DISSERTATION

SCALABLE AND DATA EFFICIENT DEEP REINFORCEMENT LEARNING METHODS FOR HEALTHCARE APPLICATIONS

Submitted by

Venkata Ratnam Saripalli

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2019

Doctoral Committee:

Advisor: Charles W Anderson

Ann Marie Hess
Peter Young
Steve John Simske

ABSTRACT

SCALABLE AND DATA EFFICIENT DEEP REINFORCEMENT LEARNING METHODS FOR HEALTHCARE APPLICATIONS

Artificial intelligence driven medical devices have created the potential for significant breakthroughs in healthcare technology. Healthcare applications using reinforcement learning are still very sparse as the medical domain is very complex and decision making requires domain expertise. High volumes of data generated from medical devices – a key input for delivering on the promise of AI, suffers from both noise and lack of ground truth. The cost of data increases as it is cleaned and annotated. Unlike other data sets, medical data annotation, which is critical for accurate ground truth, requires medical domain expertise for a high-quality patient outcome. While accurate recommendation of decisions is vital in this context, making them in near real-time on devices with computational resource constraint requires that we build efficient, compact representations of models such as deep neural networks.

While deeper and wider neural networks are designed for complex healthcare applications, model compression can be an effective way to deploy networks on medical devices that often have hardware and speed constraints. Most state-of-the-art model compression techniques require a resource centric manual process that explores a large model architecture space to find a trade-off solution between model size and accuracy. Recently, reinforcement learning (RL) approaches are proposed to automate such a hand-crafted process. However, most RL model compression algorithms are model-free which require longer time with no assumptions of the model. On the contrary, model-based (MB) approaches are data driven; have faster convergence but are sensitive to the bias in the model.

In this work, we report on the use of reinforcement learning to mimic the decision-making process of annotators for medical events, to automate annotation and labelling. The reinforce-

ment agent learns to annotate alarm data based on annotations done by an expert. Our method shows promising results on medical alarm data sets. We trained deep Q-network and advantage actor-critic agents using the data from monitoring devices that are annotated by an expert. Initial results from these RL agents learning the expert-annotated behavior are encouraging and promising. The advantage actor-critic agent performs better in terms of learning the sparse events in a given state, thereby choosing more right actions compared to deep Q-network agent. To the best of our knowledge, this is the first reinforcement learning application for the automation of medical events annotation, which has far-reaching practical use.

In addition, a data-driven model-based algorithm is developed, which integrates seamlessly with model-free RL approaches for automation of deep neural network model compression. We evaluate our algorithm on a variety of imaging data from dermoscopy to X-ray on different popular and public model architectures. Compared to model-free RL approaches, our approach achieves faster convergence; exhibits better generalization across different data sets; and preserves comparable model performance. The new RL methods' application to healthcare domain from this work for both false alarm detection and model compression is generic and can be applied to any domain where sequential decision making is partially random and practically controlled by the decision maker.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Chuck Anderson for his unconditional support and insightful, very helpful guidance throughout my Ph.D. program. I would like to thank my committee: Ann Hess, Peter Young, and Steve Simske for their willingness to guide this work, their feedback, and help in shaping my dissertation into a novel work addressing two problems important to healthcare device AI applications. I would like to thank Ingrid Bridge, our graduate student advisor, for her constant support.

Outside of my committee, I am fortunate to have great mentors and support. I would like to thank my mentors from Microsoft: David Heckerman, Kalyan Basu, Franck Dauche, and Ravi Ravichandran for their mentorship and support in my pursuit of doctoral studies. I would also like to thank my advisor from the Stanford University Biomedical Informatics program, Russ Altman, for his confidence and support of my doctoral study.

I would like to convey my deep gratitude to Keith Bigelow, Avinash Gopal, Karley Yoder, and Sharath Pasupunuti for giving me the opportunity to be part of the GE Healthcare Data Science team that created the opportunity to pursue research work. I would like to especially thank Avinash for his support in finalizing the topic and framework for my research project, and the many stimulating discussions from practical points of view.

I would also like to thank my colleagues, Ravi Soni, Jiahui Guan, and Dibyajyoti Pati at GE Healthcare for their wonderful collaboration with the Amazon AWS team in exploring and extending the model compression work. I would like to thank Michael Potter and Jiahui Guan for helpful comments and suggestions for the annotation work. My special thanks to Sneha and David for proofreading and editing help in the final stages of my dissertation preparation. I would like to thank my personal trainer, Lara Erlak, for keeping me in good health and motivating me throughout the process.

Additionally, I would like to thank all my extended family, cousins and friends for their understanding of my absence from social events and occasions. I would like to thank my parents for their

constant reminder to pursue my Ph.D. and showing me how to be cheerful in trying times. I can't thank my sons, Arvind and Krishi, enough, who not only supported me during my hectic schedule but also took care of themselves while I was pursuing my Ph.D. Last, but certainly not least, every word of this dissertation, yearn for my learning, my accomplishments, and my happiness would not have been possible without Prasad— my life partner, friend, and husband.

Finally, I express my gratitude to all who are part of my learning journey from childhood and making it a memorable one.

DEDICATION

I would like to dedicate this dissertation to my family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The healthcare domain has seen a dramatic shift in machine learning and computational methods in recent years with the rise in availability of deep learning and medical data. The need for lowering costs and improving quality in healthcare is imperative worldwide. Operating room (OR) costs represent one significant portion of the hospital's costs. Many sequential decision-making steps are involved in the OR functioning. Examples of OR decisions include transferring patients to post-anesthesia care units (PACU), scheduling staff for PACU, determining surgery end, estimating emergence phase, estimating time to extubate, and setting critical event alarms. A need and opportunity exist to develop novel, practical applications of reinforcement learning in such clinical scenarios, taking advantage of the advanced deep learning methods on one hand and enhanced data availability on the other. Below are two example healthcare use cases in which we could apply Artificial Intelligence (AI) at multiple stages of the medical intervention process.

1.1 Use-case Scenarios

John Doe, a 44-year-old patient, was taken to an emergency room (ER) after being involved in a multiple-vehicle collision. The trauma team in the ER must urgently prepare multiple procedures for trauma resuscitation of John Doe. The two most critical life threats to John Doe that should be ruled out are massive hemorrhage and critical airway compromise [1]. The trauma team might recommend a computed tomography (CT), magnetic resonance imaging (MRI), or ultrasound (US) to detect any internal bleeding. The airway compromise (obstruction to breathing) can be detected through multiple visible symptoms and a chest X-ray for critical conditions such as pneumothorax (PTX), a collapsed lung condition in which the air leaks into the chest cavity, exerting pressure on the lung. Once the life threats are evaluated and managed, hemodynamic monitoring begins to check the function of the heart to verify the blood pressure in arteries, veins, and the heart. Interventions such as intubation to induce breathing, chest compression to resuscitate heart func-

tioning, and pelvic binding to address any fractures, among others, are performed before John Doe is moved to the intensive care unit (ICU) or operation room or further procedures or continued monitoring of the resuscitation. Rapid yet accurate response throughout this process is critical for John Doe's survival.

Jane Doe, a 35-year-old patient who is 30 weeks pregnant, was brought into ER after experiencing a shortness of breath and chest pain. The pregnant patient's vital signs, laboratory results, and presence of fetus makes the case challenging in terms of interventions, therapies, and imaging choices, unlike other ER cases [2]. Jane's vitals such as heart rate, respiratory rate, and blood pressure would be different than those of a non-pregnant patient complaining of shortness of breath and chest pain. Various reasons (such as pulmonary embolus [PE], peri-partum cardiomyopathy, myocardial infraction, or aortic dissection, among others) could explain why Jane is experiencing chest pain or shortness of breath that the ER team must diagnose. Jane might display indications of heart murmurs, electrocardiogram (ECG) changes, and chest X-ray abnormalities such as elevated hemidiaphragm or plural effusion. A differential diagnosis might be recommended by her ER team to diagnose the cause of her chest pain and then treat her.

In the previously described use cases, medical professionals must interface with many test results, software programs, and medical devices in a short time span to achieve a high-quality and accurate patient outcome, usually on an emergency footing. Artificial intelligence (AI) and computational methods would help to improve the productivity of medical professionals in such critical scenarios. Deep learning (DL) methods compared with traditional machine learning are scalable and efficient in learning the data patterns when provided sufficient data. Data is the fundamental currency for solving many healthcare problems that use computational methods. While volumes of medical data are becoming increasingly available, such big data has its own unique challenges.

1.2 Brief Background

In all such scenarios, generating alarms and alerts regarding the required procedures, actions, and decisions is central to the success of the entire clinical workflow. Alarm fatigue is a well-

known issue generated by bedside monitoring systems that is a concern for medical professionals and patients [2, 3]. Patients and their families become anxious when alarms indicate a change in the patients' health and a need for medical attention. False alarms are a source of "crying wolf" behavior and tend to become silenced when repeated false alarms are output by the devices. To build smart alarm systems and use AI algorithms in practice, we must address the issue of false alarms, which require a significant volume of correctly annotated data. Medical data suffers from data privacy, sparsity, noise, quality, missing data, heterogeneity, and ground truth availability [4–7]. It is expensive and time consuming, and it requires domain expertise to obtain trustworthy annotations of medical monitoring data [2].

Medical devices such as anesthesia machines, ventilators, and monitoring systems are a rich source of data that help in processing, identifying, and alerting events that in turn serve as the basis for optimal decision making. Similarly, smart medical devices enabled by deep neural networks (DNN) for runtime decision making increase the efficiency of medical professionals, thereby improving the quality of care and lowering the healthcare costs. Medical devices have varied configurations and computational speeds. Deep and wide state-of-the-art neural networks developed in research settings might not be practical for real-world heterogenous device configurations. The need for building smaller and faster deep neural networks models within the constrained speed limits and limited computational hardware resources is imperative. Designing smaller and faster models requires domain expertise and significant manual effort to reach the optimal network to achieve desirable model performance. Recent reinforcement learning (RL) approaches, network-to-network compression (N2N), and AutoML for model compression (AMC) have paved the way for automated neural network compression methods [8, 9]. These automated compression techniques are time consuming and process intensive.

A brief introduction to RL and its advancements in recent years will be covered in the next two paragraphs. Reinforcement learning is implemented via two main approaches: model-free (MF) and model-based (MB) approaches. The MF method is a direct approach based on the trial and error of experiences. An MB method is based on a model representation of the environment, such

that an agent can predict the next state and/or reward based on previous experiences. Planning is a problem-solving method for determining an optimal policy given the model of the environment. Applications of RL and journal articles are on the rise for the MF method, in which there are no assumptions of the environment or prior knowledge of the domain or when the historical data is needed. Model-based RL applications are still sparse, as the dynamics of the environment must be accurately modeled to achieve an efficient MB RL model. Model-free approaches are often flexible and effectively learn complex policies but require many trials and training time for convergence. In contrast, MB approaches require many data samples and less training time to converge more quickly. Though practically efficient, MB approaches are sensitive to biases.

Many RL approaches have been recently seen in clinical outcome decision making based on estimating the value, given the state of the patient. In recent years, many applications of RL can be seen in hospital settings. The data generated from various health systems are yet to be tapped for their full potential. Many traditional approaches have been used to detect false alarms that depend on feature engineering and the availability of ground truth (labeled) data [3, 10, 11]. Although traditional approaches of false alarm detection have reasonable model performance, they have many limitations, such as a narrow focus on one alarm/signal type (e.g., arrhythmia) that cannot scale and generalize for various alarm types. Distant supervision methods in natural language understanding for extracting relations in sentences and generate ground truth data comes with a high cost of false positives. These methods do alleviate the ground truth problem but are limited to entity relation classification tasks in language modeling [2]. Applications of learning from experience via decision making using RL such as ventilation-weaning protocols and customized drug administration strategies have proven to be effective [12–15]. The application of RL to complex real-world examples in which the state and action spaces are highly dimensional and serve as a generalization of the resulting learning to new experiences is highly process intensive and challenging.

The first advancement in deep RL combined reinforcement learning and deep neural networks to achieve a complex state-action space, such as Atari 2600 games reaching professional human level scores without any prior domain knowledge of the game [3]. The deep Q-network (DQN)

algorithm combined reinforcement learning with a deep neural network to achieve a novel agent that can learn a complex state-action space that achieves highly accurate outcomes. The challenge of divergence due to using nonlinear function approximators such as neural networks is addressed in this approach via two methods. The first is by using experience replay, which randomizes the data and thereby removes correlations of sequential data (following i.i.d—independently and identically distributed). The second method involves updating the Q function at regular intervals rather than at every time step, as was the case in previous works. More recent RL work, the advantage actor-critic (A2C), learns the approximation of both policy and value functions, and the agent critically uses the value function to update the actions policy [16, 17]. The advantage value in A2C determines the value of a specific action compared with an average action value at a given state.

1.3 Overview and Contributions

In this work, we report on the use of reinforcement learning to address two challenges explained above: i) to mimic the decision-making process of annotators for medical events to automate annotation and labeling, and ii) to develop a model-based compression model to automate the RL model compression for faster conversion. Improvements in these two key areas together carry the promise of improving both the accuracy and efficiency of performance (i.e., computational agility) of the algorithms involved in generating alerts and alarms in ER. Toward solving the first problem, the reinforcement agent learns to annotate alarm data based on annotations performed by an expert. Our method demonstrates promising results on medical alarm data sets. We report on a comparative training and evaluation of DQN and A2C agents using the data from monitoring devices that is annotated by an expert. The initial results of the RL agents learning expert annotation behavior are promising. The A2C agent performs better in terms of learning the sparse events in a given state and thereby selects more correct actions compared with the DQN agent.

To address the second problem, we developed an RL model based on the data generated from the MF approach to predict the reward value. The neural network compression can be a long,

process-intensive task depending on the model size. A model predicting the reward value would reduce the search space of exploring different network architectures, thereby shortening the training time. We have seen training times cut by more than 65%, which significantly saves on operational costs of AI model development. Although this model-based compression can be extended to other domains, our experiments were primarily focused on medical imaging data of neural network compression.

1.4 Overarching Thesis Statements

Across the dissertation, we follow the three-primary guiding principles:

1. **T1-Data and sample efficient methods:** By training models with different downsampling of data, we can get insights into the effect on the model performance.

2. **T2-Model generalization:** By generalizing models to generic annotator behavior, we can extend this work to any events annotation.

3. **T3-Scalable models and algorithms:** By designing the models to be domain agonistic, we can scale the same model design and algorithms to broader healthcare applications.

At the end of the dissertation, we will review how the models and algorithms support these theses. Together, this work represents one of the first attempts to combine innovative RL and AI methods based on deep learning of precious data sets from actual clinical practice to address the two challenging problems involved in the application of AI in healthcare settings.

1.5 Dissertation Outline

The remainder of this dissertation is organized as described in this section. In Chapter 2, we present the summarized state-of-the-art literature for the current approaches in detecting false alarms and neural network compression techniques. The applications of RL to the healthcare domain are still sparse in the literature and practical applications. The limitations of the current

approaches and challenges encountered are presented along with the AI opportunities to improve patient outcomes and lower healthcare costs.

