Machine Learning for Physiological Time Series: Representing and Controlling Blood Glucose for Diabetes Management

by

Ian G. Fox

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2020

Ian Fox

ifox@umich.edu

ORCID iD: 0000-0002-6580-9893

Acknowledgements

With thanks to Jenna Wiens for advising me, my family for supporting me, and my friends for keeping me sane.

Contents

List of Figures

List of Tables

List of Appendices

Abstract

Type 1 diabetes is a chronic health condition affecting over one million patients in the US, where blood glucose (sugar) levels are not well regulated by the body. Researchers have sought to use physiological data (e.g., blood glucose measurements) collected from wearable devices to manage this disease, either by forecasting future blood glucose levels for predictive alarms, or by automating insulin delivery for blood glucose management. However, the application of machine learning (ML) to these data is hampered by latent context, limited supervision and complex temporal dependencies. To address these challenges, we develop and evaluate novel ML approaches in the context of i) representing physiological time series, particularly for forecasting blood glucose values and ii) decision making for when and how much insulin to deliver. When learning representations, we leverage the structure of the physiological sequence as an implicit information stream. In particular, we a) incorporate latent context when predicting adverse events by jointly modeling patterns in the data and the context those patterns occurred under, b) propose novel types of self-supervision to handle limited data and c) propose deep models that predict functions underlying trajectories to encode temporal dependencies. In the context of decision making, we use reinforcement learning (RL) for blood glucose management. Through the use of an FDA-approved simulator of the glucoregulatory system, we achieve strong performance using deep RL with and without human intervention. However, the success of RL typically depends on realistic simulators or experimental real-world deployment, neither of which are currently practical for problems in health. Thus, we propose techniques for leveraging imperfect simulators and observational data. Beyond diabetes, representing and managing physiological signals is an important problem. By adapting techniques to better leverage the structure inherent in the data we can help overcome these challenges.

Chapter 1

Introduction

Improvements in the quality and affordability of sensors have led to the widespread adoption of wearable health sensors for physiological signal monitoring [1]. The ability to easily monitor an individual's health under normal living conditions presents an unprecedented opportunity to understand, treat, and prevent chronic diseases, such as diabetes, through predicting adverse events (*e.g.*, hypoglycemic events) and suggesting interventions (*e.g.*, insulin doses) [2]. For example, blood glucose data can be collected from individuals with diabetes automatically using widely available continuous glucose monitoring (CGM) technology. However, the scale of these collected data, both on an individual and population level, renders manual interpretation infeasible. Moreover, as these data are collected under normal living conditions, they can be strongly affected by daily events such as meals and activity.

Machine learning (ML) approaches could help in transforming data collected from wearables into actionable knowledge. However, the application of ML to the problem of monitoring and managing blood glucose is not straightforward. Better approaches are needed to learn i) useful representations of longitudinal CGM data and ii) policies to automatically administer insulin and manage blood glucose levels. Learning useful representations of blood glucose data is difficult because of i) the impact of context in the form of daily events, variations in routine, and environmental factors, ii) the potential lack of large amounts of labeled data, and iii) the temporal dependencies between data points. Controlling physiological systems is challenging because of the effects of unobserved daily events on the system, moreover, policies to control these systems must be learned in a safe and sample-efficient manner. In this dissertation, we first examine the challenges of learning representations from physiological time series in greater detail, and then consider the issue of learning control policies.

Throughout this dissertation we use diabetes mellitus (referred to simply as diabetes), in particular type 1 diabetes, as our motivating application. Diabetes is a chronic health condition characterized by the inability of the body to produce sufficient insulin (a hormone) to regulate blood glucose levels. It is well-suited as a motivating application because of its prevalence (there are over 1 million people with type 1 diabetes in the U.S. alone) [3], [4] its severity [5], and because the diabetes community has developed a technical infrastructure for data collection and intervention in non-clinical settings [6]. Moreover, managing diabetes can represent a significant mental burden [7]. Individuals must carefully monitor several key aspects of their life, including meal sizes and physical activity levels, all while keeping track of their blood glucose levels and adjusting insulin doses to avoid hypo- or hyperglycemia [8]. ML can help relieve this burden by assisting people with diabetes in predicting events like hypoglycemia or determining appropriate insulin doses.

1.1 Challenges and Opportunities

In this dissertation, we develop and evaluate new methods designed to address challenges in applying machine learning to blood glucose data collected using wearable health monitors. We focus on overcoming the following challenges:

Lack of Context. To effectively manage diabetes using physiological data, we must find a time-series representation well suited to tasks such as predicting adverse events. Unprocessed time-series data may be unsuitable for standard ML techniques as the signals may be too long or noisy. Time-series motifs, subsets of the signal that are repeated many times, are a common way of representing such data [9]. However, in diabetes, external events such as meals, sleep, and physical activity strongly affect blood glucose levels, and the glucoregulatory system changes over time. Thus, each signal subset must be interpreted according to the context it occurred under. This information is not included in standard motif methods [10], [11].

In **Chapter 3** we examine methods to learn and incorporate this context and develop a contextual representation that can be used to improve adverse event prediction in type

1 diabetes. We propose to learn context in an unsupervised way by assuming it affects long term distributions of patterns in the signal. We show such an approach can learn meaningful and useful time-series representations.

Limited Supervision. Deep learning presents an avenue for representation learning from unprocessed data. While effective, it typically requires a great deal of data to achieve high performance [12]. For longitudinal health monitoring, it is possible collect large amounts of data from any individual. However, we often care about long-term outcomes, such as changes in average blood glucose, limiting the amount of outcome data available. In such cases, one might improve model performance with unsupervised pretraining on additional datasets or multi-task training with additional label sets [13], [14].

In **Chapter 4**, we demonstrate that when dealing with time-series data, the abundance of measurements for each label means neither additional unlabeled data nor additional label sets are required to improve representation learning performance. We show how auxiliary self-supervision can produce significant performance gains without requiring additional data (labeled or otherwise) across a range of tasks and datasets, in particular, patient attribute classification in people with type 1 diabetes.

Modeling Temporal Dependencies. Physiological signals can vary significantly over time for an individual, and trends can be as important as raw values. Thus, while predicting future values of these signals can help inform decision making, predicting entire trajectories of values can provide more clinically value. The task of forecasting, or signal prediction, is particularly important in diabetes, as the delay induced by sensing and actuation devices means that attempts to control changes in glucose can be mismatched with current clinical state. Standard methods to predict trajectories struggle to balance modeling temporal dependencies within the signal and making robust predictions [15], [16].

In **Chapter 5**, we develop new models that are able to navigate this trade-off without modified training schemes by explicitly learning to share information across prediction windows at training time. In addition, we constrain the output of the network, which acts as a form of regularization. We show these improvements lead to better blood glucose forecasting.

Managing Blood Glucose Levels. Continuous health monitoring holds the promise of more timely medical interventions, such as insulin administration. However, it is unreasonable to assume a clinician will be available to suggest interventions, and having patients manage interventions is burdensome. Reinforcement learning (RL) is an approach to learn good autonomous control policies from experience, potentially removing the need for direct patient decision making. In diabetes, the use of an insulin pump managed by an automated controller is called an *artificial pancreas*, and is a focus of diabetes technology research [17]. Current systems, based on classical control methods, are difficult to personalize and need additional human provided signals to handle external phenomena.

In **Chapter 6**, we examine how RL can improve system performance, particularly in the case where there is no human provided information. We demonstrate that a combination of simple and general techniques allows for excellent and safe personalized blood glucose control in a simulated setting. We also show how simulators can be combined with observational data to train RL policies without exploration in the target environment.

While there are significant challenges in the application of ML to health data from wearable devices, learning to represent and control physiological time series is a technically interesting problem that could lead to significant clinical gains. In this dissertation, we present a variety of approaches that deal with problems in representing data, such as limited context and temporal dependencies, and discuss methods to learn to control physiological processes using both simulators and observational data.

1.2 Contributions

To address the challenges discussed above, we present a series of contributions in this dissertation, summarized here:

- **Contextual Motifs**: We develop methods to incorporate context into motif representations of blood glucose data for predicting hypo- and hyperglycemic events. We demonstrate how the meaning of patterns (*i.e.,* motifs) in the signal can vary depending on the context under which they occur. This motivates us to explore methods for contextual motif discovery that jointly learn motifs and infer context, leading to the identification of more informative motifs.

- **Self-Supervised Auxiliary Tasks**: We improve patient classification using auxiliary self-supervised models. Classifying patients can serve as a tractable analogue to understanding the dynamics of an individual's glucoregulatory system. However, this task assigns only one label to a sequence of blood glucose measurements, which can exacerbate small data issues. By training our sequence classification model with additional input-derived supervision, our models learn more discriminative sequence representations.

- **Deep Multi-Output Forecasting**: We improve multi-step blood glucose forecasting with deep multi-output models. Standard methods to predict multiple steps into the future can suffer from accumulating prediction errors or lack coherence across predictions. To rectify this, we explore two complimentary multi-output architectures that learn to correct errors and jointly model the distribution of glucose trajectories.

- **Reinforcement Learning for Blood Glucose Management**: We develop a reinforcement learning (RL) system that achieves strong and safe glucose control over thousands of days of simulated data across a diverse patient population using an FDA-approved simulator. We demonstrate this system can infer meals without requiring meal announcements and show that it can beat classical control techniques that require such announcements. Moreover, we present a comprehensive evaluation of performance and stability using a realistically varying environment. We demonstrate ways to improve the sample efficiency of policy learning, presenting methods that require no patient-specific training data.

The contributions made by this dissertation could be of use to researchers interested in applying machine learning to problems in diabetes management. Concretely, this dissertation establishes effective deep architectures for representing blood glucose data, and demonstrates the promise of deep reinforcement learning for blood glucose management. More broadly, the challenges discussed and techniques developed could be used to guide applications of machine learning to better represent and manage other kinds of physiological signals, such as heart rate or mood, collected under normal living conditions.

Chapter 2

Background

In this chapter, we present a high-level overview of the clinical and technical background for this dissertation. In the first section, we focus on the clinical application of our work, managing type 1 diabetes. In the second section, we survey topics relating to machine learning for time series. More detailed notation and background is presented on a chapter-by-chapter basis for material specific to subsets of the dissertation.

2.1 Diabetes

Type 1 diabetes is a disease where machine learning with time series shows particular promise. Type 1 diabetes is a chronic health disease affecting over 1 million individuals in the US [4], and its rate of occurrence is increasing [18]. This disease is an auto-immune disorder of unknown origin, where the body begins destroying the insulin-producing beta cells found in the pancreas [19]. Insulin is a hormone that signals cells to uptake glucose in the bloodstream. The absence of this hormone is problematic in two ways: 1) without insulin signaling cells to consume glucose, the body must metabolize energy from fat cells via a process known as ketosis, prolonged use of which can lead to a life-threatening condition known as diabetic ketoacidosis [20], 2) when cells are not consuming glucose, large amounts of the sugar can accumulate in the bloodstream in a condition called hyperglycemia, chronically high levels of blood glucose can lead to nerve, eye, heart, and kidney damage [19]. The use of synthetic insulin can allow people with type 1 diabetes to artificially control their blood glucose levels, averting both issues. However, overly large administrations of insulin can lead to insufficient blood glucose levels, a condition called hypoglycemia. Hypoglycemia can be extremely serious, as long-term proper functioning

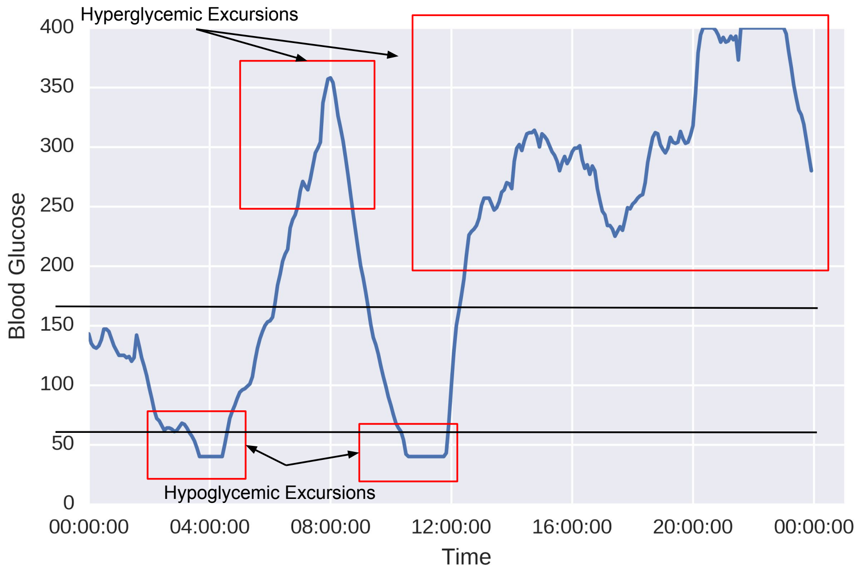


FIGURE 2.1: One day of data in the longitudinal glucose dataset.

of the brain requires a minimum concentration of blood glucose. Severe cases can lead to unconsciousness and sudden death, and chronic hypoglycemia increases risk of heart disease [21]. Tight glucose control, which can reduce the risk of complications in diabetes [22]–[24], can be challenging to maintain. There is a tremendous decision burden placed on diabetic patients who are constantly faced with decisions pertaining to food intake, activities, and insulin administration. Thus there is significant interest in the field to develop sensitive technologies and algorithms to close the loop in insulin delivery and continuous glucose monitoring [17].

2.1.1 Longitudinal Glucose Dataset

Several works in this dissertation (**Chapters 3, 4, 5**) make use of a dataset collected from people with type 1 diabetes. The study was originally designed to evaluate changes in cardiovascular health among individuals with type 1 diabetes. Data from 40 patients was collected over 3 years at 3-month intervals. Continuous glucose monitors were used to collect glucose data from each person for 2-7 days at each interval, resulting in 1.9 thousand days of blood glucose measurements, 550k measurements in total. The blood glucose data was collected at integer resolution between 40-400 mg/dL. Some individuals in

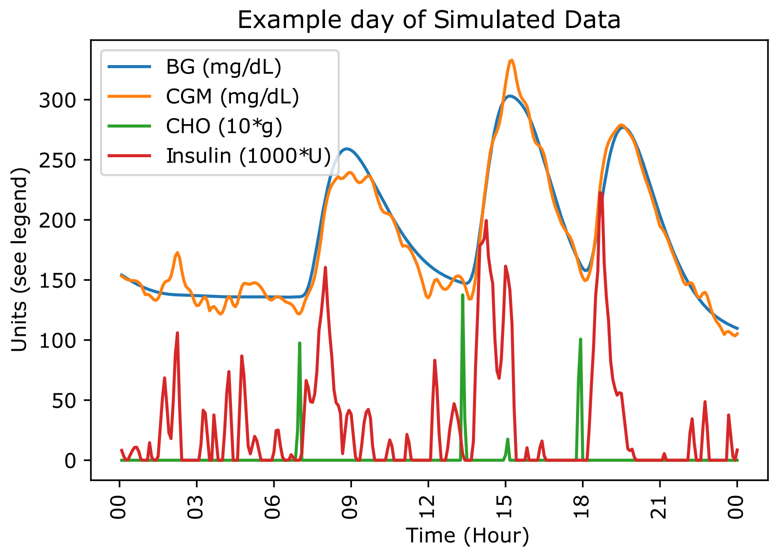


FIGURE 2.2: One day of blood glucose data from the UVA/Padova simulator.

this group used insulin pumps, others used multiple daily injections. While participants were encouraged to manually record insulin usage, physical activity, and meals, few did so, and those who did were inconsistent in usage. As a result, we focus only on the blood glucose measurements in our projects. This is a challenging proposition, as blood glucose is heavily impacted by outside phenomena such as insulin and meals. In collaboration with Dr. Rodica Pop-Busui, the PI on the original prospective study, we have made a deidentified version of this dataset publicly available[1].

2.1.2 Glucose Simulators

In the absence of large blood glucose datasets which include reliable records of daily events (such as meals), simulation strategies have been important for the analysis of blood glucose. Models of the blood glucose system have long been seen as an important component for the development and testing of an artificial pancreas [25]. Models can be used as simulation environments for testing the efficacy of control systems [26], as controllers for administering insulin [27], or as tools to gain deeper understanding of the physiological

[1]https://github.com/igfox/multi-output-glucose-forecasting

processes at work [28]. Current models are built by a combination of rigorous experimentation and expert knowledge of the underlying physiological phenomena. Typical models are built on an underlying multi-compartment model, with various sources and sinks corresponding to physiological phenomena, involving often dozens of patient-specific parameters [26]. In this dissertation, we make use of a publicly available implementation of the UVA/Padova simulator [26] known as Simglucose [29]. An example day of data generated from this simulator is presented in **Figure 2.2**. Note this data can include carbohydrate and insulin information.

2.2 Machine Learning for Time Series

A sequence of data samples from a dynamical system $\mathbf{x}_{0:T} = x_1, x_2, \ldots x_T$ indexed by time is known as a time series. Problems involving time series are found in finance [30], robotics [31], computer vision [32], speech [33], and many other domains. We focus on healthcare, where time-series data can come from sequential EHR data collected in this hospital [34], [35] or physiological time-series collected outside the hospital such as blood glucose measured by continuous glucose monitors [36]. Classical time-series analysis seeks to understand and utilize such data by finding good representations, either by identifying meaningful features, such as well-conserved patterns (motifs) [37], or by modeling the underlying dynamics of the system [38]. Time-series analysis also focuses on predicting future values (forecasting) [39]. While analyzing time series is important for understanding the dynamical system being measured, ultimately such understanding is only useful insofar as it informs action. The control of dynamical systems, also called control theory, is a problem that has been closely considered over the years [38].

Learning from physiological time series, particularly those collected using wearable health monitors, comes with a specific set of challenges. In particular, these signals are often very long and noisy, motivating the use of sparse representations such as motifs. The length of these signals also motivates methods that can draw supervision from the entire signal to learn better representations. As these data measure physiological systems, value trajectories are important to understand and predict future health. Additionally, the underlying physiological systems can be profoundly affected by daily events such as eating or physical activity which may not be recorded in data, so methods that can infer and use latent context can improve modeling. Finally, as these data represent a person's physical health, methods designed to control these systems must prioritize safety and stability. Machine learning approaches can help solve these problems, as discussed in the sections below.

2.2.1 Representation

In this dissertation, we focus on two ways to represent time series: 1) through the discovery of motifs, or repeated patterns in the signal, and 2) using a deep neural network trained end-to-end on some task of interest. Here, we give a brief overview of these approaches, how machine learning can aid in motif discovery, and how deep learning can be effective in smaller data settings.

Motifs

We can represent time series more efficiently by focusing on the most important subsequences of the signal. A subsequence is a contiguous block of measurements lying within the sequence: $\mathbf{x}_{i:j}$ where $i \geq 0$ and $j \leq T$. A meaningful component of a time series tends to be one that is repeated, as such a pattern is less likely to be the result of noise [9]. We call such patterns motifs. Though a motif is properly thought of as a subsequence $\mathbf{x}_{m_i}$, we typically refer to them using their categorical identifier m_i. Motifs are a useful abstraction for representing and analyzing sequential data, such as physiological time-series [40]. Given a sequence $\mathbf{x}_{0:T}$ and a list of n motifs $[m_1, m_2, \ldots, m_n]$, the sequence can be represented by the n-dimensional count vector $\mathbf{x}^m_{0:T} = [p_1, p_2, \ldots, p_n]$ where p_i is the number of instances of motif m_i in $\mathbf{x}_{0:T}$. Motifs exhibit several properties that make them an appealing approach for analyzing large physiological datasets, including their broad applicability across a wide range of data types, their resilience to noise, and their inherent interpretability [10], [40]–[43].

Deep Representation Learning

The assumption that time series are well represented by conserved subsequences has been useful in several domains, but more recent work has suggested the value in learning data representations end-to-end with the ultimate task of interest, typically using deep neural networks [44]. On time-series data, it is important to use models that keep a notion of temporal ordering and are able to accommodate variable input size. As a result, recurrent neural networks are a natural choice when working with time series.

A standard feed forward fully-connected neural network of length n is a defined as a sequence of layers $N = [L_1, L_2, \ldots L_n]$ where each layer consists of some number of neurons and an activation function. Each neuron i in layer j defines a linear function using weight vector $\mathbf{w}_{j,i} \in \mathcal{R}^{|L_{j-1}|+1}$ (the $+1$ is used in place of a bias term). The activation function f_j is a nonlinear sub-differentiable function from $\mathcal{R}^{|L_j|}$ to itself. The layer as a whole defines a nonlinear function $L_j(\mathbf{x}) = f(\mathbf{W}_j\mathbf{x} = [\mathbf{w}_{j,0}, \mathbf{w}_{j,1} \ldots \mathbf{w}_{j,|L_j|}])$. The network composes the functions defined at each layer, allowing for extremely flexible transformations. Importantly, such a model can be trained using gradient descent with the backpropagation algorithm.

A recurrent neural network (RNN) extends standard neural networks by incorporating layers with a notion of state. In the simplest case, this state is the output of the same layer at the previous time step. Consider a single layer RNN. The layer L at time step t takes as input both the current input $\mathbf{x}_t$ and the output at the previous time-step $\mathbf{h}_{t-1} = L(\mathbf{x}_{t-1})$. This allows information from one input to propagate to the next. The states $\mathbf{h_{0:T}}$ used to make predictions. For sequence classification problems, which learn a mapping from a sequence $\mathbf{x}_{0:T}$ to a label y, we typically feed the final state $\mathbf{h}_T$ into an output layer O. For problems involving sequential output $\mathbf{y}_{0:S}$, such as multi-step forecasting, we can use the final state $\mathbf{h}_T$ to initialize a decoder RNN. The state $\mathbf{h}_j^{Dec}$ produced at each step $0 \leq j \leq S$ can be fed to an output layer and trained such that: $O(\mathbf{h}_j^{Dec}) = y_j$. **Chapters 4** and **5** focus on deep learning with RNNs.

Unsupervised Pretraining and Multi-task Learning

Deep learning has had the greatest success where training data are abundant. In settings with limited training data, unsupervised pretraining and multi-task learning have both been shown to help. Both approaches leverage additional available information to improve performance for a task. Let $\mathcal{D}_S = \{\mathbf{x}^i_{0:T}, y^i\}_{i=0}^N$ be a supervised dataset of size N, and say we are interested in optimizing a neural network f_θ where θ represents all tunable parameters.

In unsupervised pretraining, large pools of additional unlabeled data, $\mathcal{D}_U = \{\mathbf{x}^i_{0:T}\}^M_{i=0}$ where $N < M$, are used to help learn good intermediate representations. For example, we can construct a new network $g_{\theta'}$ that is identical to f_θ except for the output layer, which is replaced with a recurrent decoder outputting sequences with the same shape as $\mathbf{x}_{0:T}$. We can train this network as an autoencoder on $\mathcal{D}_U$, then use the parameters in θ' corresponding to layers in f to initialize f. This strategy has previously been used successfully in learning with sequences [14].

Multi-task learning adds additional labeled datasets $\mathcal{D}'_S$ or new labels sets $\mathbf{y}'$ for $\mathcal{D}_S$. The model f_θ is trained on all datasets/label sets simultaneously. Learning multiple tasks jointly allows for information across tasks to be shared, this has been shown to help in a variety of ways [13], [45]. This approach has also proven useful in sequence learning [46]. In **Chapter 4** we examine how auxiliary self-supervision, which is related to unsupervised pretraining and multi-task learning, can be used to improve performance on sequence classification tasks.

2.2.2 Prediction

The representation of a time series is only useful if it improves performance on downstream tasks. Given an input time series $\mathbf{x}_{0:T}$, it is often useful to be able to predict values in the future $\mathbf{x}_{T+h}$. This problem, called forecasting, is a task with a wide variety of important applications in speech processing, energy management, and health [33], [47], [48]. A wide variety of machine learning model classes have been used successfully as forecasters, including Gaussian processes [49], gradient-boosted trees [50], and recurrent/convolutional neural networks [33], [51]. While most approaches focus on predicting a single point in the future $\mathbf{x}_{T+h}$ (single-step forecasting), predicting trajectories of data $\mathbf{x}_{T:T+h}$ (multi-step forecasting) can result in more accurate and meaningful predictions [15].

However, it is not always straightforward to predict trajectories. The two most common approaches are recursive and multi-output forecasting. Under recursive forecasting, a model f is trained to predict $\mathbf{x}_{T+1}$ given $\mathbf{x}_{0:T}$. The models prediction, $\hat{\mathbf{x}}_{T+1}$ is then appended to $\mathbf{x}_{0:T}$, approximating $\mathbf{x}_{0:T+1}$. This is then fed into the model to get a prediction for $\mathbf{x}_{T+2}$ and so on. Viewed probabilistically, this factorizes $p(\mathbf{x}_{T+1:T+h}) =$

$p(\mathbf{x}_{T+1})p(\mathbf{x}_{T+2}|\mathbf{x}_{T+1})\dots P(\mathbf{x}_{T+h}|\mathbf{x}_{T+1:T+h-1})$. While mathematically correct, this formulation is susceptible to exposure bias, or error accumulation over the prediction window [16]. The other main approach, multi-output forecasting, uses a model capable of multiple simultaneous outputs to predict $\mathbf{x}_{T+1}, \mathbf{x}_{T+2}, \dots \mathbf{x}_{T+h}$ given $\mathbf{x}_{0:T}$. In the most naive formulation, this is accomplished by training h different models, each to predict a separate value. While this formulation does not suffer from exposure bias, it also ignores relationships between values in the output window, approximating $p(\mathbf{x}_{T+1:T+h}) = p(\mathbf{x}_{T+1})p(\mathbf{x}_{T+2})\dots p(\mathbf{x}_{T+h})$. In practice, a balance must be struck between modeling dependencies between steps in the trajectory and passing prediction errors made at one step down to the next (called exposure bias). In **Chapter 5**, we explore models that are able to better navigate this trade-off, and how they can improve multi-step forecasting performance.

2.2.3 Reinforcement Learning

While prediction and representation are critical to better understand and manage physiological time series, one of the greatest promises of personal sensors and portable medical devices is the ability for machines to simulate the homeostatic processes of the body. This requires systems that can learn to act within an environment. Reinforcement learning (RL) is an approach to optimize sequential decision making in an environment that is typically assumed to follow a Markov Decision Process (MDP). An MDP is characterized by a 5-tuple (S, A, P, R, γ), where $s \in S$ are the states of the environment, $a \in A$ are actions that can be taken in the environment, the transition function $P : (s, a) \rightarrow s'$ defines the dynamics of the environment, the reward function $R : (s, a) \rightarrow r \in \mathbb{R}$ defines the desirability of state-action pairs, and the discount factor $\gamma \in [0, 1]$ determines the tradeoff between the value of immediate and delayed rewards. The goal in RL is to learn a policy $\pi : s \rightarrow a$, or function mapping states to actions, that maximizes the expected cumulative reward, or:

$$\arg\max_{\pi \in \Pi} \sum_{t=1}^{\infty} E_{s_t \sim P(s_{t-1}, a_{t-1})}[\gamma^t R(s_t, \pi(s_t))], \tag{2.1}$$

where Π is the space of possible policies and $s_0 \in S$ is the starting state. The state value function, $V(s)$, is the expected cumulative reward where $s_0 = s$. The state-action value function $Q(s,a) = R(s,a) + E_{s' \sim P(s,a)}[\gamma V(s')]$ extends the notion of value to state-action pairs. In **Chapter 6** we look at using reinforcement learning to learn a control policy for an artificial pancreas.

Batch Reinforcement Learning. One issue with reinforcement learning on health data is the danger of exploration [52]. While balancing exploration and exploitation is a fundamental problem in RL, in settings such as healthcare, excessive exploration could result in unsafe behavior. Batch RL (also called offline RL) is one solution to this problem [53]. In batch RL, instead of assuming access to an environment with associated transition function P, you assume access to a batch of data collected within this environment $\mathcal{D}_B = \{(s_i, a_i, r_i, s_{i+1}\}_{i=0}^{N}$ where actions are selected using a behavior policy $a_i \sim \pi^B(s_i)$. For the problem of blood glucose management, the behavior policy could be the behavior of the individual being monitored.

Learning policies in such a setting is nontrivial. One supervised approach is known as behavior cloning [54]. In it, the task is to, given a state s_i, predict an action a_i such that $a_i \sim \pi^B(s_i)$, learning to mimic (or clone) the actions of the behavior policy. However, if our goal is to surpass human performance on this task then behavior cloning is insufficient. Off-policy RL, methods that can learn from data collected using policies other than the one being trained, may apply. However, recent work has demonstrated such approaches fail, even when the batch was collected using similar model classes with exploration [54]. However, a special class of algorithms designed specifically for the batch setting have recently demonstrated strong performance in simulated control tasks [55], [56]. These approaches restrict the class of learnable policies, constraining them to behave similarly (according to some metric) to the behavior observed in $\mathcal{D}_B$.

Chapter 3

Predicting Adverse Events with Contextual Motifs

3.1 Introduction

Motifs exhibit several properties that make them an appealing approach for analyzing large physiological datasets, including their broad applicability across a wide range of data types, their resilience to noise, and their inherent interpretability [10], [40]–[43]. For example, we might *discover* a motif representing a rapid heartbeat within an ECG dataset. By representing an ECG signal by whether or not it contains this particular motif, we may learn an association between a rapid heartbeat and a patient's cardiac health. From this we may draw conclusions about how features of the signal are associated with health, e.g.; a rapid heartbeat may be indicative of poor cardiac health.

Despite their useful properties, Van Esbroeck et al. have shown that, as a feature representation for discriminative algorithms, motifs performed worse than other representation learning schemes [58]. We hypothesize that a key reason for this suboptimal performance is that standard motifs ignore the *context* under which they are generated. Standard motif methods assume that the system that generates the signal, and thus the meaning of the motifs, does not change over the observation period. This assumption does not always hold in the context of physiological data.

For example, suppose that the ECG data mentioned earlier was collected under a range of normal living conditions. We would expect a patient's heart rate to depend upon

The following is an adaptation of previously published work [57].

the patient's activity. In fact, the ability of the heart rate to vary to accommodate external stressors, called heart rate variability, indicates good cardiac health [59]. Rapid heartbeats would then be an indicator of good health within the context of physical activity, and an indicator of poor health outside that context.

Another, particularly relevant, example can be found in glucose data (i.e., blood sugar levels). In this domain, motifs may represent spikes in blood glucose levels. When interpreting these spikes, it is important to understand the broader context of the signal. Occasional, isolated spikes may be expected as the result of glycemic challenges (such as meals), whereas repeated spikes in a temporally localized region could indicate poor glycemic control, a risk factor for long-term health outcomes [60], [61].

Our Approach. We address these issues by introducing contextual motifs. Here, we define context as transient external structure affecting the interpretation of a signal, for example, physical activity in ECG, or meals in glucose data. We present a framework for understanding physiological time-series that incorporates time-varying context. By adding context, we hypothesize that we can more accurately model the relationship between the signal and outcomes of interest. Importantly, context can be either observed or unobserved. In our blood glucose example, if patients' meals were never recorded, the context would be unobserved. In this common setting, accounting for context is more challenging. Moreover, what is and is not context may vary depending on the physiological system under consideration. In light of these challenges, in this chapter we:

- propose a general motif framework that accounts for context, ***contextual motifs***, and
- introduce contextual motif discovery along with techniques to jointly infer context and motifs in a signal when context is unobserved.

Applied to both simulated and real physiological data, our proposed approach improves upon existing motif methods in terms of the discriminative utility of the discovered motifs. Using simulated data, we observe a consistent 10-11 percentage point improvement in AUROC when using jointly inferred contextual motifs compared with a contextless baseline. Additionally, applied to a continuous glucose monitoring (CGM) dataset of patients with type 1 diabetes (T1D), our proposed context inference techniques led to a 4 and 7.2 percentage point improvement in the AUROC when predicting hypo- and hyperglycemic events respectively, compared with a state-of-the-art motif discovery method. Upon further analysis, we demonstrate that our method is capturing a physiologically relevant context for glucose data, demonstrating the potential of joint motif-context inference. These results suggest that, even when context is unobserved, contextual motifs could lead to a better understanding of the physiological system and, therefore, more accurate predictions. Code and data for this project is publicly available .

Organization. The remainder of the chapter is organized as follows. First, we present a brief survey regarding the motivation and use of motifs, paying particular attention to the difference between data-derived and data-generating motifs. Next, we describe contextual motifs and how they can be discovered under either motif type. Finally, we present results on real and synthetic data, finding that jointly inferring motifs and context provide the most discriminative patterns.

3.2 Background and Problem Statement

Motifs have been applied across many different fields; including genetic analysis, activity monitoring, and clinical event prediction [11], [62], [63]. For an in-depth review of motifs and their applications, we refer the reader to the survey paper by Mueen [43]. Here, we limit our discussion to prior work that directly relates to our proposed approach. We begin by providing vocabulary for discussing different motif discovery approaches. In particular, we differentiate between *data-derived* and *data-generating* motifs. Then, we present our proposed framework, *contextual motifs*, and discuss related work.

3.2.1 Motifs

History, Definitions, and Notation

The study of motifs originated in the field of genetics [64]. Lin et al. ported the idea of motifs to time-series analysis [65]. While there exist several different definitions of

`https://github.com/igfox/contextual_motifs` time-series motifs, here, we focus on *support motifs* [40], [43]. Support motifs are the subsequences that are most frequently expressed throughout a dataset [43]. These types of motifs are best suited to capturing inter- and intra-sequence similarity, and are widely used as a result [11], [62], [66]–[69].

Before going further, we review some notation and definitions to make our discussion more precise. For the purpose of this chapter, we define a *signal* as an ordered observation sequence of the form $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, where T is the signal length. Signals can be any form of ordered data, but here we focus on data ordered by time, also called *time-series*. A *subsequence* of a signal is a contiguous subset of observations $\mathbf{x_{t,k}} = [x_t, x_{t+1}, \ldots, x_{t+k-1}]$, here the subsequence has length k. We call the process that generates the signal the *system*. Our goal in time-series analysis is to increase our understanding of the system using the observed signal. Some subsequences help us achieve this goal, whereas others do not. Subsequences that occur more often are less likely to be the result of noise, and thus are more likely to reflect structure in the system. This motivates the study of frequently occurring subsequences, or *motifs*.

Previous work has investigated how to increase the discriminative power of motifs. Motifs that are maximally discriminative across classes are known as shapelets [70]. This line of work extends motifs by using a measure of their predictive power during the discovery step. Our work also extends motifs to better predict outcomes. However, we approach this problem in a unsupervised manner. *I.e.,* we focus on learning a richer representation of the signal, leading to the discovery of more informative motifs. Our approach is complementary, and can be applied across different types of motifs.

