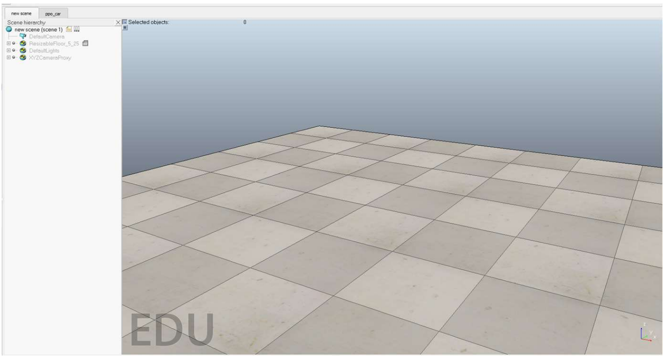

Figure 14: Base V-REP environment

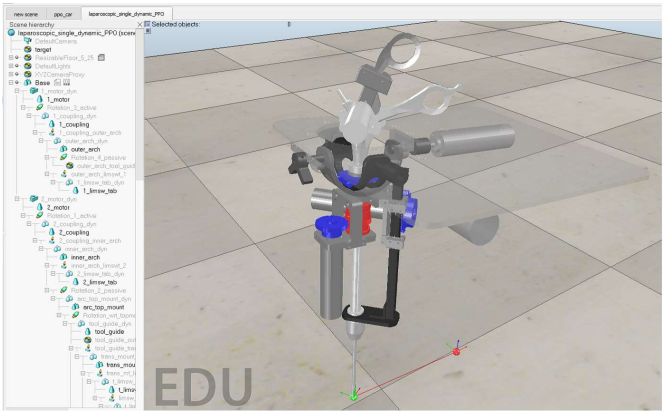

Figure 15: V-REP laparoscopic environment

The laparoscope model in V-REP is not static. As a robotic experimentation platform, V-REP is able to fully simulate the movements of model components and the mechanics that caused such motions through V-REP "joint" objects. These "joint" objects can act as a fixed link in passive mode or can be controlled directly when in force/torque mode. In force/torque mode, these joints can be operated by setting the target velocity and the maximum torque with the following behavior: the maximum torque is applied to the joint and accelerates the joint until the target velocity is reached. With this, the joint and the attached link will move either translationally or rotationally depending on the joint type. Hence, control of a model is achieved through controlling the joints of the model. This is also the goal for this project: to train a reinforcement learning agent to control the laparoscope model by setting the target velocities of certain joints to guide the model tip from the initial position to a specific goal location.

Specifically, the joints that the agent will have to control are the three joints that correspond to the three motions of the robot arm: roll, pitch and insertion. These three joints ensure that any point inside the workspace, defined by the joints movement limits, are reachable. In details, the action vector to be produced is a three-dimension vector where each element can be negative or positive. Physically, these three vector elements decide how each joint operates:

- The magnitude is the target velocity of the joint,
- The sign (negative or positive) determines the direction of the rotation or the translation of the joint.

On the first point, to control a joint in V-REP, both torque and the target velocity are needed. The reason is that when a target velocity is specified, the joint will accelerate using the provided torque until that target velocity is reached. Thus, if the either one is zero, the joint will not move. In this application, the target velocities of the controlled joints are outputted by the RL agent and the torque of these joints are set to be high (200 Nm) to ensure the responsiveness of the V-REP model. On the second point, the direction of movements of each joint is with respect to the joint own reference frame. In short, this environment has a continuous action space with the dimension of three.

Observations or input states are required for the agent to produce output actions. Generally, observations in a virtual environment would correspond to a fully observable MDP where the agent has access to all available information in a particular state. This is the same for the laparoscopy environment. The agent receives the 3D coordinate of the target and the tip of the robot arm as well as the current target velocity of each motor. This setting is reasonable even in real environment where such knowledge can easily be obtained and provide to the agent. Hence for this environment, the input is a 9-dimension vector.

The final component of the laparoscopy environment is the reward function. The reward function has a huge effect the behavior of the agent and should reflect the goal of the environment. Hence, how the rewarding system operates should be designed carefully. Since the end goal of the training is for the end effector to reach the desired target, the agent should gain a positive reward if it moves the tip closer to the target. Vice versa, a negative reward should be imposed should the agent moves the tip further from the target. In addition, the reward received for moving should be scaled according to the current tip position. That is, the positive reward magnitude is bigger the closer the tip is to the target. Finally, the agent will receive a huge positive reward if it manages to reach the target. Since there is no early termination state in the laparoscopy environment, an episode only ends when the maximum number of steps has been reached. Altogether, the reward function is laid out in equation (3.1.2).

$$r_t = \begin{cases} 0.05 + 0.1e^{-d_t} & if\ d_t \leq d_{t-1} \\ -0.05 & if\ d_t > d_{t-1} \\ 200 & if\ d_t \leq 0.015 \end{cases} \tag{3.1.2}$$

where d_t is the distance between the tip and the target at time step t.

After each trajectory, the goal position is changed randomly and the tip position is reverted back to the initial coordinates.

3.2.2. Learning Algorithm

With the action space being continuous, policy gradients algorithms stand out as more attractive than state-value or action-value algorithms. In practice, however, advanced policy gradient methods are a form of hybrid as they also approximate either state-value function as in TRPO and PPO or action-value function in DDPG. Given the good performance record of PPO while being straightforward to implement, this algorithm has become quite popular among deep reinforcement learning practitioners. In this project, a modified PPO algorithm that learn the state transition dynamics is proposed.

Proximal Policy Optimization (PPO) is introduced in 2017 and is in fact based on another popular direct policy gradient method: Trust-region policy optimization (TRPO). Traditional policy gradient algorithms typically suffer from two main drawbacks: high variance gradient estimation and destructive policy update. TRPO, and PPO, try to mitigate both problems by replacing the normal policy gradient objective function. The objective function for TRPO is laid out in equation (3.2.1) as follows:

$$\max_{\theta} \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)} \hat{A}_t \right] \tag{3.2.1}$$

$$subject\ to\ \mathbb{E}_t[KL[\pi_{\theta old}(.|s_t), \pi_\theta(.|s_t)]] \leq \delta$$

$where$:

$\delta\ is\ the\ policy\ constraint\ parameter,$

$\pi_\theta(.|s_t)\ is\ the\ policy\ with\ paramters\ \theta,$

$KL[.|.]\ is\ the\ Kullback - Leibler\ Divergence,$

$\hat{A}_t\ is\ the\ advantage\ function.$

The TRPO objective presents a constrained optimization problem that can be solved efficiently by employing dual gradient ascent and conjugate gradient algorithms. The TRPO mitigate the high variance of normal gradient by following the path of Actor-Critic algorithms: introducing a critic network. Specifically, instead of simply using Monte-Carlo returns from collected trajectory, the advantage function used in the objective describe how better, based on action-value, an action taken in a particular state compared to the current state-value of that state. This method helped reducing the variance of advantage estimation while keeping it unbiased. The second, more significant contribution of TRPO lies in the theoretical guarantee of the monotonic improvement of policy under certain conditions. This is where the constraint of the objective comes into play. The author of TRPO proved that by limiting the deviation in the distributional sense of the new policy with respect to the old policy, destructive policy updates by taking too large a gradient step can be avoided. This is the meaning of the Kullback-Leibler divergence constraint.

The theoretical support of TRPO has made it an appealing and commonly applied policy gradient method. However, implementing TRPO is not entirely straight forward, having to configure complex second order optimization algorithm such as conjugate gradient. It is this drawback of TRPO, which some may consider to be minor, that PPO aims to address. Rather than imposing a separate condition, PPO incorporate the update step size limit into the objective function (equation (3.2.2)). The PPO objective keeps the new policy close to the old one by first employing a clipping function. This clipping function limits the range of the probability ratio between new policy and old policy, which is an indicator of the difference between two policies, to $[1-\epsilon; 1+\epsilon]$. Then, the min operator turns the objective into a lower bound of the original TRPO objective. With this scheme, the hyper-parameter ϵ controls how much the new policy is allowed to deviate from the old policy. However, ϵ must be chosen carefully as a too small ϵ would cause the agent to improve slowly or not improve at all while a large ϵ would diminish the benefits of the clipping function.

$$\theta^* = \max_\theta \mathbb{E}_t \left[min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)} \hat{A}_t, clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right) \right] \tag{3.2.2}$$

In addition to the clipped objective, PPO also utilizes a novel way of estimating advantages compared to TRPO. The method is called Generalized Advantage Estimation (GAE) and is introduced in 2017 [104]. GAE aims to strike a balance between a biased but low variance one-step returns estimation with an unbiased but high variance n-step returns estimation. This is achieved by interpolating these two estimations. The final estimate is a weighted sum of all k-step advantage estimations with k ranging from 1 to infinity. All in all, PPO has made a splash in the research community with its achievements: ranging from human-level dexterity in solving the Rubik cube to beating world champions in a highly complex video game [142], [143]. Based on the foundation of PPO, this thesis now proposes two modifications/additions to the original algorithm.

First, aside from approximating the state-value by a neural network, another neural network is introduced to estimate the action-value of the agent's actions. The motivation behind this addition is to acquire a better advantage estimation than simply using state-value estimator alone. In the original PPO algorithm, the advantage function depends only on the value function. This, however, may not be sample efficient. As the value function is trained using only on-policy observations, it is constantly influenced by new incoming data and may forget knowledge gained from previous iterations. As such, while convergence is possible using only value function, this approach is slow and sample inefficient.