In Chapter 3, we describe the methods that detail the approaches of the RL algorithm used in this work for detecting false alarms. We developed a DQN and A2C algorithm for detecting false alarms using real ER data. In Chapter 4, we detail the methods for network compression. In healthcare, medical devices have varied hardware configurations and computational limitations. Our data-driven MB approach helps to automate the network compression and reduce training time.

In Chapter 5, we present the experiments and results of this work. In addition, we discuss the significance of the results. Using modest data sets, we were able to achieve promising results. In Chapter 6, we summarize the thesis contributions, limitations of our work, and outline the future directions.

Chapter 2

Background

As with recent advancements in AI, innovation in the biological domain has experienced rapid growth. Biological structures such as neural networks have historically provided excellent inspiration for the development of corresponding AI methods. In this chapter, we will broadly discuss the related work in biological domain, supervised and deep learning work in healthcare settings, background work for false alarm detection, deep learning applications for model compression and advancements in deep RL in the following sections.

2.1 Machine Learning Applications in Biological Domain

Reinforcement learning (RL) has become one of the core learning methods in recent applications of artificial intelligence (AI). Reinforcement learning is a prominent branch of AI, centered around an environment that senses, observes, and interacts with an agent in the environment. The environment, in turn, either rewards or penalizes the agent based on special conditions to attain a specific goal. Applications of MF RL in omics (such as genomics and proteomics) are seen in examples such as predicting operon in the bacterial genome, improving the accuracy of genome sequence annotation, and assembling DNA fragments (reconstructing a DNA sequence using fragments), and protein interaction networks. Hybrid RL of MF and MB approaches is applied in medical imaging for initial detection/diagnosis of prostate cancer using transrectal ultrasound images [18]. In recent years, many applications of RL have been seen in hospital settings.

Clinical outcomes such as hospital readmissions, sepsis, length of stay, and dementia, among others, have been predicted using AI methods [19]. Customized drug administration strategies using deep learning and reinforcement learning have proven to be effective [13–15]. Supervised learning prediction models of ventilation-weaning work has been developed in the past [20–22]. Niranjani et al. (2017) have introduced an off-policy RL approach to determine an optimal weaning policy based on historical data [12]. There are several challenges with the historical data, such as

its sparsity, noise due to artifacts, and interval censoring in which there is an interval range rather than an absolute observation. The authors noted interval censoring to be one challenge in learning a policy and its evaluation. The time to extubate is only available as upper bound, not the exact time. Many RL approaches were recently reported in clinical outcome decision making based on estimating the value given the patient's state. Many supervised learning methods are cited for ventilator-weaning prediction using logistic regression, naïve Bayes, and neural networks [20–22].

2.2 Machine Learning Applications in Healthcare Domain

Another important sub-domain of biological systems, Electronic health record (EHR) and electronic medical record (EMR) based healthcare systems, have matured over the past decades. The volumes of data generated from various healthcare systems are rich sources for AI applications. Many traditional supervised approaches have been used for detecting false alarms that depend on feature engineering and the availability of ground truth (labeled) data [3, 10, 11, 23–25]. Wang et al. 3 used a three-step approach of feature extraction, selection, and classification using arterial blood pressure (ABP) and electrocardiogram (ECG) signals for detecting false alarms. The authors found that direct raw signals yield poor results due to noise and unstable voltages. In response, they extracted features using statistical methods that result in improved performance. A support vector machine (SVM) is used for classifying the alarms into true and false categories. While this work presents considerable performance, the results are limited to ECG arrhythmia alarms and require significant feature engineering.

Although traditional approaches of false alarm detection have reasonable model performance, they have many limitations, such as a narrow focus on one alarm/signal type (e.g., arrhythmia) that cannot scale and is not generalized for various alarm types. Distant supervision is mapping of entity relations from a known knowledgebase to a dataset that has unlabeled data used in natural language processing [26]. Distant supervision methods alleviate the limited ground truth problem but fall short of achieving high accuracies and generalizing to a wide variety of tasks [2]. Schwab et al. [2] use a multitask network architecture to select auxiliary tasks and detect false alarms via

distant supervision. Distantly supervised learning methods achieved promising results when the labeled samples were less than 100 compared with the supervised approaches. This work is limited only to false alarm detection caused by technical errors or artifacts. Another limitation of this work is that it requires data sets that have multiple auxiliary tasks. Recent advances in deep learning and its applications, which are beginning to emerge in reinforcement learning, hold a great promise in this regard.

2.3 Deep Learning Applications in Model Compression

Embedding multiple intelligent models for healthcare applications on various medical devices is a challenge and great opportunity for automation. Medical devices such as X-ray, MR, CT, and ultrasound span a range of CPU/GPU configurations and varied inferencing speed requirements. With hardware limitations and speed constraints, reducing the size of neural networks in medical devices is becoming increasingly critical. In model compression, there is a trade-off between compression ratio and model accuracy. Over the past years, researchers have developed model compression techniques so that carefully hand-designed smaller architectures can achieve accuracy similar to that of the original model. The largest problem of these approaches is that they require manually designed network architecture, and it is a long, non-trivial manual process that often requires domain experts. Moreover, it is difficult to determine whether the hand-crafted network is designed optimally since the design space is typically large.

Automation of model compression techniques using reinforcement learning has been on the rise during the past two years. Several conventional ways exist for compressing a neural network, such as channel pruning [27], quantization [28], and knowledge distillation [29]. Pruning-based approaches remove redundant weights and keep only the weights that contribute to the final output [24]. Researchers have recently begun to use channel pruning, in which redundant channels are eliminated from feature maps [27, 30, 31]. Quantization is another approach that constrains the input from a large continuous set of values to output values in a countable or smaller set. By quantizing network weights, such as through rounding and truncation, the sizes of the resulting

networks are automatically decreased [28, 32]. The concept of knowledge distillation is employed to train a given smaller network with respect to the input teacher network so that the performance of two models is comparable [29, 33, 34]. However, these methods require manually selecting a list of layers to be removed or shrunk. As the structures of the hidden layers are often unknown, such manual processes are non-trivial.

Reinforcement learning approaches have been recently developed to automate the network compression process. Ashok et al. [8] proposed an N2N learning, and He et al. [9] used an AMC engine to reduce neural network sizes. However, most of these are model-free (MF) RL approaches, which are time consuming and require the RL agent to explore a large space. Compared with MF methods, model-based (MB) approaches require much less training time, but it is difficult to build a model with little information about the environment [35]. Dyna architecture [[36] combines MF and MB by integrating planning, acting, and learning. Under such architecture, a model learns from experience or from samples generated by MF and predicts rewards/state values that are used in value functions or policy. In the meantime, MF RL iteratively updates value/policy through trial and error. By including a learned model, Dyna RL accelerates the overall training time, since the learned model exploits the environment. The original papers on the Dyna structure use a lookup table, known also as a Q-table, to represent the states, which are less applicable to larger problems [36]. For exploring large space problems, the Q-learning approach falls short due to performance constraints of the lookup table; instead, robust functional approximation approaches are more appropriate. We use an actor and critic algorithm that combines policy search and value function estimation. Any similar functional approximation approach can be used within this framework. To reduce the RL training time while producing comparable model compression results, we developed a scalable automated network compression pipeline, particularly for healthcare medical devices, as part of our production pipeline.

Applying RL to complex real-world examples in which the state and action spaces are highly dimensional and generalizing the resulting learning to new experiences is process intensive and challenging. In the reviewed literature, many traditional regression approaches or MF RL ap-

proaches are applied. Model-based methods are seen rarely, as it is difficult to build an accurate model of the environment and its dynamics. Per our proposed data-driven RL approach, the MB component can be built based on available historical data sets and explores the MF approach as the new data come in to improve the efficiency of the RL agent. The advantages of such hybrid RL approaches include learning from the available historical data and building an efficient algorithm combining the strengths of both MB and MF to learn a better policy. Model-free approaches are often flexible and effectively learn complex policies, but their global convergence requires many trials, resulting in high computational costs and training time. On the contrary, MB techniques have a strong theoretical basis and generalize better if the dynamics of the system are known as reported in the closed-loop control of Propofol [15]. Though practically efficient, model-based approaches are sensitive to biases in the model. In addition, it is difficult to know the model's dynamics. Thus, both aforementioned approaches have their own advantages and limitations.

2.4 Deep Reinforcement Learning Advancements

Deep convolution networks are one of the approaches used as function approximators to evaluate the optimal action-value function in DQN. The challenge of divergence from using nonlinear function approximators such as neural networks is addressed in this approach via two methods: primarily by using experience replay, which randomizes the data and thereby removes correlations of sequential data (following i.i.d—independently and identically distributed) and additionally by updating the Q-function at regular intervals rather than at every time step, as was the case in previous works. The DQN agent is evaluated using an Atari 2600 platform that has approximately 49 varied tasks and compared the results against professional game tester scores. The results indicate that the DQN agent outperformed the best existing RL methods on 43 of the games without including any prior knowledge of Atari games used by other previous approaches [37]. The DQN algorithm demonstrates that experience replay and convolutional function approximator applied to an RL algorithm has achieved generalization-learning intelligence for varied tasks.

Q-learning algorithms have the challenge of overestimation and sometimes learn unrealistically high action values, as the algorithm includes a maximization step of overestimated action values, which tends to prefer overestimated to underestimated values [38]. The max operator used in Q-learning and DQN uses both same values to select and evaluate, thereby causing overestimates of the values and resulting in overoptimistic value estimates. Decoupling the estimation function into two components as action selection and action evaluation is the key difference between DQN and double DQN 38. The actor-critic (A2C) learns the approximation of both policy and value functions, and the agent critically uses the value function to update the actions policy [16, 17]. The advantage value in A2C determines the value of a specific action compared with an average action value at a given state. In our current work, we report on a generic data-driven AI-assisted annotation RL framework that can be applied for any medical event.

The scope of this research is to study the effects of an RL-based approach for detecting medical events using time-series data generated from medical machines such as anesthesia, ventilators, and monitoring systems. Domain expertise for annotating is both time consuming and expensive. Supervised and semi-supervised approaches of false alarm detection require feature engineering and domain expertise to scale and generalize, which is data intensive and expensive. In this work, we propose an RL approach to mimic medical domain expertise to annotate critical alarms and automate such annotation work with high accuracy. A renewed interest exists in the broad application of AI in clinical settings to realize its true potential. Reinforcement learning methods that can capture the essential advantages of both MB and MF methods while containing their respective disadvantages are crucial. In this context, the proposed RL methods would serve as a valuable contribution to the state of the science of RL applications. Furthermore, such methods could soon pave the way toward practical clinical applications for improved outcomes, which the healthcare industry is anticipating with much anticipation.

Chapter 3

Methods for Detecting False Alarms

Healthcare data suffers from both noise and lack of ground truth. The cost of data increases as it is cleaned and annotated in healthcare. Unlike other data sets, medical data annotation, which is critical for achieving accurate ground truth, requires medical domain expertise for an improved patient outcome. In this work, we report on the use of reinforcement learning to mimic the decision-making process of annotators for medical events to automate annotation and labeling. The reinforcement agent learns to annotate alarm data based on annotations performed by an expert. Our method reveals promising results regarding medical alarm data sets. We trained deep Q-network (DQN) and advantage actor-critic (A2C) agents using the data from monitoring devices that is annotated by an expert. The initial results of these RL agents learning expert annotation behavior are promising. The A2C agent performs better in terms of learning the sparse events in a given state, thereby choosing more correct actions compared with the DQN agent.

To keep the RL problem space simple, we merged all the ground truths into alarms (clinically significant, emergent, and urgent) and non-alarms (no clinical significance, and indeterminate) annotated by an expert. We trained simple DQN and A2C agents to learn the alarms as either true alarms or non-alarms based on the state represented by the patient's physiological signals generated by monitoring devices, as seen in Figure 3.1. Our proposed RL approach can learn from the decision making of the domain expert without any assumptions of the system or domain expertise. Once the RL agent achieves reasonable performance, we can replace the human expert with the RL agent to annotate the data and have a human involved to validate the annotations output by the RL agent.

3.1 Datasets

High-quality data sets containing annotations are critical to the development of deep learning models. We used the data from the multi-phasic Push Electronic Relay for Smart Alarms for End

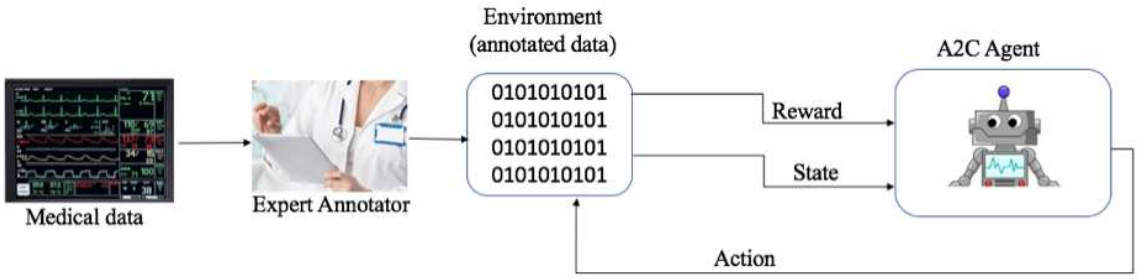


Figure 3.1: Overview of data-driven RL annotation framework for medical events

User Situational Awareness (PERSEUS) program hosted by Brown University's digital archive. This data was generated from an adult emergency department (ED) for a regional referral medical facility and level I trauma center using patient monitoring devices for a 15-bed urgent care area in the ED. The PERSEUS data set, containing 12 months of data, is in its original .json format, is de-identified, and is publicly available [39]. Each monitoring device data for a 24-hour period is recorded in a single file. The following signals are recorded in each file:

- Electrocardiogram waveform (single lead EKG , Lead II) at 250 Hz
- Pulse oximetry waveform (PPG) at 125 Hz
- Vital signs (heart rate [HR], respiratory rate [RR], systolic blood pressure [SBP], diastolic blood pressure [DBP], mean arterial blood pressure [MAP], and peripheral capillary oxygen saturation [SPO2])
- Alarm messages (institution-specified alarms)

Kobayashi et al. (2018), as part of the PERSEUS program, developed subsets with annotation for experimental (non-clinical) research known as Adjudicated/Annotated Telemetry signals for Medically Important and Clinically Significant Events [ATOMICS], which are used in this research [39]. Three non-consecutive weeks of critical alarm data are annotated by Kobayashi et al. (2018) for clinical significance and severity, as observable below.

- Clinical significance

  - Alarm messages (Clinically significant [improvement or deterioration])
  - No clinical significance
  - Indeterminate clinical significance
- Clinical severity
  - Emergent
  - Urgent
  - Non-urgent
  - Indeterminate

The annotations of red alarms are based on EKG, PPG/SPO2, and BP signals from the monitoring devices. The subset data streams consist of 10-minute slices surrounding the individual alarm event, with 5 minutes of data prior to and 5 minutes of data after the alarm. We used the ATOMICS-1 data set for training various RL agents and the ATOMICS-2 data set for testing the RL agents developed as part of this research.