Data-Derived vs. Data-Generating Motifs

Two common motif types are data-derived and data-generating motifs. Data-derived motifs are subsequences that are *derived* from the signal, data-generating motifs are latent variables that *generate* the signal.

See Table 3.1 for a summary of data-derived versus data-generating motifs.

Data-derived motifs are discovered using a measure of subsequence quality q. A motif m_t is a subsequence

$$m_t = x_{t,k} : q(x_{t,k}) > n$$

TABLE 3.1: Summary of Data-Derived vs. Data-Generating Motifs. Both are important types of motifs, and we consider both in developing contextual motifs.

	Discovery	Advantages	Baseline
Data-Derived	search	efficiency	MDLats [11]
Data-Generating	inference	quality	MMM [10]

for some threshold n. With support motifs, the quality measure q indicates the number of *approximate* occurrences of the subsequence in a dataset. Looking for exactly repeated subsequences is limiting because noise can perturb even well conserved subsequences. Thus, most motif discovery procedures attempt to find approximate motifs.

Data-generating motifs are learned using a generative model to encode assumptions about how the data were created. These motifs are defined as an ordered sequence of distributions

$$m_t = [m_{t,1}, m_{t,2} \ldots, m_{t,k}] : m_{t,i} \in \mathcal{D} \text{ for } i = 1, \ldots k$$

where $\mathcal{D}$ is some family of distributions.

The primary difference between these motif types is the method they are discovered by. Data-derived motif discovery methods focus on search. In contrast, data-generating motif discovery methods emphasize inference. Both approaches are useful to consider. Though data-derived motifs are easier to discover and thus more common in the literature [11], [43], [65], [66], [71], data-generating motifs (also called latent motifs), are typically higher quality [67], [72]. Viewing motifs as data generating also allows for a natural extension to discovering deformable motifs, which can be relevant in the analysis of physiological data [72].

Discovering Motifs - Baseline Methods

We consider two motif discovery algorithms as baselines. The first is a state-of-the-art-motif discovery algorithm, MDLats. Presented by Liu et al. [11], MDLats combines many of the methodological improvements developed in recent literature into an efficient and effective method for physiological signals [9], [11], [66]. We will show how MDLats can be applied in a contextual motif framework to further improve discriminative performance.

The second baseline we consider, based on a data-generating motif discovery algorithm proposed by Bailey et al., is a Motif Mixture Model (MMM)[10]. Their method is designed for categorical genome data, and thus fits a mixture of categorical distributions. As we are working with real-valued physiological time-series, we instead use a Gaussian Mixture Model. We assume each portion of a signal is generated by a set of distributions called a motif. For one signal, **x**, the generative model is given in **Algorithm 1**.

ALGORITHM 1: Generative model for Data Generating Motifs

```
Data: Signal length $|\mathbf{x}|$, motif length $|\mathbf{m}|$, number of motifs $n_m$, motif mixing
      parameter $\gamma$, and motif parameters $\{\boldsymbol{\theta}_j\}_{j=1}^{n_m}$
Result: motif representation of signal
for $i = 0, \ldots \frac{|\mathbf{x}|}{|\mathbf{m}|}$ do
    Pick motif $m_i \sim \text{Cat}(\gamma)$;
    for $k = 0 \ldots |\mathbf{m}| - 1$ do
        Draw observation: $x_{i|\mathbf{m}|+k} \sim \mathcal{N}(\boldsymbol{\theta}_{\mathbf{m}_i,\mu_k}, \boldsymbol{\theta}^2_{\mathbf{m}_i,\sigma_k})$;
    end
end
```

Under this definition, even noisy, poorly conserved subsequences are technically motifs. It is common to include a *background* motif, assumed to be poorly conserved, to account for such subsequences. Inference with this model is identical to inference with a Gaussian Mixture Model. Each mixture component, representing a motif, is an $|\mathbf{m}|$-dimensional Gaussian with a spherical covariance matrix.

3.2.2 Contextual Motifs- A Novel Extension

We extend the work above to incorporate context. We represent context as a categorical variable $c_t \in C$ that, at time point t, takes a discrete value $c_t \in \{1, \ldots, n_c\}$, where n_c is the number of distinct contexts. Contextual motifs are then tuples of the form (m_t, c_t), representing a motif and the context under which it occurs. Contextual motif discovery is the task of discovering these tuples. Discovery can also be viewed as the process of discovering motifs occurring within similar contexts. This distinction can be important when contexts vary from signal to signal. Without taking context into account, our measure of motif quality may mistake infrequent contexts with infrequent motifs within a context.

For example, a certain quantity of food consumption may always present a distinctive blood glucose pattern in individuals with poor glycemic control. If some signals in our CGM dataset contained large meals, and others did not, contextless motif discovery could fail to recognize the prevalence of this pattern.

We present methods that extend both data-derived and data-generating motifs to contextual motifs. When extending data-derived motifs, we focus on independently inferring context in a two-stage approach. Data-generating motifs can be extended using either a two-stage approach or by jointly inferring motifs and context. To perform this joint inference, we propose a generative model based on a subclass of dynamic Bayesian networks [73].

Despite the vast literature studying motifs, and motif discovery methods, there is relatively limited work on considering abstractions based on motifs. Still, we discuss the related work in this area and how it differs from what we propose.

Van Esbroeck et al. considered representing physiological signals using a bag of motifs (a common approach in motif discovery) [74]. The authors built upon this representation using a topic modeling approach. In their approach, each topic is associated with a distribution over motifs and each signal is then a distribution over topics. These topics are then used as an abstraction to represent the signal. While the idea of a topic is conceptually similar to context, their method differs from our proposed method in two ways. First, we jointly infer contexts and motifs, whereas they assume that motifs are first discovered and then topics are inferred. Second, because of the bag-of-motifs approach the topics they learn assume a static representation of the signal. In contrast, our approach leverages the temporal contiguity of contexts to allow for flexible variation in motif representations both within and across signals.

In other domains, where the order motifs occur is important (e.g., genetics) researchers have considered leveraging this temporal ordering to discover better motif representations. Lin et al. proposed a method based on hierarchical HMMs where the presence/absence of motifs in one part of a signal affects the presence/absence of motifs in neighboring parts [75]. Again, this is conceptually similar to context, since context can be viewed as a type of inter-motif structure. However, like the method described above, Lin et al.'s method also requires a separate motif discovery step. Our approach combines the discovery steps, enabling joint motif-context discovery and reducing the reliance on prior knowledge.

Finally, others have looked at non-motif-based approaches to learning abstractions based on time-series data. For example, Saria et al. examined the joint discovery of generating functions and temporal topics in NICU data [76]. Our approach to joint discovery is similar, though we focus on discovering motifs as opposed to generating functions, since we are interested in preserving interpretability.

3.3 Methods

In this section, we present the main technical contributions of this chapter. We begin by discussing contextual motifs applied to data-derived motifs. We introduce methods to discover data-derived contextual motifs when context is observed and when context is unobserved, using a two-stage context inference procedure. We then discuss data-generating contextual motifs. In addition to observed context methods and two-stage context inference, we introduce a method to jointly infer context and motifs.

3.3.1 Data-Derived Contextual Motifs

For completeness, we begin by briefly describing a method to discover contextual motifs when context is observed. In the more likely case of unobserved context, we present techniques to infer context in two stages: first inferring context followed by inferring motifs.

Observed Context

When context is observed, we aim to discover motifs *within* similar contexts. This results in the simple extension outlined in Algorithm 2

This framing of contextual motifs can be thought of as a specialized instance of multivariate motifs. However, by framing the problem as we have, each context can be mined for motifs independently of the others. This enables efficient, parallel motif discovery.

Compared to a standard approach that ignores context, this approach may perform worse if a motif does not occur enough times in any particular context to be discovered. However, when there is not enough per context information to discover useful motifs, it

ALGORITHM 2: Contextual Motif Discovery

```
Data: dataset of signals $\mathbf{x} \in X$, associated set of context signal $\mathbf{x}_c \in X_C$, motif
      discovery function $MD$.
Result: set of contextual motifs $cm = (\mathbf{m}_i, c_i)$
$cm = \{\}$;
for $i = 1, \ldots |C|$ do
    $\forall \mathbf{x} \in X$: find all maximal subsequences $\mathbf{x}_{c_i} \in x$ where $\mathbf{x}_c = i$ over the entire
     subsequence, denote the set of all such $\mathbf{x}_{c_i}$ as $X_{c_i}$;
    $M_{c_i} = MD(X_{c_i})$;
    for $\mathbf{m} \in M_{c_i}$ do
        append $(\mathbf{m}, i)$ to $cm$;
    end
end
```

is likely that non-contextual motif discovery techniques are already tractable, mitigating this issue.

Two-Stage Inference: Context then Motifs

The above method assumes a fully observed context signal X_C. This is not, in general, a reasonable assumption to make. The passive collection of contextual information may be impossible due to constrained resources or limitations in sensor technology, and it can be burdensome for patients to record context manually. Even if recording context is feasible, it is not always available in retrospective physiological data. Thus, we aim to infer context based on the signal.

We present two methods to accomplish this:

1. Expert-Driven: Using domain knowledge one may hand-engineer features indicative of certain contexts. For example, large spikes in blood glucose levels typically occur only after meals [77], [78]. Using this knowledge, we created an expert rule,

$$c_t = b\left(\frac{\sum_{i=1}^{k} \mathbf{x}_{t-(i-1)} - \mathbf{x}_{t-i}}{k}\right)$$

   where b is a hand-defined mapping from $\mathbb{R} \rightarrow C$. However, such methods require domain expertise, and fail to generalize to arbitrary domains.

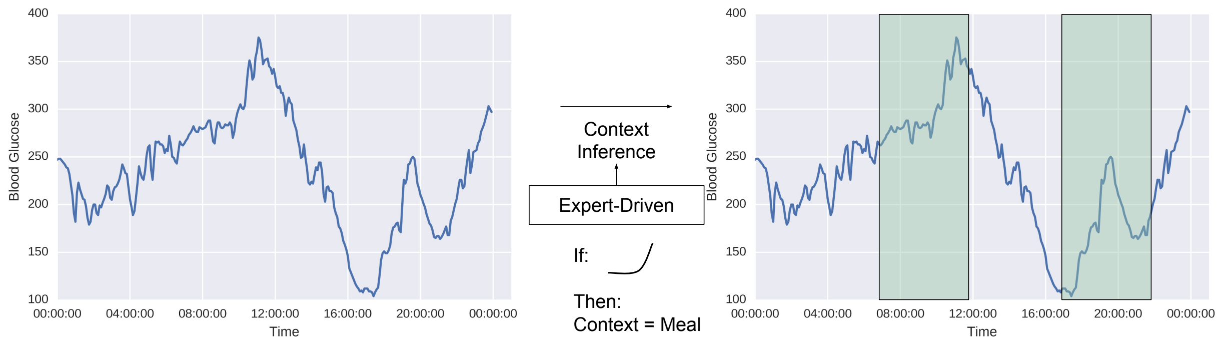


FIGURE 3.1: When expert/domain knowledge is available, we can discover context using hand-engineered rules. The rule illustrated here assigns a 'meal' context to a window (highlighted in green) around a sharp increase in blood sugar.

2. Data-Driven: Using unsupervised methods, one can infer transient structure in time-series data. For example, HMMs could be used to infer hidden states across the data. These inferred hidden states could then be used as a context for contextual motif methods.

We have presented methods to discover data-derived contextual motifs, both when context is observed and unobserved. We now explain how to discover data-generating contextual motifs.

3.3.2 Data-Generating Contextual Motifs

The use of data-generating motifs allows for the utilization of more sophisticated inference procedures when discovering motifs. These discovery methods can be used in an observed context setting identically to the data-derived search methods, using Algorithm 2. Thus, here we focus entirely on the setting where context is unobserved. We begin by briefly introducing another version of the two-stage inference presented in Section 3.3.1. Then, we present an approach to jointly discover motifs and context.

Two-Stage Inference: Motifs then Context

In Section 3.3.1, we describe two context inference procedures that infer context directly from the signal. However, as work in the genetics and activity monitoring literature indicates, motifs can be a useful primitive for deriving context [62], [75]. This motivates a reversed two-stage inference approach, where we first infer motifs and then context.

We introduce a new type of context. This context, defined over fixed window lengths of size $|\mathbf{c}|$, encodes information about the motif mixing parameter γ. A detailed view of this context discovery approach is given in Algorithm 3.

ALGORITHM 3: Data-Generating Context Discovery

```
Data: dataset $X$, where each signal $\mathbf{x} \in X$ is represented as a time-ordered list of
      motifs $[\mathbf{m}_{i_1}, \mathbf{m}_{i_2}, \ldots \mathbf{m}_{i_{\frac{|\mathbf{x}|}{|\mathbf{m}|}}}]$, context window length $|\mathbf{c}|$, number of context $n_c$
Result: Context labels $X_C$ over the dataset
protocontexts $C_p = \{\}$;
for $\mathbf{x} \in X$ do
    Partition $\mathbf{x}$ into context windows $w_i = \mathbf{x}_{|\mathbf{c}|i:|\mathbf{c}|(i+1)}$;
    for $w_i \in \mathbf{x}$ do
        Append motif frequencies in $w_i$ to $C_p$;
    end
end
Cluster $C_p$ using KMeans clustering with $k = n_c$;
Return resulting cluster labels as $X_C$;
```

We note that both variants of two-stage context inference are equally applicable to data-derived and data-generating motifs. These variants of two-stage inference imply:

1. context can improve motif discovery, and
2. motifs can improve context inference.

The combination of these two implications motivates a *joint* context and motif discovery procedure.

Joint Inference

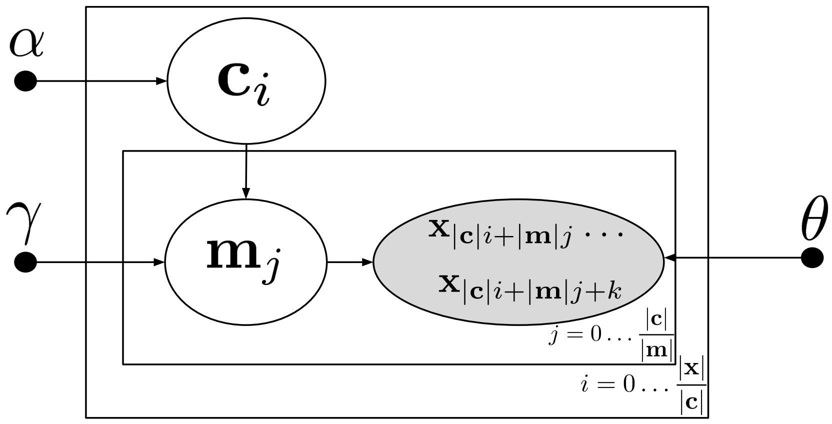
While data-generating discovery techniques carry additional computational cost, the increased model flexibility allows for the joint discovery of motifs and context. To jointly discover motifs and context, we modify the data-generating MMM (introduced in Section 3.2.1) to include a context variable. In our proposed *contextual motif mixture model* (CMMM), each context c_i defines a distribution over the set of possible motifs (Figure 3.2). The generative model is given in **Algorithm 4**.

ALGORITHM 4: Generative model for Data Generating Contextual Motifs

```
Data: Context window length $|c|$, number of contexts $n_c$, context mixing
      parameter $\alpha$, signal length $|\mathbf{x}|$, motif length $|\mathbf{m}|$, number of motifs $n_m$,
      context-dependent motif mixing parameter $\{\gamma_j\}_{j=1}^{n_c}$, and
      context-independent motif parameters $\{\boldsymbol{\theta}_k\}_{k=1}^{n_m}$
Result: contextual motif representation of signal
for $i = 0 \dots \frac{|\mathbf{x}|}{|c|}$ do
    Pick context $c_i \sim \text{Cat}(\alpha)$ ;
    for $j = 0 \dots \frac{|c|}{|m|}$ do
        Pick motif $m_j \sim \text{Cat}(\gamma_{c_i})$;
        for $k = 0 \dots |\mathbf{m}| - 1$ do
            Draw observation: $x_{j*|\mathbf{m}|+k} \sim \mathcal{N}(\boldsymbol{\theta}_{\mathbf{m}_j,\mu_k}, \boldsymbol{\theta}^2_{\mathbf{m}_j,\sigma_k})$;
        end
    end
end
```

This model can be viewed as a hierarchical HMM with a feed-forward context layer. Here, our notion of context is the context introduced in 3.3.2, which is a temporally contiguous extension of the topics used in the motif topic model of Van Esbroeck et al. [74]. In contrast to their approach, we allow the motif distribution parameter, γ, to vary over time according to a categorical context variable. We explicitly model motif mixing distributions for each context, but hold the motif parameters constant across all contexts. This allows for the identification of similar contexts across a population.

Using this model, one can infer the specific contexts under which each motif occurs at the same time one infers the motifs, allowing for the discovery of contextual motifs



α
$\mathbf{c}_i$
γ
$\mathbf{m}_j$
$\mathbf{X}_{|\mathbf{c}|i+|\mathbf{m}|j} \cdots$
$\mathbf{X}_{|\mathbf{c}|i+|\mathbf{m}|j+k}$
θ
$j = 0 \ldots \frac{|\mathbf{c}|}{|\mathbf{m}|}$
$i = 0 \ldots \frac{|\mathbf{x}|}{|\mathbf{c}|}$

FIGURE 3.2: Proposed generative model for the contextual motif generative process. Each context parameterizes a different distribution of motif frequencies.

from scratch. This model can be viewed as a mixture of mixture models, and thus exact inference on it is intractable. Note that computing the posterior of the latent variables and parameters requires computing:

$$\frac{p(\mathbf{c}, \mathbf{m}, \alpha, \gamma, \boldsymbol{\theta}, \mathbf{X})}{\int_\alpha p(\alpha) \sum_\mathbf{c} p(\mathbf{c}|\alpha) \int_\gamma p(\gamma) \sum_\mathbf{m} p(\mathbf{m}|\gamma, \mathbf{c}) \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}) p(\mathbf{x}|\boldsymbol{\theta}, \mathbf{m})}$$

To perform approximate inference on this model, we use a sampling approach, as is routine with data-generating motif systems [40], [75], [79]. Our sampling method was implemented using the probabilistic programming python module PyMC3 [80]. We sample the categorical variables using a metropolized gibbs sampler with a uniform proposal distribution [81]. The motif and context mixing parameters are assumed to have Dirichlet priors, sampled using Metropolis-Hastings with a normal proposal distribution. To sample the motif distribution parameters, we use a No-U-Turn Sampler with dual averaging [82]. These sampling schemes use the default parameters in PyMC3.

3.4 Experiments and Results

We examine the utility of contextual motifs through a series of experiments. In our experiments we seek to answer the following questions:

- Do data-derived contextual motifs improve on the discriminative ability of standard data-derived motifs? (**Section 3.4.2, Table 3.2**)
- Does considering context improve the discriminative ability of data-generating motifs? (**Section 3.4.2, Table 3.3**)
- How effectively does our contextual motif mixture model uncover known contextual structure? (**Section 3.4.3, Figure 3.4**)

To answer these questions, we discover data-derived and data-generating (contextual) motifs in both simulated and real physiological datasets using our proposed methods. We measure the utility of the discovered (contextual) motifs by using them as features in a series of classification tasks. On a separate test set, we compare the discriminative performance of contextual motifs to that of regular motifs discovered using existing, state-of-the-art methods. We begin by describing the data we use for these experiments and our evaluation procedure. We then present a series of experiments using both data-derived and data-generating motifs on the physiological data. Since the physiological data only represents a single scenario, we also present experiments on simulated data over a range of settings.

3.4.1 Dataset and Prediction Tasks

CGM Data

The dataset used for this project is the Longitudinal Blood Glucose dataset described in **Section 2.1.1**. Due to calibration and sensor errors, the sessions in this dataset contain periods of missing data. In total, the dataset contains approximately 51.6k hours of glucose data and is missing around 4.5k hours. To handle this missingness, we use linear interpolation. This is sensible for small chunks of missing data, as glucose data is locally linear. However, we exclude days with contiguous periods of missing data longer than thirty minutes. After removing days with excessive missing data, 32.4k hours of data occurring across 1,352 day remains. Each of the 40 patients contributes between 21 and 48 days to this total.

Evaluating Motifs: Prediction Task

To measure the utility of the discovered contextual motifs relative to motifs, we represent each day's data by the number of times each (contextual) motif appears in the signal and use this representation in a supervised learning task. We transform each signal into a feature vector representing motif counts. Given the fixed length feature representation, any number of machine learning techniques can be used to learn a mapping from the feature vector to the label.

Given the CGM data, we consider two different supervised learning tasks related to T1D: hypoglycemia (low blood sugar) and hyperglycemia (high blood sugar). Hypoglycemic events are caused by an over administration of insulin, skipping meals, or increased physical activity. They can result in unconsciousness, seizures, and even death [83]. Hyperglycemic events are caused by overconsumption of carbohydrates or an under administration of insulin. In the short term, severe hyperglycemic events can result in coma or can lead to life threatening diabetic ketoacidosis. In the long term, chronic hyperglycemia increases risk for cardiovascular disease, neuropathy, kidney damage, and early death [84].

To predict these events, we use one day of data to predict whether a hypo- or hyperglycemic event will occur in the next day. Due to the large number of hyperglycemic events that occur in our data, we modified the prediction task to predict the occurrence of two or more hyperglycemic events.

We selected these two tasks for their clinical relevance, but also since they differ substantially in terms of the underlying pathophysiology. Learning a representation that works well across both tasks is more challenging, but ultimately, the goal of this work.

We do not focus on the choice of supervised learning algorithm, as our contribution lies in the feature representation method. For consistency, we use a pipeline constructed using Scikit-Learn [85] that tunes hyperparameters and fits a logistic regression classifier using patient-aware stratification for cross validation and train-test splits. An outline of our evaluation process is given in Figure 3.3. We report results in term of average area under the receiver operating characteristic curve (AUROC) across train-test splits.

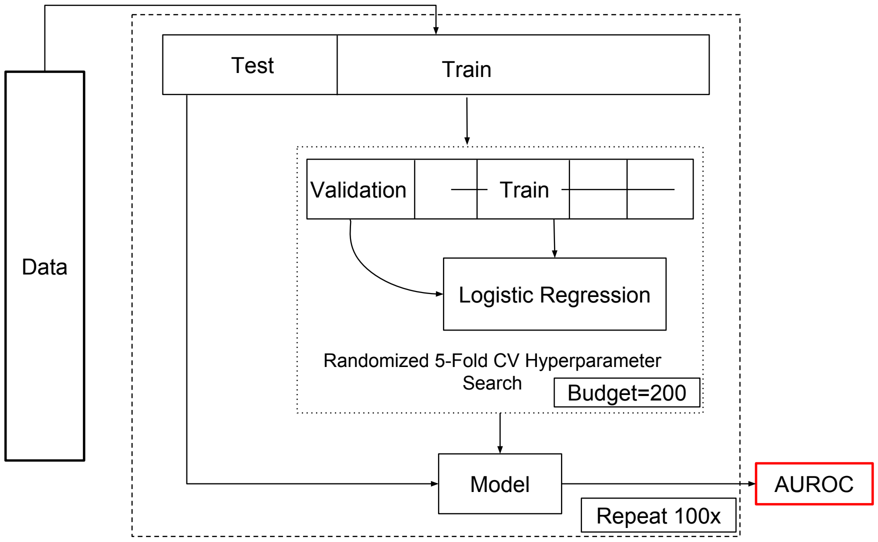


FIGURE 3.3: Outline of evaluation process. Data is split, using a patient aware scheme, into train-test splits 100 times. For each time, the training data is used to train a logistic regression classifier, using cross validation to perform randomized hyperparameter selection. The AUROC of the trained model is then calculated on the test data.

3.4.2 Experiments on Real Data

We now present the results of our contextual motif methods applied to our CGM dataset. We begin by examining the performance of our data-derived contextualize motif techniques, followed by an evaluation of our data-generating techniques.

Data-Derived Motifs

Our first set of experiments explores the utility of two-stage inference for data-derived motifs. Using the physiological data described in Section 3.4.1, we use the state-of-the-art data-derived motif discovery method MDLats to discover motifs in three ways:

- Baseline: without context,
- Expert-Driven: using two-stage inference to add context defined according to the domain rule discussed in Section 3.3.1. This rule is based on the clinically validated measure of glycemic variability, MAGE [77],

TABLE 3.2: Predictive utility of contextual data-derived motifs. We observe an increase in AUROC when including expert- and data-driven contextual information, though the data-driven context is more informative. All differences shown (except between baseline and expert-driven Hyper) are significant ($p < 0.001$).

Method	AUROC (std)	
Context	Hypo	Hyper
Baseline	0.535 (0.022)	0.527 (0.020)
Expert-Driven	0.552 (0.023)	0.534 (0.020)
Data-Driven	**0.607 (0.023)**	**0.567 (0.022)**

- Data-Driven: using two-stage inference to add context defined as the hidden state sequence in an HMM, as described in Section 3.3.1.

We use the learned motif representations to predict both hypo- and hyperglycemic events, using the pipeline discussed in 3.4.1. To faithfully re-implement MDLats, we discretized our data using the SAX representation technique[86].

The results are presented in Table 3.2. At first glance, the results of current state-of-the-art are not encouraging. MDLats (or contextless motifs) achieves an AUROC=0.535 and AUROC=0.527 for the tasks of predicting hypo- and hyperglycemic events respectively. But this level of predictive performance is not uncommon for such difficult tasks. Moreover, if we focus on the differences in performance, incorporating data-driven context leads to a significant increase in performance across *both* tasks (AUROC=0.607 and AUROC=0.567).

Data-Generating Motifs

In our second set of experiments, we shift to data-generating motifs to test our joint inference approach. Furthermore, this allows us to measure the value of contextual motifs in a different setting (i.e., data-derived vs data-generating). We compare the MMM (presented in Section 3.2.1) against our CMMM on the glucose dataset.

We applied the MMM and CMMM to the CGM data. With the MMM, we can infer context using a *two-stage* process, the first stage is motif discovery and the second stage is contextual inference. This results in a `(motifs, context)` representation that can be

TABLE 3.3: Predictive utility of contextual data-generating motifs. We note two main trends in these results. First, the CMMM consistently outperforms the MMM (difference between CMMM and MMM with no context and inferred context significant with $p < 0.001$). Second, the inferred context improves performance over the noise context (significant in CMMM with $p < 0.001$)

Method		AUROC (std)	
Model	Representation	Hypo	Hyper
MMM	`(motif)`	0.517 (0.023)	0.588 (0.026)
	`(motif, noise)`	0.486 (0.018)	0.575 (0.029)
	`(motif, context)`	0.496 (0.018)	0.578 (0.024)
CMMM	`(motif)`	**0.539 (0.026)**	**0.595 (0.025)**
	`(motif, noise)`	0.510 (0.025)	0.577 (0.023)
	`(motif, context)`	**0.533 (0.020)**	**0.591 (0.029)**

used in prediction tasks. We also consider the utility of `(motifs)` alone (without inferring context).

Recall, that for the CMMM, context and motifs are discovered jointly. To tease apart the individual contributions, we consider the performance of models trained on only `(motifs)` vs. `(motifs, context)`.

In addition to the `(motifs)` vs. `(motifs, context)` we also consider a third representation: `(motifs, noise)`. Here, we have added a 'noise' context to more explicitly compare the value added by our contextual inference procedure. By adding context to motifs, we increase the dimensionality of the feature representation. Although the standard motif representation is strictly a subset of contextual motif representation, as one increases the dimensionality of the feature space it becomes easier to overfit, especially in settings with a limited number of training examples (like our own). Thus, for the purpose of comparison we project the contextless data to a representation of equal dimension.

Discovery of data-generating motifs is much more computationally intensive than discovering data-derived motifs. As such, we perform motif discovery on a randomly chosen limited subset of the data (40 days). After fitting our models, we apply maximum likelihood estimation to the remainder of the data to find motifs and context across all data, so we do not reduce our training data size for evaluation purposes. In a follow up experiment, we used an optimized GMM package and the entire dataset. We did not, however, observe a significant improvement in predictive performance, suggesting that this subsampling has little effect on performance.

We set motif hyperparameters (number and length of motifs and contexts) for both methods to be identical. Based on a grid search over plausible values given our domain, we set the number of different motifs discovered to 20, the length of motifs $|\mathbf{m}| = 8$ (representing 40 minutes of glucose data), the length of each context window $|\mathbf{c}| = 72$, and the number of contexts to 2. We evaluated our learned motif representations identically to the method used in Section 3.4.2.

The results of our experiments are given in Table 3.3. Our proposed CMMM approach dominates the MMM approach. Focusing on the `(motif, context)` representation, when motifs and context are learned jointly as opposed to sequentially we see significant improvements in performance AUROC=0.591 vs. AUROC=0.578 for hyperglycemic events. We further explore these differences and the potential cause in the next section.

3.4.3 Experiments on Simulated Data

To evaluate our proposed contextual motif methods on data with known structure across a range of settings, we turn to experiments using simulated data. Our simulated data are created using the CMMM generative model shown in Figure 3.2. We create labels for these data following the procedure given below, varying β_1 values to simulate different types of settings. Our actual generation process is given in **Algorithm 5**.

We examine the effects of using our proposed CMM model and a variety of representations: `(motifs)`, `(motifs, noise)`, two-stage: `(motifs, context)` (where context is inferred after motifs are discovered), and joint:`(motifs, context)` (where motifs and context are jointly inferred). As an upper bound, we compare to two oracles that are based on the ground truth motifs and contextual motifs. Here, we can vary the importance of the contextual motifs (by varying the value of β_1). This allows us to directly test the efficacy of our proposed methods. We examined predictive performance using 10,000 simulated signals, with identical CMMM hyperparameters to those used in our real data-experiments, as input to the same evaluation pipeline above (with the exception that the number of train-test splits is reduced to 25).

ALGORITHM 5: Synthetic Data Generation

```
Data: The number of signals to generate $|X|$, a fully-specified contextual motif
      mixture model $CM$, including the number of contexts $n_c$, number of motifs
      $n_m$
Result: A synthetic contextual motif dataset
for $i \in [1, \dots, n_c]$ do
    for $j \in [1, \dots, n_m]$ do
        Draw a value for the contextual motif: $v_{(m_j, c_i)} \sim \mathbb{U}(-1, 1)$ ;
    end
end
for Each signal in the dataset $\mathbf{x} \in X$ do
    Draw contexts, motifs, and values for the signal according to the contextual
      motif mixture model $CM$ using Algorithm 4;
    Compute a raw score for the signal: $s_x = \sum_{(m,c) \in \mathbf{x}} v_{(m,c)}$;
    Transform the score into a Bernoulli parameter: $p_x = \frac{\exp(\beta_1 s_x)}{1+\exp(\beta_1 s_x)}$;
    Draw the signal's outcome: $y_x \sim \text{Ber}(p_x)$;
end
```

The results are presented in Figure 3.4. Motifs without context perform the worst (and almost identically to noise motifs). Here, we do not observe a meaningful decrease in performance when introducing noise, perhaps because we simulated a much larger dataset. Including separately inferred context provides a minor boost to performance, and jointly inferred context provides a major boost, surpassing the performance of the contextless oracle and approaching the performance of the oracle with context.

3.5 Discussion

Our results confirm both the importance of context in physiological time series and the efficacy of contextual motifs in capturing this context. Below, we summarize some of the main takeaways of our experiments and present a follow-up analysis where we take a closer look at what the models actually learned.

Contextual Motifs vs. Motifs. Our first set of experiments demonstrates that contextual motifs are more discriminative than their contextless counterparts, even when

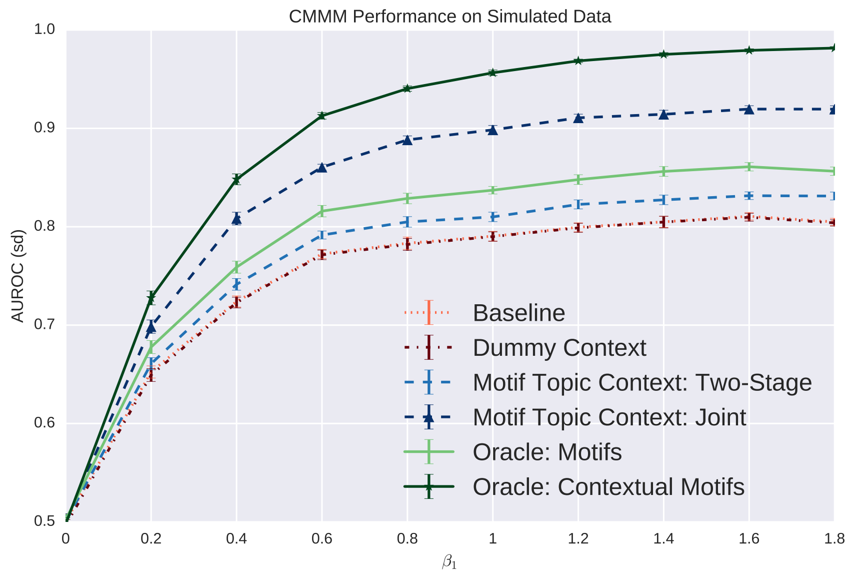


FIGURE 3.4: Here we see the efficacy of our CMMM inference method on 10,000 simulated signals as we vary the strength of relationship between contextual motifs and outcome. Using only the inferred motifs results in suboptimal performance. Two-stage context inference improves performance, but not as much as using the jointly inferred motif and context labels. Error bars on all lines signify 1 standard deviation.

context is inferred. In Table 3.2, we show that, particularly when using data-driven techniques, contextual motifs are more predictive of both hyper and hypoglycemic events. These events are, on their face, opposites, as one involves an increase in glucose levels while the other involves a decrease. However, both events are indicative of poor glycemic control. The fact that our inferred context improves the predictive utility for motifs on both tasks simultaneously suggests that contextual motifs are more informative of the physiological system than motifs alone.

Joint Inference vs. A Two-Stage Approach. Our second set of experiments suggests the utility of a joint inference approach. Both the motifs and contexts discovered by the CMMM are more discriminative than those discovered by the MMM. Although we did not observe large differences between CMMM motifs with and without context, we did observe a significant improvement compared to `(motifs, noise)`. Moreover, in our large simulated dataset we observed a performance increase when we include context. Importantly, we observe a much larger performance increase when using the jointly inferred context.

Data-Derived vs. Data-Generating. Our results also demonstrate the utility of considering both data-derived and data-generating motif frameworks. The data-derived motif discovery method, MDLats, was able to discover variable length motifs much faster than our data-generating motif discovery methods were able to discover fixed length motifs. However, we observe that, while the data-derived motif representation was more useful for predicting hypoglycemic events (max AUROC 0.607 vs max AUROC 0.533), the data-generating motifs were better at predicting hyperglycemic events (max AUROC 0.591 vs max AUROC 0.567).