Hence, another separate neural network to approximate the action-value of the agent's actions is proposed to address this issue. Similar to Q-learning style algorithms (DQN, DDPG), an action-value estimator is trained to approximate the action-value of an action. This action-value is then used along with the state-value to approximate the advantage using GAE algorithm.

Formally, by using the original GAE algorithm, the estimated advantage at time step *t*, $\hat{A}_t^{GAE}$, in a given trajectory is calculated in equation (3.2.3).

$$\hat{A}_t^{GAE} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V \tag{3.2.3}$$

$where$:

$\gamma\ is\ the\ discount\ factor$,

$\lambda\ is\ the\ GAE\ weight\ factor$,

$\delta_t^V = r_t + \gamma V_{s_{t+1}} - V_{s_t}$ is the temporal difference (TD) residual of value function estimator *V*.

Here instead of using δ_t^V, the proposed method uses both estimators Q and V in the calculation. That is:

$$\delta_t^A = Q_{s_t} - V_{s_t} \tag{3.2.4}$$

This is a valid replacement as the resulting estimation is the definition of the advantage function and would provide unbiased estimation for the true advantage function, A^π, if the estimators approaches their true values: $Q_{s_t} \approx Q^\pi$ and $V_{s_t} \approx V^\pi$. Hence, the modified estimation for the advantage function is presented in equation (3.2.5).

$$\hat{A}'_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^A \tag{3.2.5}$$

Since this Q-function also trains on data from other policies (most likely past policies), it is able to retains more information than a value function V which only depends on the state. Generally, the Q-function, Q_ϕ, is trained by utilizing the following recursive relation of action-value:

$$Q^\pi(s,a) = r + \gamma Q^\pi(s',a') \tag{3.2.6}$$

where s', a' denote the state and action at the next time step.

Thus, the Q-function can be trained to approximate the true action-value by minimizing the so-called Bellman residual as shown in equation (3.2.7) in the SARSA(0) algorithm.

$$\delta = Q_\phi(s,a) - \big(r + \gamma Q_\phi(s',a')\big) \tag{3.2.7}$$

However, directly using this training scheme while using non-linear function approximator is proven to diverge [6], [144]. Hence, algorithmic tricks applied in Q-learning style algorithms are also utilized here. First, a replay buffer $\mathcal{D}$ is created to store the state transitions information. Data that are used to train the Q-function are randomly sampled from this buffer. Once the buffer reached the limit, data are discarded randomly. Second, to avoid the moving target problem encountered by using only one Q-function, a target Q-function, $Q_{\phi_{targ}}$ is used to compute the target values for the supervised training of the main Q-function.

Mathematically, Q_ϕ is trained by minimizing the following loss function:

$$\mathcal{L}(\phi, \mathcal{D}) = \frac{1}{N}\sum_{i=0}^{N}\left(Q_\phi(s_i, a_i) - y(r_i, s_i', d_i)\right)^2 \tag{3.2.8}$$

$where$:

$N\ is\ the\ number\ of\ samples\ drawn\ from\ \mathcal{D},$

$y(.)\ is\ the\ regession\ target\ calculated\ by\ equation$ (3.2.9).

With the target values calculated in equation (3.2.9):

$$y(r_i, s_i', d_i) = r_i + \gamma(1 - d_i)Q_{\phi_{targ}}(s_i', \tilde{a}') \tag{3.2.9}$$

$where$:

$d_i\ indicates\ whether\ or\ not\ the\ state\ is\ terminal,$

$\tilde{a}'\ is\ the\ action\ for\ the\ state\ sampled\ from\ the\ lastest\ policy.$

It is worthy to note that this is different than other Q-learning style algorithms except SAC due to the fact that the target Q-values is computed directly using actions sampled from the current policy and not from the replay buffer. This is to ensure that the estimated Q values is “on-policy” and is a valid estimation of the true action-value. This is crucial as in order to estimate the advantage using equation (3.2.5), both estimated Q-values and V-values must be on-policy.

Finally, after each training iteration of the main Q-function, the target Q-function parameters are updated using Polyak averaging (equation (3.2.10)).

$$\phi_{targ}' = (1 - \rho)\phi + \rho\phi_{targ} \tag{3.2.10}$$

$where\ \rho\ is\ the\ polyak\ parameter.$

The second major modification is the addition of state transition dynamics learning. This addition effectively turns the proposed algorithm into a model-based RL algorithm. With the premise that model-based algorithms are more sample-efficient than model-free counterparts [123], [124], it is worthy to incorporate dynamics learning to the proposed algorithm.

In learning the dynamics of the environment, a neural network is trained to approximate the next state given the current state and the action taken. Thus, the loss function for the dynamic model is the mean squared error of the predicted state and the true observed state (equation (3.2.11)).

$$\mathcal{L}_{dyn} = \frac{1}{N}\sum_{i=1}^{N}(\mathcal{T}_{\omega}(s_i, a_i) - s_i')^2 \tag{3.2.11}$$

$where$:

$\mathcal{T}_{\omega}(.)\ is\ the\ dynamics\ model\ to\ be\ trained\ with\ parameters\ \omega,$

$N\ is\ the\ number\ of\ sampled\ data\ from\ \mathcal{D}.$

The dynamics model is utilized to create “what-if” scenarios during each trajectory. That is, for each trajectory, a number of states encountered is randomly selected to be seed states. From these seed states, along with actions sampled from the policy, the subsequent states are predicted using the dynamics model. Then, the predicted states become the seed states for the next time step. This procedure continues on for a number of predefined time steps or until the terminal state is reached. The data collected form these “imaginary trajectories” are used, along with data from real trajectories, to train the Q-function, V-function and the policy.

All in all, the proposed algorithm is presented in pseudocode in Table IV.

Table IV: Proposed algorithm

```
Algorithm 1: Q-PPO
1:  Input:
      - Initial policy parameters $\theta_0$
      - Initial value function parameters $\eta_0$
      - Initial Q-function parameters $\phi_0$
      - Initial dynamic model parameters $\omega_0$
2:  Set target Q-function parameters $\phi_{targ} \leftarrow \phi_0$
3:  Create empty repay buffer $\mathcal{D}$.
4:  for k = 1, 2, 3, … do
5:      Collect set of trajectories $\{\tau_k\}$ by running policy $\pi_{\theta_k}$ in environment.
6:      Generate imaginary rollouts using dynamic model $\mathcal{T}_{\omega_k}$ and add them to $\{\tau_k\}$.
7:      Add all tuples $(s, a, r, s')$ from $\{\tau_k\}$ to $\mathcal{D}$.
8:      Compute $\hat{A}'$ for all collected states $s$ using equation (3.2.5) with $Q_{\phi_k}$ and $V_{\eta_k}$.
9:      Update policy parameter $\theta_k$ by minimizing loss function defined in (3.2.2) via stochastic
        gradient ascent.
10:     Update value function parameters $\eta_k$ by minimizing the loss:
            $$\mathcal{L}(\eta, \tau_k) = \frac{1}{N}\sum_{i=1}^{N}\left(V_\eta(s_i) - \left(r_i + \gamma V_\eta(s_i')\right)\right)^2$$
11:     Sample training set $\{\mathcal{R}_k\}$ from $\mathcal{D}$.
12:     Update Q-function parameters $\phi_k$ by minimizing equation (3.2.8) using target values
        computed in equation (3.2.9) and samples from $\{\mathcal{R}_k\}$.
13:     Update $\phi_{targ}$ using equation (3.2.10).
14:     Update dynamic model parameters $\omega_k$ by minimizing equation (3.2.11) using samples
        from $\{\mathcal{R}_k\}$.
15: end for
```

3.2.3. Implementation Details

This section contains the details regarding the actual implementation of the proposed algorithm in section 3.2.2.

It is obvious that the number of hyper-parameters in most reinforcement learning algorithms, and the proposed algorithm in specific, is large. Such algorithms, in addition to having their own set of hyper-parameters, rely heavily on neural network for various function approximation tasks, which requires even more hyper-parameters. Hence, the hyper-parameters used in this project are based more on empirical experimentations and may not be optimal. As such, these hyper-parameters, especially for policy model architecture, have much rooms for tunings and optimizing. The hyper-parameters and their values is listed in Table V.

Table V: Hyper-parameters for proposed algorithm

Hyper-parameters	**Value**
Policy network	Dense(256) - Dense(256) - Dense(256) - Hyperbolic Tangent
Value network	Dense(256) - Dense(256) - Dense(256) - Linear
Q-network	Dense(128) - Dense(128) - Dense(128) - Linear
Dynamic model	Dense(128) - Dense(128) - Dense(128) - Linear
Policy learning rate	0.001 then decrease linearly with iterations
Value function learning rate	Same as policy network
Dynamic model learning rate	0.001
Q-function learning rate	0.01
Discount factor γ	0.99
GAE factor λ	0.95
Clip ratio	0.1 then decrease linearly with iterations
Polyak factor ρ	0.995
Number of trajectory per episode	5
Maximum interactions per trajectory	1400
Number imaginary partial trajectory	20
Maximum interactions per imaginary trajectory	8
Number of training iteration	1000
Policy training epochs per iteration	30
Value function training epochs per iteration	30
Dynamic model training epochs per iteration	15
Q-function training epochs per iteration	5

In the process of implementing the algorithm proposed in section 3.2.2, there are some practical details that are worth mentioning. First, the output generated by the policy during training is not directly used to interact with the environment. Rather, the output vector from the last layer is used as the mean vector of a multivariate Gaussian distribution with identity covariance matrix. Then, the actual actions are sampled from this parameterized distribution. Second, aside from the clipping function, the deviation of the new policy to the current policy is also limited by using early stopping. That is, if the parameterized actions distribution of the new policy differs from the old one above a certain threshold, the policy update process is stopped. The distributional difference is calculated using Kullback-Leibler divergence. The threshold is first set at 100 and then is decreased linearly throughout training.