3.2 Preprocessing data

All the data is preprocessed and resampled in seconds and milliseconds. The data is imputed using the mean value of the resampled window to forward fill the data. The ATOMICS-1 and ATOMICS-2 data sets for 15 bedside monitors with vitals, annotations, and alarms are preprocessed. The alarms and annotations are converted to one-hot encoding for processing. The annotations are divided into two categories of actions (alarms/non-alarms) to simplify the problem space. The clinically significant and severe alarms (emergent, urgent) are categorized as alarms, while the indeterminate and non-urgent events are categorized as non-alarms. The three pre-processed data sets—vitals, alarms, and annotations—are then merged. The merge is performed in two stages. In the first stage, alarms and annotations are merged by an inner join where in, for each row of alarm there is a matching row of annotation. In the second stage, all rows

from the vitals data are merged with matching annotated alarms data from the first stage. All rows that matched from second merge stage are alarms and rows with no match are non-alarms. The resulting rows are in a flattened file structure for use during training.

The data is highly imbalanced and sparse for critical clinically significant alarm events. Each critical alarm is surrounded by 600 non-alarm events before and after. This results in an imbalance ratio of 1:1,200; thus, for every alarm, there are 1,200 non-alarms. To rectify this imbalance, we used the following downsampling techniques: n-0, n-1, n-3, n-5, n-10, and mixed. In n-0 downsampling, we retain only alarm data. In n-1, we retain 1 non-alarm before and after an alarm. Similarly, in n-3, n-5, and n-10, we retain 3, 5, and 10 surrounding non-alarms, respectively. Mixed (0,1,3,5,10) is a random sampling of these strategies combined.

3.3 Problem formulation

We define the RL problem formulation in this section. A Markov decision process (MDP) for our alarm annotation RL problem is defined as follows:

- A finite state space S at each time step t for which the environment transitions to the next state $s_t \in S$. s_t is a vector of six physiological variables described above at a given time t.
- An action space A where an agent takes an action $a_t \in A$ at each time ste,p and the state is changed to s_{t+1}. We are using historical data to mimic the environment. In our experiments, when an agent takes the right action, the state transitions to the next state. In future work, for real use-case scenarios of a hospital setting, there is medical intervention required once the real alarm goes off, and the medical staff re-sets the alarm once the patient state reaches normalcy. The actions are alarm and non-alarm (1,0)
- A scalar reward value of 1 for a non-alarm, 10 for an alarm, and zero for an incorrect choice.

The goal of the RL agent is to maximize its expected perceived reward by using known examples to learn an optimal policy.

3.4 Learning an optimal policy

Learning the best mapping (Q-function) between actions and states is the essence of reinforcement learning. The optimal actions are learned primarily via two methods: value-based and policy-based methods. We use a deep neural network to approximate Q value which is more scalable and generalizable solution using both value and policy based approaches. We modeled two functional approximators using a DQN with experience replay (value based [37]) and Actor-Critic networks (policy based [16, 17]) for modeling the expert behavior to annotate the critical alarm events. The DQN network takes in the six physiological variables as described above as inputs, and it outputs a Q-value for each action (non-alarm, alarm) as presented in Equation 3.1.The parameters are updated after every 10 steps of training within each epoch, with a batch size of 8, learning rate of 0.001, and an Adam optimizer. The action (a) selection for both methods is based on ϵ-greedy with a starting value of 1 and annealed to 0.01 with a decay factor of 0.99975.

$$\hat{Q}_k(s_t, a_t) \leftarrow r_{t+1} + \gamma \max_{\alpha \in A} \hat{Q}_{k-1}(s_{t+1}, a, \theta) \tag{3.1}$$

The optimal policy π^* for the DQN method after k iterations is given as follows:

$$\pi^*(s) = \text{argmax}_{\alpha \in A} \hat{Q}_k(s, a) \tag{3.2}$$

The second method we used to train our agent is A2C, which has two networks: one for learning the advantage value of taking an action (actor network) given a state (s) as presented in Equation 3.3 and another for learning the goodness of the action by evaluating the state value (critic network) $v(s, w)$ where s is the state and w is the weights of the network.

$$A(s_t, a_t) \leftarrow r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t) \tag{3.3}$$

The optimal policy for the A2C method after k iterations is given as follows:

$$\pi_\theta(s, a) = P(a|s, \theta) \tag{3.4}$$

Both the actor and critic networks are updated every 10 time steps during an epoch, with a batch size of 8. The networks use an Adam optimizer with a learning rate of 0.001 for the actor network and 0.005 for the critic network. The DQN agent is a value-based algorithm and tends to not improve the accuracy for the same number of training epochs compared to A2C agent. The A2C network, on the other hand, generalizes the learning across the actions and state values independently, resulting in better performance than that of the DQN network.

3.5 Neural network architecture

The A2C architecture is implemented using a sequential dense neural network with the actor and critic as two separate networks as displayed below in Figure 3.2. We designed a shallow neural network for actor and critic to evaluate the alarm prediction problem. The actor network takes the state as input for each time step t and outputs the probability of each action. In our case, we have two actions and hence two outputs. The critic network takes in the state as the input and outputs the value of that state.

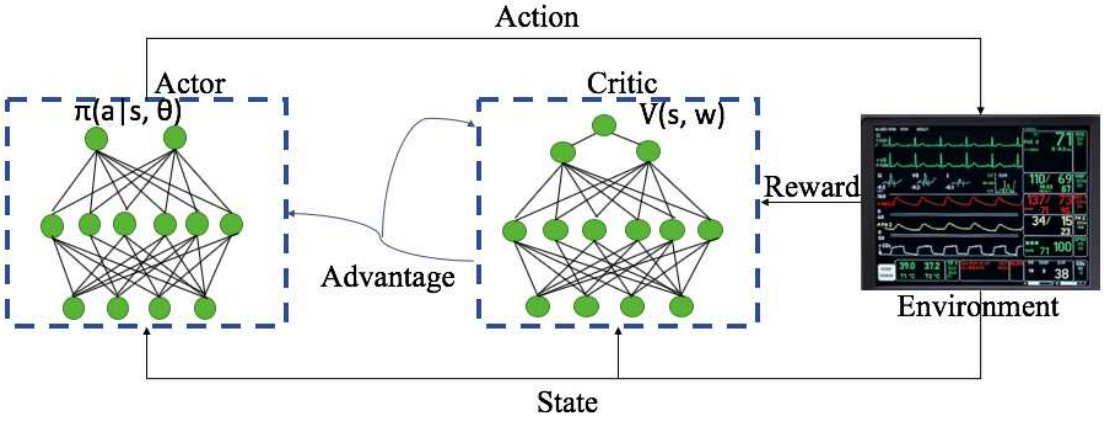


Figure 3.2: A2C–Network architecture overview

The DQN architecture is a single sequential dense neural network as displayed in Figure 3.3. The network has two dense layers, in which the first layer takes in the state as input, and the final

layer outputs the action value for each action. In our case, we have two actions and hence two outputs. The two actions are alarm and non-alarm to detect if the state is critical or normal.

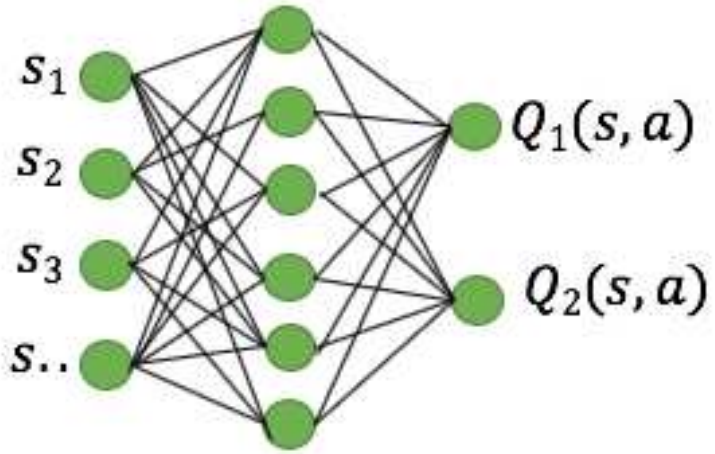


Figure 3.3: DQN-Network summary

3.6 Experiment design

All experiments were run on a MacBook Air with an Intel Core i5 1.8 GHz processor and 8 GB of RAM. Training the DQN and A2C agents was performed using ATOMICS-1 data, and all agent evaluations were conducted using ATOMICS-2 data. Table 3.1 summarizes the ground truth of the data set used in these experiments. Standard binary classification evaluation metrics such as F1 weighted score, precision, and recall are used. Precision measures the positive predictive value that is true positives out of total predicted positives. Recall measures the true positive rate that is count of true positives out of all positives. The F1 weighted score is the harmonic mean between precision and recall that is weighted by the label (alarms and non-alarms) instances. The F1 weighted score is used to compare the performance of various agents trained at different epochs. Python 3.6, Keras library, Sequential model API from Keras library are used for RL code and neural network development. The online Wilcoxon calculator [41] for statistical significance tests.

Table 3.1: Summary of ATOMICS datasets used for training and testing.

	Data subset	True Alarms	Non-Alarms	Total Alarms
Training	ATOMICS-1	437	406	843
Testing n-0	ATOMICS-2	756	468	1224
Testing n-10	ATOMICS-2	756	23035	23791

High volumes of healthcare event data generated from medical machines such as anesthesia, ventilators, and monitoring systems serve as a rich source for AI applications. Domain expertise is required to annotate this large volume of medical events data that is both time consuming and expensive. We find the RL approach for detecting false alarms to be data efficient, scalable, and generalizable for annotation tasks, which are typically costly in the healthcare domain. We are confident our RL agent can learn better and outperform our initial results with more training data when the agent is exposed to additional newer states.

3.7 Benchmarking methods

We used a neural network based classification experiments for the ATOMICS dataset to benchmark our RL based experiments. A shallow sequential dense neural network to classify alarms as a traditional binary classification problem is developed. The six features from the physiological signals described above are input and one output target class value for classifying alarms. Two hidden layers with twenty and four nodes respectively are intermediate layers. The class values for alarms are 1 for alarm and 0 for non-alarm. The neural network summary is represented as seen in Figure 3.4. Adam as an optimizer and rectified linear unit activation function are used in training the DNN.

Support vector machine (SVM), a supervised machine learning (ML) algorithm have been shown to perform well for classification tasks. SVM classifiers separate the classes using hyperplanes. We trained our ATOMICS-1 dataset using an SVM classifier to benchmark our RL agent results. All benchmarking experiments are trained with n-0, n-3, and n-10 downsampling

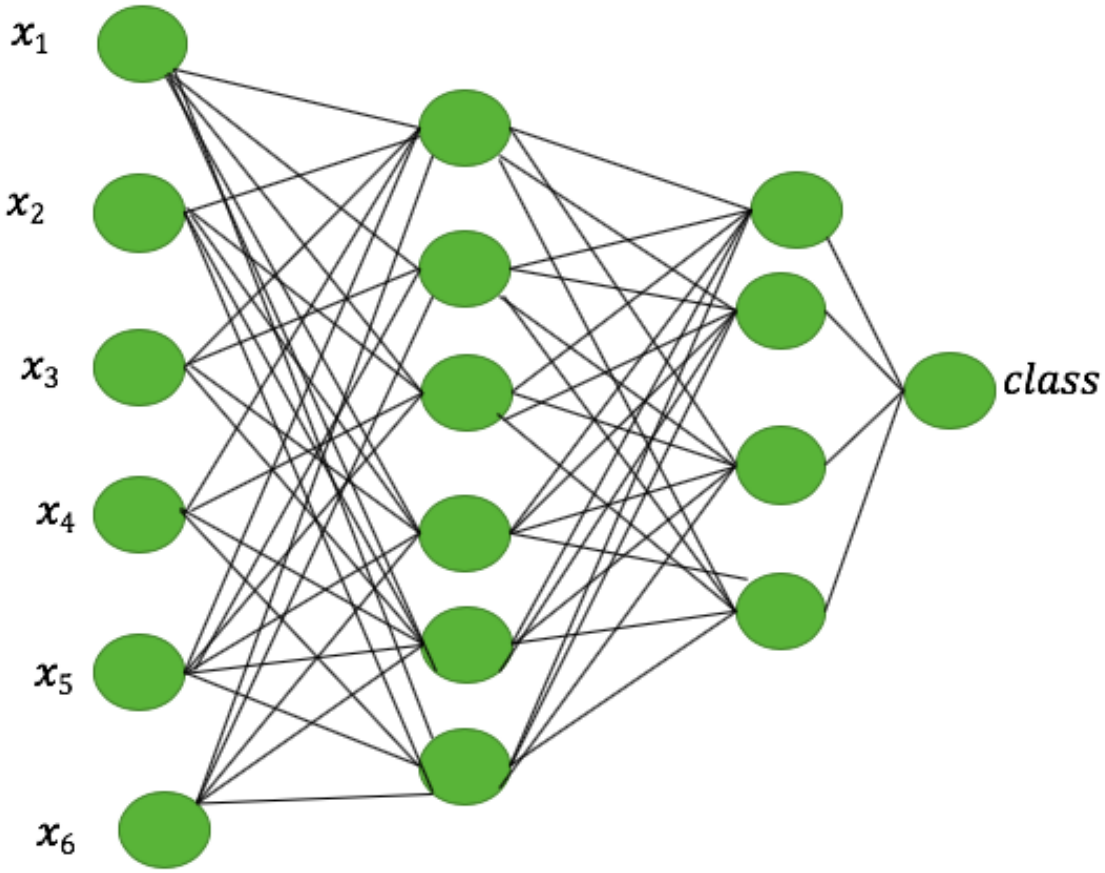


Figure 3.4: Binary classification-Network summary

ATOMICS-1 datasets. The ML algorithms are trained for 5000 epochs for fair comparison with RL agent training runs.

We used scikit-learn's sklearn.svm package, and support vector classifier (SVC) classifier for our SVM experiments. We experimented with poly, rbf, and linear kernels. We choose linear kernel to present our results that gave the maximum AUC and F1-weighted score. We used the default parameters of the SVC classifier - SVC(C=1.0, cachesize=200, classweight=None, coef0=0.0, decisionfunctionshape='ovr', degree=3, gamma='autodeprecated', kernel='linear', maxiter=-1, probability=False, randomstate=None, shrinking=True, tol=0.001, verbose=False) SVM performed the best of for n-0 downsampling and performed poorly for n-3 and n-10 downsampling datasets. DNN performance was reasonable for n-0 and performed as poorly as SVM for n-3 and n-10 downsampling datasets. More details of the benchmark comparisons are seen in Chapter 5.

Chapter 4

Methods for model compression

Artificial intelligence (AI)-driven medical devices have created new excitement in the healthcare sector. While deeper and wider neural networks are designed for complex healthcare applications, model compression can be effective in deploying networks on medical devices that often have hardware and speed constraints. Most state-of-the-art model compression techniques require a resource-centric manual process that explores a large model architecture space to find a trade-off solution between model size and accuracy.