Our third set of experiments show the promise of jointly inferred contextual motifs across a range of settings. In our large simulated dataset, we did not observe a performance difference between contextless motifs and motifs with a noise context, suggesting that our regularization issues were a result of the data and not our pipeline. In this setting we observe a performance increase when we include context. Importantly, we observe a much larger performance increase when using the jointly inferred context.

One of the main advantages of motifs is their interpretability. In Figure 3.5, we show the motifs we discovered in the CGM data. In this collection, we observe several interesting patterns, including nonlinear monotonic increases/decreases in glucose levels,

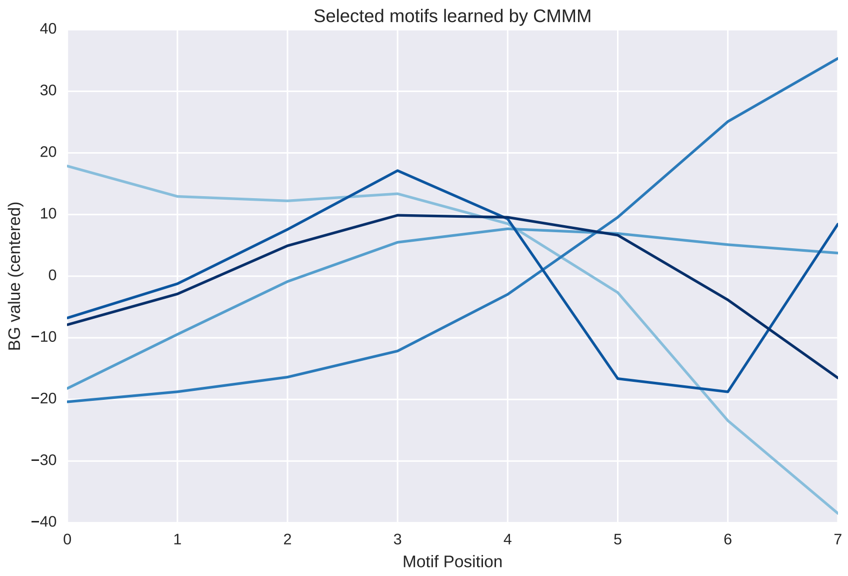
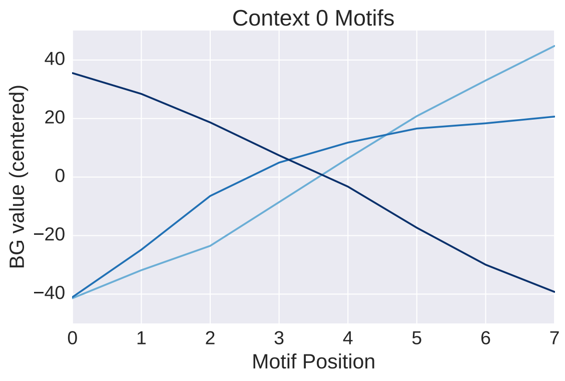


FIGURE 3.5: A hand-selected subset of motifs learned from the physiological data, selected to demonstrate the interesting range of motifs present.

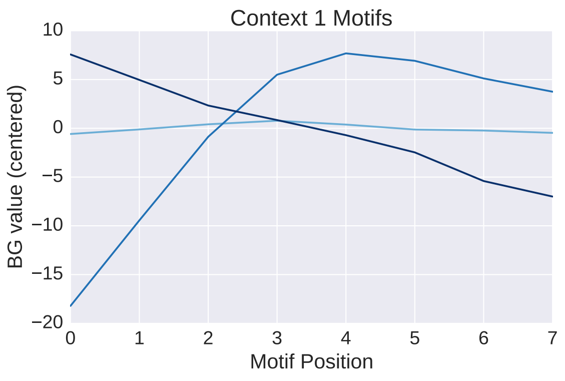
demonstrating the nonlinearity of the glucoregulatory system. Additionally, we observe 'peak' and 'trough' motifs, which appear indicative of hypo- and hyperglycemic events.

To examine the contexts we jointly inferred using the CMMM, we examine the difference in motif mixing probabilities under each context $(|\mathbf{fl}_0 - \mathbf{fl}_1|)$. The three motifs with the greatest negative and positive difference are shown in Figure 3.6. Interestingly, we observe a clinically significant increase in value range for motifs more likely under context 0 vs. 1. This suggests that context 0 represents periods of elevated glycemic variability, this indicates the unsupervised, data-driven discovery of post/pre-meal contexts.

An important assumption made by our current contextual motif methods is that all contexts are categorical. This makes sense for binary context sets, such as post/pre-meal. However, it may make sense to consider contexts on a finer scale, for example: post-large meal, post-small meal, and fasting. In these cases, one may consider extending the proposed framework to ordinal or even continuous representations. Additionally, the utility of contextual motifs depends greatly on the context under consideration. When useful context is unobserved and difficult/impossible to derive from the signal, the utility of contextual motif discovery may be limited.



(A)



(B)

FIGURE 3.6: CMMM context comparison. Figure 3.6a contains the top three motifs occurring more often in context 0, Figure 3.6b contains the top three motifs occurring more often in context 1

3.6 Summary and Conclusions

In this chapter, we described how the utility of motifs, as a representation for physiological time-series data, is limited when context is ignored. To address this limitation, we introduced the concept of *contextual motifs* and proposed methods for contextual motif discovery.

We considered two settings: when context is observed and when context is unobserved. In the first setting, we provided a simple extension that transforms any motif discovery algorithm into a contextual motif discovery algorithm. The second setting is more interesting and perhaps more common. When context is unobserved, we provided techniques to both independently and jointly infer context and motifs.

On real data, we conclude that contextual motifs are more discriminative than their contextless counterparts, even in the absence of observed context. On simulated data, we show a clear improvement in performance when motifs and context are jointly inferred compared to a two-stage approach. We conclude that given a large enough dataset, CMMM is preferred over other approaches.

Our results confirm both the importance of context in physiological time-series and the efficacy of contextual motifs in capturing this context.

Chapter 4

Improving Time-Series Classification Without Additional Data Using Limited Self-Supervision

4.1 Introduction

In **Chapter 3**, we proposed a general representation-learning approach for physiological time series based on the idea that well conserved subsequences are important. This is an example of a two-stage representation learning approach. Here, we explore an end-to-end representation learning framework, where the representation is directly optimized to be useful for the target task. More specifically, we focus on a deep end-to-end representation learning for sequence classification.

Many problems involving sequential data, such as machine translation, sentiment analysis, and mortality prediction, are naturally framed as sequence-level tasks [87]–[89]. Sequence-level tasks map a sequence of observations $\mathbf{x}_{0:T}$ to a single label y. Learning this mapping is often made challenging due to a high-D (dimension) low-N (number of samples) setting [90]. Such problems are particularly prevalent in healthcare tasks, which often involve limited quantities of labeled data captured at a high temporal resolution (*e.g.*, electrocardiogram waveforms).

In high-D low-N settings, researchers have had success with transfer learning techniques, by leveraging additional data to learn intermediate representations that are then used in the target task. Such approaches have proven effective across a number of domains including, computer vision [91], [92], speech processing [93], [94], and natural language processing [89], [95]. The assumption of additional labeled data or large amounts of unlabeled data is often reasonable in such fields, where data is abundant and human annotation for specific tasks is the bottleneck. However, in many domains collecting the data, even without annotations, is either expensive or there is an inherently limited amount of underlying data (*e.g.* personalized medicine). When additional data are unavailable, it may be possible to improve the intermediate learned representation of the data with respect to the target task by considering additional tasks intrinsic to the structure of the data. In particular, we hypothesize that the structure of sequential data provides a rich source of innate supervision. For example, signal reconstruction or forecasting could improve the intermediate representation by capturing the underlying data-generating process. Such approaches are examples of self-supervision, where labels are derived from the input (as opposed to external sources).

Our Approach. In this chapter, we show that leveraging the sequential structure of the data at-hand can lead to improved performance on sequence-level tasks (*i.e.*, the target task). More specifically, by considering self-supervised auxiliary tasks (*e.g.*, signal reconstruction), in addition to the sequence-level task, one can learn useful intermediate representations of the data. Past work investigating self-supervision for sequential data has focused on full-signal reconstruction [96], and to a lesser extent forecasting [14]. Building on past work, we examine the utility of self-supervision on sequential data when additional data are unavailable, and we propose new types of self-supervision tasks. We refer to this approach as 'limited self-supervision.' We limit the self-supervision to the labeled data and focus on self-supervised auxiliary tasks relevant to sequential data ordered by time (*i.e.*, time-series data). For applications, including those involving blood glucose data, investigating ways to improve performance without large amounts of data, labeled or unlabeled, is important. Using wearable health monitors, it is possible to collect vast amounts of longitudinal data for an individual. When these data are used for prediction tasks involving chronic diseases (which tend to progress slowly), this results in a high-D low-N learning setting. The number of individuals, N, is typically the limiting factor. In such scenarios, one might turn to unlabeled data for pretraining. However, there is no clear source of unlabeled data in this application.

Our main contributions are as follows:

- We demonstrate the efficacy of the proposed limited self-supervision framework for improving performance across datasets/tasks with no additional data.
- We compare the utility of several different existing forms of self-supervision in our limited-data setting, identify consistent trends across supervision types, and demonstrate the utility of combining multiple different forms of self-supervision.
- We propose a new form of self-supervision, piecewise-linear autoencoding, that trades off fine-grained signal modeling and long-term dependency propagation. We demonstrate that this is the best form of limited self-supervision across all tasks.
- We provide empirical insights on how auxiliary tasks improve performance under limited self-supervision.

Our work suggests that there is a wide range of time-series and sequence classification tasks where limited self-supervision could improve performance. It also shows the value of including multiple, simultaneous streams of auxiliary self-supervision. Our findings present a methodological contribution, in the form of a useful new type of self-supervision, piecewise-linear autoencoding. Further, our empirical findings on when and how auxiliary tasks help can inform future work in developing self-supervision techniques.

Organization. The remainder of this chapter is organized as follows. We first provide an overview of past work on self-supervision and multitask learning. Then, we formalize our problem and describe the self-supervised tasks we consider. We next describe the datasets we experiment with and provide our results, demonstrating the ability of self-supervised tasks to improve performance with no unsupervised data.

4.2 Related Work

Previous work has found value in self-supervised pretraining with large amounts of unlabeled data [14], [97]. In our work, we focus on time-series data instead of language data.

Several previous works have examined self-supervision for pretraining or feature extraction in the context of time series [98], [99]. We substitute the pretraining framework with a multitask learning framework, removing the requirement to train multiple individual models. However, in contrast to standard pretraining or multitask learning setups, we do not assume the availability of additional data for training. We posit that *even in the absence of additional data, self-supervision can lead to improved performance on the target task.*

Multitask learning deals with training a single model to perform well on multiple tasks. By simultaneously training on multiple related tasks and sharing representations, multitask models can improve generalization [100]. Multitask learning has been used successfully across a number of different clinical tasks [35], [87], [101], [102]. In particular, the success of multitask learning in deep learning demonstrates its value for representation learning [103]. Within the context of supervised learning, Schwab *et al.* considered a multitask framework for learning from sequential health data [104]. Though they used self-supervision, they only considered a setting where large amounts of unlabeled data were available.

Our work was inspired by the findings of Dai and Le [96]. Dai and Le compared sequence-autoencoding and language modeling as auxiliary tasks for leveraging large pools of unlabeled data for natural language tasks (*e.g.*, sentiment analysis). Their approach led to state-of-the-art performance on a range of problems. They found sequence-autoencoding led to larger improvements than language modeling. Interestingly, they found jointly training on the main and auxiliary task *decreased* performance relative to the baseline. In contrast, we focus on self-supervision for time series and do not assume access to additional data.

[105] examines self-supervision without additional data applied to hierarchical EHR data. They also demonstrate the benefit of adding auxiliary supervision. Our work differs from this by i) using sequential as opposed to hierarchical structure, ii) examining multiple streams of simultaneous supervision and iii) comparing a broad range of auxiliary tasks on general time series data. [106] find that training a sequence classification model to simultaneously forecast the data as an imputation method improves overall performance, though they suggest this is due to better handling missing data. We examine the impact of auxiliary self-supervision more generally as a way to improve supervised representation learning.

4.3 Learning with Self-Supervised Auxiliary Tasks

In this section, we present our proposed limited self-supervision approach. After describing our notation, we present our baseline encoder-decoder architecture and describe four self-supervised auxiliary tasks examined throughout this chapter.

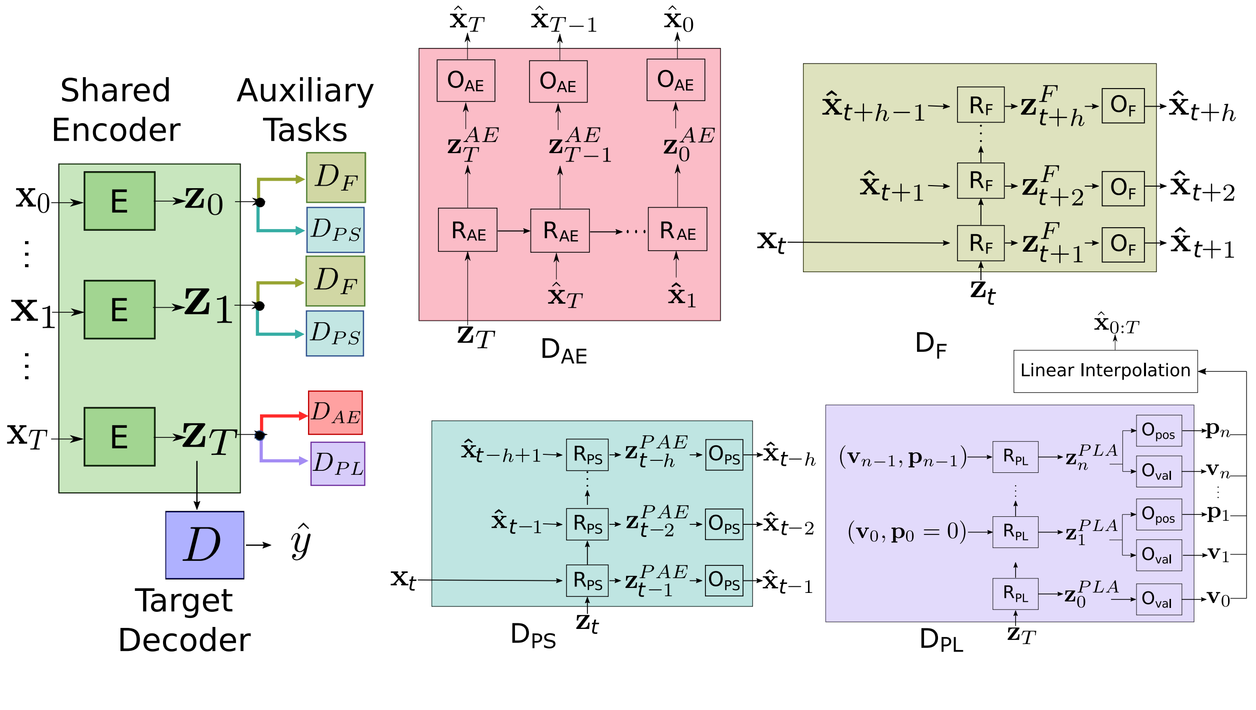
4.3.1 Problem Definition and Notation

We define a sequence as a set of observations $\{\mathbf{x}_t\}_{t=0}^{T} : \mathbf{x}_t \in \mathcal{R}^d$ ordered by the index t. We denote such a sequence as $\mathbf{x}_{0:T}$. Each observation $\mathbf{x}_t$ is a d-dimensional vector. We focus on univariate, evenly sampled time series.

We categorize time-series tasks across three dimensions: i) target *vs.* auxiliary tasks, ii) external supervision *vs.* self-supervision, and iii) sequence-level *vs.* subsequence-level tasks. A target task is the task of interest, whereas auxiliary tasks are only useful insofar as they improve performance on the target task. External supervision occurs when the task labels are provided by an external source (*e.g.*, object recognition). Self-supervision occurs when no additional supervision is required to generate the ground truth label (*e.g.*, in autoencoding, the input itself serves as the supervision). A sequence-level task is one where the supervision pertains to the entire sequence (*i.e.*, sequence classification). A subsequence-level task provides multiple instances of supervision across the signal. In our work, all the target tasks are sequence-level tasks requiring external supervision. We denote the label of the target task as y. Our auxiliary tasks are all self-supervised and may be either sequence- or subsequence-level.

4.3.2 Baseline Architecture

In this chapter, we examine the relative merits of four different self-supervised auxiliary tasks. Throughout, we consider a fixed encoder architecture, focusing on the improvements offered by the auxiliary tasks. Specifically, we focus on recurrent neural networks with LSTM cells. We use a 1-layer LSTM with the number of hidden units determined on a task-by-task basis. These architectures have proven useful for sequential tasks in many domains [14], [51], including health data [87]. We note that our techniques could work with any representation-learning gradient-based approach.



(A) (B)

FIGURE 4.1: a) Our full task architecture. The green encoder layer is shared across all tasks. The target decoder is a fully connected layer mapping to the correct label size. The auxiliary tasks connect to hidden states. Note the sequence-level tasks only connect to the final hidden state. b) Each auxiliary task decoder D_X is composed of a recurrent layer R_X and an output layer O_X. D_{AE}) Autoencoder. D_F) Forecaster. D_{PS}) Partial-Signal Autoencoder. D_{PL}) Piecewise-Linear Autoencoder.

Figure 1a depicts our baseline architecture and its relation to auxiliary tasks. The encoder is implemented as a single-layer LSTM. The target decoder is a single fully connected layer mapping from hidden state $\mathbf{z}_T$ to the output. This simple output layer purposely places a heavy burden on the encoder, since the representation learned by the encoder is shared by (and thus may improve from) all tasks. We train the model by minimizing the cross-entropy between our predictions $\hat{\mathbf{y}}$ (put through a softmax activation) and the one-hot distribution representing the correct label class (**Eqn. 1**).

$$\min_{\theta_D,\theta_E} L_{target} = -\sum_i y^{(i)} \log(D(\mathbf{z}_T;\theta_D)^{(i)}) \tag{4.1}$$

$$\mathbf{z}_T = E(\mathbf{x}_{0:T};\theta_E)$$

Where θ_E and θ_D are parameters for the encoder and decoder respectively and i indexes class labels. We compare this baseline architecture trained with no auxiliary tasks to models trained with up to four auxiliary tasks.

4.3.3 Self-Supervised Auxiliary Tasks

We consider four self-supervised auxiliary tasks: i) autoencoding (*i.e.* reconstruction), ii) forecasting, iii) partial-signal autoencoding, and iv) piecewise-linear autoencoding (shown in **Figure 1b**). The auxiliary tasks take as input the output of the encoding network. Task X is implemented as a decoder D_X, composed of a recurrent layer R_X and a fully connected output layer O_X. While our sequence-level target task requires only one prediction at the end of the sequence, some of our auxiliary tasks are subsequence-level, in which case the task takes as input the intermediate encoding representation $\mathbf{z}_t$.

i) Autoencoding (AE). This is a standard method for unsupervised training in which we seek to, given the final hidden state produced by the encoder, output a reconstruction of the complete signal that minimizes error (**Eqn. 2**). This task requires that the hidden state encode compressed version of the input, compression encourages learning latent structure and discourages learning noise. The decoder, parameterized by a single-layer autoregressive LSTM, outputs a sequence: $D_{AE}(\mathbf{z}_T) = \hat{\mathbf{x}}_{0:T}$. Note $\mathbf{z}_T = E(\mathbf{x}_{0:T};\theta_E)$, and thus depends on θ_E.

$$\min_{\theta_E,\theta_{AE}} L_{AE} = ||\mathbf{x}_{0:T} - D_{AE}(\mathbf{z}_T;\theta_{AE})||_2^2 \tag{4.2}$$

ii) Forecasting. In this task, the decoder takes the hidden state produced at any time step by the decoder and attempts to predict the next h (for example, 6) elements in the sequence. This requires the hidden state to encode the dynamics of the data-generating process. The decoder outputs $\mathbf{x}_{t+1:t+h}$ given past values $\mathbf{x}_{0:t}$, minimizing (**Eqn. 3**). We use a single-layer LSTM similar to the AE decoder, though it is used at each time-step to decode the next h observation values.

$$\min_{\theta_E,\theta_F} L_F = \sum_{t=1}^{T-h} ||\mathbf{x}_{t+1:t+h} - D_F(\mathbf{z}_t;\theta_F)||_2^2 \tag{4.3}$$

$$\mathbf{z}_t = E(\mathbf{x}_{0:t};\theta_E)$$

iii) Partial-Signal Autoencoding (PS-AE). This auxiliary task is a variant of AE that differs in three ways: 1) instead of decoding the full signal $\mathbf{x}_{0:T}$ it decodes only the previous h (for example, 6) steps of the signal $\mathbf{x}_{t-h:t-1}$, 2) instead of one prediction being made at the end of the encoder, a prediction is made at every encoding step from $\mathbf{x}_h$ onward, and 3) the input to the decoding layer includes the current value, $\mathbf{x}_t$ (**Eqn. 4**). It is implemented identically to the Forecast Decoder. The only difference is that it predicts the previous h observation values. This task allows us to examine the impact of signal reconstruction without requiring learning long-term dependencies, allowing for a more meaningful comparison with Forecasting.

$$\min_{\theta_E,\theta_{PS}} L_{PS} = \sum_{t=1}^{T-h} ||\mathbf{x}_{t-h:t-1} - D_{PS}(\mathbf{z}_t;\theta_{PS})||_2^2 \tag{4.4}$$

iv) Piecewise-Linear Autoencoding (PL-AE). A piecewise-linear approximation, or a combination of line segments, is capable of efficiently representing a wide class of signals (particularly non-periodic signals). It is a promising choice for an auxiliary task as it encourages a compact representation capturing the most important details of the signal.

A piecewise-linear representation consists of two length $n+1$ vectors (where n is the number of linear segments in the signal), a value vector $\mathbf{v}$ and a position vector $\mathbf{p}$. The positions are defined as proportions of the original signal length, between 0 and 1. The decoder produces a series of points that define a piecewise-linear reconstruction of the complete input signal. The reconstruction is defined by linear interpolation between the series of points $(p_0, v_0) \dots (p_n, v_n)$, where v_i is the value of the signal at the point i, and p_i is the relative position (or time) where the point occurs. Our decoder produces $n+1$ such points using a single-layer autoregressive LSTM where the hidden state is fed to two output layers that map $\mathbf{z}_i^{PL} \rightarrow (v_i, p_i)$. For each point $i \in [0, n+1]$ we feed the previous point value v_{i-1} and the sum of previous positions $\sum_{j=0}^{j=i-1} p_j$ to the decoder. After we have generated the target number of points, we normalize the position values to enforce $\sum_{j=0}^{n+1} p_j = T$ and perform the interpolation. As this entire process is differentiable, we optimize the decoder directly on the interpolated reconstruction loss (**Eqn. 5**).

$$\min_{\theta_E, \theta_{PL}} L_{PL} = ||\mathbf{x}_{0:T} - D_{PL}(\mathbf{z}_T; \theta_{PL})||_2^2 \tag{4.5}$$

Additional details on these tasks can be found in **Appendix A.1**.

4.3.4 Training

We optimize our model to minimize the loss:

$$L = L_{target} + L_{Aux} \tag{4.6}$$

Where L_{target} is a cross-entropy loss and

$$L_{Aux} = \alpha_{AE} L_{AE} + \alpha_F L_F + \alpha_{PS} L_{PS} + \alpha_{PL} L_{PL} \tag{4.7}$$

is a weighted summation of auxiliary MSE losses. The weighting terms α_X are defined as 0 if the auxiliary task X is not being used for training. If the task is being used, then $\alpha_X = \frac{L_{target}}{L_X}$, where the losses are calculated at the beginning of training using the newly-initialized network on the training data. This ensures that all tasks have losses of similar magnitude.

4.4 Experimental Setup

To test if auxiliary self-supervision improves performance on sequence-level tasks, we consider at a variety of sequence-level tasks across different types of synthetic and real data.

4.4.1 Target Tasks & Datasets

We consider the following three tasks (two of which are based on publicly available real datasets):

Piecewise-Linear Segment Prediction (PLA). We begin with simulated data, as it allows us to estimate the ability of self-supervised auxiliary tasks to identify long-term dependencies in the data. The dataset is composed of piecewise-linear signals, each of length 100. Point values are drawn from a uniform distribution and lie between -1 and 1. The number of line segments also varies uniformly between 1 and 6. The target task for this dataset is to estimate the number of distinct segments that occurred in the signal. 1,000 training, validation, and test sequences were generated independently.

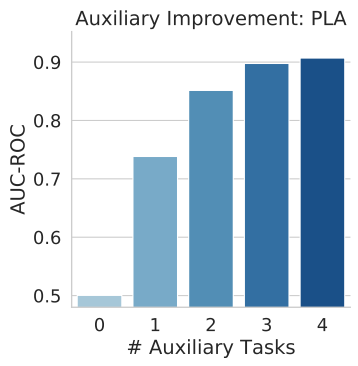
Patient Classification using Glucose Data (T1D). This task uses the glucose dataset discussed in **Section 2.1.1**. Each signal $\mathbf{x}_{0:T}$ consists of 288 glucose measurements sampled every five minutes over the course of a day. This dataset contains 1,863 days of data from 40 patients. In this task, we aim to classify patients based on their data. Here, $y \in \{1, \ldots, 40\}$ represents the patient. Classifying patients is a proxy for the important problem of identifying signal dynamics. We preprocess the data by removing physiologically implausible glucose measurements, and linearly interpolating missing chunks of data. We exclude signals where more than 20% of the measurements are missing and those that are missing a contiguous block longer than two hours. Data were collected by a series of multi-day sessions separated by three-month intervals. As our test set, we consider the final recording session from each patient. We select our validation set randomly from the remaining data.

Atrial Fibrillation Detection (AF). Our final task uses electrocardiogram (ECG) data from the publicly available 2017 PhysioNet Challenge [107], in which the goal was to automatically diagnose atrial fibrillation (AF). This dataset contains four unevenly distributed classes: normal sinus rhythm, AF, other arrhythmia, and noise. We use the training data provided for the competition (the test data are not publicly available), resulting in 8,528 samples. 771 of those signals are labeled AF. We exclude signals with less than 30 seconds of data (967 signals total, 127 with AF) and truncate all signals to exactly 30 seconds. We also downsample the data, reducing signal size from 9,000 to 125. This speeds up training time and eases memory requirements. We use the validation set provided for the challenge as the test set (removing those examples from the training set), and randomly sample 10% of the training data for use as a validation.

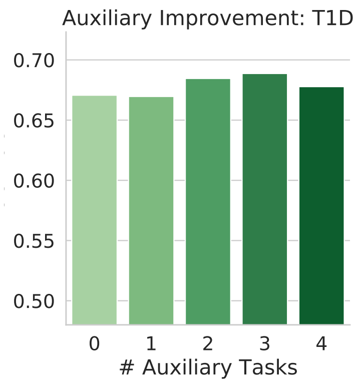
4.4.2 Implementation Details

We implement all models in PyTorch [108], and optimize model parameters using Adam [109] with an initial learning rate of $1e-3$ (the default PyTorch value). In practice we found altering the decoding horizon had little effect on performance, so we used $h = 6$ for all experiments. All encoding/decoding layers are composed of a single recurrent layer an identical number of hidden units, set on a per-task basis using performance on the validation set to balance training time, memory constraints, and target-task performance (evaluated using the early-stopping validation set). We used 128 hidden units for the PLA target task, 512 hidden units for the T1D target task, and 256 hidden units for the AF target task. The decoding layers also have 1 (or two for PL-AE) fully connected output layers. For the PL-AE, we set the number of line segments $n = 6$ as a reasonable size to approximate many signal types. We mitigate the risk of overfitting by using early stopping on a withheld validation set, training until we fail to improve performance for over 50 epochs and reporting test performance for the best performing model on the validation data.

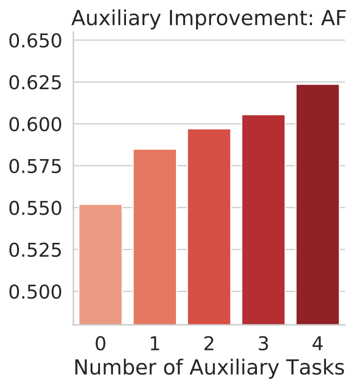
Some auxiliary tasks make predictions at multiple points in the signal. This helps prevent the vanishing gradient problem, which can impede learning with large sequences. To avoid conflating these sorts of improvements with those caused by learning better representations, we use label propagation with our target task (sequence classification), linearly annealing contributions to the loss function over the length of the signal [96]. The propagated losses are combined using a weighted average, with the weights linearly annealed from 0 to 1 over the length of the signal.



(A)



(B)



(C)

FIGURE 4.2: Performance across three target tasks by number of auxiliary tasks used (averaged over all possible orderings). In general, we observe that the greater the number of auxiliary tasks, the greater the performance for all three tasks. The marginal improvement from including additional auxiliary tasks appears to taper off as the number of tasks increases.

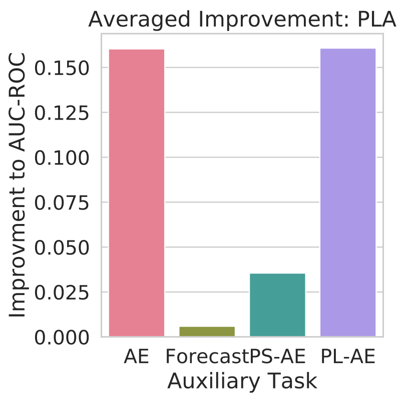
4.5 Experiments and Results

In our experiments, we seek to answer the following questions:

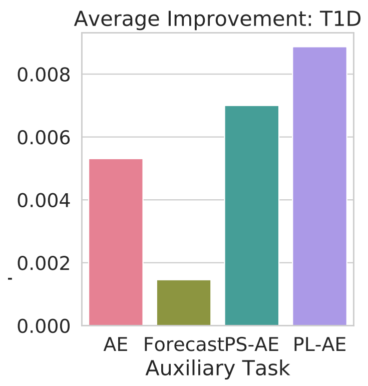
- Does limited self-supervision improve classification performance? (**Figure 4.2**)
- Are some forms of self-supervision more effective than others? (**Figure 4.3 and Table 4.1**)
- How does the amount of training data available affect self-supervision? (**Figure 4.4**)

We begin by establishing that the proposed self-supervised auxiliary task framework improves target task performance. We then look more carefully at the effects of different types of auxiliary tasks. We conclude by looking at the relationship between auxiliary task and target task performance, which sheds light on the mechanism by which auxiliary tasks improve performance.

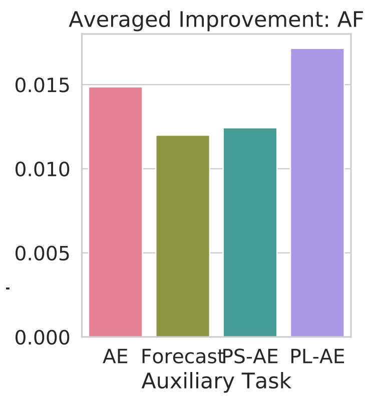
To evaluate the results of these experiments, we measure the macro-averaged AUC-ROC on the target task, and the mean absolute percent error (MAPE) for the auxiliary tasks. We repeat all experiments using three random initializations and average the results.



(A)



(B)



(C)

FIGURE 4.3: The average marginal contribution of each auxiliary task across target tasks. The PL-AE task consistently offers the greatest marginal improvement, and the forecasting task consistently provides the least.

The Benefit of Limited Self-Supervision. We begin by examining our main hypothesis, that limited self-supervision improves sequence-level task performance without additional data. In **Figure 4.2**, we plot target-task performance across our three tasks with a varying number of self-supervised auxiliary tasks. We estimate performance for a given number of auxiliary tasks by averaging the performance of all possible combinations. For all three sequence-level tasks, the inclusion of four auxiliary tasks improves performance relative to no auxiliary tasks. Moreover, we observe that performance tends to increase with the number of auxiliary tasks. The one exception is in the T1D task, where improvement peaks at three auxiliary tasks.

Relative Contribution of Different Auxiliary Tasks. We now examine how different auxiliary tasks contribute differently to performance. To measure the impact of individual auxiliary tasks, we average performance across all models that include the auxiliary task versus all models that do not. This allows us to observe an auxiliary task's individual contribution, and its ability to usefully combine with other streams of self-supervision. The averaged change in AUC-ROC indicates the marginal improvement offered by the task (**Figure 4.3**). PL-AE outperforms all other auxiliary tasks, including AE, on all three datasets. Since PL-AE limits the temporal granularity of the output, this

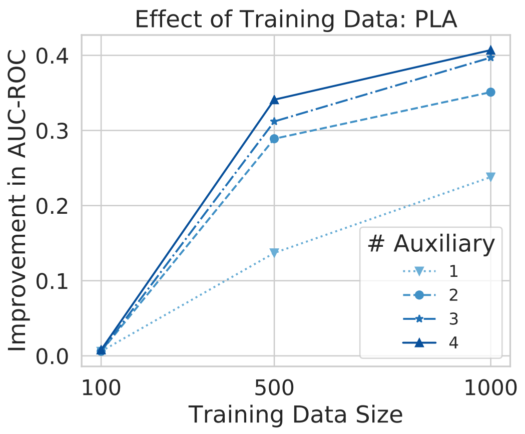
TABLE 4.1: The performance of particular combinations of auxiliary tasks. All methods include the dataset-specific target task. AE and Forecast refers to Autoencoding and Forecasting respectively, the auxiliary tasks explored in [96]. PLAE and PSAE refer to Piecewise-Linear Autoencoder and Partial-Signal Autoencoder, our novel forms of self-supervision. We see that our newly proposed forms of self-supervision outperform the other approaches on all datasets.