The data collection phase in each episode is completed by multiple workers in copies of the environment simultaneously. With this structure, the term "episode" is defined as the period for all the workers to complete their assigned number of trajectories. In essence, this allows for more samples to be collected in the same timeframe and as each environment instance is different, more exploration can be carried out. After all workers have finished their rollouts, the policy will be updated using all collected samples for a number of epoch through mini-batch gradient ascent. Then, a new episode is started and the iteration continues until the maximum number of episodes is reached. This strategy is summarized in Figure 16 and is implemented using Python and V-REP.

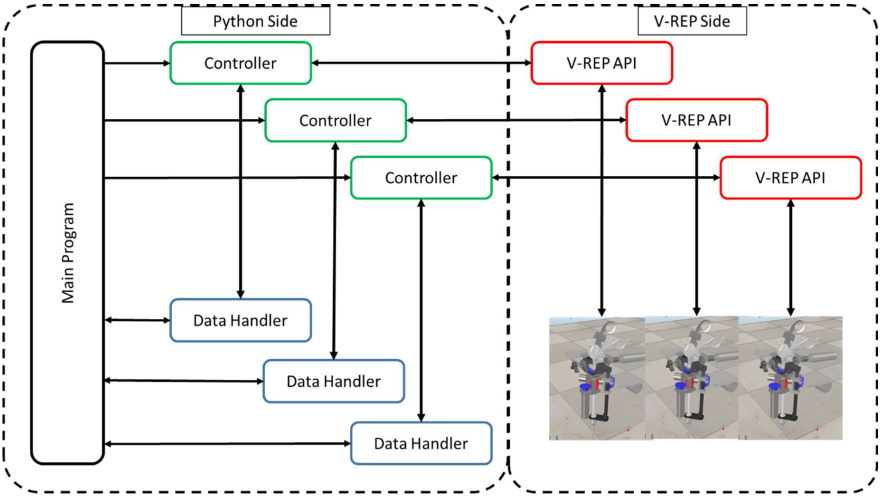

Figure 16: Synchronous training architecture (arrow indicates data flow)

In details, V-REP allows for the creation of multiple simultaneous instances of the environment with each instance is a separate Windows process. After the instances are created in the V-REP side, these instances can be controlled and interacted with from Python using V-REP remote API. V-REP API follows the client-server communication model with V-REP accepting a command from Python and responding back with environmental information. To leverage multiple instances of V-REP environments, several Python processes are spawned to control these V-REP environments independently. These Python sub-processes relay the current state of their environments and then query the policy network for actions to control their respective environments. Fortunately, the graph structure that implements the policy network in Python is thread-safe and thus, multiple threads are created from the main Python process to cater for the need of simultaneous action querying from multiple requests. These threads also serve the purpose of storing necessary data after each interactions with the environment. After each completed trajectory, these child-threads then transfer the data back to the main thread for training procedures. This architecture is believed to have fully parallelized the entirely training process and as such, significantly reduces the agent training time.

CHAPTER 4: RESULTS & DISCUSSION

4.1. Gait Assistive Wearable Device

4.1.1. Performance of Gait Rehabilitation Device Model

Model constructed in section 3.1.2 is trained using a self-collected dataset. The data collection system has similar structure to the deployment system with the differences are the added FSR sensors to generate training labels. The data collection module is shown in Figure 17.

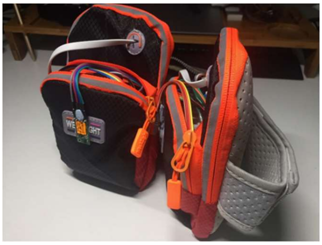

Figure 17: Data collection module (without FSR)

The data was collected in a normal walking setting. A data collector walked on a flat surface with normal pace for 10 times each with length of 12.5 meters. Since the platform was not long enough, each walk consists of one U-turn event. The total number of data points collected was 27638. In addition, a separate test dataset with the total number samples of 3500 was also collected for the purpose of model testing. The only data preprocessing step performed was rescaling data to fit inside the range of [-1;1]. A sample set of the collected data is shown in Figure 18.

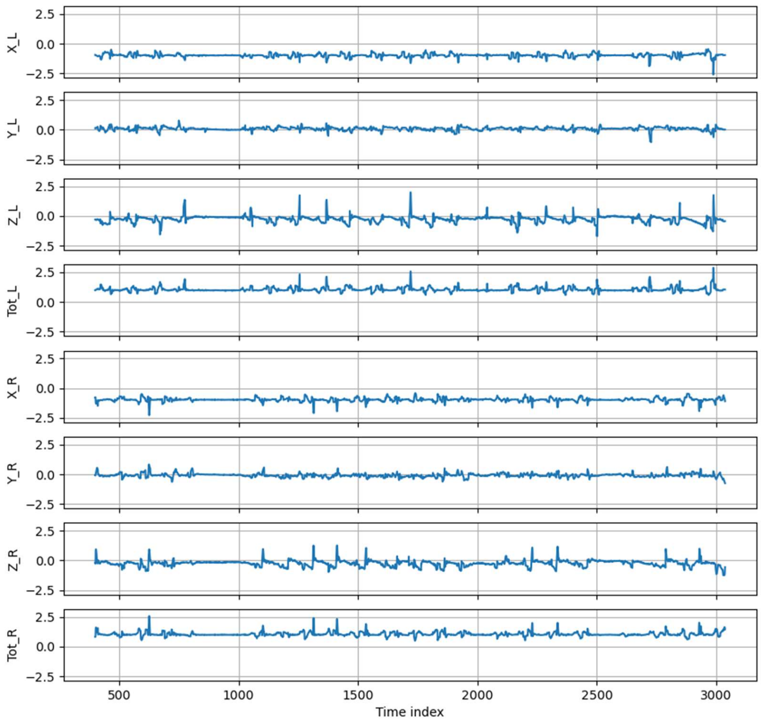

Figure 18: Unscaled acceleration data in time-series format

The training set was divided into an actual training set and a validation set with the ratio of 8:2. The validation set is used to detect whether model is overfitting to the training data. In addition, the accuracy of the model on the validation set controls the model learning rate: if the validation accuracy does not increase for a period of time, the learning rate is decreased by a factor of 5 until a minimum learning rate of 0.0001 is reached. The training progress curve is depicted in Figure 19.

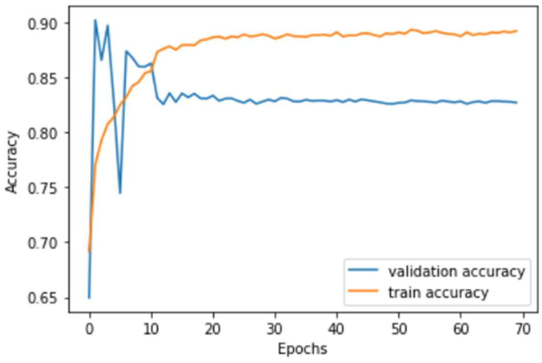

Figure 19: Training and validation curve

After the model have been trained, its performance is measured using the test set. The final performance of the model on the test set is **0.8454**. The confusion matrix of the model predictions is shown in Figure 20.

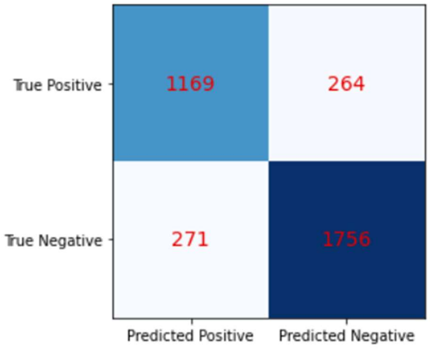

Figure 20: Confusion matrix

The constructed model is packaged into a HDF5 file and then transferred to the micro-computer for inference of unseen data. One of the key criterion of the device is the ability to produce real-time stimulation. As such, it is imperative that model inference, as the most computational intensive process, is fast enough for real-time classification. This is also the main reason that the TinkerBoard was chosen as the main computing unit. With the ability to classify around 74 data points per second depending on the current workload and condition of the processor, it is safe to say that the real-time capability has been achieved.

4.1.2. Analysis of Gait Model Performance

The model performance on the test set at 84.54% signifies that other techniques can and should be applied to improve the accuracy of the model. However, as this is a time-series problem, there are other aspects worth analyzing as well.

First, given Figure 21, it can be argued that the model would still be able to perform the desired task to a high degree of success.

