LSTMs and Deep Residual Networks for Carbohydrate and Bolus Recommendations in Type 1 Diabetes Management

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Jeremy T. Beauchamp

May 2021

This thesis titled

LSTMs and Deep Residual Networks for Carbohydrate and Bolus Recommendations in Type 1 Diabetes Management

by

JEREMY T. BEAUCHAMP

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Razvan Bunescu

Associate Professor

Mei Wei

Dean and Moss Professor of Engineering Education

Abstract

BEAUCHAMP, JEREMY T., M.S., May 2021, Computer Science

LSTMs and Deep Residual Networks for Carbohydrate and Bolus Recommendations in Type 1 Diabetes Management (61 pp.)

Director of Thesis: Razvan Bunescu

To avoid serious diabetic complications, people with type 1 diabetes must keep their Blood Glucose Levels (BGLs) as close to normal as possible. Insulin dosages and carbohydrate consumption are important considerations in managing BGLs. Since the 1960s, models have been developed to forecast blood glucose levels based on the history of BGLs, insulin dosages, carbohydrate intake, and other physiological and lifestyle factors. Such predictions can be used to alert people of impending unsafe BGLs or to control insulin flow in an artificial pancreas. In past research, an LSTM-based approach to blood glucose level prediction was developed to process "what if" scenarios, in which people could enter foods they might eat or insulin amounts they might take and then see the affect on future BGLs. In this work, the "what-if" scenario is inverted and a similar architecture is introduced, based on chaining two LSTMs that can be trained to make either insulin or carbohydrate recommendations aimed at reaching a desired BG level in the future. Leveraging a recent state-of-the-art model for time series forecasting, a novel deep residual architecture is proposed for the same recommendation task, in which the two LSTM chain is used as a repeating block. Experimental evaluations using real patient data from the OhioT1DM dataset show that the new integrated architecture compares favorably with the previous LSTM-based approach, substantially outperforming the baselines. The promising results suggest that this novel approach could potentially be of practical use to people with type 1 diabetes for self-management of BGLs.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Razvan Bunescu for his exceptional guidance, knowledge, and mostly, his patience and professionalism. I have had the pleasure of learning a lot from Dr. Bunescu and this research certainly would not have been possible without him. I would also like to thank Dr. Cindy Marling for all of her assistance and knowledge throughout the research process. In addition, I would like to thank Zhongen Li for his help implementing parts of this project.

Table of Contents

List of Tables

List of Figures

List of Acronyms

BG Blood Glucose

BGL Blood Glucose Level

BGLs Blood Glucose Levels

BGLP Blood Glucose Level Prediction

BW Bolus Wizard

CGM Continuous Glucose Monitor

FC Fully Connected

FCN Fully Connected Network

LSTM Long-Short Term Memory

MAE Mean Absolute Error

N-BEATS Neural Basis Expansion for Interpretable Time-Series Forecasting

RMSE Root Mean Square Error

RNN Recurrent Neural Network

RNNs Recurrent Neural Networks

T1D Type 1 Diabetes

T2D Type 2 Diabetes

ToD Time-of-Day

1 Introduction

Diabetes is a disease in which the body either does not produce enough insulin or cannot make sufficient use of the insulin that it does produce. Insulin is a critical hormone that allows the glucose from the blood stream to be absorbed into cells. Without it, or a way for the body to properly process it, too much glucose remains in the bloodstream. High amounts of glucose in the bloodstream can lead to several serious complications, including heart disease, vision loss, and kidney disease [8]. There are two major types of diabetes, Type 1 Diabetes (T1D) and Type 2 Diabetes (T2D), both of which require careful management of BGLs.

In T1D, which is the most severe type of diabetes, the pancreas does not produce insulin, which leads to BG levels that are too high [8]. This is thought to be caused by the destruction of cells in the pancreas by an autoimmune reaction. Of all people diagnosed with diabetes, only about 5-10% of them have type 1 [8]. Those with T1D typically are diagnosed with it early in life. As of now, there is no known cure or prevention measure for T1D.