Reinforcement learning (RL) approaches have recently been proposed to automate such a handcrafted process; however, most RL model compression algorithms are model free and require longer time with no assumptions of the model. On the contrary, model-based (MB) approaches are data driven and have faster convergence but are sensitive to bias in the model. In this study [40], a new data-driven model compression method is developed based on the data generated from the model-free (MF) method, as displayed in Figure 4.1. The MB part of the framework is the contribution and scope of this dissertation, while the MF part is not within the scope of this dissertation. The α as shown in the Figure 4.1 is a configurable parameter to choose between MB and MF. When collecting new samples, we select lower value of α to leverage MF method (where $\alpha \in [0, 1]$). We exploit the MB component once we have enough samples collected to generalize for our dataset, and the MB is performing at an expected minimum mean square error of 0.01 (MSE). Python 3.6, Keras library, Sequential model API from Keras library are used for RL code and neural network development. The online Wilcoxon calculator [41] for statistical significance tests.

We demonstrate a novel data-driven dyna like model compression (D3MC) algorithm [40] for healthcare-application-related network architectures. The teacher networks were trained on multiple imaging data sets. We then compared the D3MC framework with the model-free RL in terms of model accuracy and training speed.

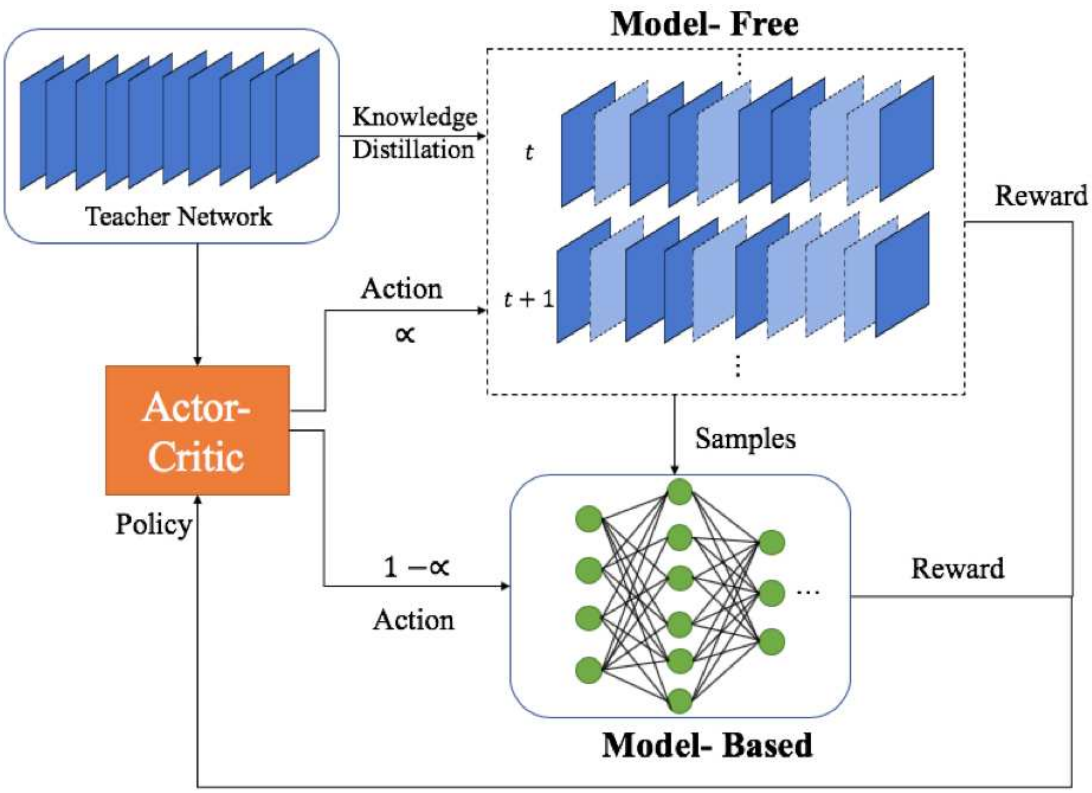


Figure 4.1: D3MC - Overview of model compression framework

Many state-of-the-art networks, such as InceptionV3, are designed for ImageNet data sets that have 1,000 classes; hence, they are wide and deep to achieve high accuracy. Many real-world applications might not be a 1000-class problem and do not require such complex networks. Adapting state-of-the-art networks to our problem is easily achieved by D3MC through exploring the state space and deriving a smaller network with similar model performance.

The D3MC network compression pipeline starts with training a deep neural network based on training datasets. Such trained network is called teacher network. A smaller and reduced network derived from the teacher network is called the student network. After creating the teacher network, we use RL techniques to explore different student networks and return the top k models based on the highest reward value. The final model is then selected based on the trade-off between model performance and the compression ratio set by the decision-maker. The trade-off decision is based on the medical device hardware specification and the model output requirement set by the decision-maker. We picked the top model with the highest reward value to present our results and further

discussion. Since the state-action space is usually huge, resulting in a large number of trial and error steps to train an RL agent, we include a model-based approach along with the model-free to speed up the RL training.

4.1 Data sets

A summary of the data sets used for developing the D3MC algorithm is provided below.

CIFAR-10: The CIFAR-10 dataset [42] consists of 10 classes of objects and is divided into 50,000 train and 10,000 test images (32x32 pixels). This data set provides an incremental level of difficulty over the MNIST data set, using multi-channel inputs to perform model compression.

Chest X-ray Pneumothorax: We used publicly available data sets of chest X-rays with pneumothorax disease from the NIH Clinical Center [43]. The scanned images represent more than 30,000 patients. Here, we used pneumothorax disease as a classification problem.

NLM Frontal: The National Library of Medicine (NLM) frontal data set can be used in a binary classification to group chest X-ray images into frontal or non-frontal position views [44]. This data set contained approximately 8,300 images of size 256x256.

Ham10K: Human Against Machine (HAM) with 100,000 training images [45] contain 10,015 dermatoscopic images to classify pigmented skin lesions.

These four publicly available healthcare data sets are used in this study to automate model compression tasks for productionizing AI models.

4.2 Problem formulation

We consider the standard RL setting in the framework (i.e., an agent interacts with an environment over a number of time steps or trials). At each time step t, the agent receives a state s_t, which is a reduced student network, and selects an action a_t based on its policy π. The policy π is a mapping from s_t to a_t. a_t is a list of binary actions (0 to keep,1 to remove) corresponding to each layer in the network. The agent then receives the next state s_{t+1} as well as a reward r_t. This

iterative process continues for N time steps, where N is sufficiently large that the reward converges (Algorithm 1). The detailed setting of RL framework is as follows:

Environment: Teacher network architectures. The environment accepts a list of layers to be removed from the RL agent.

State: Network architecture derived from the teacher model.

Action: Remove layer or not ([1,0] for each layer).

Agent: Actor-critic based agent.

Optimization: Under actor-critic architecture, the policy is directly parameterized $\pi(a|s;\theta)$. To optimize θ, the REINFORCE policy gradient algorithm from [46] is used which updates θ at each time step t with respect to its gradient ascent on $E[R_t]$. Given that $\nabla_\theta \log \pi(a_t|s_t;\theta)R_t$ is an unbiased estimator of $\nabla_\theta E(R_t)$, and subtracting a baseline can reduce its variances, we update the policy parameters θ in the direction of

$$\nabla_\theta J(\theta) = \nabla_\theta \log \pi(a_t|s_t;\theta)(R_t - b_t(s_t)) \tag{4.1}$$

We use a learned estimate of the value function $V^\pi(s_t)$, the critic, as the baseline b_t.

The MB component is a dense neural network for predicting the reward (Fig. 4.2). We use α to weigh MF and MB components (where $\alpha \in [0, 1]$). As the MB component generalizes, we decay α to reduce the MF dependency. We experimented with various α (0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3) values. Based on our experiments, α of 0.3 for ResNet-18 and 0.5 for Inception-V3 gave the best performance (highest reward) that are used for the rest of the discussion.

D3MC framework combines MF and MB approaches by integrating learning and planning. During the RL training, we add MB step to cutdown the retrain time by predicting the reward signal that was developed using the data samples generated by MF approach. As presented in Figure 4.1, on the MB path, we build a simple dense neural network model to predict the reward directly at decision time.

A decision-time planning is adapted for model based approach. We use smaller α to collect data from the MF method for new architectures and models (see Equation 4.2), where $\alpha \in [0, 1]$).

As our MB becomes more generalizable, we decay α to put more emphasis on the MB RL as a better MB model is built. We experimented with various α (0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3) values. Based on our experiments, α of 0.3 for ResNet-18 and 0.5 for Inception-V3 gave the best performance.

$$\text{D3MC} = \begin{cases} \text{Model-Free} & \text{with probability } 1-\alpha \\ \text{Model-Based} & \text{with probability } \alpha \end{cases} \tag{4.2}$$

Model-Free Reward:

MF component learns an effective policy from rewards alone. Reward is a combination of compression and accuracy ratio. We define MF reward [8] as :

$$R = C(2-C) \cdot \frac{\text{Acc}_{\text{student}}}{\text{Acc}_{\text{teacher}}} \tag{4.3}$$

where $C \in [0, 1)$ is the compression ratio defined as $C = 1 - \frac{\#\text{param}_{student}}{\#\text{param}_{teacher}}$. Acc is the accuracy of the model for evaluation set. Ashok et al. [8] use c(2-c) a non-linear compression function to bias the policy to maintain accuracy while optimizing for compression.

Model-Based Reward:

The MB reward value is computed as a function of layer description as presented in Equation 4.4 using a four-layer dense deep neural network as shown in Figure 4.2.

$$R = f(x_t) \tag{4.4}$$

where $x_t = (a_t, l, k, ks, s, p, n)$, $a_t \in \{0, 1\}_{(L_1-1)\times 1}$ is the action list, l is the layer type, k is the number of kernels, ks is the kernel size, s is stride, p is padding, and n is trainable parameters. The MB model is developed for given network architecture, and the action list for a given architecture is constant. Each input sample is a characterization of the architecture with the presence and absence of layers. When there are no layers, the relevant features are zeros. Action 1 is to remove and 0

to keep the layer. For example, if a network architecture has three layers, the action list for one sample is [0,1,0], which results in a student architecture with the second layer removed.

The MB training loss is the mean squared error (MSE), and we use cross-validation to evaluate the model. We trained the MB model for the desired output of MSE=0.01. We used 10-fold cross-validation, which is a model validation technique that divides the data into ten subsets and repeatedly trains on nine subsets retaining one holdout set for validation set until convergence. Since there are no assumed distributions, such as Gaussian distribution, this function f is driven by the data, which is more representative of the heuristic data structure.

Due to the various needs from different medical devices, we can output top k ($k \leq 30$) compressed models as well as their corresponding model sizes and accuracy. The framework gives more flexibility to choose the "best" compressed model based on the device requirements and constraints.

4.3 Neural network architecture

The MB architecture is implemented using a sequential dense neural network. A shallow network architecture is designed to evaluate the network's ability to learn the reward prediction. The summary of the neural network is displayed in Figure 4.2. The input to the network can be any network architecture model properties, as described above for x_t. Each input sample is a characterization of the architecture with the presence and absence of layers. When there are no layers the relevant layer features are zeros. The model has two hidden layers with 80 and 20 nodes, respectively. The model estimates the reward value which is input to the agent for updating the policy.

4.4 Training time comparison

To perform a fair comparison, all the experiments were trained on Tesla V100 GPUs. In the RL training, we used an Adam optimizer with a learning rate of 0.001. Based on our experiments, an α of 0.3 for Resnet-18 and 0.5 for InceptionV3 produced the best performance. We set the same

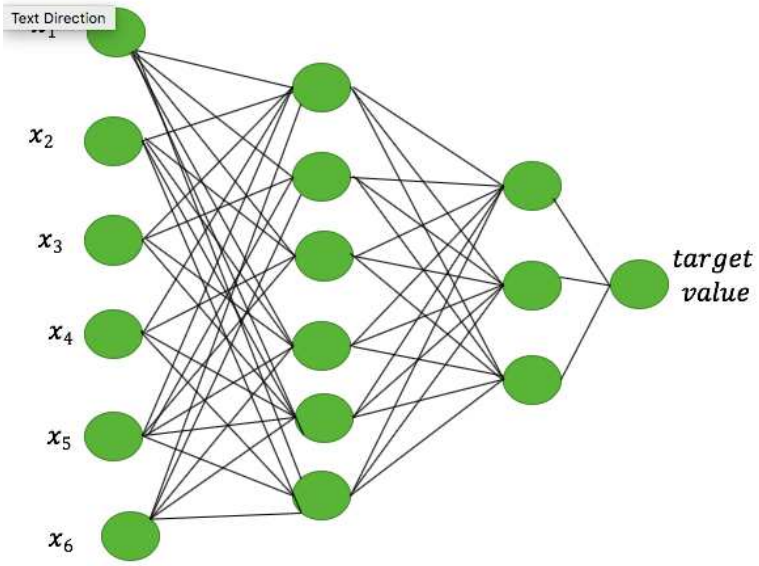


Figure 4.2: MB - Model network summary

training steps for MF RL and D3MC to avoid bias. For the same numbers of training epochs, our D3MC exhibits higher reward values than the MF method with relatively lower training time.

The goal of our D3MC method is to search and find the smallest possible network architecture from a given pre-trained teacher network architecture, while producing the best accuracy possible. We demonstrate that our method performs well on a variety of healthcare data sets and model architectures. The training time of our hybrid algorithm is significantly less for both the ResNet-18 and InceptionV3 architectures compared with MF RL. For example, it took over 120 hours (five days) to train InceptionV3 using the MF-only approach, while our algorithms shortened this time by approximately 60%, thus requiring only two days with the integration of MB component that predicts the reward to update the policy instead of re-training the student network. It is clear that our hybrid data-driven approach is more efficient than the MF RL approaches. Development and integration of the MB component is the scope of this dissertation work.

Algorithm 1 Algorithm of D3MC

$s_0 \leftarrow$ Teacher model
for $i = 1, ...N$ **do**
    for $t = 1, ..L_1$ **do**
        $a_t \sim \pi_{\text{remove}}(s_{t-1}; \theta_{\text{remove},i-1})$
        $s_t \leftarrow T(s_{t-1}, a_t)$
        Sample $u^* \sim \text{Uniform}(0, 1)$
        if $u^* > \alpha$ **then** ▷ Model-Free
            $R \leftarrow r(s_{L_1})$
        else ▷ Model-Based
            $R \leftarrow f(a_t, l, k, ks, s, p, n)$
    $\theta_{\text{remove},i} \leftarrow \nabla_{\theta_{\text{remove},i-1}} J(\theta_{\text{remove},i-1})$
Output: Student model.

Chapter 5

Results and discussion

We present the results from the implementations of enhanced RL models for the two previously defined challenges faced in healthcare AI applications: i) annotation of healthcare data by a domain expert is expensive and ii) medical devices have varied hardware configurations and limited computing resources. The results are presented and discussed in detail in this chapter.

5.1 AI assisted annotator using RL

As detailed in Chapter 1, the purpose of an AI-assisted annotator is to mimic the expert-annotated behavior. We trained DQN and A2C agents using the data from monitoring devices, annotated by an expert. More than 1,000 simulations were conducted using Python. The initial results of these RL agents learning expert-annotated behavior are discussed in detail in the following sections. The A2C agent performs better in terms of learning the sparse events in a given state, thereby choosing more correct actions compared with the DQN agent. The following sections detail the experimental results from this study.