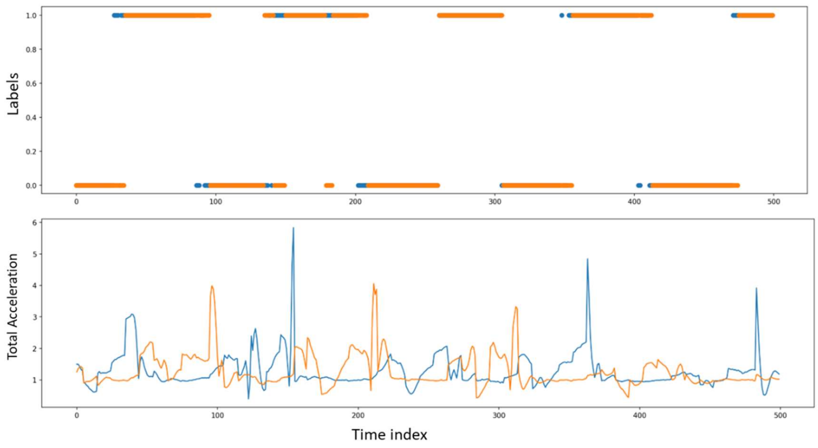

Figure 21: Sample predictions with corresponding total acceleration data (true label is in orange)(1)

The figure shows some misclassified data points along with the true labels in a temporal manner. The erroneous predictions are sparse and scattered between correct predictions. This odd-one-out type of errors would not undermine the function of the device. Indeed, with the average data point processing time of 0.0135 second, it is likely that the user would not be able to feel the effects of a few misclassified data points. What is more important for the device functionality though, is the ability to detect heel-strike events as these events signify the right window for the stimulation. In this aspect, it is safe to suggest that the current model is quite adequate. Although there is a delay of a few data points each time, this duration in terms of real time is negligible.

However, there are instances where the model cannot detect heel strike events. Such cases are rare and occur when there are abnormal pattern in data as shown in Figure 22.

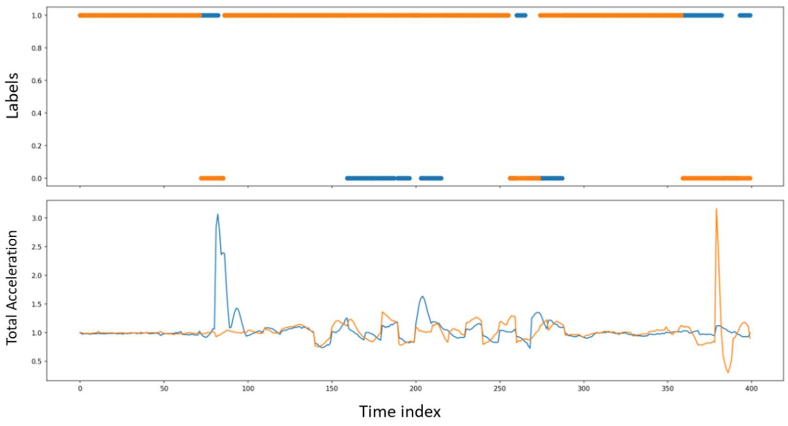

Figure 22: Sample predictions with corresponding total acceleration data (true label is in orange) (2)

This drawback can be tackle by collecting more such irregularities to be used as training data. This in fact can be considered to be a good feature of using evolvable algorithms as new information and data can be used to train the device to perform better and more tailored to each user.

4.2. Laparoscopic Surgery Agent

4.2.1. Performance of Trained Agent in Test Environments

The algorithm derived in section 3.2.2 is implemented to train RL agents not only in the laparoscopic environment but other test environments as well. Since the laparoscopy surgery simulation is a complex and challenging environment, it is important to verify beforehand that the algorithm does improve the agent performance. Thus, a number of different test environments are used to validate the proposed algorithm.

First, two popular benchmark environments from the authors of the original PPO algorithms are used. These benchmarks belong to a set of publicly available environments called “OpenAI Gym” aimed to provide a standard and unified platform of RL algorithm experimentation. The two environments used in this project are CartPole-v1 and MountainCar-v0. Sample images from these two environments are shown in Figure 23 with the details of the two environments can be found in Appendix A .

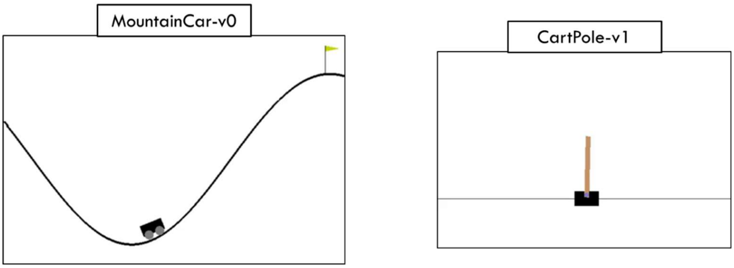

Figure 23: OpenAI Gym environments

Using the proposed algorithm, the agents in both environments were able to reach their respective desired state consistently with different random seeds: pole in upright position for CartPole-v1 and car at flagpole in MountainCar-v0. It is worthy to note that the original PPO algorithm is known to struggle with MountainCar-v0 due to the sparse rewards. This has also been the case in this project experiments with PPO.

Next, a custom environment in V-REP is created to further test the proposed algorithm capability. An image of the environment is shown in Figure 24. Again, the details of this environment is provided in Appendix B.

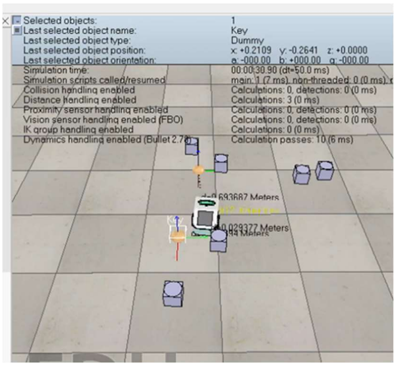
```
Selected objects:                          1
Last selected object name:                 Key
Last selected object type:                 Dummy
Last selected object position:             x: +0.2109   y: -0.2641   z: +0.0000
Last selected object orientation:          a: -000.00   b: +000.00   g: -000.00
Simulation time:                           00:00:30.90 (dt=50.0 ms)
Simulation scripts called/resumed          main: 1 (7 ms), non-threaded: 0 (0 ms),
Collision handling enabled                 Calculations: 0, detections: 0 (0 ms)
Distance handling enabled                  Calculations: 3 (0 ms)
Proximity sensor handling enabled          Calculations: 0, detections: 0 (0 ms)
Vision sensor handling enabled (FBO)       Calculations: 0, detections: 0 (0 ms)
IK group handling enabled                  Calculations: 0 (0 ms)
Dynamics handling enabled (Bullet 2.78)    Calculation passes: 10 (6 ms)
```

Figure 24: Car environment

In short, the goal of this environment is to navigate the mobile robot (car) towards a destination while keeping the number of collisions with obstacles at minimum. This environment is a considerable step-up from the previous two OpenAI Gym environments as both the action space and the observation space are continuous. Furthermore, the dynamics of the environment itself is more complicated.

After training for 1000 iterations, the agent exhibits the ability to move towards and reach the goal to a high degree of consistency. The performance of the trained agents in sample test run with 10 trials can be viewed in video format in this link (https://www.youtube.com/watch?v=dBp4IveVIIk). In this run, the agent reached the destination nine times out of ten trials.

4.2.2. Performance of Trained Agent in Laparoscopic Environment

After the algorithm has demonstrated the ability to improve the agent performance, it is finally used in the laparoscopy environment. The algorithm is implemented in Python with the training strategy devised in section 3.2.3. The hyper-parameter used is given in Table V.

After training, the agent is then tested multiple times, each with 10 trials. Overall, on a basis of ten, the agent can navigate the robot arm to reach the desired position from two to three times depending on the coordinate of the destination. The progressive movements of the laparoscope are shown in Figure 25. As with the car environment, a sample test run with 10 trials can be seen by following this link (https://www.youtube.com/watch?v=FBDMQz5KcCM).

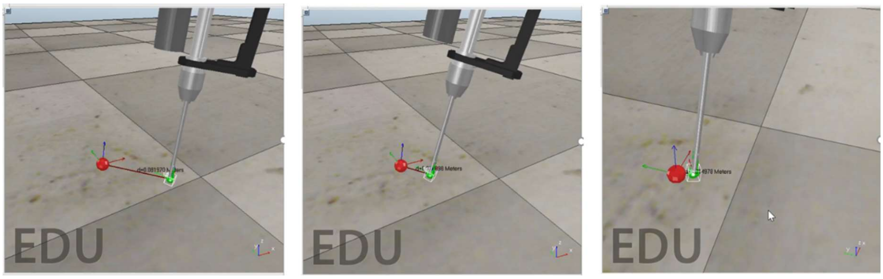

Figure 25: Movement of the laparoscope in a successful trial

Overall, the agent can recognize the direction of the target position and initiates to move towards it. However, it is observed that when the target has a high coordinate in the z-direction, the agent struggled to move the tip to the corresponding level. This may signify that the agent has not learned to fully control the movements of the laparoscope yet. This can be due to a number of reasons: the limit of the current neural network architecture, insufficient exploration or simply that the agent has not converged yet. However, exhaustive testing and tunings are computationally intensive and time-consuming and have not been tackled yet in this project. Thus, further improvements and continuations are entirely possible.

Aside from the actual behaviour of the agent in the environment, it is also useful to inspect the progress of the agent in the training process. The cumulative reward received by the agent after each episode can be an informative indicator of the training process and are shown in Figure 26.