	AUC-ROC * 100		
Auxiliary Tasks	PLA	T1D	AF
None	50.0 ± 0.4	67.1 ± 1.3	55.2 ± 4.0
AE	88.6 ± 0.3	66.7 ± 0.4	59.8 ± 1.2
Forecast	53.5 ± 5.2	66.3 ± 0.5	57.0 ± 2.0
AE+Forecast [96]	89.5 ± 0.6	67.4 ± 1.0	61.0 ± 0.7
PLAE+PSAE **(Ours)**	89.6 ± 0.4	$\mathbf{69.6 \pm 0.5}$	60.0 ± 3.3
All **(Ours)**	$\mathbf{90.7 \pm 0.3}$	67.8 ± 0.7	$\mathbf{62.4 \pm 2.1}$

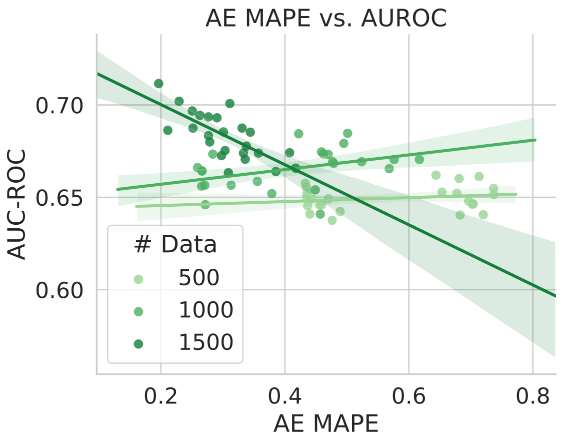
suggests that modeling fine temporal granularity does not help, and may even hurt performance. The forecasting task underperforms all other auxiliary tasks. This finding is in line with the findings of [96]. However, the explanation Dai and Le provide (that the AE encourages long-term dependency modeling) is inconsistent with the performance of PS-AE, which also does not model long-term dependencies. The fact that the PS-AE outperforms the forecasting task, but generally underperforms AE, suggests that while modeling long-term dependencies may improve performance, there is likely some other reason that AE works well. We also examine the performance of particular combinations of self-supervised auxiliary tasks in **Table 4.1**. We see that our proposed auxiliary tasks outperform and are complementary to standard auxiliary task combinations.

***Why* do Self-Supervised Auxiliary Tasks Help?** To investigate the underlying mechanism by which auxiliary tasks improve performance, we examine model performance as we vary the number of auxiliary tasks and the amount of training data (results shown in **Figure 4.4a**). As the amount of training data increases, the added value from the auxiliary tasks increases on average. This suggests that auxiliary tasks are not simply acting as a form of regularization, since otherwise we would expect to see larger improvements on smaller training sets.

If the auxiliary tasks result in better representations (not just regularized representations), their impact on performance should correlate with the auxiliary task performance. A *decrease* in auxiliary task error should lead to a better intermediate representation and



(A)



(B)

FIGURE 4.4: A) The effect training data size has on auxiliary tasks. As the amount of data increases, we tend to see an increase in the improvement afforded by the auxiliary tasks. B) The relationship between auxiliary task performance (in MAPE, lower is better) and target task performance (in AUC-ROC, higher is better) on the T1D data. Calculated over all training data sizes, there is a somewhat weak relationship (Pearson R = -0.53, line not shown). However, when we condition on the amount of training data, we find that different relationships emerge (Pearson R = 0.34 when training size = 1,000 vs. -0.72 when training size = 1,500).

an *increase* in target task performance (*i.e.*, AE MAPE and AUC-ROC should be negatively correlated). Meanwhile, if the auxiliary tasks work as regularizers, exhausting representation capacity, we would expect to see little effect or a positive correlation (if the regularization effect is too strong, and auxiliary performance comes at the *cost* of target performance).

We explore this, specifically for AE with the T1D data in **Figure 4.4b**. This relationship is highly dependent on the amount of data. When training data are limited, there is a weak positive correlation between auxiliary task error and target performance, suggesting a regularizing effect. However, with the full amount of training data, we see a strong negative correlation. Auxiliary tasks give sizable improvements at 1,000 and 1,500 training examples (averaged improvement of 0.016 and 0.010 respectively). This suggests that 1) auxiliary tasks act both to regularize *and* to enhance intermediate representations, depending on the amount of data, and 2) there is an amount of training data where they are effective in either role. These findings suggest that auxiliary self-supervised tasks may be useful across a wide range of training set sizes.

4.6 Summary and Conclusions

In this chapter, we introduced and analyzed limited self-supervised, where we sought to improve sequence-level task performance without additional data. By jointly training our target task with auxiliary self-supervised tasks, we demonstrated small but consistent improvements across three different sequence classification tasks. Our novel piecewise-linear autoencoding task emerged as the most useful auxiliary task across all datasets. In contrast, forecasting, which presents an intuitively appealing form of self-supervision, led to the smallest improvements.

Across a range of training set sizes, we showed that the value of auxiliary tasks lies in improving intermediate representations learned by the network. When limited training data are available, these tasks serve as a form of regularization. With more training data, performance on the auxiliary tasks improves and so does performance on the target task, suggesting more useful intermediate representations. In the context of time-series analysis, limited self-supervision is an effective form of supervision and comes at little cost. Going forward, researchers seeking to improve performance on sequence-level target tasks should consider incorporating self-supervised auxiliary tasks.

Chapter 5

Deep Multi-Output Blood Glucose Forecasting

5.1 Introduction

In the previous two chapters, we focused on the problem of representing sequential data as input to machine learning methods. In this chapter we shift focus to consider how we frame the machine learning task of forecasting, and the implications of the *output* representation we use.

In a typical signal forecasting problem, one aims to estimate the future value of the signal using past values. For example, one may aim to predict a blood glucose measurement occurring 30 minutes in the future, given past blood glucose measurements. This single-step setting generalizes to the multi-step setting, in which one aims to predict multiple values within a time horizon. This multi-step setting is inherently more difficult, since it requires modeling the joint probability of future measurements. While more challenging, if successful this joint modeling of observations within a sequence can improve overall performance. For example, while the word 'the' occurs often in English, the phrase 'the the the' does not.

Recursive approaches, in which a single-step forecaster predicts several values by using the current prediction to make the next prediction, are commonly used in multi-step forecasting [15]. However, such approaches often suffer from poor long-term performance, since any error introduced will enter a positive feedback loop. Alternatively,

The following is an adaptation of previously published work [110].

multi-output forecasting aims to estimate multiple values at once. While no longer susceptible to the feedback issue, multi-output forecasting may not adequately capture dependencies among predictions.

Our Approach. In this chapter we examine two complementary solutions to better model trajectories. In the first, we use a multi-output recurrent neural network where explicit temporal dependencies between outputs captures the relationship between the predictions. In the second, we propose a novel architecture that directly models the underlying generating function of the signal by learning a polynomial approximation for the outputs. The problem of error accumulation during sequence prediction has been previously studied in NLP [16], [111]. We distinguish ourselves from this past work by focusing on new models that alleviate this problem, as opposed to new training schemes. For blood glucose forecasting, approaches that predict value trajectories can be more useful than approaches that predict a single future value. This is because the overall trend and variability in the signal are important factors to determine future treatment [112].

We apply the proposed approaches to a challenging real-world forecasting problem (described below). Our main contributions can be summarized as followed:

- We propose two complementary deep multi-output forecasting architectures: an autoregressive multi-output forecaster and a polynomial "function forecasting" system.
- We improve over existing approaches by leveraging the proposed forecasting architectures on a large-scale real-world multi-step forecasting problem.

In additional analyses, we demonstrate that predicting multiple values can provide extra supervision, improving single output forecasting performance.

This work focuses on forecasting blood glucose values. Forecasting blood glucose is relevant to individuals with diabetes, as they need to consistently and tightly maintain blood glucose levels, which can be aided by predictive alarms powered by forecasting models [113].

Learning the dynamics of the glucoregulatory system is difficult because the long-term system dynamics are highly nonlinear [114]. We tackle this challenge using data from over 550K blood glucose measurements. Our proposed approaches, when used together, achieve better results in terms of Absolute Percent Error (APE) than existing shallow or deep forecasting approaches both on average (4.87 *vs.* 5.31) and particularly in periods of extreme fluctuation (12.05 *vs.* 13.34).

Organization. The remainder of the chapter is organized as follows. In the next section, we introduce notation and formally define our problem. We then present our proposed forecasting architectures and methods. After, we present a series of experiments on a real-world dataset demonstrating our approaches offer complementary solutions to the issue of temporal dependence in multi-output forecasting.

5.2 Problem Setup and Background

In signal forecasting, one aims to estimate the next value in a signal x_{t+1} given past values $\mathbf{x}_{0:t}$, $\mathbf{x}$ represents the signal of interest, and t the current time step. Here, we focus on the univariate setting, *i.e.*, $x_0, \ldots, x_t \in \mathbb{R}$, though our approaches generalize to the multidimensional setting. The most common approaches to signal forecasting focus on learning a model for $p(x_{t+1}|\mathbf{x}_{0:t})$ [15], [33]. Sometimes a prediction offset d is added to learn the model for $p(x_{t+d}|\mathbf{x}_{0:t})$. Given the recent successes of deep architectures for this problem in general [33], [47], [115], and specifically in the domain of glucose forecasting [116], we focus on building upon deep learning methods for signal forecasting.

A model that accurately predicts x_{t+1} can be used for either single- or multi-step forecasting. Applied recursively, single-step models enable multi-step forecasting, *i.e.*, predicting multiple values over a time horizon of length h (see **Figure 5.1**). Of particular note are deep conditional generative models, which model joint distributions by sequentially estimating terms in the conditional factorization of the distribution with deep neural nets [33]. This style of forecasting, where $p(\mathbf{x}_{t+1:t+h}|\mathbf{x}_{0:t})$ is estimated by the factorization: $p(x_{t+1}|\mathbf{x}_{0:t})p(x_{t+2}|\mathbf{x}_{0:t}, \hat{x}_{t+1}) \ldots p(x_{t+h}|\mathbf{x}_{0:t}, \hat{\mathbf{x}}_{t+1:t+h-1})$ is called **recursive forecasting**, and is the primary form of multi-step forecasting [15]. Recursive models have the advantage of modeling the joint probability of the signal within the prediction window. However, the re-use of predictions creates a feedback loop, amplifying potential errors and leading to lower quality predictions as the time horizon increases.

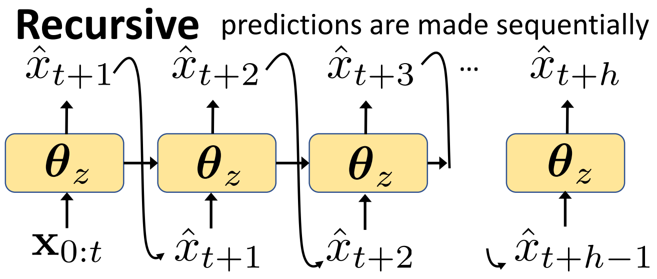


FIGURE 5.1: An example of multi-step recursive forecasting. Predictions at one time step are fed back into the network as input. This allows for single-step methods to produce multi-step forecasts.

In contrast, **multi-output** forecasting aims to estimate $p(\mathbf{x}_{t+1:t+h}|\mathbf{x}_{0:t})$ in one step [15]. Multi-output approaches sidestep the issue of error feedback by jointly estimating over the prediction window, and will be the main focus of this work.

5.3 Methods

We examine two approaches for deep multi-output forecasting: 1) propagating information across the prediction window, and 2) directly predicting the underlying generative function of the signal. We investigate both approaches, as they represent different, complementary methods to enhance multi-output forecasting. In this section, we first describe the multi-output deep learning framework shown in **Figure 5.2**, then explain both extensions, shown in **Figure 5.3**. We finish by providing additional details on how to train the models.

5.3.1 Deep Multi-Output Forecasting (DeepMO)

A neural network can function as a multi-output forecaster by using multiple output channels to infer multiple time points into the future from a shared hidden state. At time t, a standard multi-output neural network derives a hidden state vector $\mathbf{z}_t$ from input $\mathbf{x}_{0:t}$ through a series of hidden layers composed of linear combinations and nonlinear activation functions, all parameterized by $\boldsymbol{\theta}_z$. This hidden state is then translated to predictions via the network's output channels $o_{1:h}$ as follows:

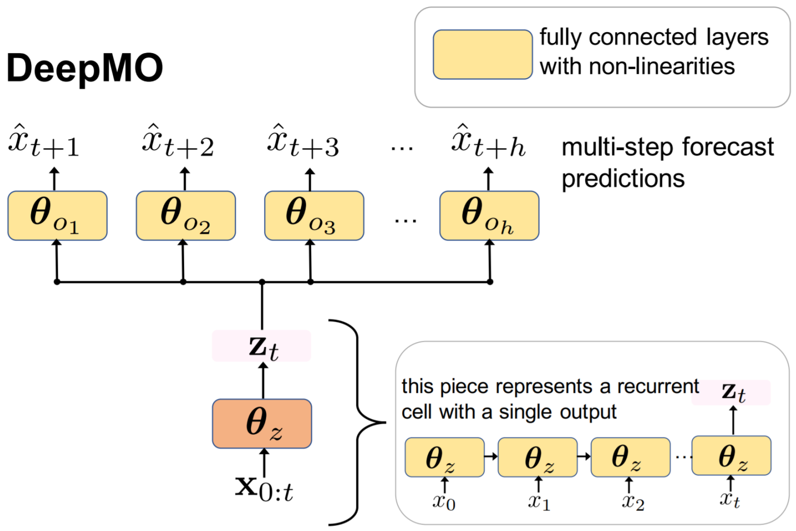
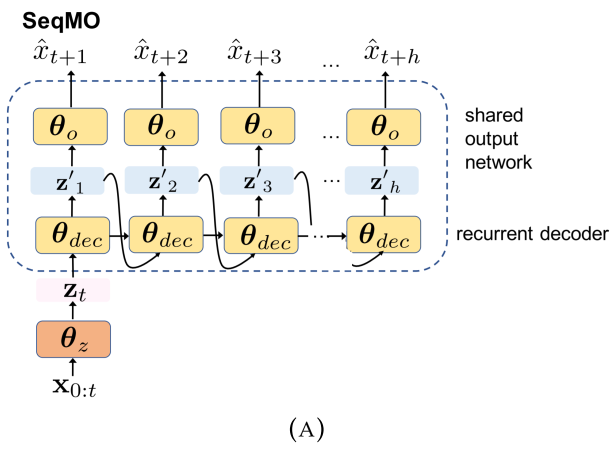

DeepMO
fully connected layers with non-linearities
$\hat{x}_{t+1}$
$\hat{x}_{t+2}$
$\hat{x}_{t+3}$
$\hat{x}_{t+h}$
multi-step forecast predictions
$\boldsymbol{\theta}_{o_1}$
$\boldsymbol{\theta}_{o_2}$
$\boldsymbol{\theta}_{o_3}$
$\boldsymbol{\theta}_{o_h}$
$\mathbf{z}_t$
$\boldsymbol{\theta}_z$
$\mathbf{x}_{0:t}$
this piece represents a recurrent cell with a single output
x_0
x_1
x_2
x_t

FIGURE 5.2: Forecasting with DeepMO involves transforming the input to a shared representation and then learning separate output networks for each time point in the prediction horizon h.

$$\hat{x}_{t+i} = o_i(\mathbf{z}_t; \boldsymbol{\theta}_i) \text{ for } i \in [1:h] \tag{5.1}$$

where o_i is defined in terms of linear combinations and nonlinear activations, parameterized by $\boldsymbol{\theta}_i$. This approach is illustrated in **Figure 5.2**. Note that the value of the predicted output at time step $t+1$ is not explicitly propagated through the remainder of the prediction window (as it would be in a recursive setting). The mapping defined by o_i has no direct impact on o_j at inference time for $j \neq i$. However, $\hat{x}_i$ is not independent of $\hat{x}_j$ since they both depend on the shared representation $\mathbf{z}_t$. Additionally, temporal dependencies among the output are implicitly propagated by the joint optimization over $\boldsymbol{\theta} = [\boldsymbol{\theta}_z, \boldsymbol{\theta}_{o_1}, \ldots \boldsymbol{\theta}_{o_h}]$ during training. However, since this approach does not explicitly encode dependencies among the outputs, learning such relationships may be more difficult.

Standard neural networks require a fixed sized input. To eliminate this limitation, we use recurrent neural networks (RNNs), which allow for variable-sized input. This allows the network to learn the amount of history that is useful for prediction and make predictions at any point in the signal. As such, this and all subsequent architectures use



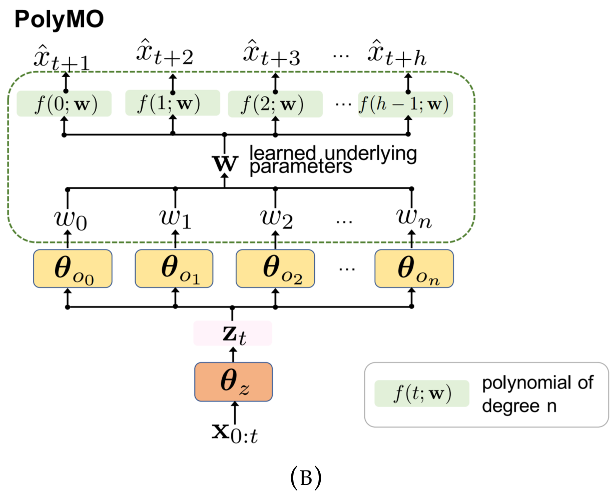


FIGURE 5.3: Two extensions to the DeepMO forecasting framework (a) SeqMO uses a decoder network to generate a representation for each time point in the prediction horizon that feeds into a shared output network to produce h predictions (b) PolyMO learns $n + 1$ separate output networks based on a shared representation z_t, to infer the parameters of an n^{th} degree polynomial that is then used to generate the predicted output.

recurrent cells to generate z_t. We use GRU cells [117], however, other recurrent cells could be used as well. The recurrent cells are depicted by the orange cell in **Figure 5.2**. We refer to the architecture described above as "DeepMO" (Deep Multi-Output Forecaster).

5.3.2 Sequential Multi-Output Forecasting (SeqMO)

We extend the approach described above by combining 1) the ability of a recursive forecaster to explicitly model temporal dependencies within a sequence with 2) the ability of a multi-output system to model multiple time steps at once. To combine the advantages of these two forecasting systems, we use the DeepMO architecture, described above, but introduce temporal dependencies between sequential outputs.

Our sequential multi-output approach modifies DeepMO by replacing the multiple output channels o_i for $i \in [1 : h]$ with a recurrent decoder network, parameterized by $\boldsymbol{\theta}_{dec}$. The decoder network unrolls the hidden state $\mathbf{z}_t$ into $[\mathbf{z}'_1, \mathbf{z}'_2, \ldots, \mathbf{z}'_h]$. Each $\mathbf{z}'_i$ is then independently passed through the same shared output channel. Specifically, we replace (5.1) with:

$$\mathbf{z}'_1, \mathbf{z}'_2, \dots, \mathbf{z}'_h = Dec(\mathbf{z}_t; \boldsymbol{\theta}_{dec}) \tag{5.2}$$

$$\hat{x}_{t+i} = o(\mathbf{z}'_i; \boldsymbol{\theta}_o) \tag{5.3}$$

With this setup, the model can learn to trade off between recursively propagating error and capturing temporal dependencies. We refer to this approach as a sequential multi-output forecaster (SeqMO) (**Figure 5.3a**). We hypothesize that by explicitly encoding a temporal relationship among predictions, we will learn a more accurate forecasting strategy. This approach uses a recurrent encoding-decoding framework [118] for time-series forecasting. Note that this involves a many-to-many mapping, since we make multiple sequential predictions at each time step.

5.3.3 Polynomial Function Forecasting (PolyMO)

Our second proposed extension reframes the forecasting task. Instead of learning the distribution of future signal values conditioned on the past, we learn to predict an underlying representation of the data. We call this function forecasting. In particular, we assume the prediction window $\mathbf{x}_{t+1:t+h} \sim f(0 : h-1; \mathbf{w})$, where $\mathbf{w}$ parameterizes the function class f. Instead of directly modeling $p(\mathbf{x}_{t+1:t+h} | \mathbf{x}_{0:t})$, we estimate the parameters to the underlying generative function $p(\mathbf{w} | \mathbf{x}_{0:t})$ (see **Figure 5.3b**). Function forecasting is analogous to SeqMO, where input data are encoded into a hidden state best parameterizing a decoder network. The key difference is that here the decoding step is restricted to the function f.

We restrict our generating function class f to polynomials of degree n. Such functions are parameterized by $n+1$ real numbers $f(t; \mathbf{w}) = w_0 + w_1 t + \cdots + w_n t^n$. We modify equation (5.1) so

$$\mathbf{w}_j = o_j(\mathbf{z}_t; \boldsymbol{\theta}_{o_j}) \text{ for } j \in [0 : n] \tag{5.4}$$

At each time step, t we predict the set of coefficients $\mathbf{w}$ parameterizing the best approximation of future values $\mathbf{x}_{t+1:t+h}$. For training, we determine the actual value of the parameter by taking the best-fit polynomial of degree n over $\mathbf{x}_{t+1:t+h}$. Since we want the generating function f to actually model the underlying signal, and not just the observations, we limit the polynomial's capacity by setting $n << h$.

We refer to this approach of polynomial function forecasting as "PolyMO" (**Figure 5.3b**). We hypothesize that focusing on estimating the underlying generative function versus the values themselves will result in improved forecasting performance. By compactly representing future data, the output complexity of the network can be lowered, reducing noise. In addition, by predicting a generating function, the network must reason about the joint distribution of the values (since each parameter w_i affects the entire output window). This helps address the error accumulation inherent to recursive forecasting.

5.3.4 Sequential Polynomial Function Forecasting (PolySeqMO)

The two extensions to DeepMO shown in **Figure 5.3** both seek to improve our estimation of $p(\mathbf{x}_{t+1:t+h}|\mathbf{x}_{0:t})$. However, these proposed techniques represent somewhat orthogonal improvements. SeqMO provides a way to learn the trade-off between relying on intermediate value estimates and avoiding recursive error accumulation. PolyMO, meanwhile, facilitates prediction by constraining the intermediate representation, predicting values parameterizing the function approximation. The prediction of these parameters is itself a multi-output prediction. While the standard PolyMO forecaster uses the DeepMO framework to generate $w_0, \ldots, w_n$, there's no reason that it couldn't use or wouldn't benefit from the SeqMO framework instead. Thus, we also examine a PolyMO forecaster with recurrent parameter decoding, denoted "PolySeqMO" (**Figure 5.4**).

5.3.5 Training and Inference Details

In the above methods, the parameters $\boldsymbol{\theta}$ can be learned using stochastic gradient descent. The standard deep forecasting formulation defines training loss based on actual values in the signal. However, previous work has found that it can be beneficial to transform the problem into a multi-class classification task [33]. Thus, we replace the task of directly predicting signal values $\hat{x}_{t+i}$ with the task of predicting the probability mass function over possible discretized values of the signal: $\hat{p}(x_{t+i})$, using a cross-entropy loss against

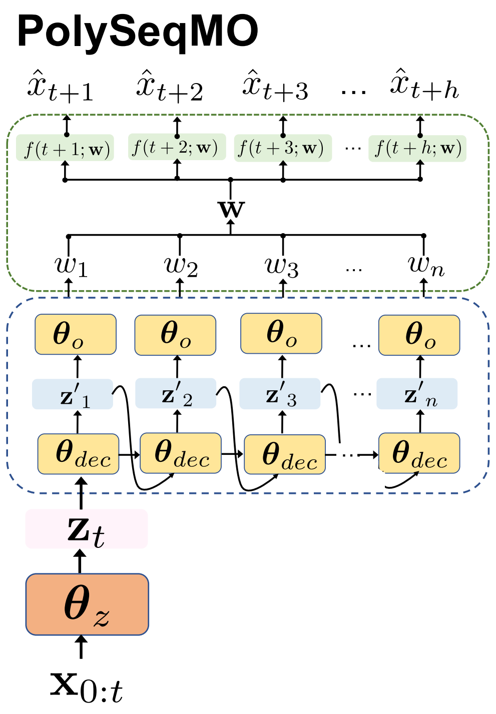


FIGURE 5.4: A combination of SeqMO and PolyMO. The function forecasting framework from PolyMO is used to predict parameters **w** for a function approximating output values. These parameters are predicted using the recurrent decoding network from SeqMO.

the one-hot distribution for the actual value. Each output channel o_i encodes not a single number, but distribution over possible values. Similarly, we predict distributions over parameter values **w** in PolyMO forecasting. While the multi-class formulation allows us to use the cross-entropy loss during training time, ultimately, we are interested in evaluating the quality of real valued forecasts. Thus, we translate these distributions to predictions by taking the value represented by the class with maximum probability in $\hat{p}$. This approach has been found to work well in the field of speech generation [33], but has not, to our knowledge, been investigated in the context of physiological signal forecasting.

Finally, at inference time, we smooth predictions by replacing the predicted values $\hat{\mathbf{x}}_{t+1:t+h}$ with the values occurring at that time in a best-fit polynomial, with the polynomial degree set using validation data. That is, we find the polynomial $f(\cdot;\mathbf{w})$ that best approximates $\hat{\mathbf{x}}_{t+1:t+h}$, and return as output the vector $[f(0), \dots f(h-1)]$. This polynomial smoothing allows for a more direct comparison between models that predict glucose values and the PolyMO approach.

In the sections that follow, we test our hypotheses and evaluate our proposed forecasting systems on a real dataset. We begin by describing the forecasting task, and then explain the experimental setup and provide the detailed implementation of the methods we evaluate.

5.4 Dataset & Forecasting Task

We consider the task for predicting future blood glucose values in patients with type 1 diabetes. These data present a challenging and clinically meaningful forecasting task.

5.4.1 The Data

We used the dataset described in **Section 2.1.1**. In total, the dataset consists of 1.9k days of blood glucose measurements, totaling nearly 550k distinct glucose measurements. Blood glucose measurements were of integer resolution in the range of 40-400 mg/dL.

5.4.2 The Task

There has been extensive work on using CGM data to predict short-term outcomes (*e.g.*, predicting hypo and hyperglycemic events, [119], [120]). In contrast, here, we focus on the more general task of glucose forecasting. More specifically, we consider the challenging task of predicting future blood glucose levels using *only* data about past blood glucose measurements. In this context, previous work has focused on using ARIMA [121] and machine learning algorithms such as Random Forests [119]. Others have proposed machine learning techniques for leveraging data pertaining to external factors [122]–[124]. While blood glucose is affected by external factors, *e.g.,* carbohydrate intake and insulin, such data are not always readily available [125] and come at the cost of increasing patient burden in the form of data collection. Through the dataset we consider does not contain these additional data, it is important to note that the proposed methods generalize to a multivariate setting.

With enough advanced warning via a forecast, one can correct blood glucose levels through the administration of either insulin or glucose. How far in advance is far enough? It is important to note that there is 1) often a delay before insulin or glucose begins to act on the glucoregulatory system and 2) a lag between changes in blood glucose levels and CGM measurements. Thus, to have the greatest impact (*e.g.*, help patients avoid hypo- and hyperglycemic events) we must be able to predict several measurements in the future. Previous work in blood glucose forecasting has settled on a 30-minute prediction window as adequate for this task [122]–[124]. Thus, to test the efficacy of our forecasting systems, we evaluate a multi-step forecast with a 30-minute ($h = 6$) prediction window.

We evaluate performance for any given prediction by calculating the mean absolute percentage error (APE) over the prediction window. This evaluation metric varies from previous work, which reports performance on only the last sample in the prediction window. We report over the entire prediction window for two reasons. First, from a clinical perspective, we are interested in the trend the values suggest. An ultimate decrease in blood glucose can have different interpretations if the rate of decline is accelerating *vs.* decelerating. Second, from a technical perspective, we are interested in evaluating our systems as multi-step forecasters, naturally suggesting evaluation over multiple steps.

5.5 Experimental Setup & Baselines

Through a series of experiments on the data and task described above, we measure the ability of our proposed methods to forecast blood glucose values. We compare against several baselines that have been used for forecasting in previous work [119]. We also investigate the advantage the extra supervision inherent in multi-output forecasting offers over single-output forecasting on a single-output task.

5.5.1 Train, Test, and Validation

In all of our experiments, we split the data into training, validation, and a held-out test dataset. This procedure is shown in **Figure 5.5**. These splits were determined using the

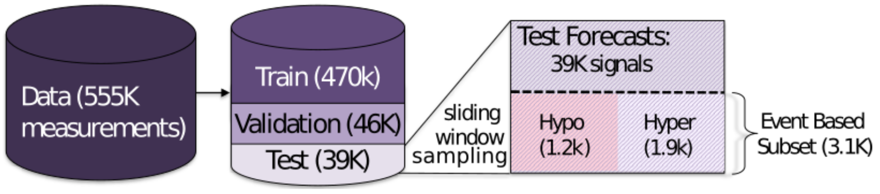


FIGURE 5.5: The splitting procedure used to train and test our models. The complete dataset is separated into three disjoint subsets: train, validation, and test. The test set is then further split into 4 non-disjoint sets: the full test set, test set examples that denote new hypo/hyperglycemic events, and subsets for each event type separately. This results in widely varying subset sizes given in the image.

CGM recording sessions across patients. For each subject, the entirety of the final recording session is added to the test set, the second to last session is added to the validation set, and the remaining data are added to the training set. Recording sessions vary in length, but this split results in approximately 85% of the data being used for training, 7.5% for validation, and 7.5% for testing. Compared to a random split, a temporal split more closely mimics how we expect the model to perform in practice. Note that several months elapse between recording sessions, so data in the training set have no immediate connection to the testing data.

We evaluate the models at any point in time where we have at least ten samples (*i.e.*, 50 minutes) of prior data. We select this minimum to ensure that there is sufficient information to make a reasonable prediction. As CGMs are used continuously in the real world, this does not restrict applicability. We remove measurements that represent physiologically unrealistic glucose fluctuations (over 40 mg/dL in under 5 minutes) to remove noisy CGM measurements. Using a sliding window sampling method with a stride of 1 results in over 39k distinct test samples.

For evaluation purposes, we divide our test set into four overlapping groups: 1) the full test set, 2) windows in the test set that contain either hypoglycemic onsets or hyperglycemic onsets, and two sets containing 3) only hypo and 4) only hyperglycemic onsets. Specifically, we examine performance on a second test set of samples filtered such that a hypo or hyperglycemic event begins in the 30-minute prediction window, and the last training sample was at a normal blood glucose level (between 70-180 mg/dL). Focusing on only hypo and hyperglycemic events reduces our test set size to 3,068 samples: 1,156 hypoglycemic events and 1,912 hyperglycemic events. We look at each of these subgroups to better understand forecaster performance across a range of relevant situations. The dynamics of the glucoregulatory system are highly nonlinear, and the dynamics can vary dramatically depending on the state of the glucoregulatory system and environmental contexts [113].

The complete test set is most representative of general model performance. However, the event test set is indicative of performance at points critical for maintaining healthy glucose levels. This event test set is further broken into a hypoglycemic event set and hyperglycemic event set. The prevention of hypo and hyperglycemic events are important for different reasons. Depending on the outcome of interest and the patient's personal history, it may be more important to effectively predict one versus the other. Thus, performance across all the test sets can be relevant.

5.5.2 Baseline Forecasting Methods

To compare the performance of our proposed approaches to existing methods, we consider the following shallow and deep architectures.

- **Baseline: Linear Extrapolation.** This baseline simply uses the most recent 30 minutes of data to extrapolate 30 minutes into the future. We chose to use the most recent 30 minutes (as opposed to a longer history) based on performance on the validation set. We include this naive baseline to give the reader a sense for how challenging the task is. It also provides an interesting comparison to the performance of the 1^{st} degree PolyMO and PolySeqMO models, as they have identical output capacity.

- **Baseline: Random Forest (RF).** Simple but effective, the RF algorithm has been successfully used in glucose forecasting [119]. A robust ensemble method, it can be parallelized for rapid training and prediction. Applied to the task of predicting hypoglycemic events, Sudharsan *et al.* achieved results competitive with state-of-the-art. We experimented with two random forest baselines: i) a random forest trained to predict one time step into the future, used to recursively generated the multi-output prediction, and ii) a true multi-output random forest.

- **Recursive RNN.** As our next baseline, we consider a recurrent neural network (RNN) that makes multi-step predictions using the recursive approach outlined in **Figure 5.1**. RNNs have recently been shown to achieve state-of-the-art results in glucose forecasting [116]. Our RNN uses two layers of GRU cells regularized via early stopping on a validation set and weight decay. While we are interested in minimizing the APE of our forecasts, we do not use APE as our loss function. Instead, as discussed in **Section 5.3.5**, our network outputs a probability mass function over a discretized set of glucose values, thus we use a cross-entropy loss.

5.5.3 Implementation Details

We implemented all deep learning models using PyTorch. We learned the model parameters using stochastic gradient descent with an ADAM optimizer [109]. We implemented the RF baseline using Scikit Learn [126].

All our models have a number of hyperparameters. To ensure fair comparison between methods, we set hyperparameters by optimizing performance on the training and validation data. Our hyperparameter search space for the deep architectures included model depth, recurrent layer size, initial learning rate, and input normalization. For the RF, we tuned the number of trees and size of input.

Values reported for the RF forecaster were obtained using 100 estimators with a 10-sample input size. The remaining hyperparameters used the default Scikit Learn values. The deep architectures were found to have performance robust to hyperparameter selection. All results reported were obtained using two recurrent layers of 512 GRU hidden units. Output channels were implemented as fully connected layers with a softmax activation. Training was run until performance on a separate validation set failed to increase for 50 epochs. A weight decay value of 10^{-5} was used for all models. All remaining model details, such as the initialization procedure and the initial learning rate for ADAM, used the PyTorch default values.

To train our PolyMO model, we tested a variety of different polynomial degrees. On the training data, we observed that blood glucose values over a 30-minute window can be well approximated with something as simple as a 1^{st} degree polynomial ($n = 1$). On average, a linear approximation of the output window incurred a reasonably small reconstruction loss. The first-degree polynomial struck a good balance between performance and capacity. On the validation data, we investigated the performance of the PolyMO model using different degrees, and found the 1^{st} degree performed best.

We found the best fit 1^{st} degree polynomials over all length six prediction windows in the training set and used the maximum and minimum values for each coefficient as the range for our categorical output prediction, except for the bias term that we limited between 40-400 to mirror the glucose monitor output limitations. Outputs were quantized into 361 equal bins both when predicting glucose and for each polynomial coefficient. This number was chosen to give the real-value network the capacity to predict any recorded value of blood glucose, as most continuous glucose monitors have integer resolution. All source code for this project is available online .