In T2D, the body does not use insulin efficiently enough to keep BGLs at normal levels. Type 2 accounts for about 90-95% of all cases of diabetes [8]. Fortunately, T2D can be prevented or delayed by living a generally healthy lifestyle.

The day-to-day self-management of diabetes is essential for someone with T1D. The patient must painstakingly manage their BGLs and attempt to prevent them from becoming too high or too low. T1D is managed by using external sources of insulin to prevent BGLs from becoming too high. However, much care must be put into the amount of insulin that is allowed into the body. Too much insulin can cause BGLs to drop too low. When BGLs become too high, patients may begin to experience symptoms of hyperglycemia. When BGLs drop too low, patients will likely start to feel the effects of hypoglycemia.

Both hyperglycemia and hypoglycemia are dangerous and can lead to severe long-term complications if BGLs are not promptly corrected.

An important part of diabetes management consists of vigilantly monitoring one's BGL. One way this can be done is by using a glucometer, which measures the amount of glucose in a blood sample, typically taken from a fingertip. Another common method of monitoring BGLs is by using a Continuous Glucose Monitor (CGM) system, a device capable of measuring blood glucose every few minutes via a subcutaneous sensor [7]. A person with T1D must also make key decisions about how much insulin to take and when to take it several times per day. They must also make decisions about the quantity and timing of their meals throughout the day.

Diabetes management primarily involves taking actions to correct BGL issues as they arise. Once BGLs become too high a person will *react* to this by taking insulin. Likewise, a person will *react* to their BGLs dropping too low by eating or taking glucose tablets. The process of diabetes management could potentially be simpler and even safer if a more proactive approach was taken instead. Rather than reacting to symptoms of hyperglycemia or hypoglycemia, preemptive steps could be taken to prevent these symptoms altogether.

An important step towards more proactive diabetes management is the ability to forecast future BGLs accurately. This information would allow people to make more informed decisions about how to prevent potential hypoglycemic or hyperglycemic events. Efforts to model BGLs can be dated back to as early as the 1960s [4]. In recent years, there has been much research into using machine learning algorithms for blood glucose level prediction [5, 22, 26, 27]. With the widespread adoption of CGM systems, and the large amounts of data that they can collect, machine learning has become a more feasible approach to create BGL prediction systems. However, even with the ability to forecast BGLs, the patient would still need to make a decision about how much to eat to raise their

BGLs to a normal range or how much to bolus to lower their BGLs to a healthy level. This kind of decisions are the primary focus of this thesis.

1.1 Research Objective

The objective of the research presented in this thesis is to design and train neural network models capable of providing people with T1D recommendations on how many grams of carbohydrates they should eat or how much insulin they should bolus to reach a target BGL in the near future. This can be thought of as essentially reversing the BGL prediction problem. Instead of predicting BGLs in the near future, the objective of this research is to recommend an action that a person should take in order to reach a desired BGL in the near future. Previous research in BGL prediction [22] has aimed to answer the question of "What will my BGL be in an hour if I eat a 30 carbs snack 10 minutes from now?", while this research attempts to answer the question "How many carbs should I eat 10 minutes from now to raise my BGL to 140 in an hour?". Two architectures are developed for these tasks, an LSTM-based architecture and an N-BEATS-based architecture. The LSTM-based architecture draws inspiration from the architecture introduced in [22] for the task of BGL prediction. The architecture was redesigned to make meal or bolus recommendations using similar input data to that of the original BGL prediction system. The N-BEATS-based architecture is a modified version of the original N-BEATS architecture [23], taking inspiration from another modified version of the N-BEATS architecture that was built for the task of BGL prediction [27]. Both architectures are trained on the OhioT1DM dataset [20], which contains data from real people with T1D. The goal of this research is to use this real-patient data and proven BGL prediction architectures to create accurate, personalized bolus and meal recommendation systems.