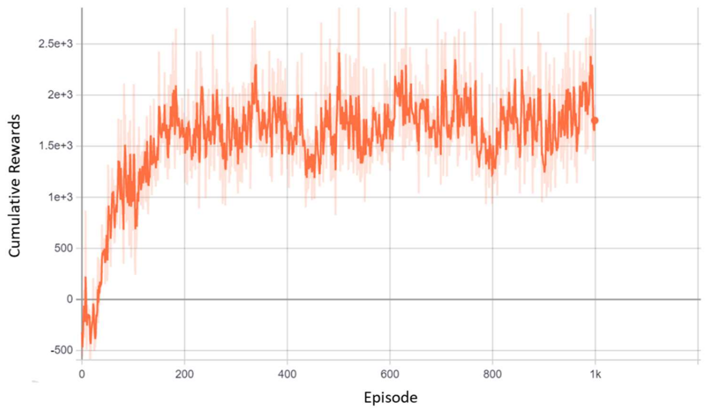

Figure 26: Training progress of RL agent

From the graph, the agent performance improves rapidly from the start until around iteration 200. From then on, the agent stagnates and did not improve significantly. Combined with the sub-optimal behaviour observed, it is likely that either the agent did converge to a sub-optimal policy or the limit of the current neural network policy was reached.

CHAPTER 5: CONCLUSION & FUTURE WORK

The thesis presented two case studies that applied machine learning approaches to control medical devices. These case studies, though distinct from each other, demonstrated the feasibility of leveraging sophisticated data-driven methods to increase the autonomy of medical operations.

Specifically, the first case study concerns with developing a wearable device for Parkinson's Disease patients that recognize the gait phase of users through accelerometer data. The device, delivering minor electrical stimulation to patients, aims to assist users in rehabilitating more continuously and independently. On the other hand, the second case study focuses on a more critical application: robot-assisted laparoscopy surgery. In this case study, a deep reinforcement learning agent is trained with the aim of achieving autonomous navigation and maneuver of the laparoscope. Such approach could have the potential to increase the movement accuracy of the instrument and thus, reducing surgery risks while enhancing laparoscopic surgery efficiency.

These two projects illustrate the feasibility as well as the importance of utilizing machine intelligence in medical sector. For instance, an application that can encompasses both presented case studies would be an exo-skeleton suit for the physically challenged that uses deep learning to infer user actions and then employs reinforcement learning to control how the suit operates to assist in completing intended movements. With such applications have yet to be fully explored, numerous other opportunities are left to be discovered, especially in the fields of operation research, resources allocation and tele-medicine.

Specifics for each case study discussed in this thesis, possible future directions include more extensive experimentations and benchmarking as well as further optimization of current algorithms and methods.

In details, for the gait model, with limited computational power as a hard constraint, converting the model into embedding programming language objects would improve inference speed considerably, which in turn allows for more expressive models to be deployed for better classification performance. In addition, the classification problem can also be expanded to incorporate more classes to represent more gait phases. Once the gait model has become more complete and performant, clinical trials involving human participants can be launched to study and validate the extents of improvements to PD patients' gait function by using the device.

Regarding the laparoscopy surgery reinforcement leaning agent, further testing using benchmark environments would verify the capability of proposed algorithm and identify potential issues or errors. Naturally, better performance on the laparoscopy environment is likely to be obtained with appropriate measures such as hyper-parameter tunings and ensemble learning. Moreover, with the ultimate goal of achieving actual autonomous navigation of laparoscopes physically, transferring from experiments in simulated environment to experiments involving real systems and mechanisms is the direction that should be extensively pursued.

BIBLIOGRAPHY

[1] K. K. Boyer, “Flexibility, in manufacturingFLEXIBILITY IN MANUFACTURING,” *Encyclopedia of Production and Manufacturing Management*. Springer US, Boston, MA, pp. 209–213, 2000, [Online]. Available: https://doi.org/10.1007/1-4020-0612-8_340.

[2] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[3] A. M. Legendre, *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[7] “CADUCEUS: An Experimental Expert System for Medical Diagnosis,” in *The AI Business: Commercial Uses of Artificial Intelligence*, MITP, 1986, pp. 67–80.

[8] E. H. Shortliffe and B. G. Buchanan, “A model of inexact reasoning in medicine,” *Math. Biosci.*, vol. 23, no. 3, pp. 351–379, Apr. 1975, doi: 10.1016/0025-5564(75)90047-4.

[9] F. Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[10] A. G. Ivakhnenko, A. G. Ivakhnenko, V. G. Lapa, V. G. Lapa, V. G. A. LAPA, and R. N. McDonough, *Cybernetics and Forecasting Techniques*. American Elsevier Publishing Company, 1967.

[11] S. M. McKinney *et al.*, “International evaluation of an AI system for breast cancer screening,” *Nature*, vol. 577, no. 7788, pp. 89–94, Jan. 2020, doi: 10.1038/s41586-019-1799-6.

[12] M. P. McBee *et al.*, “Deep Learning in Radiology,” *Acad. Radiol.*, vol. 25, no. 11, pp. 1472–1480, Nov. 2018, doi: 10.1016/j.acra.2018.02.018.

[13] T. Burki, “A new paradigm for drug development,” *Lancet Digit. Health*, vol. 2, no. 5, pp. e226–e227, May 2020, doi: 10.1016/S2589-7500(20)30088-1.

[14] M. Ragoza, J. Hochuli, E. Idrobo, J. Sunseri, and D. R. Koes, “Protein–ligand scoring with convolutional neural networks,” *J. Chem. Inf. Model.*, vol. 57, no. 4, pp. 942–957, 2017.

[15] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, “Deep neural nets as a method for quantitative structure–activity relationships,” *J. Chem. Inf. Model.*, vol. 55, no. 2, pp. 263–274, 2015.

[16] A. Mayr, G. Klambauer, T. Unterthiner, and S. Hochreiter, “DeepTox: toxicity prediction using deep learning,” *Front. Environ. Sci.*, vol. 3, p. 80, 2016.

[17] G. E. Dahl, N. Jaitly, and R. Salakhutdinov, “Multi-task neural networks for QSAR predictions,” *ArXiv Prepr. ArXiv14061231*, 2014.

[18] M. H. S. Segler and M. P. Waller, “Neural-Symbolic Machine Learning for Retrosynthesis and Reaction Prediction,” *Chem. – Eur. J.*, vol. 23, no. 25, pp. 5966–5971, May 2017, doi: 10.1002/chem.201605499.

[19] M. H. Segler, M. Preuss, and M. P. Waller, “Learning to plan chemical syntheses,” *ArXiv Prepr. ArXiv170804202*, 2017.

[20] “Parkinson’s Disease Information Page.” National Institute on Aging (NIA), May 16, 2017, [Online]. Available: https://www.ninds.nih.gov/Disorders/All-Disorders/Parkinsons-Disease-Information-Page.

[21] R. P. Lisak, D. D. Truong, W. M. Carroll, and R. Bhidayasiri, *International Neurology*. Wiley, 2016.

[22] E. R. Dorsey *et al.*, “Projected number of people with Parkinson disease in the most populous nations, 2005 through 2030,” *Neurology*, vol. 68, no. 5, p. 384, Jan. 2007, doi: 10.1212/01.wnl.0000247740.47667.03.
[23] M. Morris, R. Iansek, T. Matyas, and J. Summers, “Abnormalities in the stride length-cadence relation in parkinsonian gait,” *Mov. Disord.*, vol. 13, no. 1, pp. 61–69, Jan. 1998, doi: 10.1002/mds.870130115.
[24] Y. Okada, T. Fukumoto, K. Takatori, K. Nagino, and K. Hiraoka, “Variable initial swing side and prolonged double limb support represent abnormalities of the first three steps of gait initiation in patients with Parkinson’s disease with freezing of gait,” *Front. Neurol.*, vol. 2, pp. 85–85, Dec. 2011, doi: 10.3389/fneur.2011.00085.
[25] R. L. Braddom, *Physical Medicine and Rehabilitation E-Book*. Elsevier Health Sciences, 2010.
[26] N. Giladi *et al.*, “Motor blocks in Parkinson’s disease,” *Neurology*, vol. 42, no. 2, pp. 333–333, 1992, doi: 10.1212/WNL.42.2.333.
[27] N. Giladi *et al.*, “Freezing of gait in patients with advanced Parkinson’s disease,” *J. Neural Transm.*, vol. 108, no. 1, pp. 53–61, Jan. 2001, doi: 10.1007/s007020170096.
[28] B. R. Bloem, J. M. Hausdorff, J. E. Visser, and N. Giladi, “Falls and freezing of gait in Parkinson’s disease: A review of two interconnected, episodic phenomena,” *Mov. Disord.*, vol. 19, no. 8, pp. 871–884, Aug. 2004, doi: 10.1002/mds.20115.
[29] R. Moretti, P. Torre, R. M. Antonello, F. Esposito, and G. Bellini, “The On-Freezing Phenomenon: Cognitive and Behavioral Aspects,” *Park. Dis.*, vol. 2011, p. 746303, Jul. 2011, doi: 10.4061/2011/746303.
[30] S. Perez-Lloret *et al.*, “Prevalence, Determinants, and Effect on Quality of Life of Freezing of Gait in Parkinson Disease,” *JAMA Neurol.*, vol. 71, no. 7, pp. 884–890, Jul. 2014, doi: 10.1001/jamaneurol.2014.753.
[31] J. B. Rowe *et al.*, “Parkinson’s disease and dopaminergic therapy--differential effects on movement, reward and cognition,” *Brain J. Neurol.*, vol. 131, no. Pt 8, pp. 2094–2105, Aug. 2008, doi: 10.1093/brain/awn112.
[32] Tomlinson CL, Herd, CP, Clarke, CE, Meek, C, Patel, S, Stowe, R, Deane, KHO, Shah, L, Sackley, CM, Wheatley, K. and N. Ives, “Physiotherapy for Parkinson’s disease: a comparison of techniques,” *Cochrane Database Syst. Rev.*, no. 6, 2014, doi: 10.1002/14651858.CD002815.pub2.
[33] A. Abbott, “Levodopa: the story so far,” *Nature*, vol. 466, no. 7310, pp. S6–S7, Aug. 2010, doi: 10.1038/466S6a.
[34] W. B. Abrams, C. B. Coutinho, A. S. Leon, and H. E. Spiegel, “Absorption and Metabolism of Levodopa,” *JAMA*, vol. 218, no. 13, pp. 1912–1914, Dec. 1971, doi: 10.1001/jama.1971.03190260028007.
[35] J. E. Hardebo and C. Owman, “Barrier mechanisms for neurotransmitter monoamines and their precursors at the blood-brain interface,” *Ann. Neurol.*, vol. 8, no. 1, pp. 1–11, Jul. 1980, doi: 10.1002/ana.410080102.
[36] “PubChem Compound Summary for CID 34359, Carbidopa,” *National Center for Biotechnology Information*. [Online]. Available: https://pubchem.ncbi.nlm.nih.gov/compound/Carbidopa.
[37] P. Ambermoon, A. Carter, W. D. Hall, N. N. W. Dissanayaka, and J. D. O’Sullivan, “Impulse control disorders in patients with Parkinson’s disease receiving dopamine replacement therapy: evidence and implications for the addictions field,” *Addiction*, vol. 106, no. 2, pp. 283–293, Feb. 2011, doi: 10.1111/j.1360-0443.2010.03218.x.
[38] J. S. Perlmutter and J. W. Mink, “DEEP BRAIN STIMULATION,” *Annu. Rev. Neurosci.*, vol. 29, no. 1, pp. 229–257, Jun. 2006, doi: 10.1146/annurev.neuro.29.051605.112824.
[39] A. L. Benabid, “Deep brain stimulation for Parkinson’s disease,” *Curr. Opin. Neurobiol.*, vol. 13, no. 6, pp. 696–706, Dec. 2003, doi: 10.1016/j.conb.2003.11.001.