5.1.1 Comparing the optimizers Adam and RMSProp

The performance of various agents trained at different epochs was evaluated and compared for DQN and A2C using F1-weighted scores, as seen in Figure 5.1. The x-axis in Figure 5.1 represents the training epochs at which the agent was saved and evaluated with respect to the test data set. The y-axis is the F1 weighted score, which is the harmonic mean between precision and recall that is weighted by the label (alarms and non-alarms) instances. F1-score is used in binary classification to compute the combined metric of precision and recall. For example, if the classifier is randomly making choices the f1-score would be 0.5, and if the classifier is only predicting alarms always, then the f1-score would be 0.33. We used n-3 downsampling (three non-alarms surrounding an alarm) for training and tested on ATOMICS-2 dataset.

As can be seen from Figure 5.1, the Adam optimizer was found to be more stable than RMSProp with n-mixed downsampling. The RMSProp in red and yellow lines fluctuate resulting in low F1 scores, whereas Adam in blue and green seems steadier in terms of F1 score. While both Adam and RMSProp are learning rate adaptive methods, we see Adam has smaller fluctuations compared to RMSProp. We therefore continued with Adam for the rest of the experiments.

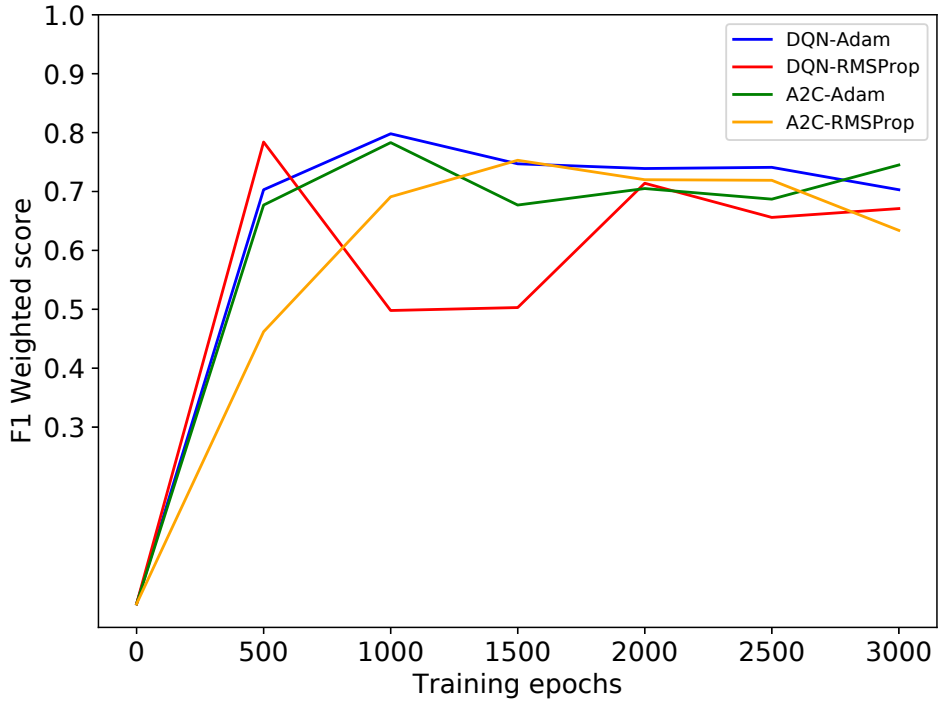


Figure 5.1: Comparison of Adam and RMSProp optimizers for the DQN and A2C networks.

5.1.2 Comparison of results of agents from downsampling ranges

As described in the methods section, the alarms data is highly imbalanced, and we have experimented with various downsampling ranges, such as n-0, n-1, n-3, n-5, n-10, and mixed. We trained A2C agents with the various downsampling ranges data and evaluated the agents saved after the first 1,000 epochs. The agents' performance is compared for the various ranges, as displayed in Figure 5.2, using an F1 weighted score against an n-10 test data set to determine how well the agents generalize for non-alarm data.

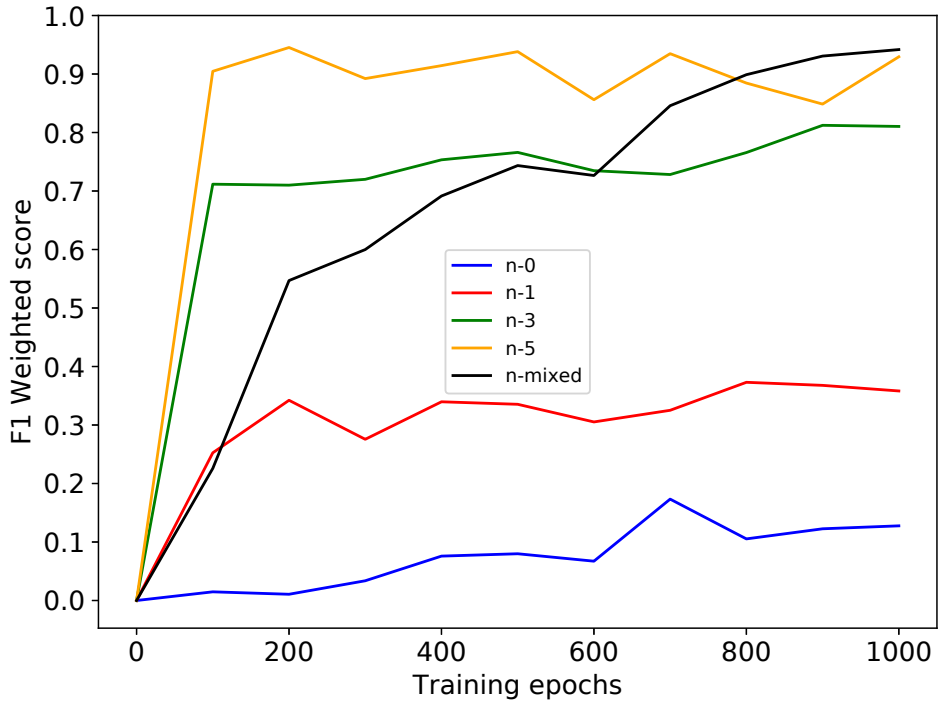


Figure 5.2: Comparison of agents trained on downsampling sample ranges against an F1 score tested on n-10 sampling alarms, excluding n-10 trained agents (10 non-alarms surrounding each alarm).

The A2C agents generalize better when the training data has mixed downsampling ranges compared with a single downsampling, as displayed in Figure 5.2 for the first 1,000 epochs. The agents trained on n-0, n-1, and n-3 sampling ranges perform poorly when tested with n-10 downsampling dataset. The agents trained on n-5 and n-mixed downsampling datasets perform better when tested with n-10 sampling.

These results reveal that exposing the agents to newer and more number of states improve the performance and generalize better with higher sampling dataset. The blue line indicates agents trained on only alarms data and tested on n-10 (10 non-alarms surrounding an alarm) downsampled data display an F1-score of 0.1. In contrast, when agents are trained with n-mixed downsampling, we see that the black line improves over the training epochs. We chose to use mixed downsampling for the section's remaining experiments, as it can scale to any data sampling strategy. In the future, when we extend this work to predictions, for real-time monitoring scenarios, we will encounter both non-alarm data and alarm data.

5.1.3 Comparing the results of A2C and DQN agents

The A2C agent performs better than DQN in our initial results comparing the sensitivity. In analyzing the DQN agent's results, the sparse critical alarm events are not detected, and the agent settles in a local minimum of maximizing the rewards. In the following section, Table 5.1 displays the results, summarizing the performance metrics for the top three agents for the A2C and DQN algorithms. An agent is saved after every 100 epochs. Top three agents results from 50 agents are summarized as seen in the Table 5.1. We ran the experiments for 5,000 epochs. The Table 5.1 columns summarizing the agents performance are defined as TP-true positive; FN-false negative; FP-false positive; TN-true negative and AUC-area under the curve. F1-score is the harmonic mean between precision and recall. Sensitivity is the true positive rate as seen in Equation 5.1 and specificity is the true negative rate Equation 5.2. All the performance metrics of RL agent (TP, FN,FP,TN) are treated equally. The top three agents' results based on the highest AUC and sensitivity are displayed in tables for further discussion.

$$sensitivity = \frac{\text{TP}}{\text{TP + FN}} \tag{5.1}$$

$$specificity = \frac{\text{TN}}{\text{TN + FP}} \tag{5.2}$$

The significance of the results, as presented in Table 5.1, is as follows. The A2C consistently have higher sensitivity scores, greater than 0.774 and with a high of 0.896. This indicates that the A2C agent can detect 90% of the true alarms, whereas the DQN agent can detect only 76% true alarms. The DQN agent has low false positives which might reduce the alarm fatigue at a cost of missing many critical alarms. It can be seen that overall, A2C performs better in terms of sensitivity, which is important in detecting critical alarms.

The DQN agent consistently performs better on the specificity, so it detects the non-alarms more accurately, which is the secondary objective in terms of model performance. We cannot miss

any critical alarms in the healthcare domain. We emphasize high sensitivity, which means fewer critical alarms are missed.

Table 5.1: Comparing the results of A2C and DQN agents with n-mixed downsampling and milliseconds as sampling period tested on ATOMICS-2 alarms dataset. The top three agents' results are listed per group. Best results are highlighted in bold.

Agent	TP	FN	FP	TN	AUC	F1-score	Sensitivity	Specificity
A2C	586	171	192	276	0.681	0.702	0.774	0.589
	599	158	210	258	0.671	0.695	0.791	0.551
	679	**78**	268	200	0.662	0.697	**0.896**	0.427
DQN	576	181	165	303	**0.704**	**0.718**	0.760	0.647
	348	409	63	405	0.662	0.609	0.459	0.865
	254	503	**38**	**430**	0.627	0.533	0.335	**0.918**

5.1.4 Statistical significance tests for A2C and DQN performance results

We used the Wilcoxon rank-sum test to measure the median population difference between the two sample groups (A2C, DQN). The A2C and DQN agents are generated by saving the model for every 100 training epochs of 5,000 total epochs. We tested these 50 agent samples against the ATOMICS-2 data set to generate AUC values. The AUC summary statistics for A2C and DQN are described in Table 5.2. The hypothesis tests if there is significant difference between the two agent's AUC values for 50 agents. Our hypothesis is presented below in Equation 5.3:

$$\begin{aligned} H_0 &= Median(Difference) \leq 0 \\ H_a &= Median(Difference) > 0 \end{aligned} \tag{5.3}$$

Table 5.2: Summary statistics of A2C and DQN agents.

Agent	Samples	Mean	STDev
A2C	50	0.578	0.069
DQN	50	0.521	0.043

The significance level is $\alpha = 0.05$, and the p-value is $p = 0$. Since it is observed that $p = 0 < 0.05$, we reject the null hypothesis. Therefore, we conclude that there is enough evidence to claim the population median of differences is greater than 0, at the 0.05 significance level which means A2C has higher median AUC than DQN for the 50 samples.

5.1.5 A2C Agents training curves and reward signals

All results discussed in this section are based on A2C agents, as they perform better than DQN agents as seen in Table 5.1. The training curves for the first 200 epochs in Figure 5.3 a) displays the average score per epoch during training using n-1 downsampling (one non-alarm surrounding a real alarm). Figure 5.3 b) displays the average score per episode during training using n-5 downsampling (five non-alarms surrounding a real alarm). Figure 5.3 c) displays the average score per episode during training using n-10 downsampling (10 non-alarms surrounding a real alarm). Figure 5.3 d) displays the average score per episode during training using n-mixed downsampling (1,3,5, or 10 non-alarms surrounding a real alarm).

The score is a cumulative average reward gained by the agent at the end of each epoch. The agent is learning to choose better actions steadily. The scalar rewards are different for each of the figures due to the different number of non-alarms in each data set.

5.1.6 Performance metrics for the best A2C agents

In the following section, Table **??**table:a2cresults displays the best agents trained on the ATOMICS-1 data set and tested on the ATOMICS-2 data set for only alarms data. The results are displayed for n-1 and n-mixed downsampling. Although n-1 samples appear to perform reasonably compared

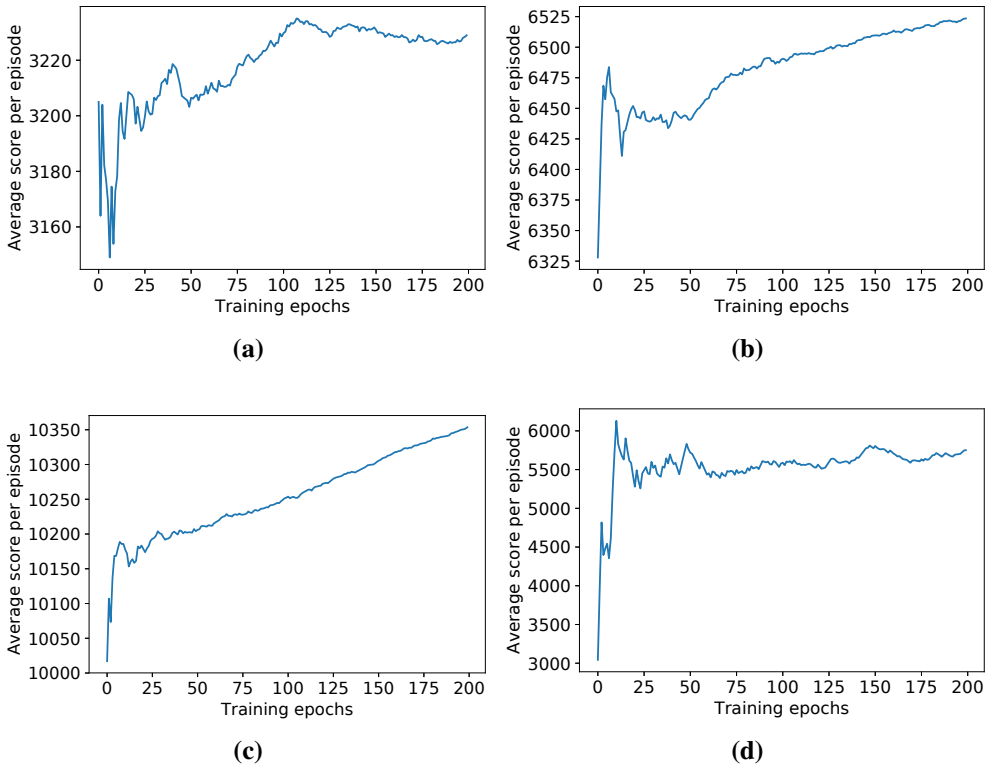
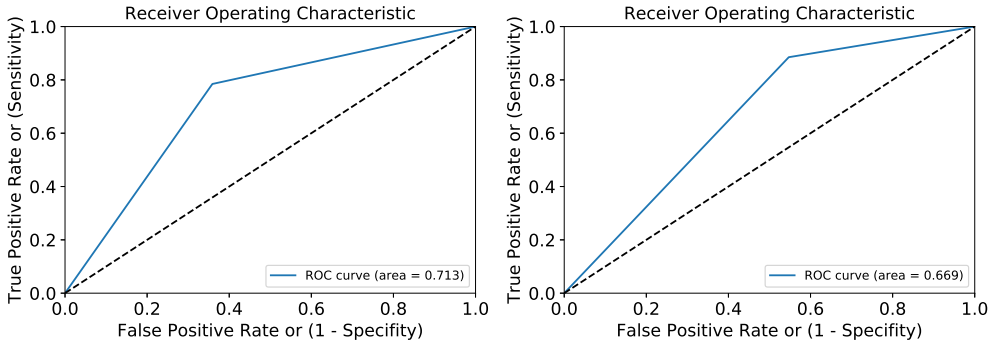


Figure 5.3: Four examples of the A2C training curves tracking the agent's average score. (a) Average score per episode using n-1 downsampling. (b) Average score per episode using n-5 downsampling. (c) Average score per episode using n-10 downsampling. (d) Average score per episode using n-mixed downsampling.

with n-10, when we test our n-1 agents against n-10 (i.e., exposing them to many non-alarms), they perform poorly (blue line of Figure 5.1). All our experiments in the following sections use only n-mixed downsampling with a sampling period in milliseconds.