5.6 Experiments & Results

In our experiments, we seek to answer the following questions:

- Do deep forecasters outperform strong shallow approaches? (**Section 5.6.1**)
- Is multi-output or recursive forecasting a better multi-step forecasting baseline? (**Section 5.6.2**)
- What improvements come from explicitly modeling sequential dependencies in the prediction window? (**Section 5.6.3**)
- Is there added benefit to predicting underlying functional representations? (**Section 5.6.4**)
- Are there performance benefits from explicitly predicting blood glucose trajectories compared to single-step forecasting? (**Section 5.6.6, Figure 5.7**)

In total, we tested eight distinct forecasting systems: 1) Linear Extrapolation, 2) a recursive RF (RF: Rec), 3) a multi-output RF (RF: MO), 4) a recursive RNN (Recursive), 5) a multi-output RNN (DeepMO), 6) a sequential multi-output RNN (SeqMO), 7) a polynomial multi-output RNN (PolyMO), and 8) a sequential polynomial multi-output RNN (PolySeqMO).

`https://github.com/igfox/multi-output-glucose-forecasting`

TABLE 5.1: Results. We examine the performance of our eight forecasting approaches across different subsets of the CGM test data. Results are reported as 50th percentile APE over the prediction window, values in parentheses are 2.5th − 97.5th percentiles. Underlined results indicate the best single-model performance. Bold results demonstrate the best overall (single or ensembled) performance.

	Architecture	Full 39k	Event 3.1k	Hypo 1.2k	Hyper 1.9k
Shallow Baseline	Extrapolation	6.48 (0.21-42.12)	10.76 (1.42-63.98)	14.85 (1.89-86.81)	8.73 (1.30-36.87)
	RF: Rec	8.00 (0.62-40.83)	10.45 (1.99-65.21)	14.31 (2.73-91.12)	8.82 (1.85-30.42)
	RF: MO	5.18 (0.71-30.16)	10.64 (1.41-55.28)	17.88 (2.70-75.46)	8.20 (1.14-28.00)
Deep Baseline	Recursive	5.31 (0.00-29.32)	10.00 (1.45-46.22)	13.34 (2.17-62.86)	8.43 (1.24-30.49)
	DeepMO	5.01 (0.00-28.74)	9.93 (1.62-41.67)	12.91 (2.26-56.04)	8.52 (1.43-30.02)
Proposed	SeqMO	4.91 (0.00-28.95)	9.69 (1.51-41.54)	12.48 (2.28-54.02)	8.37 (1.29-29.46)
	PolyMO	4.95 (0.51-28.30)	9.79 (1.48-43.67)	12.49 (1.93-60.78)	8.46 (1.31-30.75)
	PolySeqMO	<u>4.87</u> (0.48-27.80)	<u>9.57</u> (1.43-43.59)	<u>12.05</u> (2.03-60.90)	<u>8.31</u> (1.24-29.76)
	PolySeqMO Ensemble	**4.59** (0.41-21.12)	**9.38** (1.35-42.34)	**11.61** (1.99-59.89)	**8.13** (1.18-29.49)

Table 1 presents the forecasting model performance, in terms of APE over the prediction window in the held-out test data. We noted the error distribution was non-normal, so we report the median APE and the 2.5th − 97.5th percentile errors. That said, all observed trends hold when instead considering the mean APE, with the exception that SeqMO outperforms PolySeqMO on the Event and Hypo subtasks (12.63 *vs.* 12.79 and 16.60 *vs.* 17.04 respectively). These results illustrate the strengths (and weaknesses) of the proposed forecasting systems applied to the task of predicting blood glucose. We discuss the implications of these results in the sections that follow.

5.6.1 Deep *vs.* Shallow.

While RF: MO achieves good performance on the full test set, it does worse than our three improved deep multi-output methods. Moreover, it underperforms the deep approaches on the event subset. This is due mainly to very poor performance on hypoglycemic predictions. These results suggest that, compared to shallow models, deep models can more accurately learn the underlying dynamics of the glucoregulatory system from raw data.

Still, we note that RF is a competitive forecaster, in line with previous work [119]. In particular, it achieves lower APE on the hyperglycemic test set than all models except the PolySeqMO Ensemble.

5.6.2 Multi-Output *vs.* Recursive.

We observe that among both deep (DeepMO *vs.* Recursive: 5.01 *vs.* 5.31) and shallow approaches (RF: MO *vs.* Rec: 5.18 *vs.* 8.00), multi-output forecasting offers significant advantages over recursive forecasting. We also observe that all deep multi-output models improve on the Recursive model in the hypoglycemic task (12.05-12.91 *vs.* 13.34).

We highlight these differences further in **Figure 5.6**. We plot the average performance at each of the six time points within the 30-minute prediction window. The difference in the approaches is amplified as we predict further out. At the first place in the prediction window, corresponding to predicting x_{t+1}, the recursive approach outperforms most other approaches. As the target moves further in the future, we observe two trends. First, the problem becomes more difficult for all approaches (*i.e.,* MO Error increases). This makes sense, as it is inherently more difficult to predict events further in the future. Second, the relative performance of the Recursive forecaster degrades with respect to the multi-output approaches. By the final step, the recursive model is far and away the worst predictor (8.38 *vs.* 7.51-7.74).

5.6.3 Adding Sequential Dependencies

Examining the difference in performance between the DeepMO and SeqMO models, we note the autoregressive connections improve performance across every subset of the data (4.91 *vs.* 5.01 on the full test set). While the multi-output approach under-performs the deep recursive forecaster on the hyperglycemic task, once we add the sequential decoding, the resulting model beats the recursive forecaster on every task. This indicates that, while multi-output forecasting represents a step in the right direction, it is important to consider sequential dependencies between outputs.

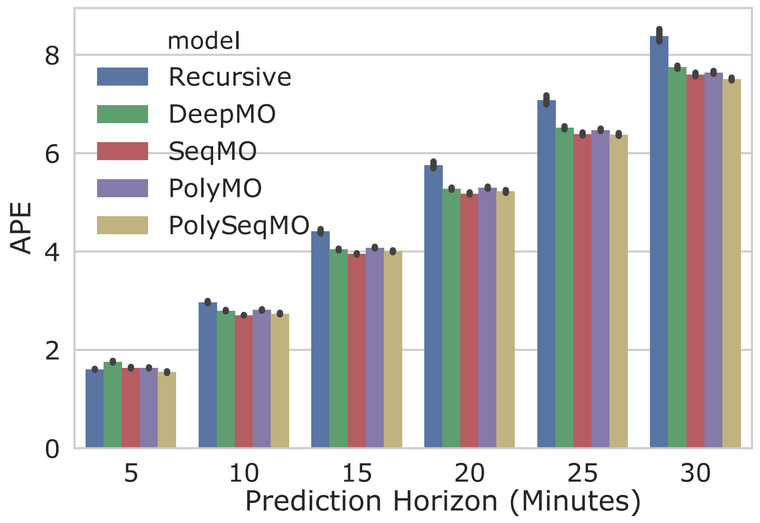


FIGURE 5.6: A comparison of per-step error between the various forecasters. While the multi-output models initially perform worse, they do not accumulate error as rapidly as the recursive approach, achieving lower error at later prediction steps.

5.6.4 Predicting Underlying Function *vs.* Values.

In our investigation of PolyMO, we began by looking at the performance attained using a range of polynomial function classes. We looked at four different degree settings for the best-fit polynomial we predict (0-3 degree polynomials). We found the degree 1 model achieved the best performance on the validation set (4.87 *vs.* next best 5.14). While higher degree polynomials allow for strictly better approximations of prediction windows, they also allow for more variation in output. Moreover, minor errors in high-degree coefficients rapidly compound to large errors.

Focusing on the 1st degree PolyMO, we see it is advantageous to rephrase the value forecasting problem as a function forecasting one. We find that PolyMO beats DeepMO on every task.

Moreover, we find that the improvements from function forecasting are complimentary to those achieved by accounting for sequential output dependencies. By combining the SeqMO output decoder with the PolyMO function forecasting, resulting in PolySeqMO, we achieve better performance than all other non-ensemble models in every task

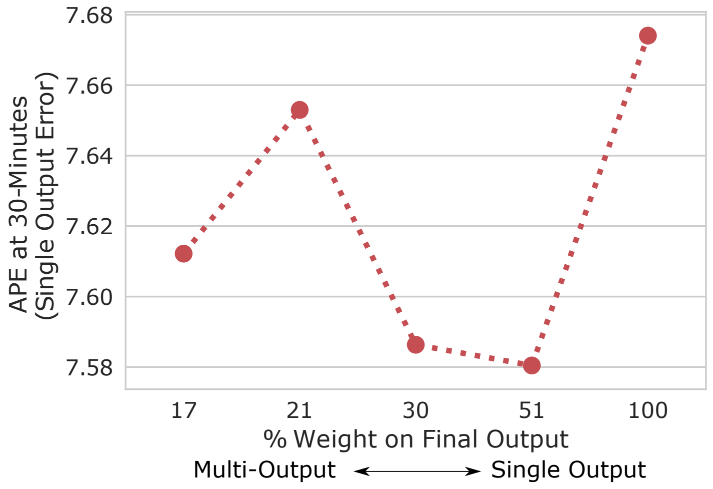


FIGURE 5.7: We examine the single output error across a range of different model types, determined using an exponential loss weighting. We derive the proportion of weight allocated to the final output (which represents the evaluation target). Surprisingly, we observe that a multi-output loss improves *single*-output performance, suggesting that it is helpful to model forecast trajectories even when you only care about the final value.

under consideration. The fact that PolySeqMO does well across all subsets of the data suggests that it encodes a more accurate and complete view of the underlying dynamics of the glucoregulatory system.

Both PolyMO and PolySeqMO focus entirely on modeling value trajectories as opposed to the values themselves. Given that we are evaluating using a multi-output metric, this built-in assumption may appear to drive the boost in performance. However, upon inspecting the performance of the Linear Extrapolation baseline, we conclude that this is not the case. Degree 1 PolyMO and PolySeqMO are equivalent in output capacity to the Linear Extrapolation baseline, and both inherently emphasize trajectories. However, the Linear Extrapolation does far worse. PolySeqMO significantly reduces the error of the Linear Extrapolation approach on the full dataset (4.87 *vs.* 6.48). This demonstrates that the value of PolySeqMO is its ability to predict the future, not its assumption of linear trajectories. However, the fact that all models improve performance under polynomial smoothing suggests there is some value in the trajectory assumption.

5.6.5 Ensembling

While PolySeqMO is the best individual model in every task, it under-performs RF: MO on hyperglycemic prediction (8.31 *vs.* 8.20). While this could be due to the fact that RF: MO is simply better suited to that task, it could also be a result of the general improvements observed when ensembling different model performances. To test this effect, we trained 10 PolySeqMO models on the same training set, varying only the random seed for initialization and training batch ordering. We then averaged the results of each models prediction on the test set by taking the mean.

We found that even this simple ensembling scheme with few models (relative to the 100-model ensemble used in RF: MO) leads to a sizeable increase in performance across all tasks. In particular, we find the PolySeqMO ensemble outperforms RF: MO on the hyperglycemic prediction task (8.13 *vs.* 8.20).

5.6.6 Multi-output *vs.* Direct Forecasting

There are many forecasting problems in which an accurate single-output forecast may suffice. In such cases, it is common to use a direct forecaster [15], where one *directly* estimates $p(\mathbf{x}_{t+h}|\mathbf{x}_{0:t})$. In a follow-up analysis, we demonstrate that even in cases where only a single output is desired, it can be beneficial to consider a multi-output forecasting framework.

To demonstrate this, we begin by introducing a method to transition from multi-output to direct forecasting, focusing on SeqMO. This model operates by predicting multiple values at each time step. The training loss is the average across each of the six time steps in the prediction window. A direct forecaster, aiming to make a single prediction at the final time step, can be approximated by zeroing out all losses except those incurred at the final step. This focuses the full capacity of the network on predicting the final value. We can flexibly transition between direct and multi-output forecasting by manipulating the per-step loss weighting, transitioning from a one-hot vector on the final output (direct) to a uniform allocation of weight across the window (multi-output). We encapsulate this transition in a single hyperparameter, $0 \leq b_w \leq 1$, or the base for the per-step loss weight. For each step i in a prediction window of length h, we set the loss weight $w_{l,i} = \frac{b_w^{h-i}}{\sum_{i=0}^{h} b_w^{h-i}}$.

In **Figure 5.7** we show the single output performance (predicting 30 minutes into the future) of SeqMO with different settings of b_w. We observe that modeling the full trajectory does not worsen performance, and in fact appears to slightly improve it (MO 7.67 *vs.* Direct 7.61). Interestingly, we found that best single-output performance was achieved using an intermediate value for b_w (7.58 with $b_w = 0.5$). Mixing multi-output and direct forecasting strategies could be a promising direction for improving single-output forecasting performance.

5.7 Summary and Conclusions

In this chapter, we investigated methods for deep multi-output blood glucose forecasting. Applied to the challenging task of predicting blood glucose, we demonstrated that: 1) multi-output methods outperform recursive alternatives, 2) modeling underlying dependencies among outputs using explicit connections and function forecasting leads to better performance, and 3) the proposed approaches are complementary, and combining them significantly improves performance. Additionally, we demonstrated that multi-output forecasting improves performance, even on a single-output task.

These experimental results suggest a multi-output approach can effectively capture the underlying dynamics of the glucoregulatory system. While we focus on blood glucose, forecasting real valued signals is a problem with applications across a number of different domains including speech processing, weather prediction, and medicine [33], [47], [127]. Our proposed methods may be applied to any forecasting problem requiring multi-step predictions.

Chapter 6

Deep Reinforcement Learning for Blood Glucose Management

6.1 Introduction

In the previous chapter, we discussed glucose forecasting, an important application for predictive alarms. If machine learning approaches can model the glucoregulatory system reasonably well, a natural question is how well could a machine learning approach control such a system? In this chapter, we examine that question in detail.

People with type 1 diabetes (T1D) need to manage their blood glucose levels consistently using insulin to achieve good glycemic control. Getting the correct dose requires careful measurement of glucose levels and carbohydrate intake, resulting in at least 15-17 data points a day. When using a continuous glucose monitor (CGM), this can increase to over 300 data points, or a blood glucose reading every 5 minutes [129].

CGMs with an insulin pump, a device that delivers insulin, can be used with a closed-loop controller as an 'artificial pancreas' (AP). Though the technology behind CGMs and insulin pumps has advanced, there remains significant room for improvement when it comes to the control algorithms [130], [131]. Current hybrid closed-loop approaches require accurate meal announcements to maintain glucose control.

Our Approach. In this chapter, we investigate deep reinforcement learning (RL) for blood glucose management. RL is a promising solution, as it is well-suited to learning

The following is an adaptation and expansion of previously published work [128].

complex behavior and readily adapts to changing domains [132]. In particular, RL can learn effective policies when the environment is not fully observed. For the problem of blood glucose management, this is an important benefit. Blood glucose values are strongly affected by events, such as meals, that cannot be automatically recorded. We hypothesize that deep RL, the combination of RL with a deep neural network, will be able to accurately infer latent meals to control insulin. Furthermore, as RL is a learning-based approach, we hypothesize that RL will adapt to predictable meal schedules better than baseline approaches.

The fact that RL is learning-based means it requires data to work effectively. Unlike many other health settings, there are credible simulators for blood glucose management [133]. Having a simulator alleviates many concerns of applying RL to health problems [52], [134]. However, that does not mean RL for blood glucose control is straightforward, and, in this paper, we identify and address several challenges. To the best of our knowledge, we present the first deep RL approach that achieves human-level performance in controlling blood glucose without requiring meal announcements. Accomplishing this required several important considerations, outlined below.

- The range of insulin and carbohydrate requirements across patients makes it difficult to find a single action space that balances the needs of rapid insulin administration and safety. To solve this problem, we present a robust patient-specific action space that naturally encourages safer policies.

- We found several pitfalls in evaluating our proposed approach that led to unrealistic performance estimates. To address this issue, we used validation data to perform careful model selection, and used extensive test data to evaluate the quality of our models.

- Deep RL has been shown to be unstable [135], [136], often achieving poor worst-case performance. This is unacceptable for safety-critical tasks, such as those in healthcare. We found that a combination of simple and widely applicable approaches stabilized performance. In particular, we used a safety-augmented reward function, realistic randomness in training data, and random restarts to train models that behaved safely over thousands of days of evaluation.

- Finally, unlike game settings where one has ability to learn from hundreds of thousands of hours of interaction, any patient-specific model must be able to achieve strong performance using a limited amount of data. We show that a simple transfer learning approach can be remarkably sample efficient and can even surpass the performance of models trained from scratch.

In a followup analysis, we consider a more realistic setting in which we have access to data collected by a behavior policy, known as *batch* reinforcement learning. Due to the presence of reasonable physiological models in our application, we also assume access to an imperfect simulator. This setting addresses the safety concerns of training RL approaches on real individuals. We compare policies learned using only data collected by a behavior policy, such as automatically logged human behavior, to ones learned using a simulator. We present an approach to augment common batch RL algorithms with an imperfect simulator. We demonstrate that batch RL is applicable to the task of glucose control, and that batch RL augmented with a simulator can perform even better.

Organization. The remainder of this chapter is organized as follows. We describe current approaches to manage and simulated blood glucose. We frame the problem of blood glucose management as an MDP and describe our approaches to solve it. We then present our results using a simulator, demonstrating how we achieved strong and consistent blood glucose control without requiring meal announcements. Finally, we describe our application of batch reinforcement learning to this problem. We demonstrate how augmenting batch RL with an imperfect simulator can lead to improvements over standard algorithms.

6.2 Background and Related Work

In recent years, researchers have started to explore RL in healthcare. Examples include matching patients to treatment in the management of sepsis [137], [138] and mechanical ventilation [139]. In addition, RL has been explored to provide contextual suggestions for behavioral modifications [140]. Despite its successes, RL has yet to be fully explored as a solution for a closed-loop AP system [130]. To provide necessary background, we discuss how RL and other approaches have been used for blood glucose control and present an overview on blood glucose simulation.

6.2.1 Algorithms for Blood Glucose Control

Among recent commercial AP products, proportional-integral-derivative (PID) control is the most common backbone [141]. The simplicity of PID controllers make them easy to use, and in practice they achieve strong results [142]. The Medtronic Hybrid Closed-Loop system, one of the few commercially available, is built on a PID controller [143], [144]. A hybrid closed-loop controller adjusts baseline insulin rates but requires human intervention to control for the effect of meals. The main weakness of PID controllers is their reactivity. As a result, they often cannot react fast enough to meals, and thus rely on meal announcements [143]. Additionally, without safety modifications, PID controllers can deliver too much insulin, triggering hypoglycemia [144]. In contrast, we hypothesize that an RL approach will be able to leverage patterns associated with mealtimes, resulting in more responsive and safer policies.

Previous works have examined RL for different aspects of blood glucose control. See [145] for a recent survey. Many of these works investigated the use of RL to adapt existing insulin treatment regimens to learn a 'human-in-the-loop' policy [146]–[148]. This contrasts with our setting, where we aim to learn a fully closed-loop policy.

Similar to our work, [149] and [150] focus on the task of closed-loop glucose control. [149] use direct future prediction to aid PID-style control, substituting the problem of RL with prediction. [150] use a policy-iteration framework with Gaussian process approximation and Bayesian active learning. While they tackle a similar problem, these works use a simple simulator and a fully deterministic meal routine for training and testing. In our experiments, we use an FDA-approved glucose simulator and a realistic non-deterministic meal schedule, significantly increasing the challenge.

6.2.2 Glucose Models and Simulation

Models of the glucoregulatory system are important for the development and testing of an AP [25]. In our experiments, we use the UVA/Padova model [26]. This simulator models the glucoregulatory system as a nonlinear multi-compartment system, where glucose is generated in the liver, absorbed through the gut, and controlled by external insulin. A more detailed explanation can be found in [26]. For reproducibility, we use an open-source version of the simulator that comes with 30 virtual patients [29]. The parameter distributions of the patient population are determined by age, and the simulator comes with 10 children, adolescents, and adults each [26]. We combine the simulator with a non-deterministic meal schedule (**Appendix B.1**) to realistically simulate patient behavior.

6.3 Online RL

In this section, we consider the problem of learning blood glucose management policies in an online fashion. We provide a problem formulation and a set of methods that can improve glucose control over standard baselines while maintaining safety.

6.3.1 Methods

We present a deep RL approach well suited for blood glucose control. In framing our problem, we pay special attention to the concerns of partial observability and safety. The issue of partial observability motivates us to use a maximum entropy control algorithm, soft actor-critic, combined with a recurrent neural network. Safety concerns inspire many aspects of our experimental setup, including our choice of action space, reward function, and evaluation metrics. We also introduce several strong baselines, both with and without meal announcements, to which we compare.

Problem Setup

We frame the problem of closed-loop blood glucose control as a partially-observable Markov decision process (POMDP) consisting of the 7-tuple $(S^*, O, S, A, P, R, \gamma)$. A POMDP generalizes the MDP setting described in **Section 2.2.3** by assuming we do not have access to the true environment states, here denoted $s^* \in S^*$, but instead observe noisy states $s \in S$ according to the (potentially stochastic) observation function: $O : s^* \to s$. This setting applies given the noise inherent in CGM sensors [151] and our assumption of unobserved meals.

In our setting, the true states $\mathbf{s}_t^* \in S^*$ are the 13-dimensional simulator states, as described in [26]. The stochastic observation function $O : \mathbf{s}_t^* \rightarrow b_t, i_t$ maps from the simulator state to the CGM observation b_t and insulin i_t administered. To provide temporal context, we augment our observed state space $\mathbf{s}_t \in S$ to include the previous 4 hours of CGM $\mathbf{b}^t$ and insulin data $\mathbf{i}^t$ at 5-minute resolution: $\mathbf{s}_t = [\mathbf{b}^t, \mathbf{i}^t]$ where:

$$\mathbf{b}^t = [b_{t-47}, b_{t-46}, \ldots, b_t], \mathbf{i}^t = [i_{t-47}, i_{t-46}, \ldots, i_t],$$

$b_t \in \mathbb{N}_{40:400}$, $i_t \in \mathbb{R}_+$, and $t \in \mathbb{N}_{1:288}$ represents a time index for a day at 5-minute resolution. Note that in our augmented formulation, the observation function no longer directly maps to observed states, as observed states incorporate significant amounts of historical data. We chose a 4-hour window, as we empirically found it led strong performance. We use a time resolution of 5 minutes to mimic the sampling frequency of many common CGMs. Actions $a_t \in \mathbb{R}_{\geq 0}$ are real positive numbers, denoting the size of the insulin bolus in medication units.

The transition function P, our simulator, consists of two elements: i) $M : t \rightarrow c_t$ is the meal schedule, and is defined in **Appendix B.1**, and ii) $G : (a_t, c_t, \mathbf{s}_t^*) \rightarrow (b_{t+1}, i_{t+1}, \mathbf{s}_{t+1}^*)$, where $c_t \in \mathbb{R}_{\geq 0}$ is the amount of carbohydrates input at time t and G is the UVA/Padova simulator [26]. Note that our meal schedule is patient-specific, and includes randomness in the daily number, size, and timing of meals.

The reward function $R : (\mathbf{s}_t, a_t) \rightarrow \mathbb{R}$ is defined as negative $-risk(b_t)$ where $risk$ is the Magni risk function:

$$risk(b) = 10 * (c_0 * \log(b)^{c_1} - c_2)^2, \tag{6.1}$$

$c_0 = 3.35506$, $c_1 = 0.8353$, and $c_2 = 3.7932$ [152]. These values are set such that $risk(70) = risk(280) = 25$, see **Figure 6.1**. Finally, we set $\gamma = 0.99$ for our experiments, a value we determined empirically on validation data in combination with the early termination penalty.

Considered in isolation, our reward could lead to dangerous behavior. As it is always negative, cumulative reward is maximized by ending the episode as quickly as possible, which occurs when glucose reaches unsafe levels. To avoid this, we add a termination penalty of $-1e5$ to trajectories that enter dangerous blood glucose regimes (blood glucose

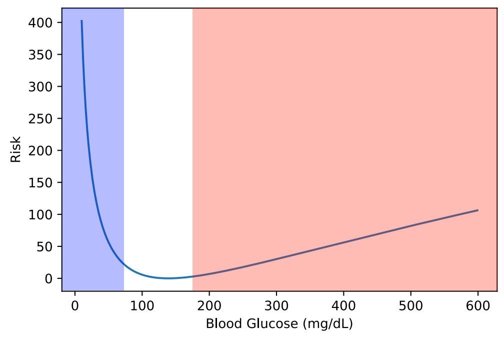


FIGURE 6.1: The risk function proposed in [152]. The mapping between blood glucose values (in mg/dL, x-axis) and risk values (y-axis). Blood glucose levels corresponding to hypoglycemia are shown in the blue shaded region, the glucose range corresponding to hyperglycemia is shown in the red shaded region. This function identifies low blood glucose values as higher risk than high blood glucose values, which is sensible given the rapidity of hypoglycemia.

levels less than 10 or more than 1,000 mg/dL), countering the advantage of early termination. We investigated other reward functions, such as time in range or distance from a target blood glucose value, but found this reward worked best. It led to control schemes with less hypoglycemia, as low blood glucose is penalized more heavily than high glucose. Low glucose can occur quickly when large amounts of insulin are given without an accompanying meal. Given the lack of meal announcements and sensor noise in our setting, avoiding hypoglycemia was a significant challenge.

Soft Actor-Critic

We chose to use the soft actor-critic (SAC) algorithm to learn glucose control policies. We initially experimented with a Deep-Q Network approach [153]. However, choosing a discretized action space (as is required by Q-learning) that accounted for the range of insulin values across a day and allowed exploration proved impractical, as large doses of inappropriately timed insulin can be dangerous. Among continuous control algorithms, we selected SAC as it has been shown to be sample efficient and competitive [154]. Additionally, maximum entropy policies like the ones produced by SAC can do well in partially observed settings like our own [155].

SAC produces a stochastic policy $\pi : \mathbf{s}_t \to p(a)\ \forall a \in A$, which maps a state to a distribution over possible actions. Under SAC, the policy (or actor) is represented by a neural network with parameters ϕ. Our network generates outputs $\mu, \log(\sigma)$ which parameterize a normal distribution $\mathcal{N}(\mu, \sigma)$. The actions are distributed according to a TanhNormal distribution, or $tanh(z), z \sim \mathcal{N}(\mu, \sigma)$. π_ϕ is trained to maximize the maximum entropy RL objective function:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, a_t) \sim P(\mathbf{s}_{t-1}, \pi_\phi(\mathbf{s}_{t-1}))}[R(\mathbf{s}_t, a_t) + \alpha H(\pi_\phi(\cdot|\mathbf{s}_t))], \tag{6.2}$$

where entropy, H, is added to the expected cumulative reward to improve exploration and robustness [154]. Intuitively, the return in **Equation 6.2** encourages a policy that can obtain a high reward under a variety of potential actions. The temperature hyperparameter α controls the tradeoff between reward and entropy. In our work, we set this using automatic temperature tuning [156]. **Equation 6.2** is optimized by minimizing the KL divergence between the action distribution and the distribution induced by the state-action values:

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathrm{D_{KL}} \left(\pi_\phi \left(\cdot | \mathbf{s}_t \right) \| \frac{\exp\left(Q_\theta\left(\mathbf{s}_t, \cdot\right)\right)}{Z_\theta\left(\mathbf{s}_t\right)} \right) \right] \tag{6.3}$$

where $\mathcal{D}$ is a replay buffer containing previously seen $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ tuples, Z_θ is a partition function, and Q_θ is the state-action value function parameterized by a neural network (also called a critic). This means that our learned policy engages in probability matching, selecting an action with probability proportional to its expected value. This requires an accurate value function. To achieve this, Q_θ is trained by minimizing the temporal difference loss:

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\theta\left(\mathbf{s}_t, \mathbf{a}_t\right) - \hat{Q}\left(\mathbf{s}_t, \mathbf{a}_t\right) \right)^2 \right], \tag{6.4}$$

$$\hat{Q}\left(\mathbf{s}_t, \mathbf{a}_t\right) = r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p} \left[V_{\overline{\psi}}\left(\mathbf{s}_{t+1}\right) \right]. \tag{6.5}$$

V_ψ is the soft value function parameterized by a third neural network, and $V_{\overline{\psi}}$ is the running exponential average of the weights of V_ψ over training. This is a continuous variant of the hard target network replication in [153]. V_ψ is trained to minimize:

$$J_V(\psi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V_\psi \left(\mathbf{s}_t \right) - \mathbb{E}_{\mathbf{a}_t \sim \pi_\rho} \left[Q_\theta \left(\mathbf{s}_t, \mathbf{a}_t \right) - \log \pi_\phi \left(\mathbf{a}_t | \mathbf{s}_t \right) \right] \right)^2 \right]. \tag{6.6}$$

In summary: we learn a policy that maps from states to a probability distribution over actions, the policy is parameterized by a neural network π_ϕ. Optimizing this network (**Equation 6.3**) requires an estimation of the soft state-action value function, we learn such an estimate Q_θ (**Equation 6.4**) together with a soft value function V_ψ (**Equation 6.6**). Additional details of this approach, including the gradient calculations, are given in [154]. In keeping with previous work, when testing our policy we remove the sampling component, instead selecting the mean action $tanh(\mu)$. We replace the MSE temporal difference loss in **Equation** 6.4 with the Huber loss, as we found this improved convergence.

Recurrent Architecture

Our network π_ϕ takes as input the past 4 hours of CGM and insulin data, requiring no human input (*i.e.*, no meal announcements). To approximate the true state $\mathbf{s}_t^*$ from the augmented state s we parameterize Q_θ, V_ψ, and π_ϕ using gated-recurrent unit (GRU) networks [118], as GRUs have been successfully used for glucose modeling previously [110], [157].

Patient-Specific Action Space

After the network output layer, actions are squashed using a *tanh* function. Note that this results in half the action space corresponding to negative values, which we interpret as administering no insulin. This encourages sparse insulin utilization and makes it easier for the network to learn to safely control baseline glucose levels. To ensure that the maximum amount of insulin delivered over a 5-minute interval is roughly equal to a normal meal bolus for each individual, we use the average ratio of basal to bolus insulin in a day [158] to calculate a scale parameter for the action space, $\omega_b = 43.2 * bas$, where *bas* is the default patient-specific basal insulin rate provided by [29].

Efficient Policy Transfer

One of the main disadvantages of deep RL is sample efficiency. Thus, we explored transfer learning techniques to efficiently transfer existing models to new patients. We refer to our method trained from scratch as RL-Scratch, and the transfer approach as RL-Trans. For RL-Trans, we initialize Q_θ, V_ψ and π_ϕ for each class of patients (children, adolescents, and adults) using fully trained networks from one patient of that source population (see **Appendix B.4**). We then fine-tune these networks on data collected from the target patient.

When fine-tuning, we modify the reward function by removing the termination penalty and adding a constant positive value (100) to all rewards. This avoids the negative reward issue discussed in **Section 6.3.1 - Problem Setup**. Removing the termination penalty increased the consistency of returns over training, allowing for a more consistent policy gradient. The additional safety provided by a termination penalty is not required as we begin with policies that are already stable. We found this simple approach for training patient-specific policies attains good performance while using far less patient-specific data.

Baselines

We examine three baseline methods for control: basal-bolus (**BB**), **PID** control, and PID with meal announcements (**PID-MA**). BB reflects an idealized human-in-the-loop control strategy, and PID reflects a common closed-loop AP algorithm. PID with meal announcements is based on current AP technology, and is a combination of the two, requiring regular human intervention. Finally, we consider an 'oracle' approach that has access to the true state s_t^*. This decouples the task of learning a policy from state inference, serving as a pseudo-upper bound on performance for our proposed approach.

Basal-Bolus Baseline. This baseline is designed to mimic human control and is an ideal depiction of how an individual with T1D controls their blood glucose. In this setting, we modify the state representation $\mathbf{s}_t$ to include a carbohydrate signal and a cooldown signal (explained below), and to remove the historical data, $\mathbf{s}_t = [b_t, i_t, c_t, cooldown]$. Note that the inclusion of a carbohydrate signal, or meal announcement, places the burden of providing accurate and timely estimates of meals on the person with diabetes. Each virtual patient in the simulator comes with the parameters necessary to calculate a reasonable basal insulin rate *bas* (the same value used in our action space definition), correction factor CF, and carbohydrate ratio CR. These three parameters, together with a glucose target b_g, define a clinician-recommended policy $\pi(s_t) = bas + (c_t > 0) * (\frac{c_t}{CR} + cooldown * \frac{b_t - b_g}{CF})$ where *cooldown* is 1 if there have been no meals in the past three hours, otherwise it is 0. The cooldown ensures that each meal is only corrected for once. Appropriate settings for these parameters can be estimated by endocrinologists, using previous glucose and insulin information [159]. The parameters for our virtual patient population, which are derived from a distribution validated by clinical trials [26], are given in **Appendix B.2**.

PID Baseline. PID controllers are a common and robust closed-loop baseline [142]. A PID controller operates by setting the control variable, a_t, to the weighted combination of three terms $a_t = k_P P(b_t) + k_I I(b_t) + k_D D(b_t)$ such that the process variable b_t (t is the time index) remains close to a specified setpoint b_g. The terms are calculated as follows: i) the proportional term $P(b_t) = \max(0, b_t - b_g)$ increases the control variable proportionally to the distance from the setpoint, ii) the integral term $I(b_t) = \sum_{j=0}^{t}(b_j - b_g)$ corrects long-term deviations from the setpoint, and iii) the derivative term $D(b_t) = |b_t - b_{t-1}|$ uses the rate of change as a basic estimate of the future. The set point and the weights (also called gains) k_P, k_I, k_D are hyperparameters. To compare against the strongest PID controller possible, we tuned these hyperparameters using multiple iterations of grid-search with exponential refinement per-patient, minimizing risk. Our final settings are presented in **Appendix B.3**.

PID with Meal Announcements. This baseline, which is similar to available hybrid closed loop AP systems [143], [144], combines BB and PID into a control algorithm we call PID-MA. During meals, insulin boluses are calculated and applied as in the BB approach. The basal rate, instead of being fixed, is controlled by a PID algorithm, allowing for adaptation between meals. As above, we tune the gain parameters for the PID algorithm using sequential grid search to minimize risk.

Oracle Architecture. A deep RL approach to learning AP algorithms requires that the representation learned by the network contain sufficient information to control the system. As we are working with a simulator, we can explore the difficulty of this task in isolation, by replacing the observed state $\mathbf{s}_t$ with the ground-truth state $\mathbf{s}_t^*$. Though unavailable in real-world settings, this representation decouples the problem of learning a policy from that of inferring the state. Here, Q_θ, V_ψ, and π_ϕ are fully connected with two hidden layers, each with 256 units. The network uses ReLU nonlinearities and BatchNorm [160].

6.3.2 Experimental Setup & Evaluation

To measure the utility of deep RL for the task of blood glucose control, we trained and tested our policies on data with different random seeds across 30 different simulated individuals.

Training and Hyperparameters.