1.2 Thesis Outline

The rest of this thesis is structured as follows:

- Chapter 2 will compare and contrast this research with other similar research, as well as provide background on some of the core deep learning models that are utilized in this work.
- Chapter 3 will characterize the problems that this research aims to solve as well as the neural architectures that have been designed for each recommendation scenario.
- Chapter 4 will introduce the dataset that is used for this research as well as the pre-processing procedures that have been developed for mitigating the noise in the original raw data.
- Chapter 5 will explain the procedures for training and evaluating the models, as well as discuss the results of several experimental evaluations.
- Chapter 6 will end this thesis with concluding remarks and suggestions for future research.

2 Background

This chapter will provide background on the major neural network architectures that are utilized in this research, as well as summarize and discuss research related to artificial intelligence-based T1D management systems. Two of the most common T1D management systems are BGL prediction systems and insulin recommendation systems. Research into improving the accuracy and reliability of these systems could have a significant positive impact on the lives of those with T1D. These systems can give people with T1D more confidence in their decisions about how much insulin they should take and provide insight into how these decisions will affect their BGLs over the following hours. As artificial intelligence and machine learning algorithms continue to evolve and improve, so too can the performance of these T1D management systems.

2.1 Time Series Prediction Models

Neural networks are powerful machine learning models capable of learning to perform a wide variety of tasks across many domains given sufficient training data. In this research, the goal is to create neural network models capable of performing time series forecasting, where the task consists of predicting a future point in a time series given previous data from the series. However, due to the temporal nature of the data, a generic fully connected neural network will have difficulty processing time series data. As described in the following sections, Recurrent Neural Networks (RNNs) and Deep Residual Networks are two types of networks that are capable of processing and learning from such data.

2.1.1 Recurrent Neural Networks

RNNs are neural networks capable of learning from time series data. RNNs process time series data in discrete time-steps, rather than processing all of it at once like a fully connected network would. RNNs have three sets of weights, W, U, and V, that are used to

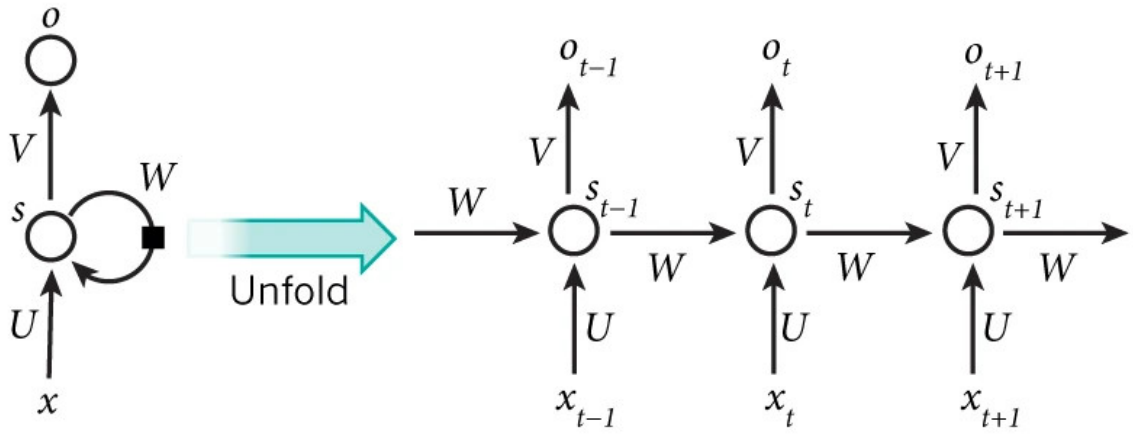

Figure 2.1: A visualization of an RNN [18].