In our work, we focus on learning the decision making of an expert annotator to discern non-alarms from critical alarms. Our initial results reveal that we can achieve a 72.9% F1-score performance level compared with an expert annotator. Agents perform better when they are exposed to many new states. We see the agents perform best with mixed event downsampling and a sampling period of milliseconds. It can be seen from the results in Table 5.3 that the results of n-mixed downsampling with a sampling period of milliseconds perform the best. The two best performing agents' receiver operating characteristic (ROC) are displayed in Figure 5.4. The decision maker will choose the operating point based on the business application using these ROC curves.



(a) **(b)**

Figure 5.4: Top two agents ROC curves. (a) Agent with 0.8 sensitivity. (b) Agent with 0.88 sensitivity.

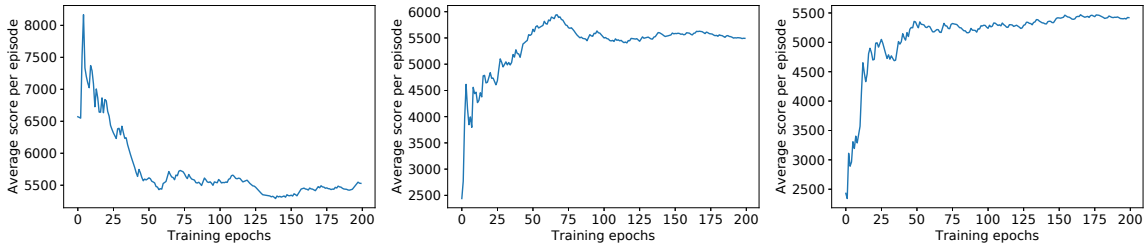
5.1.7 Reward signal of A2C agents for different discount factors

The discount factor in reinforcement learning considers the relative importance of the future rewards, where a 0 value indicates that the agent learns only immediate rewards as opposed to a value of 1, for which it considers future actions. Figure 5.5 a, b, and c illustrate the training curve examples of the average score for the three discount factor values 0.0, 0.5, and 0.9, respectively, for the first 200 training epochs.

Table 5.3: Summary of results for various agents tested on ATOMICS-2 alarms dataset. The top three agents' results are listed per group. Best results are highlighted in bold

Training sample range	Sampling period	TP	FN	FP	TN	AUC	F1-score	Sensitivity	Specificity
n-1 (1 non-alarm surrounding alarm)	milli seconds	599	158	215	253	0.665	0.691	0.791	0.540
		602	155	233	251	0.665	0.691	0.795	0.536
		599	158	217	251	0.663	0.689	0.791	0.536
	seconds	659	97	220	248	0.700	0.731	0.871	0.529
		612	144	197	271	0.694	0.717	0.809	0.579
		617	139	206	262	0.687	0.713	0.816	0.559
n-mixed (0,1,3,5,10 non-alarm surrounding alarm)	milli seconds	**670**	**87**	256	212	0.669	0.703	**0.885**	0.452
		606	151	196	272	0.690	0.713	0.800	0.581
		594	163	**168**	**300**	**0.712**	**0.729**	0.784	**0.641**
	seconds	625	131	274	194	0.620	0.653	0.826	0.414
		595	161	199	269	0.680	0.703	0.787	0.574
		586	170	178	290	0.697	0.715	0.775	0.619

The y-axis values vary as we use n-mixed downsampling with a sampling period of milliseconds. The average cumulative score received by the agent is the same across the three discount factor values for the first 200 training epochs. The significance of these training curves is that the RL agent steadily learns the reinforcement signal and therefore the correct action. Training curve in Figure 5.5 a with a discount factor of 0.0, considers immediate reward only and does not take future rewards into account. The agents with discount factor 0.0 are called myopic.



(a) **(b)** **(c)**

Figure 5.5: Three examples of the A2C training curves tracking the agent's average score for different discount factors using n-mixed downsampling. (a) Average score per episode with 0.0 discount factor of future rewards. (b) Average score per episode with 0.5 as discount factor. (c) Average score per episode with 0.9 as discount factor.

5.1.8 Performance metrics of A2C agents for different discount factors

In the following section, Table 5.4 displays the performance metrics for the best trained agents for the three different discount factors. The discount factor of 0.5 has the highest true positives and sensitivity compared with the 0.0 and 0.9 values. The discount factor is determined by the decision maker of the application regarding how much importance should be given to future reward signals.

We do observe a significant difference in the choice of the discount factor. We see that the discount factor of 0.5 results in the best sensitivity of 89.6% but has relatively low specificity, which results in high false alarm rate. The trade-off between specificity and sensitivity is performed by the decision maker for the application. We provide the range of options in the results for the decision maker to make an informed decision.

Table 5.4: Summary of A2C agents results for various discount factors with n-mixed downsampling and milliseconds as sampling period tested on ATOMICS-2 dataset. The top three agents' results are listed per group. Best results are highlighted in bold.

Discount factor	TP	FN	FP	TN	AUC	F1-score	Sensitivity	Specificity
0	599	158	192	276	0.690	0.712	0.791	0.589
	602	155	200	268	0.683	0.707	0.795	0.572
	457	300	**123**	**345**	0.670	0.659	0.603	**0.737**
0.5	586	171	192	276	0.681	0.702	0.774	0.589
	599	158	210	258	0.671	0.695	0.791	0.551
	679	**78**	268	200	0.662	0.697	**0.896**	0.427
0.9	670	87	256	212	0.669	0.703	0.885	0.452
	606	151	196	272	0.690	0.713	0.800	0.581
	594	163	168	300	**0.712**	**0.729**	0.784	0.641

5.1.9 Comparison of agents with rewards only and TD option

The A2C algorithm of Equations 3.1 and 3.3 have temporal difference (TD) error components that contribute to the transition probabilities when time steps are involved in the environment. Since our data samples are i.i.d and not correlated, we experimented in removing the TD error component. In this section, Table 5.5 displays the performance summary results for the two approaches. IIt can be seen that both approaches have similar AUC and F1-score and do not have a significant difference. In such cases we can go with the original A2C with a TD error, which performs better with a higher sensitivity of 89.6%. The reward-only approach appears to perform reasonably well but not as well as the original approach with the TD component. The reward-only approach and the discount factor of 0.0 are equivalent. When the TD component is multiplied by discount factor of 0.0, it results in a reward only option.

Table 5.5: Comparing the results of A2C agents with only reward and with reward + TD error tested on ATOMICS-2 dataset. The top three agents' results are listed per group. Best results are highlighted in bold.

Approach	TP	FN	FP	TN	AUC	F1-score	Sensitivity	Specificity
Reward only	589	168	190	278	0.686	**0.706**	0.778	0.594
	548	209	160	308	**0.691**	0.701	0.723	0.658
	523	234	**147**	**321**	0.688	0.692	0.690	**0.685**
Reward + TD error	586	171	192	276	0.681	0.702	0.774	0.589
	599	158	210	258	0.671	0.695	0.791	0.551
	679	**78**	268	200	0.662	0.697	**0.896**	0.427

5.1.10 Statistical significance tests with rewards + TD error and rewards

We used the Wilcoxon rank-sum test to measure the difference between the sample groups (reward + TD error, rewards only) of the two populations. The two groups of agents are generated by saving the model for every 100 training epochs for a total of 5,000 epochs. We tested these 50 agent samples against the ATOMICS-2 data set to generate AUC values. Our hypothesis is presented below:

$$\begin{aligned} H_0 &= Median(Difference) \leq 0 \\ H_a &= Median(Difference) > 0 \end{aligned} \tag{5.4}$$

The significance level is $\alpha = 0.05$, and the p-value is $p = 0.986$. Since it is observed that $p = 0.986 \not< 0.05$, we fail to reject the null hypothesis H_o. Therefore, we conclude that there is not enough evidence to claim that the population median of differences is greater than 0, at the 0.05 significance level which means the agent is not learning much from the TD error component in this case.

5.1.11 Testing model generalizability

The results of the agent trained on a single sample range of n-1 and tested on n-10 downsampling using ATOMICS-2 dataset are displayed in Table 5.6. are promising compared to traditional classification approaches seen in Table 5.7 group n-10.

Table 5.6: The results of A2C agents trained on n-1 downsampling and tested on n-10 downsampling ATOMICS-2 dataset. The top three agents' results are listed per group. Best results are highlighted in bold.

TP	FN	FP	TN	AUC	F1-score	Sensitivity	Specificity
594	**163**	13430	10609	**0.613**	0.593	**0.784**	0.441
481	276	**10592**	**13447**	0.597	**0.692**	0.635	**0.559**
547	210	12747	11292	0.596	0.618	0.722	0.469

5.1.12 Model interpretability

Healthcare domain demands model interpretability as it involves human safety. In the recent years, many techniques such as saliency maps, activation heat maps and visualization techniques have been developed to explain model predictions to gain users trust. Neural network learning is represented in terms of the weights learned by the network per each layer. Figure 5.6 displays the weights of the six physiological signals across the twenty-four nodes of the layer contributing to the Q-value prediction. We see all the input signals have weights greater than zero, thereby indicating influence on the detection of alarms outcome. We see the EKG (blue line), and the SpO2Pleth (orange line) signals have more influence than the other input features. The two waveform signals (EKG, SpO2Pleth) capture more information at a lower temporal granularity (millisecond) than the other features that are captured at higher temporal granularity (second).

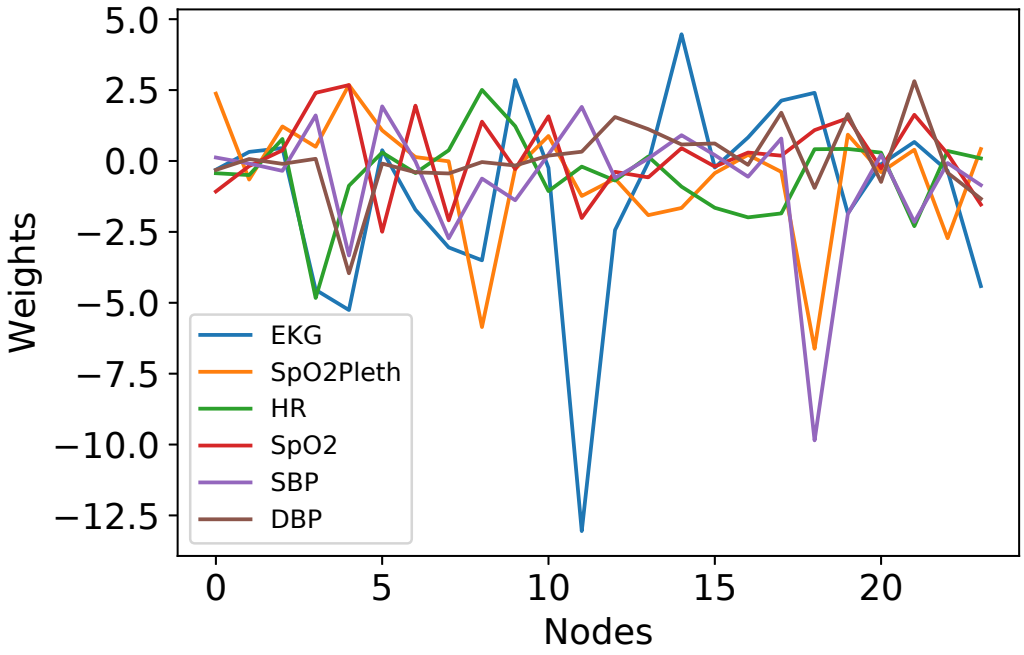


Figure 5.6: A2C neural network model weights.

5.1.13 Q plots

In this section, we show four examples of the Q plots with PPG waveform signals. The best A2C RL agent is used to infer alarms for a single bed monitor data that has alarms and false alarms. We picked 200 milliseconds window slices to plot these results. The x-axis is the time steps in milliseconds. The y-axis is the normalized value of the PPG waveform signal in blue and the Q-values of alarms predicted by the RL agent are in red. The green value is the ground truth as annotated by the expert. The annotations are provided only for events that occur at the trigger of an alarm and are not continuous values as shown in the plot. We represented our actions as 0 for non-alarm and 1 for alarm. The other five variables (EKG, HR, SpO2, SBP, and DBP) do not vary much across the 200 millisecond window and thus not represented in these plots.

The Q-values are probabilities that the RL agent learns during the training for each action. Figure 5.7 a) displays the plot for the true positive example, where the Q-value varies along with the varying PPG signal with a high average Q-value indicating an alarm. In Figure 5.7 b), for the true negative example we do see a lower Q-value. Figure 5.7 c) displays an average Q-value around 0.5 suppressing the false alarm. Figure 5.7 displays the false positive example where the Q-value has a higher average than 0.5. The default threshold of 0.5 is used to discuss the results. In reality, this threshold is decided by the decision-maker and the type of problem.

The Q-values are learnt from the state representation of all the six physiological signals. The Q-plot in Figure 5.7 a) displays the Q-values trending upward towards the alarm event by estimating the future reward, which is the significant value of using RL approach in alarm detection problem. The preliminary experimental results show promising results. The action taken by the agent will get medical attention and intervention in a real use-case scenario. In this research work, we use historical data for the agent to learn the state transitions, and we see the change in q-values as shown in the Q-plot after the alarm to go down, indicating there is a state change to the non-alarm state.