[40] N. Allert, J. Volkmann, S. Dotse, H. Hefter, V. Sturm, and H.-J. Freund, “Effects of bilateral pallidal or subthalamic stimulation on gait in advanced Parkinson’s disease,” *Mov. Disord.*, vol. 16, no. 6, pp. 1076–1085, Nov. 2001, doi: 10.1002/mds.1222.

[41] J. Volkmann, J. Herzog, F. Kopper, and G. Deuschl, “Introduction to the programming of deep brain stimulators,” *Mov. Disord.*, vol. 17, no. S3, pp. S181–S187, Mar. 2002, doi: 10.1002/mds.10162.

[42] A. Mostofi, J. M. Evans, L. Partington-Smith, K. Yu, C. Chen, and M. A. Silverdale, “Outcomes from deep brain stimulation targeting subthalamic nucleus and caudal zona incerta for Parkinson’s disease,” *Npj Park. Dis.*, vol. 5, no. 1, p. 17, Aug. 2019, doi: 10.1038/s41531-019-0089-1.

[43] Suzhen Lin *et al.*, “Deep brain stimulation of the globus pallidus internus versus the subthalamic nucleus in isolated dystonia,” *J. Neurosurg. JNS*, vol. 132, no. 3, pp. 721–732, 2019, doi: 10.3171/2018.12.JNS181927.

[44] R. F. Dallapiazza *et al.*, “Considerations for Patient and Target Selection in Deep Brain Stimulation Surgery for Parkinson’s Disease.,” in *Parkinson’s Disease: Pathogenesis and Clinical Aspects*, T. B. Stoker and J. C. Greenland, Eds. Brisbane (AU): Codon Publications, 2018.

[45] D. Soh, T. R. Ten Brinke, A. M. Lozano, and A. Fasano, “Therapeutic Window of Deep Brain Stimulation Using Cathodic Monopolar, Bipolar, Semi-Bipolar, and Anodic Stimulation.,” *Neuromodulation J. Int. Neuromodulation Soc.*, vol. 22, no. 4, pp. 451–455, Jun. 2019, doi: 10.1111/ner.12957.

[46] Omar K. Bangash *et al.*, “Drowning hazard with deep brain stimulation: case report,” *J. Neurosurg. JNS*, vol. 124, no. 5, pp. 1513–1516, 2016, doi: 10.3171/2015.5.JNS15589.

[47] P. K. Doshi, “Long-Term Surgical and Hardware-Related Complications of Deep Brain Stimulation,” *Stereotact. Funct. Neurosurg.*, vol. 89, no. 2, pp. 89–95, 2011, doi: 10.1159/000323372.

[48] G.-M. Hariz, P. Limousin, and K. Hamberg, “‘DBS means everything - for some time’. Patients’ Perspectives on Daily Life with Deep Brain Stimulation for Parkinson’s Disease,” *J. Park. Dis.*, vol. 6, no. 2, pp. 335–347, Mar. 2016, doi: 10.3233/JPD-160799.

[49] *Parkinson’s disease in adults: diagnosis and management*. London: National Institute for Health and Care Excellence (UK), 2017.

[50] C. L. Tomlinson *et al.*, “Physiotherapy intervention in Parkinson’s disease: systematic review and meta-analysis,” *BMJ*, vol. 345, p. e5004, Aug. 2012, doi: 10.1136/bmj.e5004.

[51] D. G. Rutz and D. H. Benninger, “Physical Therapy for Freezing of Gait and Gait Impairments in Parkinson Disease: A Systematic Review.,” *PM R*, Jan. 2020, doi: 10.1002/pmrj.12337.

[52] C. L. Tomlinson *et al.*, “Physiotherapy versus placebo or no intervention in Parkinson’s disease.,” *Cochrane Database Syst. Rev.*, vol. 2013, no. 9, p. CD002817, Sep. 2013, doi: 10.1002/14651858.CD002817.pub4.

[53] P. Ginis, E. Nackaerts, A. Nieuwboer, and E. Heremans, “Cueing for people with Parkinson’s disease with freezing of gait: A narrative review of the state-of-the-art and novel perspectives.,” *Ann. Phys. Rehabil. Med.*, vol. 61, no. 6, pp. 407–413, Nov. 2018, doi: 10.1016/j.rehab.2017.08.002.

[54] D. S. Peterson and K. Smulders, “Cues and Attention in Parkinsonian Gait: Potential Mechanisms and Future Directions,” *Front. Neurol.*, vol. 6, pp. 255–255, Dec. 2015, doi: 10.3389/fneur.2015.00255.

[55] S. J. Lee, J. Y. Yoo, J. S. Ryu, H. K. Park, and S. J. Chung, “The Effects of Visual and Auditory Cues on Freezing of Gait in Patients with Parkinson Disease,” *Am. J. Phys. Med. Rehabil.*, vol. 91, no. 1, 2012, [Online]. Available: https://journals.lww.com/ajpmr/Fulltext/2012/01000/The_Effects_of_Visual_and_Auditory_Cues_on.2.aspx.

[56] R. Constantinescu, C. Leonard, C. Deeley, and R. Kurlan, “Assistive devices for gait in Parkinson’s disease,” *Parkinsonism Relat. Disord.*, vol. 13, no. 3, pp. 133–138, Apr. 2007, doi: 10.1016/j.parkreldis.2006.05.034.
[57] ARAVANTINOS DIMITRIOS, ARAVANTINOS KATINA, and MURPHY HAROLD R, “Visual Stimulation Cane For Parkinson’s Disease Sufferers,” US 6330888 B1, Dec. 18, 2001.
[58] E. Cubo, C. G. Moore, S. Leurgans, and C. G. Goetz, “Wheeled and standard walkers in Parkinson’s disease patients with gait freezing,” *Parkinsonism Relat. Disord.*, vol. 10, no. 1, pp. 9–14, Oct. 2003, doi: 10.1016/S1353-8020(03)00060-9.
[59] K. Kompoliti, C. G. Goetz, S. Leurgans, M. Morrissey, and I. M. Siegel, “‘On’ freezing in Parkinson’s disease: Resistance to visual cue walking devices,” *Mov. Disord.*, vol. 15, no. 2, pp. 309–312, Mar. 2000, doi: 10.1002/1531-8257(200003)15:2<309::AID-MDS1016>3.0.CO;2-P.
[60] S. R. Koehler-McNicholas, L. Danzl, A. Y. Cataldo, and L. I. E. Oddsson, “Neuromodulation to improve gait and balance function using a sensory neuroprosthesis in people who report insensate feet – A randomized control cross-over study,” *PLOS ONE*, vol. 14, no. 4, p. e0216212, Apr. 2019, doi: 10.1371/journal.pone.0216212.
[61] S. J. M. Bamberg, R. J. Carson, G. Stoddard, P. S. Dyer, and J. B. Webster, “The Lower Extremity Ambulation Feedback System for Analysis of Gait Asymmetries: Preliminary Design and Validation Results,” *JPO J. Prosthet. Orthot.*, vol. 22, no. 1, 2010, [Online]. Available: https://journals.lww.com/jpojournal/Fulltext/2010/01000/The_Lower_Extremity_Ambulation_Feedback_System_for.6.aspx.
[62] C. Barthel *et al.*, “The laser shoes: A new ambulatory device to alleviate freezing of gait in Parkinson disease.,” *Neurology*, vol. 90, no. 2, pp. e164–e171, Jan. 2018, doi: 10.1212/WNL.0000000000004795.
[63] C. Punin *et al.*, “A Non-Invasive Medical Device for Parkinson’s Patients with Episodes of Freezing of Gait,” *Sensors*, vol. 19, no. 3, p. 737, Feb. 2019, doi: 10.3390/s19030737.
[64] S. L. Ho, M. Xie, and T. N. Goh, “A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction,” *Comput. Ind. Eng.*, vol. 42, no. 2, pp. 371–375, Apr. 2002, doi: 10.1016/S0360-8352(02)00036-0.
[65] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.
[66] A. Limarc, “What is the Difference Between CNN and RNN?,” Mar. 09, 2020. https://lionbridge.ai/articles/difference-between-cnn-and-rnn/.
[67] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, “Dropout Improves Recurrent Neural Networks for Handwriting Recognition,” in *2014 14th International Conference on Frontiers in Handwriting Recognition*, Sep. 2014, pp. 285–290, doi: 10.1109/ICFHR.2014.55.
[68] A. Graves, A. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6645–6649, doi: 10.1109/ICASSP.2013.6638947.
[69] M. Hüsken and P. Stagge, “Recurrent neural networks for time series classification,” *Neurocomputing*, vol. 50, pp. 223–235, Jan. 2003, doi: 10.1016/S0925-2312(01)00706-8.
[70] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, “Deep learning for time series classification: a review,” *Data Min. Knowl. Discov.*, vol. 33, no. 4, pp. 917–963, 2019.
[71] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
[72] C. Olah, “Understanding LSTM Networks,” *Colah’s Blog*, Aug. 27, 2015. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
[73] M. Tang and J. Zhu, “Text-To-Speech quality evaluation based on LSTM Recurrent Neural Networks,” in *2019 International Conference on Computing, Networking and Communications (ICNC)*, Feb. 2019, pp. 260–264, doi: 10.1109/ICCNC.2019.8685619.