compute a hidden state, $\mathbf{s}_t$ and an output $\mathbf{o}_t$ for every time step t. The state contains values that the network has recursively computed based on inputs from all previous time-steps. Along with the hidden state of the previous time-step, $\mathbf{s}_{t-1}$, the input at the current step in the time series, $\mathbf{x}_t$, is used to calculate both the output of the cell and the hidden state values. The hidden state at time t, $\mathbf{s}_t$, is calculated by the following formula:

$$\mathbf{s}_t = f(W\mathbf{s}_{t-1} + U\mathbf{x}_t + \mathbf{b}_s)$$

where $\mathbf{b}_s$ is a bias vector, and f is a non-linear activation function. The tanh function or the sigmoid function are common choices for f. The output of the network at time t, $\mathbf{o}_t$, is calculated as follows:

$$\mathbf{o}_t = f(V\mathbf{s}_t + \mathbf{b}_o)$$

where f is again a non-linear activation function and $\mathbf{b}_o$ is a different bias vector. Note that f does not need to be the same in both equations. The weight matrices W, U, V, are learned via the Backpropagation Through Time (BPTT) algorithm [30]. While RNNs are capable of learning from time series data, they can have difficulty learning long distance

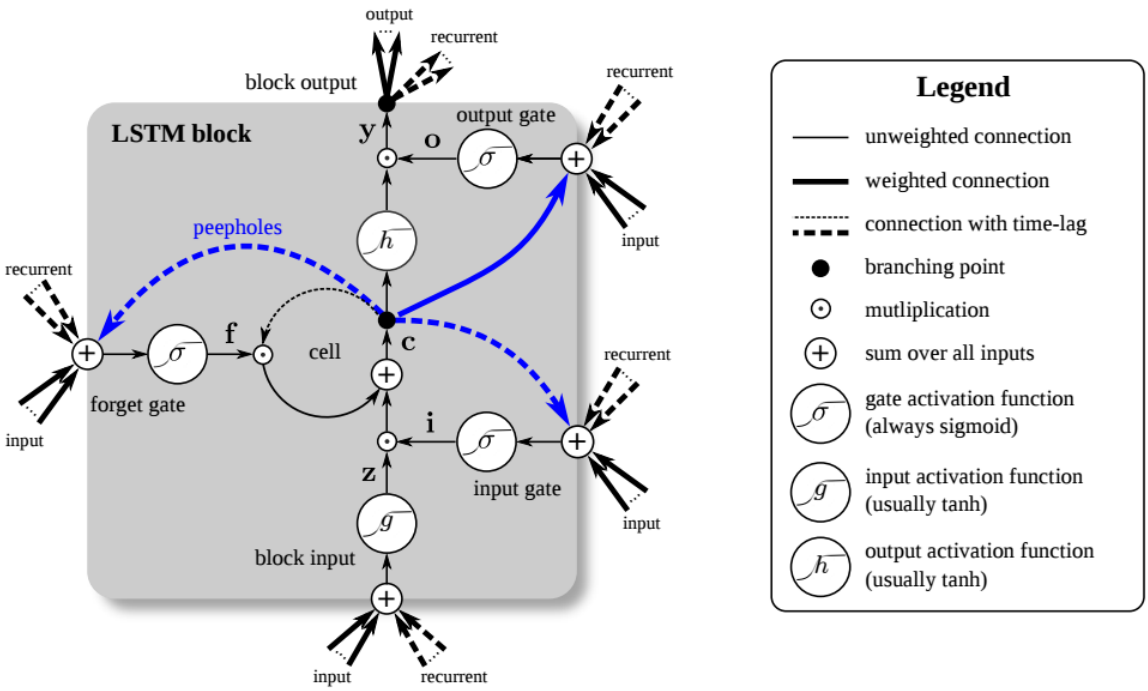

Figure 2.2: A diagram of an LSTM cell. The peephole connections represented by the blue lines are not part of the original LSTM architecture. Image from [11] was slightly modified.

dependencies due to the vanishing gradient problem [3]. The vanishing gradient problem refers to norm of the gradient rapidly shrinking to 0 for long-term dependencies during training, which makes it very difficult for RNNs to learn the relationships between events that are temporally distant [24]. While there have been several proposed Recurrent Neural Network (RNN) modifications that aim to remedy this issue, one of the most widely used is the Long-Short Term Memory (LSTM) network [14].