We trained our models separately for each patient. They were trained from scratch for 300 epochs for RL-Scratch, and fine-tuned for 50 epochs for RL-Trans. They were trained with batch size 256 and an epoch length of 20 days. We used an experience replay buffer of size 1e6 and a discount factor of 0.99. We found that extensive training from-scratch was required to obtain consistent performance across test runs. We also found that too small of an epoch length could lead to dangerous control policies. We optimized the parameters of Q_θ, V_ψ and π_ϕ using Adam with a learning rate of $3E-4$. All deep networks were composed of two layers of GRU cells with a hidden state size of 128, followed by a fully-connected output layer. All network hyperparameters, including number and size of layers, were optimized on training seeds on a subset of the simulated patients for computational efficiency. Our networks were initialized using PyTorch defaults.

Evaluation.

We measured the performance of π_ϕ on 10 days of validation data after each training epoch. After training, we evaluated on test data using the model parameters from the best epoch as determined by the validation data. While this form of model selection is not typical for RL, we found it led to significant changes in performance (see **Section 6.3.5 - Potential Pitfalls in Evaluation**). Our model selection procedure first filters out runs that could not control blood glucose within safe levels over the validation run (glucose between 30-1000 mg/dL), then selects the epoch that achieved the lowest risk. We tested each patient-specific model on 1000 days of test data, broken into 100 independent 10 day rollouts. We trained and evaluated each approach 3 times, resulting in 3000 days of evaluation per method per person.

We evaluated approaches using i) risk, the average Magni risk calculated over the 10-day test rollout, ii) % time spent euglycemic (blood glucose levels between 70-180 mg/dL), iii) % time hypo/hyperglycemic (blood glucose lower than 70 mg/dL or higher than 180 mg/dL respectively), and iv) % of rollouts that resulted in a catastrophic failure, which we define as a run that achieves a minimal blood glucose level below 5 mg/dL (at which point recovery becomes highly unlikely). Note that while catastrophic failures are a major concern, our simulation process does not consider consuming carbohydrates in reaction to low blood glucose levels. This is a common strategy to avoid dangerous hypoglycemia in real life, and thus catastrophic failures, while serious, are manageable. The random seeds controlling noise, meals, and all other forms of randomness, were different between training, validation, and test runs. We test the statistical significance of differences between methods using Mood's median test for all metrics except for catastrophic failure rate, for which we use a binomial test.

6.3.3 Overview of Experiments

Our experiments are divided into two broad categories: i) experiments showing the benefits of deep RL for blood glucose control relative to baseline approaches and ii) the challenges of using deep RL in this setting, and how to overcome them.

Throughout our experiments, we consider 3 variants of RL methods: i) RL-Scratch, our approach trained from scratch on every individual, ii) RL-Trans, which fine-tunes an RL-Scratch model from an arbitrary child/adolescent/adult, and iii) RL-MA, which uses RL-Scratch trained using the automated meal boluses from BB or PID-MA. We also report results on an Oracle approach, which is trained and evaluated using the ground truth simulator state.

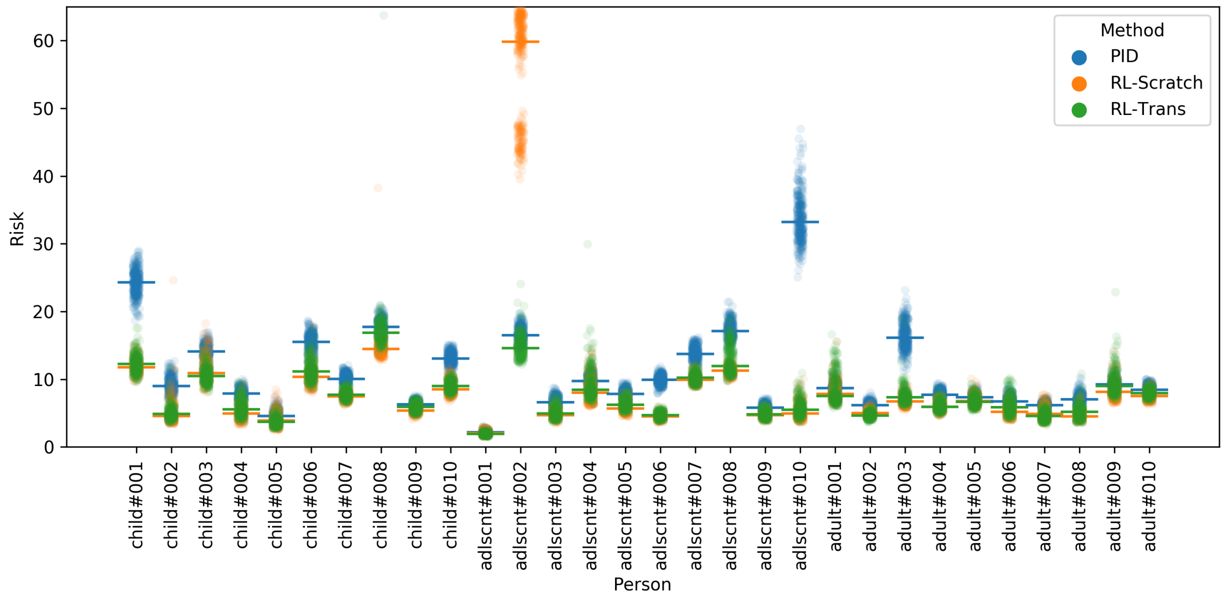


FIGURE 6.2: The risk over 10 days for different simulated patients using methods that do not require meal announcements. Each point corresponds to a different random test seed that controls the meal schedule and sensor noise, and the line indicates the median performance for each method on each patient. Results are presented across 3 random training seeds, controlling model initialization and randomness in training. We observe that, although there is a wide range in performance across and within individuals, The RL approaches tend to outperform PID.

6.3.4 Advantages of Deep RL

We compare our deep RL approaches to baselines with and without meal announcements across several metrics (**Section 6.3.4 - Deep RL *vs.* Baseline Approaches**). We then investigate two hypotheses for why deep RL is well suited to the problem of glucose management without meal announcements:

- the high-capacity neural network, integral to the RL approaches, is able to quickly infer when meals occur (**Section 6.3.4 - Detecting Latent Meals**), and

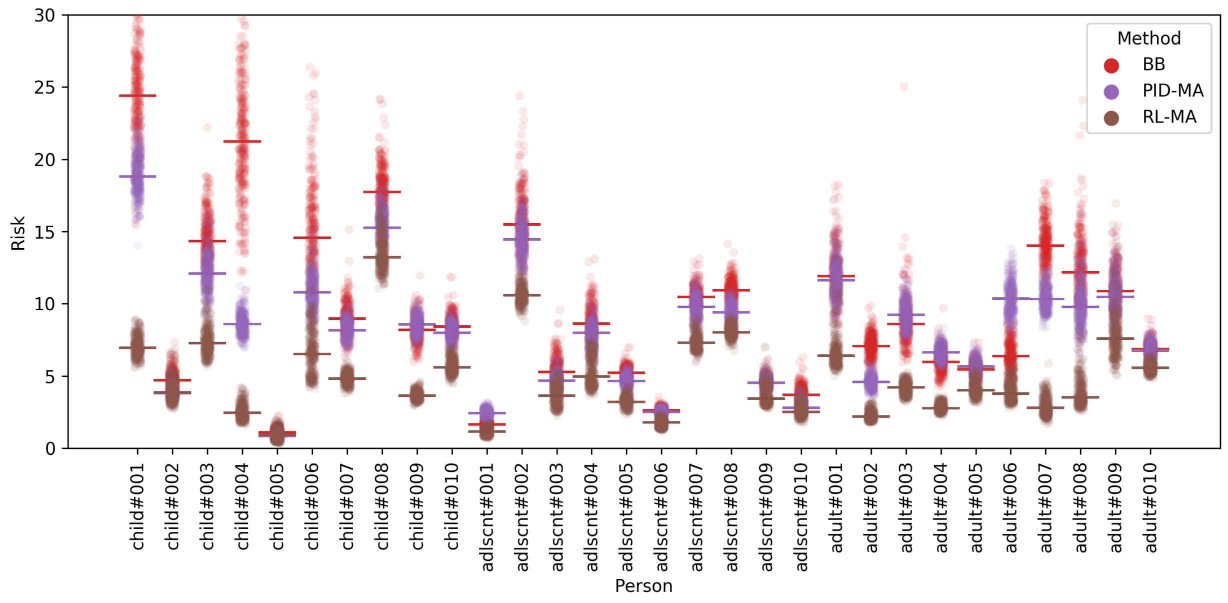


FIGURE 6.3: The risk over 10 days using methods that require meal announcements. PID-MA tends to outperform BB, and RL-MA outperforms PID-MA.

- the learning-based approach is able to adapt to behavioral patterns better than a PID controller (**Section 6.3.4 - Ability to Adapt to Behavioral Schedules**).

Deep RL vs. Baseline Approaches

A comparison between the PID baseline to the RL approaches is presented in **Figure 6.2**. Each point represents a different test rollout by a policy. For the RL approaches, the performance of each method is reported as the combination of 3 random training restarts. Among the 3 methods that do not require meal announcements, RL-Scratch performs best across patients (average rank 1.33), followed by RL-Trans (average rank 1.7), then PID (average rank 2.97). For each individual, we rank the 3 approaches in terms of median risk. We calculate average rank by taking the mean of each approach's rankings across all 30 individuals. Note that RL-Scratch, while achieving strong performance overall, reliably performs poorly on adolescent#002. We discuss this issue in **Appendix B.5**.

One major advantage of our proposed approach is its ability to achieve strong performance without meal announcements. This does not mean that it does not benefit from meal announcements, as shown in **Figure 6.3**. Among the 3 methods that require meal announcements, RL-MA performs best (average rank 1.07), followed by PID-MA (average rank 2.13) then BB (average rank 2.8).

We examine additional metrics in the results presented in **Table 6.1**. The difference between results that are bold, or bold and underlined, and the next best non-bold result (excluding RL-Oracle) are statistically significant with $p < 0.001$. We observe that RL-MA equals or surpasses the performance of all non-oracle methods on all metrics, except for % time spent hyperglycemia. Interestingly, all RL variants achieve lower median risk than PID-MA, which requires meal announcements. This is because the RL approaches achieve low levels of hypoglycemia, which the risk metric heavily penalizes (see **Figure 6.1**). Note that all methods, including PID-MA, were optimized to minimize this metric. Across patients, the RL methods achieve approximately 60-80% time euglycemic, compared with 52% $\pm$ 19.6% observed in real human control [161]. These results suggest that deep RL could be a valuable tool for closed-loop or hybrid closed-loop AP control.

TABLE 6.1: Median risk, percent of time Eu/Hypo/Hyperglycemic, and failure rate calculated using 1000 days of simulation broken into 100 independent 10-day rollouts for each of 3 training seeds for 30 patients, totaling 90k days of evaluation (with interquartile range). Lower Magni Risk, Hypoglycemia, and Hyperglycemia are better, higher Euglycemia is better. Hybrid and Non-closed loop approaches (requiring meal announcements) are indicated with *. Approaches requiring a fully observed simulator state are indicated with †. The non-oracle approach with the best average score is in bold and underlined, the best approach that does not require meal announcements is in bold.

		Risk $\downarrow$	Euglycemia (%) $\uparrow$	Hypoglycemia (%) $\downarrow$	Hyperglycemia (%) $\downarrow$	Failure (%) $\downarrow$
No MA	PID	8.86 (6.8-14.3)	71.68 (65.9-75.9)	1.98 (0.3-5.5)	**24.71 (21.1-28.6)**	0.12
	RL-Scratch	**6.50 (4.8-9.3)**	**72.68 (67.7-76.2)**	**0.73 (0.0-1.8)**	26.17 (23.1-30.6)	**0.07**
	RL-Trans	6.83 (5.1-9.7)	71.91 (66.6-76.2)	1.04 (0.0-2.5)	26.60 (22.7-31.0)	0.22
MA	BB*	8.34 (5.3-12.5)	71.09 (62.2-77.9)	2.60 (0.0-7.2)	23.85 (17.0-32.2)	0.26
	PID-MA*	8.31 (4.7-10.4)	76.54 (70.5-82.0)	3.23 (0.0-8.8)	**18.74 (12.9-23.2)**	**0.00**
	RL-MA*	**4.24 (3.0-6.5)**	**77.12 (71.8-83.0)**	**0.00 (0.0-0.9)**	22.36 (16.6-27.7)	**0.00**
	RL-Oracle†	3.58 (1.9-5.4)	78.78 (73.9-84.9)	0.00 (0.0-0.0)	21.22 (15.1-26.1)	0.01

Detecting Latent Meals

Our approach achieves strong blood glucose control without meal announcements, but how much of this is due to the ability to react to meals? To investigate this, we looked at the total proportion of insulin delivered on average after meals for PID and RL-Scratch, shown in **Figure 6.4a**. A method able to infer meals should use insulin rapidly after meals, as the sooner insulin is administered the faster glycemic spikes can be controlled while avoiding hypoglycemia. We observe that RL-Scratch administers the majority of its post-meal bolus within one hour of a meal, whereas PID requires over 90 minutes, suggesting RL-Scratch can indeed better infer meals. Interestingly, when considering the percentage of total daily insulin administered in the hour after meals, RL-Scratch responds even more aggressively than BB or PID-MA (54.7% *vs.* 48.5% and 47.3% respectively). This demonstrates that our RL approach is able to readily react to latent meals shortly after they have occurred.

Ability to Adapt to Behavioral Schedules

We hypothesize that one advantage of RL is its ability to compensate for predictable variations (such as meals) in the environment, improving control as the environment becomes more predictable. To test this benefit, we changed the meal schedule generation procedure outlined in **Algorithm 7** (**Appendix B.1**) for Adult 1. We removed the small 'snack' meals, set all meal occurrence probabilities to 1, and made meal amounts constant (*i.e.*, each day Adult 1 consumes an identical set of meals). We then evaluated both the PID model and RL-Scratch on 3 variations of this environment, characterized by the standard deviation of the mealtimes (either 0.1, 1, or 10 hours). This tests the ability of each method to take advantage of patterns in the environment. As the standard deviation decreases, the task becomes easier for two reasons: i) there are fewer instances where two meals occur in quick succession, and ii) the meals become more predictable. The results are presented in **Figure 6.4b**. We observe that both methods improve in performance as the standard deviation decreases, likely due to (i). However, while RL-Scratch outperforms PID under all settings, the difference increases as the standard deviation of mealtimes decreases, suggesting RL is better able to leverage the predictability of meals. Specifically, mean risk decreases by roughly 12% for the PID approach (form 9.65 to 8.54), whereas it

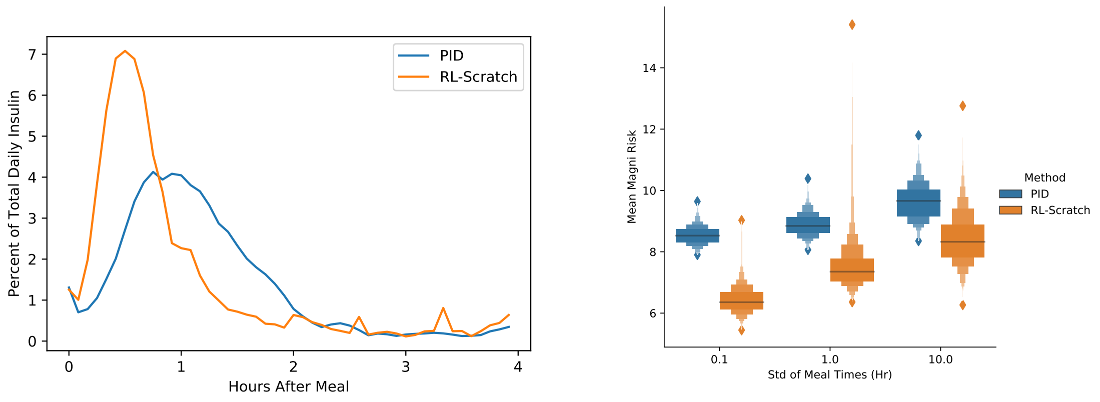


FIGURE 6.4: a) The average amount of insulin (in percent of total daily insulin) provided after meals for PID and RL-Scratch (note: RL-Trans, unshown, is very similar to RL-Scratch). RL-Scratch is able respond to meals more quickly than PID, with insulin peaking 30 minutes post-meal as opposed to roughly 60 minutes for PID. Additionally, the RL approach finishes delivering most post-meal insulin after 1hr, PID takes over 90 minutes. b) The distribution of average risk scores over 300 10-day rollouts for Adult 1 using meal schedules with varying amounts of predictability (mealtime standard deviation). While PID performs better with more regularly spaced meals (median risk lowers from 9.66 at std=10 to 8.53 at std=0.1, a 12% decrease), RL-Scratch sees a larger proportional and absolute improvement (median risk lowers from 8.33 at std=10 to 6.36 at std=0.1, a 24% decrease).

decreases nearly 24% for the RL approach (from 8.40 to 6.42). This supports our hypothesis that RL is better able to take advantage of predictable meal schedules.

6.3.5 Challenges for Deep RL

While in the previous section we demonstrated several advantages of using deep RL for blood glucose control, here we emphasize that the application of deep RL to this task and its evaluation are non-trivial. Specifically, in this section we:

- demonstrate the importance of our action space formulation for performance (**Section 6.3.5 - Developing an Effective Action Space**),
- illustrate the critical need for careful and extensive validation, both for model selection and evaluation (**Section 6.3.5 - Potential Pitfalls in Evaluation**).

- show that, applied naively, deep RL leads to an unacceptable catastrophic failure rate, and present three simple approaches to improve this (**Section 6.3.5 - Reducing Catastrophic Failures**),
- address the issue of sample complexity with simple policy transfer (**Section 6.3.5 - Sample Efficiency and Policy Transfer**).

Developing an Effective Action Space

One challenging element of blood glucose control in an RL setting is defining the action space. Insulin requirements vary significantly by person (from 16 to 60 daily units in the simulator population we use), and throughout most of the day, insulin requirements are much lower than after meals. To account for these challenges, we used a patient-specific action space, where much of the space corresponds to delivering no insulin (discussed in **Section 6.3.1 - Recurrent Architecture**). We perform an ablation study to test the impact of these decisions. On an arbitrarily chosen patient (child#001), we shifted the *tanh* output to remove the negative insulin space. This increased the catastrophic failure rate from 0% (on this patient) to 6.6%. On a challenging subset of 4 patients (indicated in **Appendix B.2**), we looked at the effect of removing the patient-specific action scaling ω_b. This resulted in a 13% increase in median risk from 9.92 to 11.23. These results demonstrate that a patient-specific action space that encourages conservative behavior can improve performance.

Potential Pitfalls in Evaluation

In our experiments, we observed two key points for model evaluation: i) while often overlooked in RL, using validation data for model selection during training was key to achieving good performance, and ii) without evaluating on far more data than is typical, it was easy to underestimate the catastrophic failure rate.

Performance instability necessitates careful model selection. Off-policy RL with function approximation, particularly deep neural networks, is known to be unstable [52], [162]. As a result, we found it was extremely important to be careful in selecting which network (and therefore policy) to evaluate. In **Figure 6.5a**, we show the fluctuation in validation performance over training for Adult#009. While performance increases on average over the course of training (at least initially), there are several points where performance degrades considerably. **Figure 6.5b** shows how performance averaged over all patients changes depending on the approach used to select the policy for evaluation. When we simply evaluate using the final epoch, almost half of test rollouts end in a catastrophic failure. Surprisingly, even when we select the model that minimized risk on the validation set, nearly a fifth of rollouts fail. However, by first limiting our pool of models to those that achieve a minimum blood glucose level of at least 30 mg/dL over the validation data, we reduce the catastrophic failure rate to 0.07%. As performance instability has been noted in other RL domains [136], [163], this observation is likely relevant to other applications of deep RL in healthcare.

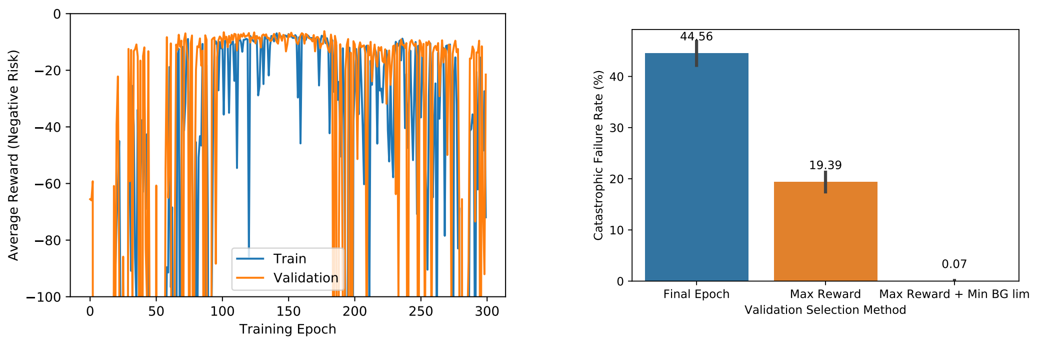


FIGURE 6.5: a) The training and validation curve (average reward) for adult#009. Note the periods of instability affect both training and validation performance. b) Catastrophic failure rate over all patients for 3 methods of model selection: i) selecting the final training epoch, ii) selecting the epoch that achieved minimal risk, and iii) selecting the minimal risk epoch that maintained blood glucose above 30 mg/dL. We see large differences in performance depending on the model selection strategy.

Extensive evaluation is necessary. Health applications are often safety critical. Thus, the susceptibility of deep RL to unexpected or unsafe behavior can pose a significant risk [164], [165]. While ongoing work seeks to provide safety guarantees in control applications using deep learning [165], [166], it is important that practitioners take every step to evaluate the safety of their approaches. While it is typical to evaluate RL algorithms on a small number of rollouts [136], in our work we saw how easy it can be to miss unsafe behavior, even with significant testing. We examined the number of catastrophic failures that occurred using RL-Trans using different evaluation sets. Over our full evaluation set of 9,000 10-day rollouts, we observed 20 catastrophic failures (a rate of 0.22%). However, when we only evaluated using the first 5 test seeds, which is still over 12 years of data, we observed 0 catastrophic failures. Additionally, when we evaluated using 3-day test rollouts instead of 10, we only observed only 5 catastrophic failures (a rate of .05%), suggesting that longer rollouts result in a higher catastrophic failure rate. These results demonstrate that, particularly when dealing with noisy observations, it is critical to evaluate potential policies using many different, lengthy rollouts.

Reducing Catastrophic Failures

Due to their potential danger, avoiding catastrophic failures was a significant goal of this work. The most direct approach we used was to modify the reward function, using a large termination penalty to discourage dangerous behavior. While unnecessary for fine-tuning policies, when training from scratch this technique was crucial. On a subset of 6 patients (see **Appendix B.4**), including the termination penalty reduced the catastrophic failure rate from 4.2% to 0.2%.

We found varying the training data also had a major impact on the catastrophic failure rate. Every time we reset the environment during training, we used a different random seed (which controls meal schedule and sensor noise). Note that the pool of training, validation, and test seeds were non-overlapping. On a challenging subset of 7 patients (described in **Appendix B.4**), we ran RL-Scratch with and without this strategy. The run that varied the training seeds had a catastrophic failure rate of 0.15%, the run that didn't had a 2.05% rate (largely driven by adult#009, who reliably had the highest catastrophic failure rate across experiments).

Other approaches can further improve stability. In **Table 6.1**, our RL results are averaged over three random restarts in training. This was done to demonstrate that our learning framework is robust to randomness in training data and model initialization. However, in a real application it would make more sense to select (using validation data)

TABLE 6.2: Risk and percent of time Eu/Hypo/Hyperglycemic calculated for the RL approaches treating the 3 training seeds as different random restarts. The stability of the Scratch and Trans approaches improves relative to performance in Table 6.1.

	Risk $\downarrow$	Euglycemia (%) $\uparrow$	Hypoglycemia (%) $\downarrow$	Hyperglycemia (%) $\downarrow$	Failure Rate (%) $\downarrow$
RL-Scratch	6.39 (4.7-8.9)	72.96 (69.1-76.6)	0.62 (0.0-1.7)	25.96 (22.7-29.6)	0.00
RL-Trans	6.57 (5.0-9.3)	71.56 (67.0-75.7)	0.80 (0.0-1.9)	27.19 (23.4-31.2)	0.13
RL-MA	3.71 (2.7-6.3)	77.36 (72.7-83.2)	0.00 (0.0-0.5)	22.45 (16.7-26.9)	0.00

one model for use out of the random restarts. We apply this approach in **Table 6.2**, choosing the seed that obtained the best performance according to our model selection criteria. This improves all the RL methods. Most notably, it further reduces the catastrophic failure rate for the approaches without meal announcements (0.07% to 0% for RL-Scratch, and 0.22% to 0.13% for RL-Trans).

Sample Efficiency and Policy Transfer

While RL-Scratch achieves strong performance on average, it requires a large amount of patient-specific data: 16.5 years per patient. While RL-Trans reduced this amount, it still required over 2 years of patient-specific data, which for most health applications would be infeasible. Thus, we investigated how performance degrades as less data is used.

In **Figure 6.6**, we show the average policy performance by epoch for both RL-Scratch and RL-Trans relative to the PID controller. Note the epoch determines the maximum possible epoch for our model selection, not the actual chosen epoch. We see that far less training is required to achieve good performance with RL-Trans. In over 40% of rollouts, RL-Trans outperforms PID with no patient-specific data (median risk 10.31), and with 10 epochs of training (roughly half a year of data) RL-Trans outperforms PID in the majority of rollouts (59.6%; median risk 7.95).

Interestingly, the lack of patient-specific data does not appear to cause excessive catastrophic failures. With no patient-specific data the failure rate is 0%, after 5 epochs of training it has risen to .5%, and then declines over training to the final value of .22%. This implies two things: i) patient-specific training can increase the catastrophic failure rate,

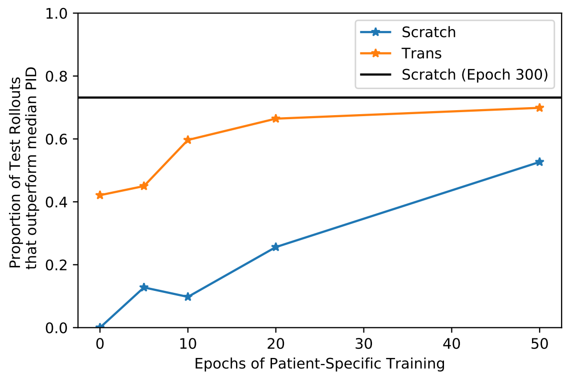


FIGURE 6.6: The proportion of test rollouts where RL-Scratch and RL-Trans outperform the median PID risk with different amounts of patient-specific training. We see that without any patient-specific data RL-Trans performs better than PID in 40% of rollouts. RL-Scratch requires a significant amount of patient-specific data before achieving comparable performance.

possibly by learning overly aggressive treatment policies, and ii) our current model selection procedure does not minimize the catastrophic failure rate. We do not observe this for RL-Scratch, where all epochs under 50 achieve a catastrophic error rate of over 17%. These results suggest that our simple transfer approach can be effective even with limited amounts of patient-specific data.

6.4 SABR: Simulation Augmented Batch Reinforcement Learning

In **Section 6.3.5 - Sample Efficiency and Policy Transfer**, we evaluated a simple approach to reduce the sample complexity of deep RL for blood glucose management. While we observed a large decrease in the amount of training data required to reliably beat the PID controller (from 1000 days to less than 200 to beat the PID $\geq$ 50%), even the reduced amount of training data presents a serious safety issue. Any amount of training that requires our policy to actively explore or interact with its environment could lead to catastrophic failures during training.

One solution to the problem of safety during training is to attempt to learn using observational data. High-quality observational data can be easily obtained by logging the manual glucose management strategies of people with diabetes. Learning policies from a batch of observational data collected by a behavior policy, *i.e.,* **batch RL** (or offline RL), has been explored in several recent works [55], [56]. The fundamental problem in this setting is constraining learned policies to avoid exploiting inadequate state/action coverage in the batch while maintaining good performance. *E.g.,* the goal is to learn a policy such that, if the behavior policy does not sufficiently explore part of the state space to develop accurate value estimates, the learned policy will avoid those states when possible. Standard off-policy methods such as soft actor-critic, while theoretically capable of learning in such settings, fail in practice [54]. Such techniques fail even when the behavior policy contains highly diverse data generated from a similar policy class (*i.e.*: a batch of data generated by training an identical soft actor critic on the same task) [54].

To address this issue, recent work has focused on using various distance metrics to constrain a learned policy from deviating significantly from the behavioral policy [54], [56]. In this work, we examine supplementing these approaches assuming access to an imperfect simulator.

Motivation. While approaches in batch RL have demonstrated more stable performance relative to off-policy learning in the batch setting, they can still struggle to achieve performance that surpasses the behavior policy, let alone the performance of online training [55]. Meanwhile, while blood glucose simulators are known to imperfectly model real-world phenomenon, they are sufficiently realistic to qualify for FDA approval as a replacement for animal trials in closed loop control studies [167]. It seems sensible then to assume that the knowledge encoded in simulators could help correct some of the flaws in policies learned using batch RL.

Our Approach. Naively, one might train policies entirely on the simulator. However, we assume the simulator is imperfect, and the behavior batch contains information from the true environment. We propose an approach that combines the imperfect simulator with the batch data. We draw inspiration from past work in batch RL that limits divergence from a behavior policy and extend this notion to a simulator by initializing our simulator using a batch and limiting simulated rollout lengths. We propose simulation-augmented batch RL, or SABR, a simple and general technique to augment batch RL algorithms with a simulator.

6.4.1 Batch RL Setup

We assume access to a batch of data $\mathcal{D}_B$ collected using a behavior policy, that we call the "behavior batch". At training time, this behavior batch is used in place of the experience replay buffer used in traditional off-policy learning approaches. SABR makes use of a second, regularly updated batch of data $\mathcal{D}_R$ used as the replay buffer, that we call the "replay batch". We also assumes access to a simulator $\hat{P}$ that approximates the transition function discussed in **Section 6.3.1 - Problem Setup**.

6.4.2 Proposed Algorithm

Given this setup, at each training iteration, a data sample is drawn from the behavior batch $(s_i, a_i, r_i, s_{i+1})\} \sim \mathcal{D}_B$, then s_i is used to initialize the state of the simulator and a short trajectory is rolled out: $s_{i+k} \sim \hat{P}(\cdot|s_{i+k-1}, \pi_\phi(s_{i+k-1})$ for $k = 1$ to the rollout length h (typically 10 in our experiments). The rolled out trajectory $\{(s_{i+k}, a_{i+k}, r_{i+k}, s_{i+k+1})\}_{k=0}^{h-1}$ is then appended to $\mathcal{D}_R$. The full algorithm is given in **Algorithm 6**.

Note that SABR can be used with any RL or batch RL algorithm that uses an experience replay buffer. For approaches, such as BCQ and BEAR, that involve training a behavior policy, we still train using the behavior batch. The limited length of the rollouts helps prevent the dynamics of the simulator from washing out information contained in the behavior batch, and the use of the batch to initialize the environment allows our experience replay buffer to contain diverse trajectories, despite the short rollout lengths.

6.4.3 Experimental Setup

To test the efficacy of SABR (while allowing ground truth evaluation), we created a batch RL problem using the UVA/Padova simulator used for our previous experiments. To

ALGORITHM 6: Simulation Augmented Batch RL

```
Data: Behavior Batch $\mathcal{D}_B$, Simulator $\hat{P}$
Result: Trained policy $\pi_\phi$
Initialize policy $\pi_\phi$ and empty experience replay buffer $\mathcal{D}_R$;
while Not trained do
    Draw observation $(s_i, a_i, r_i, s_{i+1})\} \sim \mathcal{D}_B$;
    for $k = 1 \dots h$ do
        $a_{i+k-1} \sim \pi_\phi(\cdot|s_{i+k-1})$;
        $s_{i+k} \sim \hat{P}(\cdot|s_{i+k-1}, a_{i+k})$;
        $r_{i+k-1} = R(s_{i+k-1}, a_{i+k-1})$;
    end
    Append trajectory $\{(s_{i+k}, a_{i+k}, r_{i+k}, s_{i+k+1})\}_{k=0}^{h-1}$ to $\mathcal{D}_R$;
    if Training behavior policy then
        Perform iteration of behavior cloning using $\mathcal{D}_B$;
    Perform iteration of training $\pi_\phi$ using standard RL or Batch RL algorithm with
      buffer $\mathcal{D}_R$;
end
```

initialize the simulator, we use the oracle state space described in **Section 6.3.1 - Oracle Architecture**. We also consider applying SABR to our normal state space by using a separate round of state inference, trained using our imperfect simulator.

We collect a batch of data from Adult#001 using a basal bolus control policy to simulate human behavior. The batch is collected using independent 20-day rollouts with varying random seeds for meals and sensor noise. We then use this batch of data together with a copy of the simulator for adult#004 to train control policies for adult#001. We use the same learning and validation procedure described in **Section 6.3.1**. We measure performance using mean risk and compare the effect of using different batch sizes. We evaluate using batches containing 1 month, 2 months, or 2 years of observational data.

We evaluate the following approaches:

- **Simulation-Only**: The policy is learned using the imperfect simulator in a standard online fashion (using soft actor-critic). This approach serves as a baseline, testing the importance of considering batch data.
- **Behavior Cloning (BC)**: We use the variational autoencoder behavioral cloning approach of [54]. This approach measures the efficacy of the behavioral policy we can

learn from the batch.

- **Batch Constrained Q Learning (BCQ)** [54]: This approach constrains the learned policy by learning to modify a cloned behavior policy using a perturbation network thresholded on the L_∞-norm measured in the action space of the policies. The threshold is set proportional to the maximum absolute action. We use the original continuous formulation from the authors implementation with a perturbation threshold of 5%.
- **BEAR** [56]: This approach constrains the learned policy to lie within the support of the behavior policy using an MMD loss. We use the authors implementation and tune hyperparameters (specifically the MMD variance term and lagrange threshold parameter) for our task. This method, along with BCQ, test the efficacy of state-of-the-art batch RL on the new problem of blood glucose management.
- **SABR-BCQ/BEAR**: We augment SAC/BCQ/BEAR respectively using our proposed approach outlined in **Algorithm 6**.
- **Online Training (Oracle)**: This serves as a form of oracle performance. We train our RL approach using the simulator for the true patient, adult#001.