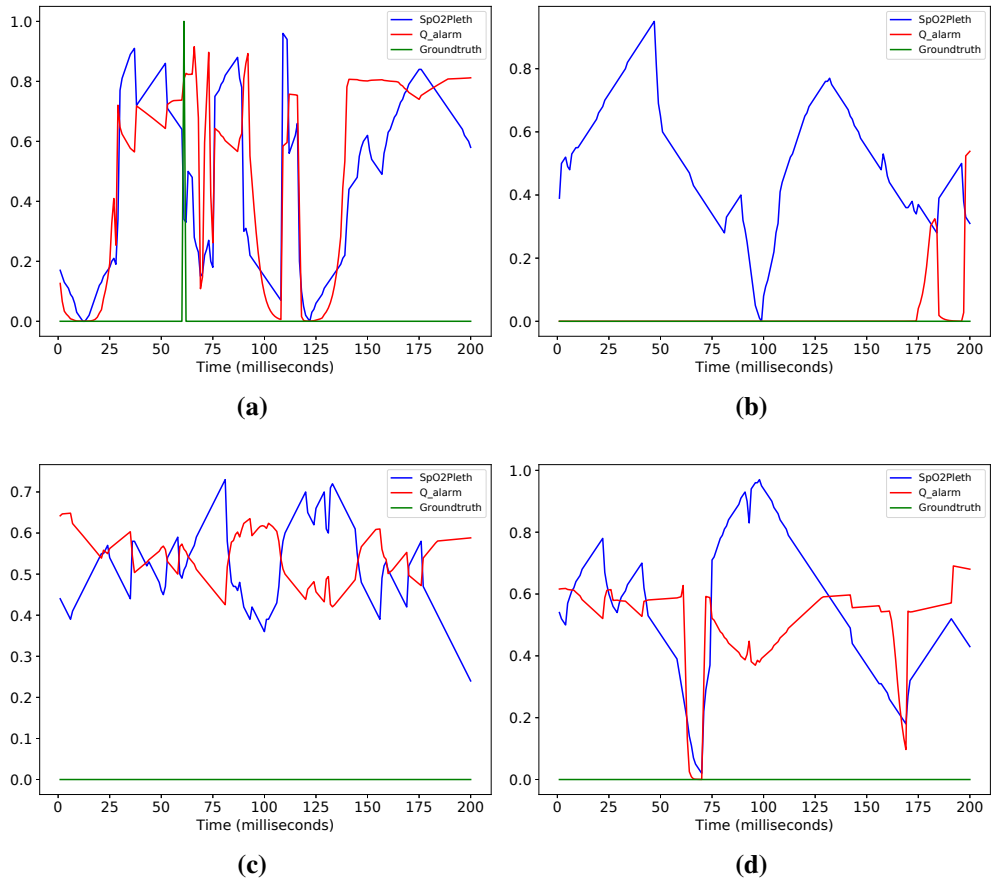


Figure 5.7: Four examples of the alarm Q plots with pulse oximetry (PPG) waveform signals (SpO2Pleth). (a) True positive example. (b) True negative example. (c) False alarm suppressed example. (d) False positive example.

5.2 Benchmarking RL and ML results

We benchmarked our top A2C agent results against DNN and SVM binary classification results for the ATOMICS dataset. We visualized the data using a correlation matrix before training the ML classifiers (DNN and SVM). The correlation heatmap is displayed in Figure 5.8. Correlation heatmaps are one of the visualization technique that helps to see the association between the variables. The SpO2 feature is strongly correlated with the class variable as seen in the heatmap. The DNN is a shallow four-layer network with two hidden layers and an output layer to predict alarm and non-alarm. We used SVM with a linear kernel and default parameters in the svm library.

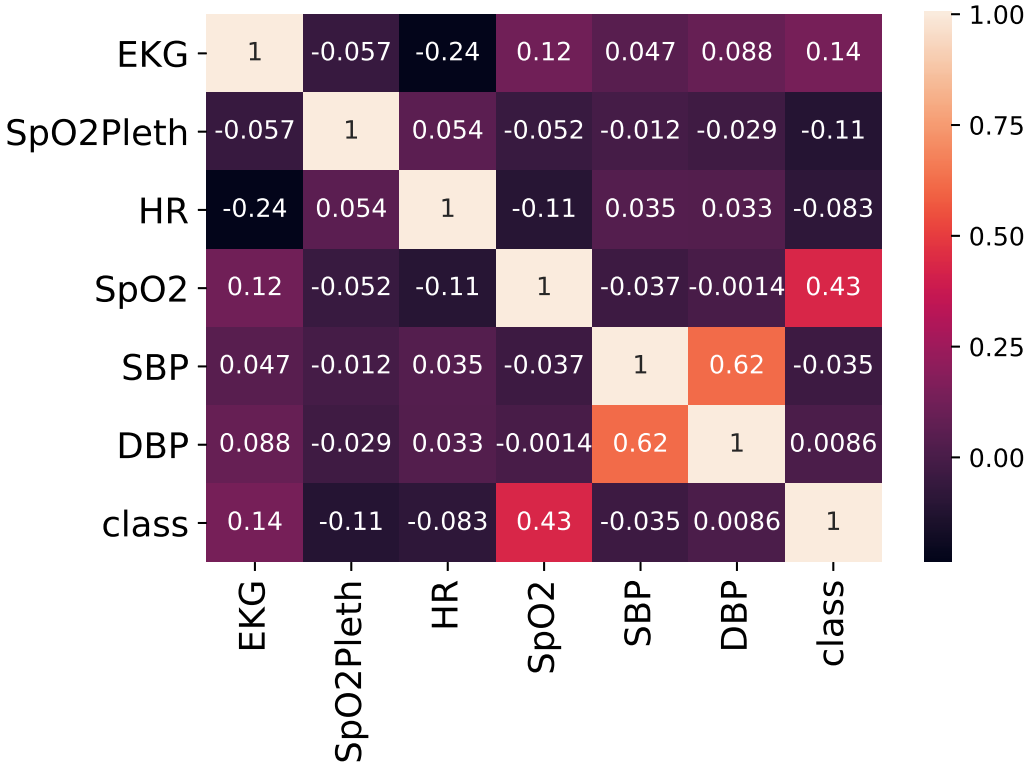


Figure 5.8: Correlation heatmap.

The benchmarking experiments are trained on n-0, n-3, and n-10 downsampling ATOMICS-1 dataset and tested on the ATOMICS-2 alarms dataset. The results are presented in Table 5.7. SVM performs the best for the n-0 sampling range compared to the RL A2C and DNN methods. SVM is a well-known ML non-probabilistic classifier and predominant classification method in

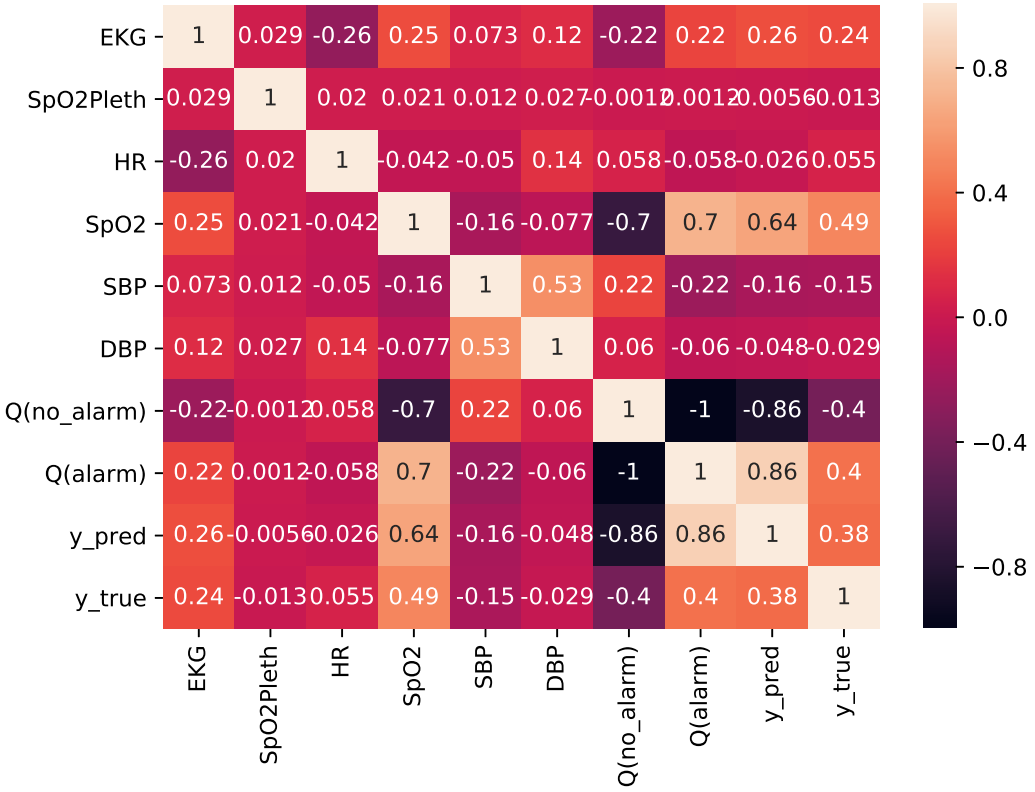


Figure 5.9: Correlation heatmap.

ML literature. For a simple classification task, SVM is probably the best option as seen from the n-0 group results. If we want to scale the problem, meaning if we have more data with non-alarm as seen in n-3, and n-10, SVM performs poorly, indicating that SVM cannot generalize better on newer datasets. A2C performs better as the sample range is increased and learns the new state space and action mapping. We found that the n-mixed downsampling approach with different, randomly selected ranges gives the best results as seen in the last row of the Table 5.7. The correlation heatmap for A2C agent results is displayed in Figure 5.9. The SpO2 feature is strongly correlated with the Q-value of the alarms as seen in the heatmap.

A2C agent with n-mixed downsampling has the highest sensitivity, which is critical in medical domain while keeping the false positives low. A2C agent detects true alarms with 88.5% sensitivity and 45.2% specificity. SVM is a non-probabilistic binary linear classifier that separates the two classes based on the trained examples. A2C methods use a probabilistic approach with a non-linear neural network representation of the feature space. From the initial results we see A2C does much better when we increase the sampling ranges as shown in Table 5.7.

We analyzed our best agent performance with high sensitivity as a target and measured the reduction of false alarms. IIn summary, our promising results for the best agent with a sensitivity of 88.5% is able to reduce the false alarms rate to 45.2%, as seen in Table 5.7. We conclude this RL application of AI-assisted annotation results and discuss our second challenge of model compression.

5.3 Model compression results for model-based approach

Model compression techniques facilitating an efficient deep neural network on hardware and speed constraint devices have more demand and opportunities in the healthcare industry. In this study, we introduced a hybrid RL model compression framework that trains a reinforcement learning agent, iteratively collects samples/experience, and integrates the MB and MF approaches. This automated compression algorithm significantly reduces RL training time and outputs optimal compressed models. The scope of this dissertation is limited to the MB methods employed in this study.

Table 5.7: Comparing the performance results of RL and ML methods tested on ATOMICS-2 dataset. The top three agents' results are listed per group (n-0,n-1,n-10). Best results are highlighted in bold.

Sample range	Network	TP	FN	FP	TN	AUC	F1-score	Sensitivity	Specificity
	A2C	627	130	255	213	0.641	0.673	0.828	0.455
n-0	DNN	505	252	**125**	**343**	0.70	0.696	0.801	0.516
	SVM	**657**	**100**	193	275	**0.727**	**0.754**	**0.867**	**0.587**
	A2C	**403**	**354**	98	370	**0.661**	**0.633**	**0.532**	0.790
n-3	DNN	8	749	**7**	461	0.497	0.222	0.101	0.985
	SVM	0	757	0	**468**	0.5	0.211	0.00	**1.00**
	A2C	**108**	**649**	17	451	**0.553**	**0.371**	**0.142**	0.963
n-10	DNN	8	749	**7**	461	0.497	0.222	0.101	0.985
	SVM	0	757	0	**468**	0.5	0.211	0.00	**1.00**
n-mixed proposed	**A2C**	**670**	87	256	212	0.669	0.703	**0.885**	0.452

5.3.1 Training time comparisons

We benchmarked our data-driven dyna-like model compression (D3MC) to MF RL on ResNet-18 and InceptionV3. The training time comparison is displayed in Figure 5.10.

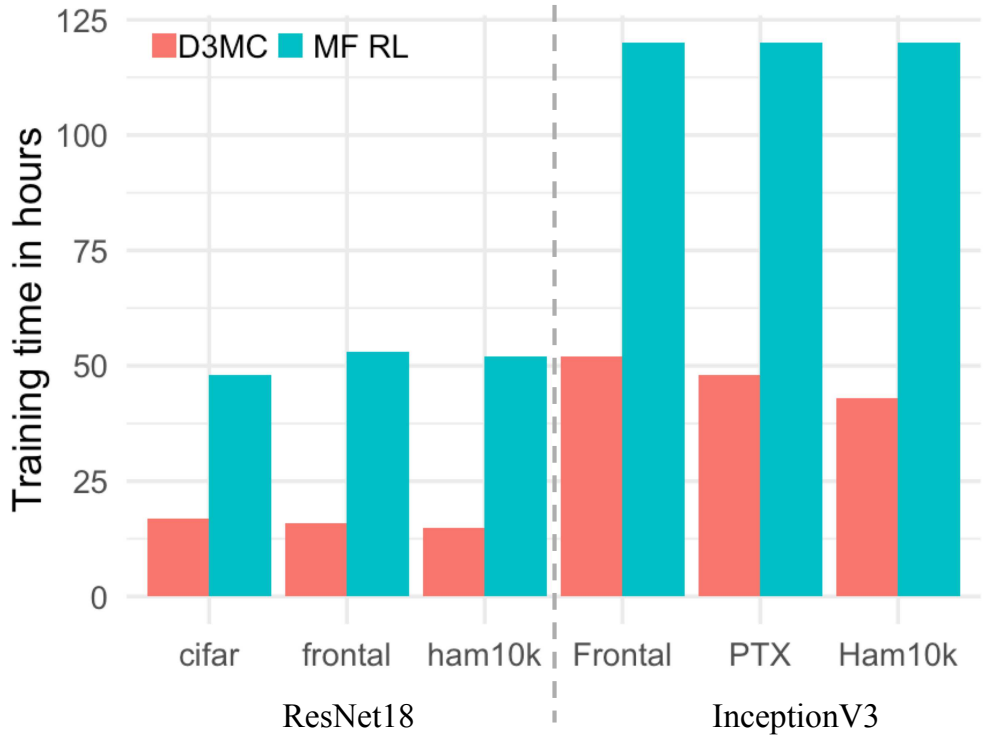


Figure 5.10: Comparison of training time in hours.

The training time of hybrid RL is significantly less for both the ResNet-18 and InceptionV3 architectures compared with MF RL. For example, it required more than 120 hours (five days) to train InceptionV3 using the MF-only approach, while our D3MC shortened this time by approximately 60% (to two days). It is clear that our D3MC is more efficient than the MF RL approaches.

5.3.2 Model compression performance

Our experimental results demonstrate comparable model performance. Table 5.8 illustrates the network compression ratio and model accuracy of the optimal compressed networks. The MF RL rows has an α = 1 (no MB component). The D3MC rows, α is 0.3 for ResNet18 group and

Table 5.8: Summary of results from model compression experiments.