[74] A. Graves, N. Jaitly, and A. Mohamed, “Hybrid speech recognition with deep bidirectional LSTM,” in *2013 IEEE workshop on automatic speech recognition and understanding*, 2013, pp. 273–278.

[75] S. Wang and J. Jiang, “Machine comprehension using match-lstm and answer pointer,” *ArXiv Prepr. ArXiv160807905*, 2016.

[76] M. Chen, G. Ding, S. Zhao, H. Chen, Q. Liu, and J. Han, “Reference based LSTM for image captioning,” presented at the Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[77] S. Saadatnejad, M. Oveisi, and M. Hashemi, “LSTM-based ECG classification for continuous monitoring on personal wearable devices,” *IEEE J. Biomed. Health Inform.*, vol. 24, no. 2, pp. 515–523, 2019.

[78] B. Wiese and C. Omlin, “Credit card transactions, fraud detection, and machine learning: Modelling time with LSTM recurrent neural networks,” in *Innovations in neural information paradigms and applications*, Springer, 2009, pp. 231–268.

[79] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, “Stock price prediction using LSTM, RNN and CNN-sliding window model,” in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1643–1647.

[80] S. Y. Tang, N. S. Hoang, C. K. Chui, J. H. Lim, and M. C. H. Chua, “Development of Wearable Gait Assistive Device Using Recurrent Neural Network,” in *2019 IEEE/SICE International Symposium on System Integration (SII)*, Jan. 2019, pp. 626–631, doi: 10.1109/SII.2019.8700415.

[81] S. Sauerland *et al.*, “Laparoscopy for abdominal emergencies,” *Surg. Endosc. Interv. Tech.*, vol. 20, no. 1, pp. 14–29, Jan. 2006, doi: 10.1007/s00464-005-0564-0.

[82] “Laparoscopy (keyhole surgery).” United Kingdom National Health Service, Aug. 01, 2018, [Online]. Available: https://www.nhs.uk/conditions/laparoscopy/what-happens/.

[83] L. L. Swanstrom and N. J. Soper, *Mastery of Endoscopic and Laparoscopic Surgery*. Wolters Kluwer Health, 2013.

[84] S. Sgouros, *Neuroendoscopy: Current Status and Future Trends*. Springer Berlin Heidelberg, 2013.

[85] R. Vecchio, B. MacFayden, and F. Palazzo, “History of laparoscopic surgery.,” *Panminerva Med.*, vol. 42, no. 1, pp. 87–90, 2000.

[86] “LAPAROSCOPE.” New York City Advanced Laparoscopy and Bariatric Surgery, [Online]. Available: https://www.laparoscopic.md/surgery/instruments/laparoscope.

[87] H. Li *et al.*, “Laparoscopic VS open hepatectomy for hepatolithiasis: An updated systematic review and meta-analysis,” *World J. Gastroenterol.*, vol. 23, no. 43, pp. 7791–7806, Nov. 2017, doi: 10.3748/wjg.v23.i43.7791.

[88] C.-D. Zhang, S.-C. Chen, Z.-F. Feng, Z.-M. Zhao, J.-N. Wang, and D.-Q. Dai, “Laparoscopic Versus Open Gastrectomy for Early Gastric Cancer in Asia: A Meta-Analysis,” *Surg. Laparosc. Endosc. Percutan. Tech.*, vol. 23, no. 4, 2013, [Online]. Available: https://journals.lww.com/surgical-laparoscopy/Fulltext/2013/08000/Laparoscopic_Versus_Open_Gastrectomy_for_Early.1.aspx.

[89] S. J. S. Bajwa and A. Kulshrestha, “Anaesthesia for laparoscopic surgery: General vs regional anaesthesia.,” *J. Minimal Access Surg.*, vol. 12, no. 1, pp. 4–9, Mar. 2016, doi: 10.4103/0972-9941.169952.

[90] S. Wang, N. Shi, L. You, M. Dai, and Y. Zhao, “Minimally invasive surgical approach versus open procedure for pancreaticoduodenectomy: A systematic review and meta-analysis.,” *Medicine (Baltimore)*, vol. 96, no. 50, p. e8619, Dec. 2017, doi: 10.1097/MD.0000000000008619.

[91] D. M. Shabanzadeh and L. T. Sørensen, “Laparoscopic surgery compared with open surgery decreases surgical site infection in obese patients: a systematic review and meta-analysis.,” *Ann. Surg.*, vol. 256, no. 6, pp. 934–945, Dec. 2012, doi: 10.1097/SLA.0b013e318269a46b.

[92] E. P. Westebring-van der Putten, R. H. M. Goossens, J. J. Jakimowicz, and J. Dankelman, “Haptics in minimally invasive surgery--a review.,” *Minim. Invasive Ther. Allied Technol. MITAT Off. J. Soc. Minim. Invasive Ther.*, vol. 17, no. 1, pp. 3–16, 2008, doi: 10.1080/13645700701820242.

[93] A. G. Gallagher, N. McClure, J. McGuigan, K. Ritchie, and N. P. Sheehy, “An ergonomic analysis of the fulcrum effect in the acquisition of endoscopic skills.,” *Endoscopy*, vol. 30, no. 7, pp. 617–620, Sep. 1998, doi: 10.1055/s-2007-1001366.
[94] C. Nezhat, N. S. Saberi, B. Shahmohamady, and F. Nezhat, “Robotic-assisted laparoscopy in gynecological surgery.,” *JSLS*, vol. 10, no. 3, pp. 317–320, Sep. 2006.
[95] P. J. Coronado, M. A. Herraiz, J. F. Magrina, M. Fasero, and J. A. Vidart, “Comparison of perioperative outcomes and cost of robotic-assisted laparoscopy, laparoscopy and laparotomy for endometrial cancer,” *Eur. J. Obstet. Gynecol. Reprod. Biol.*, vol. 165, no. 2, pp. 289–294, Dec. 2012, doi: 10.1016/j.ejogrb.2012.07.006.
[96] Ilic D, Evans, SM, Allan, CA, Jung, JH, Murphy, D. and M. Frydenberg, “Laparoscopic and robotic-assisted versus open radical prostatectomy for the treatment of localised prostate cancer,” *Cochrane Database Syst. Rev.*, no. 9, 2017, doi: 10.1002/14651858.CD009625.pub2.
[97] J. A. Eandi *et al.*, “Robotic Assisted Laparoscopic Salvage Prostatectomy for Radiation Resistant Prostate Cancer,” *J. Urol.*, vol. 183, no. 1, pp. 133–137, Jan. 2010, doi: 10.1016/j.juro.2009.08.134.
[98] F. M. A. Melfi, G. F. Menconi, A. M. Mariani, and C. A. Angeletti, “Early experience with robotic technology for thoracoscopic surgery.,” *Eur. J. Cardio-Thorac. Surg. Off. J. Eur. Assoc. Cardio-Thorac. Surg.*, vol. 21, no. 5, pp. 864–868, May 2002, doi: 10.1016/s1010-7940(02)00102-1.
[99] C. Nezhat *et al.*, “Robotic versus standard laparoscopy for the treatment of endometriosis,” *Fertil. Steril.*, vol. 94, no. 7, pp. 2758–2760, Dec. 2010, doi: 10.1016/j.fertnstert.2010.04.031.
[100] R. A. Howard, *Dynamic programming and Markov processes.* Oxford, England: John Wiley, 1960, pp. viii, 136.
[101] M. Van Otterlo and M. Wiering, “Reinforcement learning and markov decision processes,” in *Reinforcement Learning*, Springer, 2012, pp. 3–42.
[102] P. A. Gagniuc, *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.
[103] G. F. Rose, “A. A. Markov. Téoriá algorifmov (Theory of algorithms). Trudy Matématičéskogo Instituta iméni V. A. Stéklova, vol. 42. Izdatél’stvo Akadémii Nauk SSSR, Moscow-Leningrad1954, 375 pp.,” *J. Symb. Log.*, vol. 22, no. 1, pp. 77–79, 1957, doi: 10.2307/2964063.
[104] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. 2015.
[105] R. Bellman, “A Markovian Decision Process,” *Indiana Univ. Math. J.*, vol. 6, no. 4, pp. 679–684, 1957.
[106] R. Bellman, “On the Theory of Dynamic Programming,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 38, no. 8, pp. 716–719, Aug. 1952, doi: 10.1073/pnas.38.8.716.
[107] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[108] N. Metropolis, “THE BEGINNING of the MONTE CARLO METHOD.”
[109] R. S. Sutton, “Learning to predict by the methods of temporal differences,” *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, Aug. 1988, doi: 10.1007/BF00115009.
[110] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*, vol. 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.
[111] C. J. Watkins and P. Dayan, “Q-learning,” *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, 1992.
[112] G. Tesauro, “Temporal difference learning and TD-Gammon,” *Commun. ACM*, vol. 38, no. 3, pp. 58–68, 1995.
[113] S.-L. Chen and Y.-M. Wei, “Least-squares SARSA (Lambda) algorithms for reinforcement learning,” in *2008 Fourth International Conference on Natural Computation*, 2008, vol. 2, pp. 632–636.
[114] F. Wang, J. Decker, X. Wu, G. Essertel, and T. Rompf, “Backpropagation with Continuation Callbacks: Foundations for Efficient and Expressive Differentiable Programming.”