6.5 Experiments and Results

In our experiments, we seek to answer the following questions:

- Do standard batch RL approaches learn effective glucose control policies? (**Table 6.3**)
- Can imperfect simulators be used to augment the performance of these approaches when ignoring issues of state inference? (**Table 6.3**)
- Can these approaches work even in the presence of biased state inference? (**Table 6.4**)

Our results using the oracle state space are presented in **Table 6.3**. We observe that the simulation augmented approaches, SABR-BCQ and SABR-BEAR, improve over vanilla BCQ/BEAR, reducing risk on a 2-month batch from 7.94 with BEAR to 5.52. Interestingly, we observe a consistent decrease in performance from all methods except BC when increasing the amount of batch data. Thus, in our next experiment we considered further reducing the batch data to 1 month.

TABLE 6.3: Median risk and interquartile range over 100 10-day rollouts and 3 training seeds, using the ground truth simulator state. We evaluate methods trained with either 2 months of batch data or 2 years (note this does not affect the simulation-only methods). We observe the SABR approaches, particularly SABR-BEAR, perform better than the baselines.

Method	Risk: 2 Month Batch	Risk: 2 Year Batch
Sim-Only	6.82 (6.68, 6.96)	6.82 (6.68, 6.96)
BC [54]	9.12 (8.68, 9.80)	7.65 (7.06, 7.78)
BCQ [54]	95.42 (95.23, 96.32)	76.57 (76.12, 77.83)
BEAR [56]	7.94 (7.22, 8.60)	9.39 (8.63, 10.15)
SABR-BCQ (Ours)	6.54 (6.17, 6.95)	9.13 (6.39, 9.78)
SABR-BEAR (Ours)	**5.52**(**5.29**, **5.89**)	**5.94**(**5.63**, **6.22**)
Oracle*	3.58 (1.88, 5.37)	3.58 (1.88, 5.37)

Evaluated on our observed state space, consisting of the past four hours of carbs and insulin, the task becomes considerably more difficult. Using simulator augmentation requires a separate state inference step. Given the observed data, we predict the true simulator state. We train this approach in a supervised fashion, using data collected from adult#004. We describe our process for doing this more fully in **Appendix B.6**. The results are presented in **Table 6.4**. The additional state inference step has a negative impact on performance. Neither SABR-BCQ nor SABR-BEAR consistently outperform their vanilla counterparts. We continue to observe general increases in performance when reducing the quantity of batch data, contrary to our expectations.

While our results using simulator augmentation were mixed, helping most in a setting where state inference was not required, this work does suggest that batch RL can be reasonably applied to the problem of blood glucose control. In the results presented in **Table 6.3** and **Table 6.4** the oracle risk performance was within 2 points or less of the next best approach. This is encouraging as the batch RL approaches require no interaction with the environment and is thus safer than an online approach.

Given the difference in results between **Table 6.3** and **Table 6.4**, additional work in state inference may further reduce the gap between batch RL and interactive training.

TABLE 6.4: Median risk and interquartile range using our standard state space. We observe that SABR-BEAR outperforms all baselines in the low data setting, but with higher quantities of data underperforms BEAR. Curiously, large behavior batches reduce overall performance across most methods.

Method	Risk: 1 Month Batch	Risk: 2 Month Batch	Risk: 2 Year Batch
BC [54]	21.99 (21.03, 22.90)	17.32 (16.32, 18.17)	20.31 (19.32, 21.14)
BCQ [54]	16.71 (13.41, 22.71)	18.33 (13.48, 26.59)	17.33 (15.75, 43.24)
BEAR [56]	8.43 (7.94, 9.02)	**8.78**(**8.43**, **9.19**)	**8.79**(**8.40**, **9.26**)
SABR-BCQ (Ours)	18.08 (12.90, 38.28)	28.97 (17.70, 102.55)	30.73 (7.82, 93.78)
SABR-BEAR (Ours)	**8.31**(**7.76**, **9.07**)	10.49 (8.67, 11.41)	9.93 (9.48, 10.50)
Oracle*	7.83 (7.15, 8.64)	7.83 (7.15, 8.64)	7.83 (7.15, 8.64)

Recent work has demonstrated how the adjoint method allows backpropagation through arbitrary ODE-solvers [168], this could be used with our simulator to optimize state inference using observational data. However, in both cases the best performing non-oracle approach was simulation-augmented BEAR, which is encouraging for the overall efficacy of our proposed simulation augmentation.

6.6 Summary and Conclusions

In this work, we demonstrate how deep RL can lead to significant improvements over alternative approaches to blood glucose control, with and without meal announcements (**Section 6.3.4 - Deep RL *vs.* Baseline Approaches**). We provide insight into why (**Section 6.3.4 - Detecting Latent Meals**) and when (**Section 6.3.4 - Ability to Adapt to Behavioral Schedules**) we would expect to see RL perform better. We demonstrate the importance of a robust action space in patient-specific tasks (**Section 6.3.5 - Developing an Effective Action Space**), show how careful and extensive validation is necessary for realistic evaluation of performance (**Section 6.3.5 - Potential Pitfalls in Evaluation**), investigate several simple and general approaches to improving the stability of deep RL (**Section 6.3.5 - Reducing Catastrophic Failures**), and develop a simple approach to reduce the requirements of patient-specific data (**Section 6.3.5 - Sample Efficiency and Policy Transfer**). To go further, we provide the first application of batch RL to the problem of blood glucose management (**Section 6.4**), and demonstrate how biased simulators could be combined with suboptimal observational data to learn blood glucose management policies. While the main goal of this work is to advance a clinical application, the challenges we encountered and the solutions they inspired are likely to be of interest to researchers applying RL to healthcare more broadly.

While our results in applying deep RL to blood glucose control are encouraging, they come with several limitations. First, our results are based on simulation. The simulator may not adequately capture variation across patients or changes in the glucoregulatory system over time. In particular, the virtual patient population the simulator comes equipped with does not differentiate individuals based on demographic information, such as gender and ethnicity. Thus, the applicability of our proposed techniques to all relevant patient populations cannot be assessed. However, as an FDA-approved substitute for animal trials [26], success in using this simulator is a nontrivial accomplishment. Second, we define a reward function based on risk. Though optimizing this risk function should lead to tight glucose control, it could lead to excess insulin utilization (as its use is unpenalized). Future work could consider resource-aware variants of this reward. Third, our choice of a 4-hour state space discourages learning long-term patterns or trends. In our environment, this did not reduce performance relative to a longer input history, but this could be important for managing blood glucose levels in more realistic simulators or real-world cases [167]. Finally, we emphasize that blood glucose control is a safety-critical application. An incorrect dose of insulin could lead to life-threatening situations. Many of the methods we examined, even those that achieve good average performance, are susceptible to catastrophic failures. We have investigated several ways to minimize such failures, including modifying the reward function and selecting models across multiple random restarts. While the results from **Table 6.2** suggest these approaches mitigate catastrophic failures, the results of **Section 6.3.5 - Potential Pitfalls in Evaluation** show such empirical evaluations can miss failure cases. To enable researchers to better explore and correct these limitations, we evaluated on an open-source simulator and made all the code required to reproduce our experiments publicly available.

Chapter 7

Conclusion

This dissertation addressed how to learn useful representations of physiological time series with a focus on algorithms for monitoring and managing blood glucose levels in people with type 1 diabetes. Physiological data collected under normal living conditions can provide great insight into the progression and management of chronic diseases. However, learning from such data presents several challenges. The fact that the data are collected under normal living conditions means that unobserved daily events and environmental factors can have large effects on the data which are difficult to model. Additionally, when dealing with chronic diseases, outcomes can occur over time horizons spanning years, thus huge amounts of longitudinal data are often available for each labelled example, resulting in a challenging high-D low-N learning problem. Additionally, physiological time series inherit the problems (and advantages) of more general time-series data, requiring approaches that model temporal dependencies. Finally, as with all problems in health, safety and reliability are absolute necessities, introducing serious challenges when learning automated control policies.

This dissertation builds on work spanning several fields, including time-series analysis, machine learning with sequential data, and deep reinforcement learning. Identifying representations of time-series data that can then be used to make predictions (*e.g.*, predict hypoglycemia) has been a longstanding problem. Motif representations, which emphasize repeated patterns in the data, have been used to find efficient and discriminative representations of sequences [43]. However, these techniques can ignore important context in the signal. In recent years, a great deal of work has been done on end-to-end representation learning using deep learning. These approaches can struggle without access to large amounts of labeled data, though work has shown the value of self-supervision on large unlabeled datasets [96]. For predicting future values in time series, approaches in multi-step forecasting have emphasized modeling trajectories in using recursive and multi-output models, which can ignore temporal dependencies [15]. Finally, deep reinforcement learning has made great strides in recent years on several complex domains [153], [169], but past work has not explored how it could be used for closed loop blood glucose management.

Building on this past work, our dissertation has contributed several new approaches for learning useful representations of and controlling physiological time series. For representing signals, **we proposed contextual motifs**, an approach that combines well conserved subsequences with global context to learn discriminative representations (**Chapter 3**). For end-to-end representation learning with sequences, **we showed self-supervision does not require unlabeled data** to improve classification performance, as it can draw on the sequential structure inherent to the input (**Chapter 4**). Relating to both representing and controlling physiological signals, our work on multi-output forecasting demonstrated the importance of modeling dependencies in the output window, and **we showed that incorporating functional priors into output representations improved performance** (**Chapter 5**). While our work in forecasting could serve as a component of a control scheme, we also **used RL to learn policies that manage blood glucose levels**. We observed that stable personalized policies could be learned directly from data, and that these policies could be used without any human intervention, even without any access to the true environment (**Chapter 6**). While we demonstrated that our contributions often lead to quantitative gains in relevant metrics, one unmeasured benefit is the potential to reduce patient burden. Even if automated methods for blood glucose management do not significantly improve performance relative to human control, removing the stress of managing glucose levels is important.

There are several areas touched upon in this dissertation that could be interesting for further examination. Here we outline four possibilities.

First, the ability to incorporate extremely long temporal contexts in sequence prediction is still lacking. Approaches in this dissertation only considered using input lengths of up to one day for prediction problems, which would certainly be insufficient for problems involving long-term disease progression. Learning to connect minute-level input data with trends that emerge over years is not feasible with the approaches we have laid out. While this problem in general may be infeasible, the fact that patterns in blood glucose data are largely present at a daily level suggests that hierarchical representations that track slow changes in daily patterns may be feasible.

Second, there is still a clear divide between methods that use expert definitions and those that seek to learn solely from data. While recent history in ML suggests that the latter is preferable, it would be surprising if existing expert information on physiological processes had no role to play in understanding and managing these systems. In particular, using data to improve existing simulators is an exciting way to combine expert definitions and data. While existing glucose simulators are capable of generating realistic days of glucose traces [167], the ability to easily generate accurate personalized simulators using only observational data would be an exciting development for advancing blood glucose management.

Third, while we observed good performance using deep RL to control blood glucose levels, it is unclear how such models could be used in practice. The fundamental issue with deep RL in safety-critical control problems is the lack of provable guarantees on behavior for general methods. With the recent surge of interest in RL for autonomous driving and robotics, there has been an increase in work done on safety in RL [170]–[172]. However there has been limited work using RL to learn provably safe control policies for physiological time series. Related to this issue is the problem of sample efficiency. Even our transfer-learning approach required 6 months of patient-specific data to achieve good performance. Clinicians can recommend changes to insulin regimes using only a few weeks of data. This suggests that better modeling could lead to improved sample efficiency.

Finally, one major limitation of this dissertation was a lack of evaluation on real people in a prospective manner. While we worked with data collected from real individuals where possible, evaluations were performed retrospectively and thus their applicability to real-world settings cannot be presumed. This problem is particularly notable in **Chapter 6**, in which we only use simulated data. While the simulator we evaluate on is strong, it does not model several important aspects of the glucoregulatory system, such as changes in insulin sensitivity [167]. Before this work could be deployed broadly, it must be carefully evaluated prospectively on individuals under real-world settings. Such trials have been done for existing commercial hybrid closed loop systems, suggesting this is feasible [141]. However, due to the expense of such trials and our finding that extensive evaluation is needed to properly evaluate approaches (**Section 6.3.5**), as an initial step it would be valuable to explore methods to efficiently generate adversarial meal schedules for control algorithms. This would help determine worst case controller performance without requiring prohibitive amounts of real world evaluation.

The main contributions of this dissertation are: 1) approaches to learn useful representations of blood glucose data for predicting adverse events, and 2) methods to learn blood glucose management policies from online or batch data. Important to these contributions is the identification of overarching problems that arise when learning from these data. By emphasizing problems such as latent context and the mismatch of input data size to labels, this dissertation could help to clarify issues in other application areas using data from wearable health monitors. Another important aspect of this dissertation is the emphasis on problems in diabetes management, a societally important and technologically interesting but often overlooked application area for machine learning research. By highlighting applications of machine learning approaches to problems in this field, particularly the closed-loop management of blood glucose levels, we hope this dissertation inspires more interest from the machine learning community in this area.

Appendix A

Appendix for Improving Time-Series Classification Without Additional Data Using Limited Self-Supervision

A.1 Auxiliary Tasks: Additional Motivation and Details

A.1.1 Autoencoding

Motivation. Autoencoding encourages the hidden state to retain all relevant information from the original input. By reducing the size of the hidden state, our model must learn a compressed representation of the signal. Compression encourages learning latent structure and discourages learning noise.

Details. We use a single-layer autoregressive LSTM for the recurrent layer R_{AE}, which is initialized using the hidden and cell states from the encoder LSTM and sequentially decodes hidden states $\mathbf{z}_t^{AE}$. These hidden states are fed into a fully-connected output layer, O_{AE}, that maps $\mathbf{z}_t^{AE} \to \hat{\mathbf{x}}_t$. The output $\hat{\mathbf{x}}_t$ is then fed into R_{AE}, generating $\mathbf{z}_{t+1}^{AE}$, continuing until $\hat{\mathbf{x}}_{0:T}$ is fully generated. To provide a shorter path for gradient flow, we decode in reverse order, generating $\hat{\mathbf{x}}_{T:0}$ instead of $\hat{\mathbf{x}}_{0:T}$ [44].

A.1.2 Forecasting

Motivation. Forecasting encourages E to encode the dynamics of the data-generating process. Without information about the underlying dynamics, future value prediction is either challenging or trivial (if the signal does not change). Such dynamics may carry valuable information for a range of sequence-level tasks, though the aspects of the dynamics most relevant to the target task may differ from those that predict future values.

Details. We focus on a multi-output forecasting architecture, predicting several future values simultaneously [173]. This is done using a recurrent decoder, similar to the autoencoder, but applied at each encoding step and expanded only h steps, where h is a hyperparameter.

A.1.3 Partial-Signal Autoencoding

Motivation. Previous work has found advantages to reconstruction over prediction [96]. Dai and Le hypothesize the observed superiority of sequence autoencoding over forecasting may result from the short-term nature of the language modeling task (only predicting the next word). To investigate the effect of short-term dependency auxiliary tasks, we use a multi-step forecasting system, where we can examine the effect of varying the prediction horizon. Analogously, we use PS-AE to examine the effect of producing short-term reconstructions.

Details. PS-AE is implemented using a setup identical to our forecasting approach, except we estimate the *previous* h values instead of the *subsequent* values (**Figure 4.1 a**, D_{PS}).

A.1.4 Piecewise-Linear Autoencoding

Motivation. AE encourages fine-grained modeling of the signal over long periods, whereas PS-AE encourages fine-grained modeling over a short period. While PS-AE can control the range of temporal dependencies modeled using the decoding horizon h, it does not vary the granularity at which the output is modeled. What level of signal granularity is required for reasonable target-task performance? To explore this question, we introduce Piecewise-Linear (PL) Autoencoding.

Details. To generate a PL representation of a signal with n distinct pieces, we take the encoded representation of the signal, $\mathbf{z}_T$, and feed it into a recurrent layer R_{PL}. **Figure 4.1 a**, D_{PL} illustrates our point-generation system. The sequential output of the decoder, $\mathbf{z}_j^{PL}$, generates the j^{th} point value and position. Specifically, we use two fully-connected output layers, O_{val} and O_{pos}, that map $\mathbf{z}_j^{PL} \rightarrow v_j, p_j$ respectively. Since we know $p_0 = 0$, at the first step we generate only an initial value. We continue the decoding process, feeding $\mathbf{v}_j, \sum_{k=0}^{j} \mathbf{p}_k$ to the decoder to generate $\mathbf{z}_{j+1}^{PL}$, until we have generated $n+1$ points. We then use linear interpolation to map $(\mathbf{v}_0, \mathbf{p}_0) \ldots (\mathbf{v}_n, \mathbf{p}_n) \rightarrow \hat{\mathbf{x}}_{0:T}$. We normalize the position vector and use the cumulative summation to determine segment positions.

A.2 All Combinations of Auxiliary Tasks

In Figure 5 we present plots showing average AUC-ROC for all possible combinations of auxiliary tasks, for each data set. Here the contribution of each task can be examined.

A.3 Application to UCR Times Series data-sets

In order to evaluate the use of self-supervised auxiliary tasks in an easily replicable setting, and in order to facilitate comparisons with state-of-the are methods, we performed classification on 7 data sets from the UCR Time Series Archive `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`. We included tasks that had fixed length sequences and a sample to label ratio greater than 100. We only included the subset of these tasks for which our baseline architecture performed better than random chance. These selections were to ensure that the tasks were reasonably well suited to our baseline architecture. For each dataset, the last 20% of the training sample was used as a validation set, with the testing sample used as a test set.

Table A.1 shows baseline results for each task (accuracy when our model was run with no auxiliary tasks), the highest accuracy achieved with any combination of auxiliary

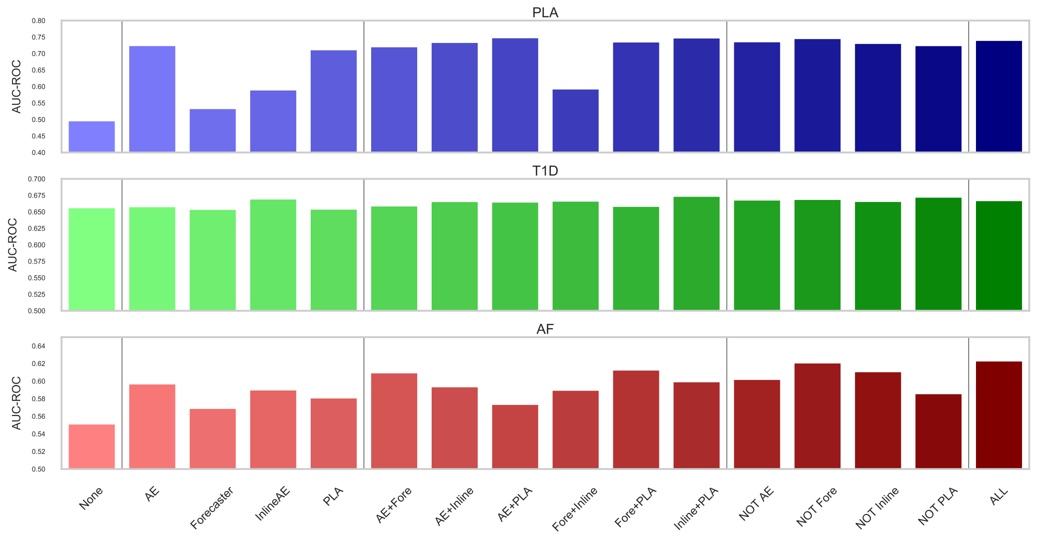


FIGURE A.1: Average AUC-ROC for every combination of Auxiliary task for all three analyses. Here, columns marked 'NOT' included all but the indicated auxiliary task.

tasks, and state-of-the-art performance according to `timeseriesclassification.com`. Although the baseline architecture is routinely and drastically outperformed by state-of-the-art methods, the addition of auxiliary tasks improved performance to be significantly closer to the state-of-the-art level in most cases, and our model achieved at- state-of-the-art accuracy for the Two Patterns task. Although our model was unable to achieve state-of-the-art performance on most tasks, the improvement over baseline indicates that the addition of self-supervised auxiliary tasks could enhance better suited baseline architectures on these and other tasks.

TABLE A.1: Accuracy on 7 data sets from the UCR Archive.

Data Set	Baseline-ACC	Best-ACC	Best-Tasks	SOTA- ACC
ElectricDevices	57.5	59.6	Forecaster	89.5
FordB	60.3	75.4	Inline + PLA	92.9
FordA	61.4	86.6	Not Forecaster	96.5
Wafer	89.1	95.2	AE	100.0
HandOutlines	54.7	54.7	None	92.4
TwoPatterns	64.4	100	Forecaster+Inline	100
StarLightCurves	84.6	89.0	Not PLA	98.0

Appendix B

Appendix for Deep Reinforcement Learning for Blood Glucose Management

B.1 Harrison-Benedict Meal Generation Algorithm

See **Algorithm 7**. In short, this meal schedule calculates BMR for each simulated individual using the Harrison-Benedict equation [174]. This is used to estimate expected daily carbohydrate consumption, assuming individuals eat a reasonably low-carb diet where 45% of calories come from carbohydrates. Daily carbohydrates were divided between 6 potential meals: breakfast, lunch, dinner, and 3 snacks. The occurrence probability of the meal and expected size of the meal was set such that the expected number of carbs eaten per day matched the BMR-derived estimate. Note, in our experiments our implementation of of this meal schedule incorrectly estimated ages for many patients, particularly adults. This effect was fairly minor, resulting in no more than a 10% change in estimated BMR.

ALGORITHM 7: Generate Meal Schedule

```
Data: body weight $w$, age $a$, height $h$, number of days $n$
  $BMR = 66.5 + (13.75 * w) + (5.003 * h) - (6.755 * a)$ ;
  $ExpectedCarbs = (BMR * 0.45)/4$ ▷ 45% of calories assumed from carbs, 4 calories per carb;
  $MealOcc = [0.95, 0.3, 0.95, 0.3, 0.95, 0.3]$;
  $TimeLowerBounds = [5, 9, 10, 14, 16, 20] * 12$;
  $TimeUpperBounds = [9, 10, 14, 16, 20, 23] * 12$;
  $TimeMean = [7, 9.5, 12, 15, 18, 21.5] * 12$;
  $TimeStd = [1, .5, 1, .5, 1, .5] * 12$;
  $AmountMean = [0.250, 0.035, 0.295, 0.035, 0.352, 0.035] * ExpectedCarbs * 1.2$;
  $AmountStd = AmountMean * 0.15$;
  $Days = []$;
  for $i \in [1, \ldots, n]$ do
    $M = [0]_{j=1}^{288}$;
    for $j \in [1, \ldots, 6]$ do
      $m \sim Binomial(MealOcc[j])$;
      $lb = TimeLowerBounds[j]$;
      $ub = TimeUpperBounds[j]$;
      $\mu_t = TimeMean[j]$;
      $\sigma_t = TimeStd[j]$;
      $\mu_a = AmountMean[j]$;
      $\sigma_a = AmountStd[j]$;
      if $m$ then
        $t \sim Round(TruncNormal(\mu_t, \sigma_t, lb, ub))$;
        $c \sim Round(max(0, Normal(\mu_a, \sigma_a)))$;
        $M[t] = c$;
      end
    end
    $Days.append(M)$;
  end
```

B.2 BB Parameters

The parameters are presented in **Table B.1**. The simulator provides age and TDI for each individual. To calculate CR and CF we use equations $CR = 500/TDI$ and $CF = 1800/TDI$, which were provided to us via a clinical consultation. The resulting CR and CF we used differed from those provided in the baseline Simglucose download from [29], in practice we found our values led to better performance.

TABLE B.1: Basal-Bolus Parameters

Person	CR	CF	Age	TDI
child#001	28.62	103.02	9	17.47
child#002	27.51	99.02	9	18.18
child#003	31.21	112.35	8	16.02
child#004	25.23	90.84	12	19.82
child#005	12.21	43.97	10	40.93
child#006	24.72	89.00	8	20.22
child#007	13.81	49.71	9	36.21
child#008	23.26	83.74	10	21.49
child#009	28.75	103.48	7	17.39
child#010	24.21	87.16	12	20.65
adolescent#001	13.61	49.00	18	36.73
adolescent#002	8.06	29.02	19	62.03
adolescent#003	20.62	74.25	15	24.24
adolescent#004	14.18	51.06	17	35.25
adolescent#005	14.70	52.93	16	34.00
adolescent#006	10.08	36.30	14	49.58
adolescent#007	11.46	41.25	16	43.64
adolescent#008	7.89	28.40	14	63.39
adolescent#009	20.77	74.76	19	24.08
adolescent#010	15.07	54.26	17	33.17
adult#001	9.92	35.70	61	50.42
adult#002	8.64	31.10	65	57.87
adult#003	8.86	31.90	27	56.43
adult#004	14.79	53.24	66	33.81
adult#005	7.32	26.35	52	68.32
adult#006	8.14	29.32	26	61.39
adult#007	11.90	42.85	35	42.01
adult#008	11.69	42.08	48	42.78
adult#009	7.44	26.78	68	67.21
adult#010	7.76	27.93	68	64.45

B.3 PID and PID-MA parameters

TABLE B.2: PID parameters

	k_p	k_i	k_d
child#001	-3.49E-05	-1.00E-07	-1.00E-03
child#002	-3.98E-05	-2.87E-08	-3.98E-03
child#003	-6.31E-05	-1.74E-08	-1.00E-03
child#004	-6.31E-05	-1.00E-07	-1.00E-03
child#005	-1.00E-04	-2.87E-08	-6.31E-03
child#006	-3.49E-05	-1.00E-07	-1.00E-03
child#007	-3.98E-05	-6.07E-08	-2.51E-03
child#008	-3.49E-05	-3.68E-08	-1.00E-03
child#009	-3.49E-05	-1.00E-07	-1.00E-03
child#010	-4.54E-06	-3.68E-08	-2.51E-03
adolescent#001	-1.74E-04	-1.00E-07	-1.00E-02
adolescent#002	-1.00E-04	-1.00E-07	-6.31E-03
adolescent#003	-1.00E-04	-1.00E-07	-3.98E-03
adolescent#004	-1.00E-04	-1.00E-07	-4.79E-03
adolescent#005	-6.31E-05	-1.00E-07	-6.31E-03
adolescent#006	-4.54E-10	-1.58E-11	-1.00E-02
adolescent#007	-1.07E-07	-6.07E-08	-6.31E-03
adolescent#008	-4.54E-10	-4.54E-12	-1.00E-02
adolescent#009	-6.31E-05	-1.00E-07	-3.98E-03
adolescent#010	-4.54E-10	-4.54E-12	-1.00E-02
adult#001	-1.58E-04	-1.00E-07	-1.00E-02
adult#002	-3.98E-04	-1.00E-07	-1.00E-02
adult#003	-4.54E-10	-1.00E-07	-1.00E-02
adult#004	-1.00E-04	-1.00E-07	-3.98E-03
adult#005	-3.02E-04	-1.00E-07	-1.00E-02
adult#006	-2.51E-04	-2.51E-07	-1.00E-02
adult#007	-1.22E-04	-3.49E-07	-2.87E-03
adult#008	-1.00E-04	-1.00E-07	-1.00E-02
adult#009	-1.00E-04	-1.00E-07	-1.00E-02
adult#010	-1.00E-04	-1.00E-07	-1.00E-02

TABLE B.3: PID-MA parameters

	k_p	k_i	k_d
child#001	-5.53E-09	-1.00E-07	-3.49E-04
child#002	-1.00E-04	-2.87E-08	-1.00E-03
child#003	-1.00E-05	-2.87E-08	-1.00E-03
child#004	-6.31E-05	-1.00E-07	-1.00E-03
child#005	-1.00E-04	-1.00E-07	-3.31E-03
child#006	-1.00E-05	-3.68E-08	-1.00E-03
child#007	-2.35E-07	-1.00E-07	-1.00E-03
child#008	-4.72E-06	-2.87E-08	-1.00E-03
child#009	-1.00E-05	-1.00E-07	-3.49E-04
child#010	-3.49E-05	-4.72E-08	-1.00E-03
adolescent#001	-1.00E-04	-4.72E-08	-6.31E-03
adolescent#002	-1.00E-05	-1.00E-07	-3.49E-03
adolescent#003	-6.31E-05	-1.00E-07	-2.09E-03
adolescent#004	-6.31E-05	-1.00E-07	-2.51E-03
adolescent#005	-4.79E-05	-1.00E-07	-3.98E-03
adolescent#006	-1.00E-04	-1.00E-07	-2.75E-03
adolescent#007	-1.00E-05	-1.00E-07	-3.02E-03
adolescent#008	-1.58E-09	-1.00E-07	-2.75E-03
adolescent#009	-3.98E-05	-1.00E-07	-1.91E-03
adolescent#010	-1.00E-04	-1.00E-07	-4.37E-03
adult#001	-1.07E-07	-1.00E-07	-4.37E-03
adult#002	-1.58E-04	-1.00E-07	-4.37E-03
adult#003	-7.59E-05	-1.00E-07	-2.51E-03
adult#004	-1.00E-04	-1.00E-07	-1.00E-03
adult#005	-1.07E-07	-1.00E-07	-6.31E-03
adult#006	-1.00E-04	-1.00E-07	-1.00E-02
adult#007	-1.58E-04	-2.51E-07	-3.02E-03
adult#008	-1.58E-05	-1.00E-07	-3.98E-03
adult#009	-4.54E-10	-1.00E-07	-6.31E-03
adult#010	-3.98E-05	-1.00E-07	-4.37E-03

B.4 Relevant Patient Subgroups

For tuning our models and selecting hyperparameters, we focused on one challenging but representative individual from each category. In particular, we used child#001, adolescent#004, and adult#001.

For our action space ablation we examined the subset of 4 individuals who were most prone to catastrophic failures: child#006, child#008, adolescent#002, and adult#009

For the termination penalty experiment, we included child#001, child#003, adolescent#002, adolescent#008, adult#008, and adult#009. We used these patients as they contained a mix of regular and hard to control patients.

For seed progression experiments, we focused on a subset of patients we found reliably challenging for different models to control (in terms of risk and catastrophic failure rate). We included child#001, child#006, child#008, adolescent#002, adolescent#003, adult#001, and adult#009 in this analysis.

B.5 RL-Scratch on Adolescent#002

RL-Scratch has a distinct form of degenerate behavior that occurs only in adolescent#002, drastically lowering performance (see **Figure 6.2**). We reliably observe that models trained on adolescent#002 do not administer *any* insulin, and thus achieve chronic hyperglycemia. This is because, unlike any other virtual patient, adolescent#002 does not require any insulin to achieve blood glucose levels below 1000 mg/dL (the threshold at which we apply the hyperglycemic termination penalty) over a 10 day period. As a result of the large disparity between average rewards and the termination penalty, the network quickly learns to never administer insulin and the learned exploration rate collapses to 0. This approach can be easily avoided by warm-starting using an environment from another individual, as then the network has already learned to administer generally safe amounts of insulin. Adolescent#002 spends, on average, 97% of their time hyperglycemic under RL-Scratch, and only 39.2% of time hyperglycemic under RL-Trans.

B.6 SABR State Inference

We train a random forest regressor implemented in Scikit-Learn to predict the 13-dimensional oracle state given the past hour and future of data. We include the future hour as it improves the performance, particularly when inferring meals. The model is trained using a paired dataset of observed and ground truth states generated using the simulator for adult#004.

Bibliography

[1] "Literature on Wearable Technology for Connected Health: Scoping Review of Research Trends, Advances, and Barriers", vol. 21, e14017, DOI: `10.2196/14017`. [Online]. Available: `https://www.jmir.org/2019/9/e14017/`.

[2] T. Nakamura, K. Kiyono, H. Wendt, P. Abry, and Y. Yamamoto, "Multiscale analysis of intensive longitudinal biomedical signals and its clinical applications", *Proceedings of the IEEE*, vol. 104, no. 2, pp. 242–261, 2016.

[3] B. Tao, M. Pietropaolo, M. Atkinson, D. Schatz, and D. Taylor, "Estimating the Cost of Type 1 Diabetes in the U.S.: A Propensity Score Matching Method", en, *PLOS ONE*, vol. 5, no. 7, e11501, Jul. 2010, ISSN: 1932-6203.

[4] W.-P. You and M. Henneberg, "Type 1 diabetes prevalence increasing globally and regionally: The role of natural selection and life expectancy at birth", *BMJ Open Diabetes Research and Care*, vol. 4, no. 1, e000161, Mar. 1, 2016, ISSN: 2052-4897. DOI: `10.1136/bmjdrc-2015-000161`. [Online]. Available: `https://drc.bmj.com/content/4/1/e000161` (visited on 01/04/2019).

[5] CDC, *National diabetes statistics report: Estimates of diabetes and its burden in the united states. atlanta, ga: Centers for disease control and prevention; 2014*, 2017.

[6] R. M. Bergenstal, W. V. Tamborlane, A. Ahmann, J. B. Buse, G. Dailey, S. N. Davis, C. Joyce, T. Peoples, B. A. Perkins, J. B. Welsh, S. M. Willi, and M. A. Wood, "Effectiveness of Sensor-Augmented Insulin-Pump Therapy in Type 1 Diabetes", *New England Journal of Medicine*, vol. 363, no. 4, pp. 311–320, Jul. 2010, ISSN: 0028-4793. DOI: `10.1056/NEJMoa1002853`. [Online]. Available: `https://doi.org/10.1056/NEJMoa1002853` (visited on 10/29/2018).

[7] E. A. Rogers, K. J. Yost, J. K. Rosedahl, M. Linzer, D. H. Boehm, A. Thakur, S. Poplau, R. T. Anderson, and D. T. Eton, “Validating the patient experience with treatment and self-management (pets), a patient-reported measure of treatment burden, in people with diabetes”, *Patient related outcome measures*, vol. 8, p. 143, 2017.

[8] K. Bohlen, E. Scoville, N. D. Shippee, C. R. May, and V. M. Montori, “Overwhelmed patients: A videographic analysis of how patients with type 2 diabetes and clinicians articulate and address treatment burden during clinical encounters”, *Diabetes care*, vol. 35, no. 1, pp. 47–49, 2012.

[9] B. Chiu, E. Keogh, and S. Lonardi, “Probabilistic discovery of time series motifs”, in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2003, pp. 493–498.

[10] T. L. Bailey, C. Elkan, *et al.*, “Fitting a mixture model by expectation maximization to discover motifs in bipolymers”, 1994.

[11] B. Liu, J. Li, C. Chen, W. Tan, Q. Chen, and M. Zhou, “Efficient motif discovery for large-scale time series in healthcare”, *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 583–590, 2015.

[12] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, “Revisiting unreasonable effectiveness of data in deep learning era”, in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[13] R. Caruana, “Multitask learning”, *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[14] P. Ramachandran, P. J. Liu, and Q. Le, “Unsupervised pretraining for sequence to sequence learning”, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 383–391.

[15] S. B. Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, “A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition”, *Expert systems with applications*, vol. 39, no. 8, pp. 7067–7083, 2012.