Architecture	Training Data	Method	MB Training Data	Training Time(hrs.)	Teacher Acc.	Δ Acc.
Resnet -18	Cifar10	Model-Free	–	48	86.4%	2.94%
		D3MC	Cifar10	**17**	84.4%	2.75%
	Frontal	Model-Free	–	53	99.5%	0.3%
		D3MC	Cifar10	**16**	99.0%	0.4%
	Ham10k	Model-Free	–	52	82.55%	3.29%
		D3MC	Cifar10 +Frontal	**14.8**	81.47%	2.073%
Inception -v3	Frontal	Model-Free	–	120	99.6%	0.28%
		D3MC	Frontal	**48**	99.59%	0.21%
	PTX	Model-Free	–	120	82.2%	1.96%
		D3MC	Frontal	**52**	81.99%	2.85%
	Ham10k	Model-Free		83	83.69%	3.16%
		D3MC	Frontal +PTX	**27**	81.99%	3.042%

0.5 for InceptionV3 group. Both MF RL and D3MC heavily reduced the size of ResNet-18 and InceptionV3, with minimal impact to model performance. The differences between MF RL and D3MC are small compared with the teacher accuracy. With a slight loss of compression ratio and model accuracy, D3MC provides a significant gain in training time (presented in Table 5.8), faster convergence, and improved generalization across different data sets.

We further conducted a paired Wilcoxon rank significance test of the layer removal across the data sets. We failed to reject the null hypothesis that the paired three groups are identical, with p-values of 0.72 (PTX versus Ham10k), 0.93 (Frontal versus PTX), and 0.74 (Frontal versus Ham10k). This observation suggests that a common layer removal pattern exists across the tested healthcare data sets.

Our model-based compression neural network learned to accurately predict the accuracy for a given student architecture with an MSE of 0.01. The overall D3MC network learned that initial layers are more critical in learning the class information and the end layers are not contributing

to any learning. In InceptionV3 network we see the last two big blocks of layers are removed that contribute to more than 40% of the compression factor. The D3MC approach enables us to streamline our model development across medical devices and network architectures. A renewed interest exists in broad application of AI in healthcare settings to realize its true potential. Reinforcement learning methods that can capture the essential advantages of both MB and MF methods while containing their respective disadvantages are crucial. In this context, our promising results can contribute to breakthroughs for many healthcare applications and contribute to the state of the science of RL applications.

Chapter 6

Conclusions and future direction

Key contributions of this work centered around designing and demonstrating that application of deep reinforcement learning to healthcare domain can significantly improve the quality of real-time decision making in care for better patient outcomes. This work serves as a step toward solving the challenges of medical domain data, such as ground truth availability, for building deep learning models. With the increase of data availability and AI applications, deploying deep learning models on hardware constraint medical devices can be automated with a data-driven model compression framework using reinforcement learning. Our methods are data efficient, scalable, and generalizable.

Healthcare applications using RL are still sparse, as the medical domain is highly complex and requires domain expertise. Domain expertise is needed to annotate high volumes of medical events data, which is both time consuming and expensive. Supervised and semi-supervised approaches of false alarm detection require feature engineering and domain expertise to scale and generalize, which is data intensive and expensive. In this work, we propose an RL approach to mimic medical domain expertise to annotate critical alarms and automate such annotation work with high accuracy. We find the RL approach to be data efficient, scalable, and generalizable for annotation tasks, which are typically costly in the healthcare domain.

Compared with other medical event time series data, this work is distinct, as it does not depend on any longitudinal data. The modeled data pertains to the events at the time of hospitalization. Supervised and semi-supervised learning approaches for identifying clinically false alarms are significantly more difficult than those caused by artifacts and technical errors, as clinical reasoning requires deep knowledge of a patient's high-level physiological state and a significant amount of domain knowledge.

We analyzed our A2C best agent performance with high sensitivity as a target and measured the reduction of false alarms. Our promising results for the best agent with a sensitivity of 78.4% can

reduce the rate of false alarms by 64.1%. Furthermore, our best agent can achieve 88.5% sensitivity in detecting true alarms and 45.2% specificity in identifying false alarms compared with domain experts after analyzing only one week's worth of data. We are confident that our RL agent can learn even better and outperform our initial results with additional training data when the agent is exposed to more newer states.

Artificial intelligence (AI)-driven medical devices have created new excitement in the healthcare sector. While deeper and wider neural networks are designed for complex healthcare applications, model compression can be effective in deploying networks on medical devices that often have hardware and speed constraints. Most state-of-the-art model compression techniques require a resource-centric manual process that explores a large model architecture space to find a trade-off solution between model size and accuracy. Reinforcement learning (RL) approaches have recently been proposed to automate such a hand-crafted process; however, most RL model compression algorithms are model free, which require more time with no assumptions of the model. On the contrary, model-based (MB) approaches are data driven and have faster convergence.

In conclusion, to build smart medical devices, a need exists for efficient model compression techniques. Our experiments have demonstrated promising results on two standard networks used for classification. In this research, we developed an MB model compression algorithm integrated with an MF compression framework to automate such a model compression effort to productionalize AI model development. We evaluated our algorithm on a variety of imaging data, from dermoscopy to X-ray, on different popular and public model architectures. Compared with model-free RL approaches, our approach achieves faster convergence, exhibits better generalization across different data sets, and preserves comparable model performance.

We demonstrate that our data-driven framework integrates the MB and MF approaches to significantly reduce RL training time and output optimally compressed models. We reveal that our method performs well on a variety of healthcare data sets and model architectures. The D3MC framework improved our compression pipeline efficiency and reduced the training time by more than 65%. We demonstrated that our RL agent generalizes across different data sets for a given

architecture and compresses the InceptionV3 network by more than 55% while maintaining comparable model performance.

6.1 Limitations and challenges

Reinforcement learning applications have been seen more advanced in the gaming community than in real-world applications. Many RL examples exist for games, while few real-world application examples exist. Reinforcement learning problem formulation is the most challenging task regarding how to design an environment and agent to learn the expected behavior.

In this false alarm detection work, we are limited by two weeks of ground truth data availability. We limited our experiments to train and test our agents to one week's worth of data. Due to a lack of any similar work of RL application to annotate data, we were unable to benchmark our results with respect to any prior work. To simplify the RL problem space, we merged our critical alarms categories into alarms and non-alarms. The results listed in Chapter 5 are on test data; the actual results may vary on the new dataset. Domain expert annotators used duration as the key criteria to discern the alarms. We did not include this key missing piece of information that should be included in future work.

Similarly, in our MB model compression method, we were unable to benchmark our experiments due to a lack of prior MB methods for compression. In addition, we limited our experiment to only a few architectures and one machine learning problem, classification.

6.2 Key contributions from this work

The overarching research principles this dissertation offers is follows:

T1 - Data and sample efficient methods: We believe that the big-data generated from medical devices can be rich source of AI healthcare applications for a high quality patient outcomes. We demonstrate how to efficiently use the data for broader healthcare applications. Mixed downsampling trained agents have superior performance metrics compared with any single downsampling

approach in this highly imbalanced data set of false alarms. Our approach is more data efficient and is scalable to other tasks.

T2 - Model generalization: In the past, false alarm detection has been focused on specific alarm types requiring domain knowledge. We believe that mimicking the domain expert-annotation behavior, our models generalize better compared to the traditional ML models. We show the agents performance promising initial performance on a completely new set of data.

T3- Scalable models and algorithms: Throughout this dissertation, we believe it is important to scale models and demonstrate that by analyzing the model performance results and the significance, we can scale models and algorithms for broader applications.

Our approach is data efficient, scalable to multiple tasks, and less computationally intensive. All experiments were run on a MAC Book air laptop. Furthermore, such methods could soon pave the way to many practical, non-clinical applications for an improved process to lower the costs of annotation and generate more labeled data for healthcare applications.

6.3 Future direction

Our initial results for detecting false alarms are promising, and we would like to extend this work to specific alarm types (emergent, urgent, indeterminate) and prediction tasks in our future work. We would like to incorporate the duration of events to achieve improved performance, as this was the key expert annotation behavior missing from the current work. We would like to extend this work to other data sets that are publicly available for false alarm detection. The current research work can be applied to any event detection problem that follows the Markov decision process.

In our model compression automation framework, to avoid potential overfitting, we intend to incorporate early stopping in our RL algorithm. One idea is to adopt a compression ratio and/or accuracy constraint. As healthcare projects have different requirements in terms of model sizes and accuracies, such constraints can be used as a terminal state for early stopping. Additionally, we will explore further individual network components that drive compression factors to improve

the efficiency and generalization of the RL agent across various network architectures. We would like to explore compression techniques for segmentation and other machine learning problems. Our RL methods for both false alarm detection and model compression work are generic and can be applied to any domain in which sequential decision making is partially random and partially controlled by the decision maker.

Bibliography

[1] Anton Helman and Shaun Mehta. Em cases summary. https://emergencymedicinecases.com/wp-content/uploads/2018/12/Episode-118-Dec2018-Trauma-1st-Last-15-mins-Part-1.pdf, December 2018.

[2] Patrick Schwab, Emanuela Keller, Carl Muroi, David J Mack, Christian Strässle, and Walter Karlen. Not to cry wolf: Distantly supervised multitask learning in critical care. *arXiv preprint arXiv:1802.05027*, 2018.

[3] Xing Wang, Yifeng Gao, Jessica Lin, Huzefa Rangwala, and Ranjeev Mittu. A machine learning approach to false alarm detection for critical arrhythmia alarms. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pages 202–207. IEEE, 2015.

[4] Lina Zhou, Shimei Pan, Jianwu Wang, and Athanasios V Vasilakos. Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361, 2017.

[5] Cao Xiao, Edward Choi, and Jimeng Sun. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. In *JAMIA*, 2018.

[6] Marzyeh Ghassemi, Tristan Naumann, Peter Schulam, Andrew L Beam, and Rajesh Ranganath. Opportunities in machine learning for healthcare. *arXiv preprint arXiv:1806.00388*, 2018.

[7] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24, 2019.

[8] Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M Kitani. N2n learning: Network to network compression via policy gradient reinforcement learning. *arXiv preprint arXiv:1709.06030*, 2017.

[9] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[10] Omid Sayadi and Mohammad B Shamsollahi. Life-threatening arrhythmia verification in icu patients using the joint cardiovascular dynamical model and a bayesian filter. *IEEE Transactions on Biomedical Engineering*, 58(10):2748–2757, 2011.

[11] Rebeca Salas-Boni, Yong Bai, Patricia Rae Eileen Harris, Barbara J Drew, and Xiao Hu. False ventricular tachycardia alarm suppression in the icu based on the discrete wavelet transform in the ecg signal. *Journal of electrocardiology*, 47(6):775–780, 2014.

[12] Niranjani Prasad, Li-Fang Cheng, Corey Chivers, Michael Draugelis, and Barbara E Engelhardt. A reinforcement learning approach to weaning of mechanical ventilation in intensive care units. *arXiv preprint arXiv:1704.06300*, 2017.

[13] Pablo Escandell-Montero, Milena Chermisi, Jose M Martinez-Martinez, Juan Gomez-Sanchis, Carlo Barbieri, Emilio Soria-Olivas, Flavio Mari, Joan Vila-Francés, Andrea Stopper, Emanuele Gatti, et al. Optimization of anemia treatment in hemodialysis patients via reinforcement learning. *Artificial intelligence in medicine*, 62(1):47–60, 2014.

[14] Shamim Nemati, Mohammad M Ghassemi, and Gari D Clifford. Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2978–2981. IEEE, 2016.

[15] Regina Padmanabhan, Nader Meskin, and Wassim M Haddad. Closed-loop control of anesthesia and mean arterial pressure using reinforcement learning. *Biomedical Signal Processing and Control*, 22:54–64, 2015.

[16] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT press Cambridge, 2nd edition, 1998.

[17] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[18] M. Mahmud, M. S. Kaiser, A. Hussain, and S. Vassanelli. Applications of deep learning and reinforcement learning to biological data. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2063–2079, June 2018.

[19] Oliver Faust, Yuki Hagiwara, Tan Jen Hong, Oh Shu Lih, and U Rajendra Acharya. Deep learning for healthcare applications based on physiological signals: A review. *Computer methods and programs in biomedicine*, 161:1–13, 2018.

[20] Martina Mueller, Jonas S Almeida, Romesh Stanislaus, and Carol L Wagner. Can machine learning methods predict extubation outcome in premature infants as well as clinicians? *Journal of neonatal biology*, 2, 2013.

[21] Hung-Ju Kuo, Hung-Wen Chiu, Chun-Nin Lee, Tzu-Tao Chen, Chih-Cheng Chang, and Mauo-Ying Bien. Improvement in the prediction of ventilator weaning outcomes by an artificial neural network in a medical icu. *Respiratory care*, 60(11):1560–1569, 2015.

[22] Yuanyuan Gao, Anqi Xu, Paul Jen-Hwa Hu, and Tsang-Hsiang Cheng. Incorporating association rule networks in feature category-weighted naive bayes model to support weaning decision making. *Decision Support Systems*, 96:27–38, 2017.

[23] Joachim Behar, Julien Oster, Qiao Li, and Gari D Clifford. Ecg signal quality during arrhythmia and its application to false alarm reduction. *IEEE transactions on biomedical engineering*, 60(6):1660–1666, 2013.

[24] Gari D Clifford, Ikaro Silva, Benjamin Moody, Qiao Li, Danesh Kella, Abdullah Shahin, Tristan Kooistra, Diane Perry, and Roger G Mark. The physionet/computing in cardiology

challenge 2015: reducing false arrhythmia alarms in the icu. In *2015 Computing in Cardiology Conference (CinC)*, pages 273–276. IEEE, 2015.

[25] F Plesinger, P Klimes, J Halamek, and P Jurak. Taming of the monitors: reducing false alarms in intensive care units. *Physiological measurement*, 37(8):1313, 2016.

[26] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[27] Adam Polyak and Lior Wolf. Channel-level acceleration of deep face representations. *IEEE Access*, 3:2163–2175, 2015.

[28] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[29] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[30] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[31] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *CoRR, abs/1611.06440*, 2016.

[32] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[33] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[34] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[35] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838, 2016.

[36] Richard S Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael P Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. *arXiv preprint arXiv:1206.3285*, 2012.

[37] Mnih Volodymyr, Kavukcuoglu Koray, Silver David, A Rusu Andrei, and Veness Joel. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[38] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[39] Leo Kobayashi, Adewole Oyalowo, Uday Agrawal, Shyue-Ling Chen, Wael Asaad, Xiao Hu, Kenneth A Loparo, Gregory D Jay, and Derek L Merck. Development and deployment of an open, modular, near-real-time patient monitor datastream conduit toolkit to enable healthcare multimodal data fusion in a live emergency department setting for experimental bedside clinical informatics research. *IEEE Sensors Letters*, 3(1):1–4, 2018.

[40] Jiahui Guan, Ravi Soni, Dibyajyoti Pati, Gopal Avinash, and V.Ratna Saripalli. *Large-Scale Annotation Of Biomedical Data And Expert Label Synthesis And Hardware Aware Learning For Medical Imaging And Computer Assisted Intervention.*, chapter D3MC: A Reinforcement Learning based Data-driven Dyna Model Compression, pages 98–105. Springer International ublishing, 2019.

[41] Wilcoxon rank-sum test. https://mathcracker.com/wilcoxon-rank-sum.

[42] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[43] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 3462–3471. IEEE, 2017.

[44] Zhiyun Xue, Daekeun You, Sema Candemir, Stefan Jaeger, Sameer Antani, L Rodney Long, and George R Thoma. Chest x-ray image view classification. In *Computer-Based Medical Systems (CBMS), 2015 IEEE 28th International Symposium on*, pages 66–71. IEEE, 2015.

[45] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5:180161, 2018.

[46] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.