[115] J. Baxter and P. L. Bartlett, “Direct gradient-based reinforcement learning,” in *2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No. 00CH36353)*, 2000, vol. 3, pp. 271–274.

[116] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[117] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Mach. Learn.*, vol. 8, no. 3–4, pp. 229–256, 1992.

[118] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[119] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” 2014.

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

[121] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *ArXiv Prepr. ArXiv170706347*, 2017.

[122] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” *ArXiv Prepr. ArXiv180101290*, 2018.

[123] C. G. Atkeson and J. C. Santamaria, “A comparison of direct and model-based reinforcement learning,” in *Proceedings of international conference on robotics and automation*, 1997, vol. 4, pp. 3557–3564.

[124] A. S. Polydoros and L. Nalpantidis, “Survey of model-based reinforcement learning: Applications on robotics,” *J. Intell. Robot. Syst.*, vol. 86, no. 2, pp. 153–173, 2017.

[125] D. Baek, M. Hwang, H. Kim, and D.-S. Kwon, “Path planning for automation of surgery robot based on probabilistic roadmap and reinforcement learning,” in *2018 15th International Conference on Ubiquitous Robots (UR)*, 2018, pp. 342–347.

[126] L. Yu, X. Yu, X. Chen, and F. Zhang, “Laparoscope arm automatic positioning for robot-assisted surgery based on reinforcement learning,” *Mech. Sci.*, vol. 10, no. 1, pp. 119–131, 2019.

[127] C. López-Casado, E. Bauzano, I. Rivas-Blanco, V. F. Muñoz, and J. C. Fraile, “Collaborative robotic system for hand-assisted laparoscopic surgery,” in *Iberian Robotics conference*, 2017, pp. 548–553.

[128] X. Tan, C.-B. Chng, Y. Su, K.-B. Lim, and C.-K. Chui, “Robot-assisted training in laparoscopy using deep reinforcement learning,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 485–492, 2019.

[129] B. Thananjeyan, A. Garg, S. Krishnan, C. Chen, L. Miller, and K. Goldberg, “Multilateral surgical pattern cutting in 2d orthotropic gauze with deep reinforcement learning policies for tensioning,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2371–2378.

[130] M. N. Alam, A. Garg, T. T. K. Munia, R. Fazel-Rezai, and K. Tavakolian, “Vertical ground reaction force marker for Parkinson’s disease,” *PloS One*, vol. 12, no. 5, pp. e0175951–e0175951, May 2017, doi: 10.1371/journal.pone.0175951.

[131] I. González, J. Fontecha, R. Hervás, and J. Bravo, “An ambulatory system for gait monitoring based on wireless sensorized insoles,” *Sensors*, vol. 15, no. 7, pp. 16589–16613, 2015.

[132] T. Seel, J. Raisch, and T. Schauer, “IMU-based joint angle measurement for gait analysis,” *Sensors*, vol. 14, no. 4, pp. 6891–6909, 2014.

[133] Y. Jiang and K. E. Norman, “Effects of visual and auditory cues on gait initiation in people with Parkinson’s disease,” *Clin. Rehabil.*, vol. 20, no. 1, pp. 36–45, 2006.

[134] M. P. Pereira, L. T. Gobbi, and Q. J. Almeida, “Freezing of gait in Parkinson’s disease: Evidence of sensory rather than attentional mechanisms through muscle vibration,” *Parkinsonism Relat. Disord.*, vol. 29, pp. 78–82, 2016.

[135] R. Marchese, M. Diverio, F. Zucchi, C. Lentino, and G. Abbruzzese, “The role of sensory cues in the rehabilitation of parkinsonian patients: a comparison of two physical therapy protocols,” *Mov. Disord.*, vol. 15, no. 5, pp. 879–883, 2000.
[136] J. Han, E. Kim, J. Jung, J. Lee, H. Sung, and J. Kim, “Effect of muscle vibration on spatiotemporal gait parameters in patients with Parkinson’s disease,” *J. Phys. Ther. Sci.*, vol. 26, no. 5, pp. 671–673, 2014.
[137] A. M. De Nunzio, M. Grasso, A. Nardone, M. Godi, and M. Schieppati, “Alternate rhythmic vibratory stimulation of trunk muscles affects walking cadence and velocity in Parkinson’s disease,” *Clin. Neurophysiol.*, vol. 121, no. 2, pp. 240–247, 2010.
[138] C. P. Donnellan and K. Caldwell, “TENS and FES for sensory impairment and gait dysfunction following removal of spinal cord ependymoma—a case report,” *Physiother. Res. Int.*, vol. 14, no. 4, pp. 234–241, 2009.
[139] K. Kita *et al.*, “A pilot study of sensory feedback by transcutaneous electrical nerve stimulation to improve manipulation deficit caused by severe sensory loss after stroke,” *J. Neuroengineering Rehabil.*, vol. 10, no. 1, p. 55, 2013.
[140] Y. Qi, C. B. Soh, E. Gunawan, K.-S. Low, and R. Thomas, “Assessment of foot trajectory for human gait phase detection using wireless ultrasonic sensor network,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 24, no. 1, pp. 88–97, 2015.
[141] M. Li, T. Zhang, Y. Chen, and A. J. Smola, “Efficient mini-batch training for stochastic optimization,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 661–670.
[142] I. Akkaya *et al.*, “Solving rubik’s cube with a robot hand,” *ArXiv Prepr. ArXiv191007113*, 2019.
[143] C. Berner *et al.*, “Dota 2 with large scale deep reinforcement learning,” *ArXiv Prepr. ArXiv191206680*, 2019.
[144] L. Sergey, “Value Function Methods,” [Online]. Available: http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-7.pdf.

APPENDIX

A. OpenAI Gym Environments

1. CartPole-v1

Documentation: https://gym.openai.com/envs/CartPole-v1/

Observations:

Number	Information	Value Range
0	Cart position	$[-4.8; +4.8]$
1	Cart velocity	$[-\infty; +\infty]$
2	Pole angle (rad)	$[-0.418; +0.418]$
3	Pole angular velocity	$[-\infty; +\infty]$

Actions:

Number	Action
0	Push to left
1	Push to right

Rewards: 1 for every step taken, including the termination step

2. MountainCar-v0

Documentation: https://gym.openai.com/envs/MountainCar-v0/

Observations:

Number	Information	Value Range
0	Car position	$[-1.2; +0.6]$
1	Car velocity	$[-0.07; +0.07]$

Actions:

Number	Action
0	Accelerate to left

1	No acceleration
2	Accelerate to right

Rewards:

- 0 is awarded if the agent reached the flag (position = 0.5) on top of the mountain.
- -1 is awarded if the position of the agent is less than 0.5.

B. Car Environment

Observations:

Number	**Information**	**Value Range**
0	Collision information	$\{0,1,2,3\}$
1	Previous action	$[-1;+1]$
2		
3		
4	Obstacles positions	$[-0.95;+0.95]$
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17	Car orientation (rad)	$[-\pi;+\pi]$
18	Direction to target (rad)	$[-\pi;+\pi]$
19	Car position	$[-\infty;+\infty]$
20		

21	Target position	$[-0.95; +0.95]$
22		

Actions:

Number	Actions	Value Range
0	Left wheel velocity	$[-1; +1]$
1	Right wheel velocity	$[-1; +1]$

Early termination conditions:

- Car position is smaller than -0.95 or larger than +0.95
- Collision information is 3 (critical collision)

Rewards:

If early termination:

$$r_t = -50$$

else:

$$r_t = -0.5(Col) + \begin{cases} 0.05(2.7 - d_t) & if\ d_t \leq d_{t-1} \\ -0.04 & if\ d_t > d_{t-1} \\ 80 & if\ d_t \leq 0.05 \end{cases}$$

$where\ d_t\ is\ the\ distance\ between\ the\ tip\ and\ the\ target\ position, Col\ is\ the\ collision\ infos$