[16] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks", in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[17] C. Cobelli, E. Renard, and B. Kovatchev, "Artificial pancreas: Past, present, future", *Diabetes*, vol. 60, no. 11, pp. 2672–2682, 2011.

[18] J. Tuomilehto, "The Emerging Global Epidemic of Type 1 Diabetes", en, *Current Diabetes Reports*, vol. 13, no. 6, pp. 795–804, Dec. 2013, ISSN: 1539-0829. DOI: 10.1007/s11892-013-0433-5. [Online]. Available: https://doi.org/10.1007/s11892-013-0433-5 (visited on 10/29/2018).

[19] K. G. M. M. Alberti, R. A. DeFronzo, and P. Zimmet, Eds., *International textbook of diabetes mellitus*, English, 2nd ed. New York: J. Wiley, 1997, ISBN: 978-0-471-93930-6.

[20] M. E. Kerl, "Diabetic ketoacidosis: Pathophysiology and clinical and laboratory presentation", *Compendium*, vol. 23, no. 3, pp. 220–228, 2001.

[21] J. K. Snell-Bergeon and R. P. Wadwa, "Hypoglycemia, Diabetes, and Cardiovascular Disease", *Diabetes Technology & Therapeutics*, vol. 14, no. S1, S–51, May 2012, ISSN: 1520-9156. DOI: 10.1089/dia.2012.0031. [Online]. Available: https://www.liebertpub.com/doi/abs/10.1089/dia.2012.0031 (visited on 10/29/2018).

[22] Writing Team for the Diabetes Control and Complications Trial/Epidemiology of Diabetes Interventions and Complications Research Group, "Sustained effect of intensive treatment of type 1 diabetes mellitus on development and progression of diabetic nephropathy: The Epidemiology of Diabetes Interventions and Complications (EDIC) study", eng, *JAMA*, vol. 290, no. 16, pp. 2159–2167, Oct. 2003, ISSN: 1538-3598. DOI: 10.1001/jama.290.16.2159.

[23] "Intensive Diabetes Therapy and Glomerular Filtration Rate in Type 1 Diabetes", *New England Journal of Medicine*, vol. 365, no. 25, pp. 2366–2376, Dec. 2011, ISSN: 0028-4793. DOI: 10.1056/NEJMoa1111732. [Online]. Available: https://doi.org/10.1056/NEJMoa1111732 (visited on 05/23/2018).

[24] D. M. Nathan, M. Bayless, P. Cleary, S. Genuth, R. Gubitosi-Klug, J. M. Lachin, G. Lorenzi, B. Zinman, and f. t. D. R. Group, "Diabetes Control and Complications Trial/Epidemiology of Diabetes Interventions and Complications Study at 30 Years: Advances and Contributions", en, *Diabetes*, vol. 62, no. 12, pp. 3976–3986, Dec. 2013, ISSN: 0012-1797, 1939-327X. DOI: `10.2337/db13-1093`. [Online]. Available: `http://diabetes.diabetesjournals.org/content/62/12/3976` (visited on 05/23/2018).

[25] C. Cobelli, G. Federspil, G. Pacini, A. Salvan, and C. Scandellari, "An integrated mathematical model of the dynamics of blood glucose and its hormonal control", *Mathematical Biosciences*, vol. 58, no. 1, pp. 27–60, 1982.

[26] B. P. Kovatchev, M. Breton, C. Dalla Man, and C. Cobelli, *In silico preclinical trials: a proof of concept in closed-loop control of type 1 diabetes*. SAGE Publications Sage CA: Los Angeles, CA, 2009.

[27] B. W. Bequette, "Algorithms for a Closed-Loop Artificial Pancreas: The Case for Model Predictive Control", en, *Journal of Diabetes Science and Technology*, vol. 7, no. 6, pp. 1632–1643, Nov. 2013, ISSN: 1932-2968, 1932-2968. (visited on 11/09/2018).

[28] R. N. Bergman, "Toward physiological understanding of glucose tolerance: Minimal-model approach", *Diabetes*, vol. 38, no. 12, pp. 1512–1527, 1989.

[29] J. Xie, *Simglucose v0.2.1*, Available: https://github.com/jxx123/simglucose, version v0.2.1, 2018. (visited on 01/20/2019).

[30] M. Kumar and M. Thenmozhi, "Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest", Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 876544, Jan. 2006.

[31] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction", in *Advances in Neural Information Processing Systems*, 2016, pp. 64–72.

[32] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, “Video frame synthesis using deep voxel flow”, in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4463–4471.

[33] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio”, in *9th ISCA Speech Synthesis Workshop*, pp. 125–125.

[34] Y. Luo, Y. Xin, R. Joshi, L. Celi, and P. Szolovits, “Predicting icu mortality risk by grouping temporal trends from a multivariate panel of physiologic measurements”, in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[35] J. Wiens, J. Guttag, and E. Horvitz, “Patient risk stratification with time-varying parameters: A multitask learning approach”, *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2797–2819, 2016.

[36] R. Bunescu, N. Struble, C. Marling, J. Shubrook, and F. Schwartz, “Blood Glucose Level Prediction Using Physiological Models and Support Vector Regression”, in *2013 12th International Conference on Machine Learning and Applications*, vol. 1, Dec. 2013, pp. 135–140.

[37] S. Saria, A. Duchi, and D. Koller, “Discovering deformable motifs in continuous time series data”, in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, 2011, p. 1465. [Online]. Available: `http://ijcai.org/Proceedings/11/Papers/247.pdf`.

[38] K. J. Aström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[39] A. Cowles, “Stock Market Forecasting”, *Econometrica*, vol. 12, no. 3/4, pp. 206–214, 1944, ISSN: 0012-9682. DOI: `10.2307/1905433`. [Online]. Available: `https://www.jstor.org/stable/1905433` (visited on 10/24/2018).

[40] Z. Syed, C. Stultz, M. Kellis, P. Indyk, and J. Guttag, “Motif discovery in physiological datasets: A methodology for inferring predictive elements”, *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 4, no. 1, p. 2, 2010.

[41] M. Breton and B. Kovatchev, "Analysis, modeling, and simulation of the accuracy of continuous glucose sensors", *Journal of Diabetes Science and Technology*, vol. 2, no. 5, pp. 853–862, 2008.

[42] C. K. Chui, *An introduction to wavelets*. Academic press, 2014, vol. 1.

[43] A. Mueen, "Time series motif discovery: Dimensions and applications", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 2, pp. 152–159, 2014.

[44] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[45] Y. S. Abu-Mostafa, "Learning from hints in neural networks.", *J. Complexity*, vol. 6, no. 2, pp. 192–198, 1990.

[46] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task Sequence to Sequence Learning", *arXiv:1511.06114 [cs, stat]*, Nov. 2015, arXiv: 1511.06114.

[47] A. Gensler, J. Henze, B. Sick, and N. Raabe, "Deep Learning for solar power forecasting #x2014; An approach using AutoEncoder and LSTM Neural Networks", in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2016, pp. 002 858–002 865.

[48] M. Fiterau, S. Bhooshan, J. Fries, C. Bournhonesque, J. Hicks, E. Halilaj, C. Ré, and S. Delp, "Shortfuse: Biomedical time series representations in the presence of structured information", *Proceedings of machine learning research*, vol. 68, p. 59, 2017.

[49] P. Schulam and S. Saria, "Reliable decision support using counterfactual models", in *Advances in Neural Information Processing Systems*, 2017, pp. 1697–1708.

[50] C. Midroni, P. J. Leimbigler, G. Baruah, M. Kolla, A. J. Whitehead, and Y. Fossat, "Predicting glycemia in type 1 diabetes patients: Experiments with XGBoost", p. 6,

[51] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks”, in *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, 2016, pp. 1747–1756.

[52] O. Gottesman, F. Johansson, J. Meier, J. Dent, D. Lee, S. Srinivasan, L. Zhang, Y. Ding, D. Wihl, X. Peng, *et al.*, “Evaluating reinforcement learning algorithms in observational health settings”, *arXiv preprint arXiv:1805.12298*, 2018.

[53] S. Levine, A. Kumar, G. Tucker, and J. Fu, “Offline reinforcement learning: Tutorial, review, and perspectives on open problems”, *arXiv preprint arXiv:2005.01643*, 2020.

[54] S. Fujimoto, D. Meger, and D. Precup, “Off-policy deep reinforcement learning without exploration”, in *International Conference on Machine Learning*, 2019, pp. 2052–2062.

[55] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, “Benchmarking Batch Deep Reinforcement Learning Algorithms”, en, *arXiv:1910.01708 [cs, stat]*, Oct. 2019. [Online]. Available: `http://arxiv.org/abs/1910.01708` (visited on 02/06/2020).

[56] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, “Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction”, in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 11 761–11 771. [Online]. Available: `http://papers.nips.cc/paper/9349-stabilizing-off-policy-q-learning-via-bootstrapping-error-reduction.pdf` (visited on 02/06/2020).

[57] I. Fox, L. Ang, M. Jaiswal, R. Pop-Busui, and J. Wiens, “Contextual motifs: Increasing the utility of motifs using contextual data”, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 155–164.

[58] A. Van Esbroeck, “Learning better clinical risk models”, PhD thesis, The University of Michigan, 2015.

[59] R. E. Kleiger, J. P. Miller, J. T. Bigger, and A. J. Moss, “Decreased heart rate variability and its association with increased mortality after acute myocardial infarction”, *The American journal of cardiology*, vol. 59, no. 4, pp. 256–262, 1987.

[60] M. Jaiswal, K. McKeon, N. Comment, J. Henderson, S. Swanson, C. Plunkett, P. Nelson, and R. Pop-Busui, “Association between impaired cardiovascular autonomic function and hypoglycemia in patients with type 1 diabetes”, *Diabetes care*, vol. 37, no. 9, pp. 2616–2621, 2014.

[61] M. Muggeo, G. Zoppini, E. Bonora, E. Brun, R. C. Bonadonna, P. Moghetti, and G. Verlato, “Fasting plasma glucose variability predicts 10-year survival of type 2 diabetic patients: The verona diabetes study.”, *Diabetes care*, vol. 23, no. 1, pp. 45–50, 2000.

[62] D. Minnen, T. Starner, I. Essa, and C. Isbell, “Discovering characteristic actions from on-body sensor data”, in *2006 10th IEEE international symposium on wearable computers*, IEEE, 2006, pp. 11–18.

[63] Q. Zhou and W. H. Wong, “Cismodule: De novo discovery of cis-regulatory modules by hierarchical mixture modeling”, *Proceedings of the national academy of sciences of the United States of America*, vol. 101, no. 33, pp. 12 114–12 119, 2004.

[64] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins”, *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[65] J. Lin, E. Keogh, S. Lonardi, and P. Patel, “Finding motifs in time series”, in *Proc. of the 2nd Workshop on Temporal Data Mining*, 2002, pp. 53–68.

[66] J. Buhler and M. Tompa, “Finding motifs using random projections”, *Journal of computational biology*, vol. 9, no. 2, pp. 225–242, 2002.

[67] J. Grabocka, N. Schilling, and L. Schmidt-Thieme, “Latent time-series motifs”, *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 11, no. 1, p. 6, 2016.

[68] K. L. Jensen, M. P. Styczynski, I. Rigoutsos, and G. N. Stephanopoulos, "A generic motif discovery algorithm for sequential data", *Bioinformatics*, vol. 22, no. 1, pp. 21–28, 2006.

[69] T. Oates, "Peruse: An unsupervised algorithm for finding recurring patterns in time series", in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, IEEE, 2002, pp. 330–337.

[70] L. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining", in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2009, pp. 947–956.

[71] M. Shokoohi-Yekta, Y. Chen, B. Campana, B. Hu, J. Zakaria, and E. Keogh, "Discovery of meaningful rules in time series", in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2015, pp. 1085–1094.

[72] S. Saria, A. Duchi, and D. Koller, "Discovering deformable motifs in continuous time series data", in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, 2011, p. 1465.

[73] K. P. Murphy, "Dynamic bayesian networks", *Probabilistic Graphical Models, M. Jordan*, vol. 7, 2002.

[74] A. Van Esbroeck, C.-C. Chia, and Z. Syed, "Heart rate topic models", in *The Twenty-Sixth AAAI Conference on Artificial Intelligence*, vol. 1001, 2012, p. 48 109.

[75] T.-h. Lin, P. Ray, G. K. Sandve, S. Uguroglu, and E. P. Xing, "Baycis: A bayesian hierarchical hmm for cis-regulatory module decoding in metazoan genomes", in *Annual International Conference on Research in Computational Molecular Biology*, Springer, 2008, pp. 66–81.

[76] S. Saria, D. Koller, and A. Penn, "Learning individual and population level traits from clinical temporal data", in *Proc. Neural Information Processing Systems (NIPS), Predictive Models in Personalized Medicine workshop*, Citeseer, 2010.

[77] P. A. Baghurst, “Calculating the mean amplitude of glycemic excursion from continuous glucose monitoring data: An automated algorithm”, *Diabetes technology & therapeutics*, vol. 13, no. 3, pp. 296–302, 2011.

[78] G. D. Molnar, J. W. Rosevear, E. Ackerman, L. C. Gatewood, W. F. Taylor, *et al.*, “Mean amplitude of glycemic excursions, a measure of diabetic instability”, *Diabetes*, vol. 19, no. 9, pp. 644–655, 1970.

[79] R. Siddharthan, E. D. Siggia, and E. Van Nimwegen, “Phylogibbs: A gibbs sampling motif finder that incorporates phylogeny”, *PLoS Comput Biol*, vol. 1, no. 7, e67, 2005.

[80] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, “Probabilistic programming in python using pymc3”, *PeerJ Computer Science*, vol. 2, e55, 2016.

[81] J. S. Liu, “Metropolized gibbs sampler: An improvement”, Technical report, Dept. Statistics, Stanford Univ, Tech. Rep., 1996.

[82] M. D. Hoffman and A. Gelman, “The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo.”, *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1593–1623, 2014.

[83] S. Zoungas, A. Patel, J. Chalmers, B. E. de Galan, Q. Li, L. Billot, M. Woodward, T. Ninomiya, B. Neal, S. MacMahon, *et al.*, “Severe hypoglycemia and risks of vascular events and death”, *New England Journal of Medicine*, vol. 363, no. 15, pp. 1410–1418, 2010.

[84] R. M. Bergenstal, “Glycemic variability and diabetes complications: Does it matter? simply put, there are better glycemic markers!”, *Diabetes care*, vol. 38, no. 8, pp. 1615–1621, 2015.

[85] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, “Scikit-learn: Machine learning in python”, *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[86] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: A novel symbolic representation of time series", *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[87] H. Harutyunyan, H. Khachatrian, D. C. Kale, G. Ver Steeg, and A. Galstyan, "Multitask learning and benchmarking with clinical time series data", *Scientific data*, vol. 6, no. 1, pp. 1–18, 2019.

[88] H. Hassan, A. Aue, C. Chen, V. Chowdhary, J. Clark, C. Federmann, X. Huang, M. Junczys-Dowmunt, W. Lewis, and M. Li, "Achieving Human Parity on Automatic Chinese to English News Translation", *arXiv preprint arXiv:1803.05567*, 2018.

[89] A. Radford, R. Jozefowicz, and I. Sutskever, "Learning to Generate Reviews and Discovering Sentiment", *arXiv:1704.01444 [cs]*, Apr. 2017. (visited on 04/13/2018).

[90] N. M. Nasrabadi, "Pattern recognition and machine learning", *Journal of electronic imaging*, vol. 16, no. 4, p. 049 901, 2007.

[91] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations", in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09, New York, NY, USA: ACM, 2009, pp. 609–616, ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553453. (visited on 04/13/2018).

[92] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning", in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[93] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups", *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov. 2012, ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2205597.

[94] A. van den Oord and O. Vinyals, "Neural discrete representation learning", in *Advances in Neural Information Processing Systems*, 2017, pp. 6309–6318.

[95] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality”, in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[96] A. M. Dai and Q. V. Le, “Semi-supervised sequence learning”, in *Advances in neural information processing systems*, 2015, pp. 3079–3087.

[97] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding”, in *NAACL-HLT (1)*, 2019.

[98] A. Hyvarinen and H. Morioka, “Unsupervised feature extraction by time-contrastive learning and nonlinear ica”, in *Advances in Neural Information Processing Systems*, 2016, pp. 3765–3773.

[99] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding”, *arXiv preprint arXiv:1807.03748*, 2018.

[100] R. Caruana, “Multitask learning”, in *Learning to learn*, Springer, 1998, pp. 95–133.

[101] B. Ahmed, T. Thesen, K. Blackmon, R. Kuzniecky, O. Devinsky, J. Dy, and C. Brodley, “Multi-task learning with weak class labels: Leveraging iEEG to detect cortical lesions in cryptogenic epilepsy”, in *Machine Learning for Healthcare Conference*, 2016, pp. 115–133.

[102] N. Razavian, J. Marcus, and D. Sontag, “Multi-task prediction of disease onsets from longitudinal laboratory tests”, in *Machine Learning for Healthcare Conference*, 2016, pp. 73–100.

[103] S. Ruder, “An Overview of Multi-Task Learning in Deep Neural Networks”, *arXiv:1706.05098 [cs, stat]*, Jun. 2017.

[104] P. Schwab, E. Keller, C. Muroi, D. J. Mack, C. Strässle, and W. Karlen, “Not to cry wolf: Distantly supervised multitask learning in critical care”, in *International Conference on Machine Learning*, 2018, pp. 4518–4527.

[105] E. Choi, C. Xiao, W. Stewart, and J. Sun, "Mime: Multilevel medical embedding of electronic health records for predictive healthcare", in *Advances in Neural Information Processing Systems*, 2018, pp. 4547–4557.

[106] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "Brits: Bidirectional recurrent imputation for time series", in *Advances in Neural Information Processing Systems*, 2018, pp. 6775–6785.

[107] G. Clifford, C. Liu, B. Moody, L. Lehman, I. Silva, Q. Li, A. Johnson, and R. Mark, "AF classification from a short single lead ECG recording: The Physionet Computing in Cardiology Challenge 2017", *Computing in Cardiology*, vol. 44, 2017.

[108] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch", 2017.

[109] D. Kingma and J. Ba, "Adam: A method for stochastic optimization", *International Conference for Learning Representations (ICLR)*, 2015.

[110] I. Fox, L. Ang, M. Jaiswal, R. Pop-Busui, and J. Wiens, "Deep Multi-Output Forecasting: Learning to Accurately Predict Blood Glucose Trajectories", in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18, New York, NY, USA: ACM, 2018, pp. 1387–1395, ISBN: 978-1-4503-5552-0. DOI: `10.1145/3219819.3220102`. [Online]. Available: `http://doi.acm.org/10.1145/3219819.3220102` (visited on 11/05/2018).

[111] A. M. Lamb, A. G. A. P. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks", in *Advances in neural information processing systems*, 2016, pp. 4601–4609.

[112] G. Cappon, G. Acciaroli, M. Vettoretti, A. Facchinetti, and G. Sparacino, "Wearable continuous glucose monitoring sensors: A revolution in diabetes treatment", *Electronics*, vol. 6, no. 3, p. 65, 2017.

[113] C. D. Man, F. Micheletto, D. Lv, M. Breton, B. Kovatchev, and C. Cobelli, "The UVA/PADOVA type 1 diabetes simulator: New features", *Journal of diabetes science and technology*, vol. 8, no. 1, pp. 26–34, 2014.

[114] R. Hovorka, V. Canonico, L. J. Chassin, U. Haueter, M. Massi-Benedetti, M. O. Federici, T. R. Pieber, H. C. Schaller, L. Schaupp, T. Vering, *et al.*, "Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes", *Physiological measurement*, vol. 25, no. 4, p. 905, 2004.

[115] W. Wu, K. Chen, Y. Qiao, and Z. Lu, "Probabilistic short-term wind power forecasting based on deep neural networks", in *2016 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, Oct. 2016, pp. 1–8.

[116] S. Mirshekarian, R. Bunescu, C. Marling, and F. Schwartz, "Using LSTMs to Learn Physiological Models of Blood Glucose Behavior", *EMBC*, 2017.

[117] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches", *Syntax, Semantics and Structure in Statistical Translation*, p. 103, 2014.

[118] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation", in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[119] B. Sudharsan, M. Peeples, and M. Shomali, "Hypoglycemia prediction using machine learning models for patients with type 2 diabetes", *Journal of diabetes science and technology*, vol. 9, no. 1, pp. 86–90, 2014.

[120] S. Oviedo, J. Vehí, R. Calm, and J. Armengol, "A review of personalized blood glucose prediction strategies for T1dm patients", *International journal for numerical methods in biomedical engineering*, vol. 33, no. 6, 2017.

[121] M. Eren-Oruklu, A. Cinar, and L. Quinn, *Hypoglycemia prediction with subject-specific recursive time-series models*. SAGE Publications, 2010.

[122] C. Zecchin, A. Facchinetti, G. Sparacino, and C. Cobelli, "Jump Neural Network for Real-Time Prediction of Glucose Concentration", English, in *Artificial Neural Networks*, ser. Methods in Molecular Biology, DOI: 10.1007/978-1-4939-2239-0_15, Springer New York, Jan. 2015, pp. 245–259, ISBN: 978-1-4939-2238-3.

[123] K. Plis, R. Bunescu, C. Marling, J. Shubrook, and F. Schwartz, “A machine learning approach to predicting blood glucose levels for diabetes management”, *Modern Artificial Intelligence for Health Analytics. Papers from the AAAI-14*, 2014.

[124] K. Turksoy, E. S. Bayrak, L. Quinn, E. Littlejohn, D. Rollins, and A. Cinar, “Hypoglycemia early alarm systems based on multivariable models”, *Industrial & engineering chemistry research*, vol. 52, no. 35, pp. 12 329–12 336, 2013.

[125] C. Zecchin, A. Facchinetti, G. Sparacino, and C. Cobelli, “How Much Is Short-Term Glucose Prediction in Type 1 Diabetes Improved by Adding Insulin Delivery and Meal Content Information to CGM Data? A Proof-of-Concept Study”, en, *Journal of Diabetes Science and Technology*, vol. 10, no. 5, pp. 1149–1160, Sep. 2016, ISSN: 1932-2968.

[126] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, “Scikit-learn: Machine learning in Python”, *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[127] M. Ghassemi, M. A. Pimentel, T. Naumann, T. Brennan, D. A. Clifton, P. Szolovits, and M. Feng, “A multivariate timeseries modeling approach to severity of illness assessment and forecasting in icu with sparse, heterogeneous clinical data”, in *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, vol. 2015, NIH Public Access, 2015, p. 446.

[128] I. Fox, J. Lee, R. Busui, and J. Wiens, “Deep reinforcement learning for closed-loop blood glucose control”, in *Machine Learning for Healthcare Conference*, 2020.

[129] R. D. Coffen and L. M. Dahlquist, “Magnitude of type 1 diabetes self-management in youth health care needs diabetes educators”, *The Diabetes Educator*, vol. 35, no. 2, pp. 302–308, 2009.

[130] M. K. Bothe, L. Dickens, K. Reichel, A. Tellmann, B. Ellger, M. Westphal, and A. A. Faisal, “The use of reinforcement learning algorithms to meet the challenges of an artificial pancreas”, en, *Expert Review of Medical Devices*, vol. 10, no. 5, pp. 661–673, Sep. 2013, ISSN: 1743-4440, 1745-2422. DOI: 10.1586/17434440.2013.827515.

[Online]. Available: http://www.tandfonline.com/doi/full/10.1586/17434440.2013.827515 (visited on 10/31/2018).

[131] J. E. Pinsker, J. B. Lee, E. Dassau, D. E. Seborg, P. K. Bradley, R. Gondhalekar, W. C. Bevier, L. Huyett, H. C. Zisser, and F. J. Doyle, "Randomized Crossover Comparison of Personalized MPC and PID Control Algorithms for the Artificial Pancreas", en, *Diabetes Care*, p. dc152344, Jun. 2016, ISSN: 0149-5992, 1935-5548. DOI: 10.2337/dc15-2344. [Online]. Available: http://care.diabetesjournals.org/content/early/2016/06/10/dc15-2344 (visited on 11/08/2018).

[132] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt: Meta-learning for model-based control", *Proc. of ICLR*, 2019.

[133] R. Visentin, C. Dalla Man, B. Kovatchev, and C. Cobelli, "The university of virginia/padova type 1 diabetes simulator matches the glucose traces of a clinical trial", *Diabetes technology & therapeutics*, vol. 16, no. 7, pp. 428–434, 2014.

[134] O. Gottesman, F. Johansson, M. Komorowski, A. Faisal, D. Sontag, F. Doshi-Velez, and L. A. Celi, "Guidelines for reinforcement learning in healthcare", *Nature Medicine*, vol. 25, no. 1, pp. 16–18, 2019.

[135] A. Irpan, *Deep reinforcement learning doesn't work yet*, https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018.

[136] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters", in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[137] W.-H. Weng, M. Gao, Z. He, S. Yan, and P. Szolovits, "Representation and reinforcement learning for personalized glycemic control in septic patients", *NeurIPS 2017 ML4H Workshop*, Dec. 2, 2017. (visited on 09/20/2019).

[138] M. Komorowski, L. A. Celi, O. Badawi, A. C. Gordon, and A. A. Faisal, "The Artificial Intelligence Clinician learns optimal treatment strategies for sepsis in intensive care", *Nature Medicine*, p. 1, 2018.

[139] N. Prasad, L. F. Cheng, C. Chivers, M. Draugelis, and B. E. Engelhardt, "A reinforcement learning approach to weaning of mechanical ventilation in intensive care units", in *33rd Conference on Uncertainty in Artificial Intelligence, UAI 2017*, 2017.

[140] P. Klasnja, S. Smith, N. J. Seewald, A. Lee, K. Hall, B. Luers, E. B. Hekler, and S. A. Murphy, "Efficacy of contextually tailored suggestions for physical activity: A micro-randomized optimization trial of HeartSteps", *Annals of Behavioral Medicine*, vol. 53, no. 6, pp. 573–582, May 3, 2019. (visited on 09/20/2019).

[141] S. Trevitt, S. Simpson, and A. Wood, "Artificial pancreas device systems for the closed-loop control of type 1 diabetes", *Journal of Diabetes Science and Technology*, vol. 10, no. 3, pp. 714–723, Nov. 20, 2015. (visited on 09/06/2019).

[142] G. M. Steil, "Algorithms for a closed-loop artificial pancreas: The case for proportional-integral-derivative control", *Journal of diabetes science and technology*, vol. 7, no. 6, pp. 1621–1631, 2013.

[143] S. K. Garg, S. A. Weinzimer, W. V. Tamborlane, B. A. Buckingham, B. W. Bode, T. S. Bailey, R. L. Brazg, J. Ilany, R. H. Slover, S. M. Anderson, R. M. Bergenstal, B. Grosman, A. Roy, T. L. Cordero, J. Shin, S. W. Lee, and F. R. Kaufman, "Glucose Outcomes with the In-Home Use of a Hybrid Closed-Loop Insulin Delivery System in Adolescents and Adults with Type 1 Diabetes", *Diabetes Technology & Therapeutics*, vol. 19, no. 3, pp. 155–163, Jan. 2017, ISSN: 1520-9156. DOI: `10.1089/dia.2016.0421`. [Online]. Available: `https://www.liebertpub.com/doi/full/10.1089/dia.2016.0421` (visited on 11/08/2018).

[144] J. L. Ruiz, J. L. Sherr, E. Cengiz, L. Carria, A. Roy, G. Voskanyan, W. V. Tamborlane, and S. A. Weinzimer, "Effect of Insulin Feedback on Closed-Loop Glucose Control: A Crossover Study", en, *Journal of Diabetes Science and Technology*, vol. 6, no. 5, pp. 1123–1130, Sep. 2012, ISSN: 1932-2968. DOI: `10.1177/193229681200600517`. [Online]. Available: `https://doi.org/10.1177/193229681200600517` (visited on 11/09/2018).

[145] M. Tejedor, A. Z. Woldaregay, and F. Godtliebsen, "Reinforcement learning application in diabetes blood glucose control: A systematic review", *Artificial Intelligence in Medicine*, p. 101 836, Feb. 21, 2020, ISSN: 0933-3657. DOI: `10.1016/j.artmed.`

2020.101836. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0933365718304548 (visited on 03/04/2020).

[146] P. D. Ngo, S. Wei, A. Holubová, J. Muzik, and F. Godtliebsen, "Reinforcement-learning optimal control for type-1 diabetes", in *2018 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, Mar. 2018, pp. 333–336. DOI: 10.1109/BHI.2018.8333436.

[147] M. Oroojeni Mohammad Javad, S. Agboola, K. Jethwani, I. Zeid, and S. Kamarthi, "Reinforcement learning algorithm for blood glucose control in diabetic patients", in *Volume 14: Emerging Technologies; Safety Engineering and Risk Analysis; Materials: Genetics to Structures*, Houston, Texas, USA: ASME, 2015, V014T06A009, ISBN: 978-0-7918-5757-1. (visited on 11/10/2018).

[148] Q. Sun, M. Jankovic, J. Budzinski, B. Moore, P. Diem, C. Stettler, and S. G. Mougiakakou, "A dual mode adaptive basal-bolus advisor based on reinforcement learning", *IEEE Journal of Biomedical and Health Informatics*, pp. 1–1, 2018.

[149] E. Daskalaki, L. Scarnato, P. Diem, and S. G. Mougiakakou, "Preliminary results of a novel approach for glucose regulation using an actor-critic learning based controller", in *UKACC International Conference on Control 2010*, ISSN: null, Sep. 2010, pp. 1–5. DOI: 10.1049/ic.2010.0287.

[150] M. De Paula, G. G. Acosta, and E. C. Martínez, "On-line policy learning and adaptation for real-time personalization of an artificial pancreas", *Expert Syst. Appl.*, vol. 42, no. 4, pp. 2234–2255, Mar. 2015. (visited on 11/16/2018).

[151] M. Vettoretti, S. Del Favero, G. Sparacino, and A. Facchinetti, "Modeling the error of factory-calibrated continuous glucose monitoring sensors: Application to dexcom g6 sensor data", in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2019, pp. 750–753.

[152] L. Magni, D. M. Raimondo, L. Bossi, C. D. Man, G. De Nicolao, B. Kovatchev, and C. Cobelli, "Model predictive control of type 1 diabetes: An in silico trial", *Journal of diabetes science and technology (Online)*, vol. 1, no. 6, pp. 804–812, Nov. 2007. (visited on 02/26/2020).

[153] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning", en, *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, ISSN: 0028-0836. DOI: `10.1038/nature14236`. [Online]. Available: `http://www.nature.com/nature/journal/v518/n7540/abs/nature14236.html` (visited on 12/02/2016).

[154] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", en, in *International Conference on Machine Learning*, Jul. 2018, pp. 1861–1870. (visited on 09/22/2019).

[155] B. Eysenbach and S. Levine, "If maxent rl is the answer, what is the question?", *arXiv preprint arXiv:1910.01913*, 2019.

[156] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft Actor-Critic Algorithms and Applications", *arXiv:1812.05905 [cs, stat]*, Dec. 2018. (visited on 09/23/2019).

[157] T. Zhu, K. Li, P. Herrero, J. Chen, and P. Georgiou, "A deep learning algorithm for personalized blood glucose prediction", *IJCAI Knowledge Discovery in Healthcare Data Workshop*, 2018.

[158] A. Kuroda, H. Kaneto, T. Yasuda, M. Matsuhisa, K. Miyashita, N. Fujiki, K. Fujisawa, T. Yamamoto, M. Takahara, F. Sakamoto, T.-a. Matsuoka, and I. Shimomura, "Basal insulin requirement is 30% of the total daily insulin dose in type 1 diabetic patients who use the insulin pump", *Diabetes Care*, vol. 34, no. 5, pp. 1089–1090, May 1, 2011. (visited on 09/23/2019).

[159] J. Walsh, R. Roberts, and T. Bailey, "Guidelines for optimal bolus calculator settings in adults", *Journal of Diabetes Science and Technology*, vol. 5, no. 1, pp. 129–135, Jan. 1, 2011. (visited on 09/23/2019).

[160] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *International Conference on Machine Learning*, 2015, pp. 448–456.

[161] S. Ayano-Takahara, K. Ikeda, S. Fujimoto, K. Asai, Y. Oguri, S.-i. Harashima, H. Tsuji, K. Shide, and N. Inagaki, "Carbohydrate intake is associated with time spent in the euglycemic range in patients with type 1 diabetes", *Journal of diabetes investigation*, vol. 6, no. 6, pp. 678–686, 2015.

[162] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, Second edition, ser. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018, ISBN: 978-0-262-03924-6.

[163] R. Islam, P. Henderson, M. Gomrokchi, and D. Precup, "Reproducibility of benchmarked deep reinforcement learning tasks for continuous control", *ICML Reproducibility in Machine Learning Workshop*, 2017.

[164] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg, "Ai safety gridworlds", *arXiv preprint arXiv:1711.09883*, 2017.

[165] J. Futoma, M. A. Masood, and F. Doshi-Velez, "Identifying distinct, effective treatments for acute hypotension with soda-rl: Safely optimized diverse accurate reinforcement learning", *AMIA Summits on Translational Science Proceedings*, vol. 2020, p. 181, 2020.

[166] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization", in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 22–31.

[167] R. Visentin, E. Campos-Náñez, M. Schiavon, D. Lv, M. Vettoretti, M. Breton, B. P. Kovatchev, C. Dalla Man, and C. Cobelli, "The uva/padova type 1 diabetes simulator goes from single meal to single day", *Journal of diabetes science and technology*, vol. 12, no. 2, pp. 273–281, 2018.

[168] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations", in *Advances in neural information processing systems*, 2018, pp. 6571–6583.

[169] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, and T. Graepel, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm", *arXiv preprint arXiv:1712.01815*, 2017.

[170] X. Xiong, J. Wang, F. Zhang, and K. Li, "Combining deep reinforcement learning and safety based control for autonomous driving", *arXiv preprint arXiv:1612.00147*, 2016.

[171] A. Jain, K. Khetarpal, and D. Precup, "Safe option-critic: Learning safety in the option-critic architecture", *ICML ALA Workshop*, 2018.

[172] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning", in *Advances in neural information processing systems*, 2018, pp. 8092–8101.

[173] S. B. Taieb, A. Sorjamaa, and G. Bontempi, "Multiple-output modeling for multi-step-ahead time series forecasting", *Neurocomputing*, vol. 73, no. 10-12, pp. 1950–1957, 2010.

[174] J. A. Harris and F. G. Benedict, *A biometric study of basal metabolism in man*, 279. Carnegie institution of Washington, 1919.