2.1.1.1 Long Short-Term Memory

In 1997, LSTM networks were introduced as a solution to the vanishing gradient problem [14]. LSTM networks use multiplicative gates on the input and output for each cell. The purpose of the input gate is to prevent the information that has already been

learned from the sequence to be perturbed by input events that are not relevant. Similarly, the output gate prevents irrelevant information from negatively affecting the output of a particular cell. In addition to gating the inputs and outputs of a cell, LSTM cells also include a forget gate, which allows the cell to determine how much of the information captured in the previous hidden state should be remembered or forgotten [9]. These three gates allow the network to learn how much the previous hidden state and the input should affect the internal state of the cell, as well as how much of the internal state should be used when determining the output of the cell.

The values of the three gates at time t are calculated by the following formulas:

$$\mathbf{f}_t = \sigma(W^{(f)}\mathbf{s}_{t-1} + U^{(f)}\mathbf{x}_t + \mathbf{b}^{(f)})$$
$$\mathbf{i}_t = \sigma(W^{(i)}\mathbf{s}_{t-1} + U^{(i)}\mathbf{x}_t + \mathbf{b}^{(i)})$$
$$\mathbf{o}_t = \sigma(W^{(o)}\mathbf{s}_{t-1} + U^{(o)}\mathbf{x}_t + \mathbf{b}^{(o)})$$

In these formulas, $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ represents the values of the forget gate, input gate, and output gate at time t, respectively. The σ in these formulas represents the sigmoid function. Once these gate values have been calculated, the output of the cell, $\mathbf{z}_t$ and the values of the internal state, $\mathbf{c}_t$ are calculated as follows:

$$\mathbf{z}_t = \tanh(W^{(z)}\mathbf{s}_{t-1} + U^{(z)}\mathbf{x}_t + b^{(z)})$$
$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{z}_t$$

Finally, once the internal state $\mathbf{c}_t$ has been calculated, the version of the hidden state the will be used as input for the next cell, $\mathbf{s}_t$, can be calculated as follows:

$$\mathbf{s}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

This improved logic allows LSTM networks to not only learn from time series data, but also learn what information should be remembered, and what can safely be forgotten. The

element-wise multiplication of the forget gate with the previous value of the cell provides a path for the gradient to pass to previous cell states without being repeatedly diminished. This represents an elegant and powerful solution to the vanishing gradient problem and as such seen widespread adoption for time series prediction tasks.

2.1.2 Deep Residual Networks

Research on the generalization performance of neural networks has demonstrated that increasing the depth of a network often results in improved performance [17, 28]. However, simply stacking a large number of layers in a network is not an infallible solution to improving its performance. Increasing the depth of a network not only increases the computational cost of training it but can also increase the difficulty of training [10]. One solution to this problem are shortcut (also called residual) connections, which skip at least one layer in a network. In [12], it was shown that using shortcut connections to simply add the original input of a stack of layers to that stack's output and pass that as input to the next stack allowed for very deep networks to achieve great performance on image recognition tasks [12, 13, 15]. Additionally, the inclusion of these shortcut connections made the networks easier to train. These types of networks are known as residual networks, or ResNets. Very recent work has shown that deep residual networks can also achieve state-of-the-art performance on the task of time series forecasting, even without the aid of RNNs [23].

2.1.2.1 N-BEATS

Oreshkin et al. have recently introduced a new architecture for time series forecasting, the Neural Basis Expansion for Interpretable Time-Series Forecasting (N-BEATS)[23]. The N-BEATS architecture is a deep residual model consisting of basic blocks that are stacked in a doubly residual manner. The basic building block of N-BEATS is a fully connected structure that initially takes as input a fixed-size lookback period of past values

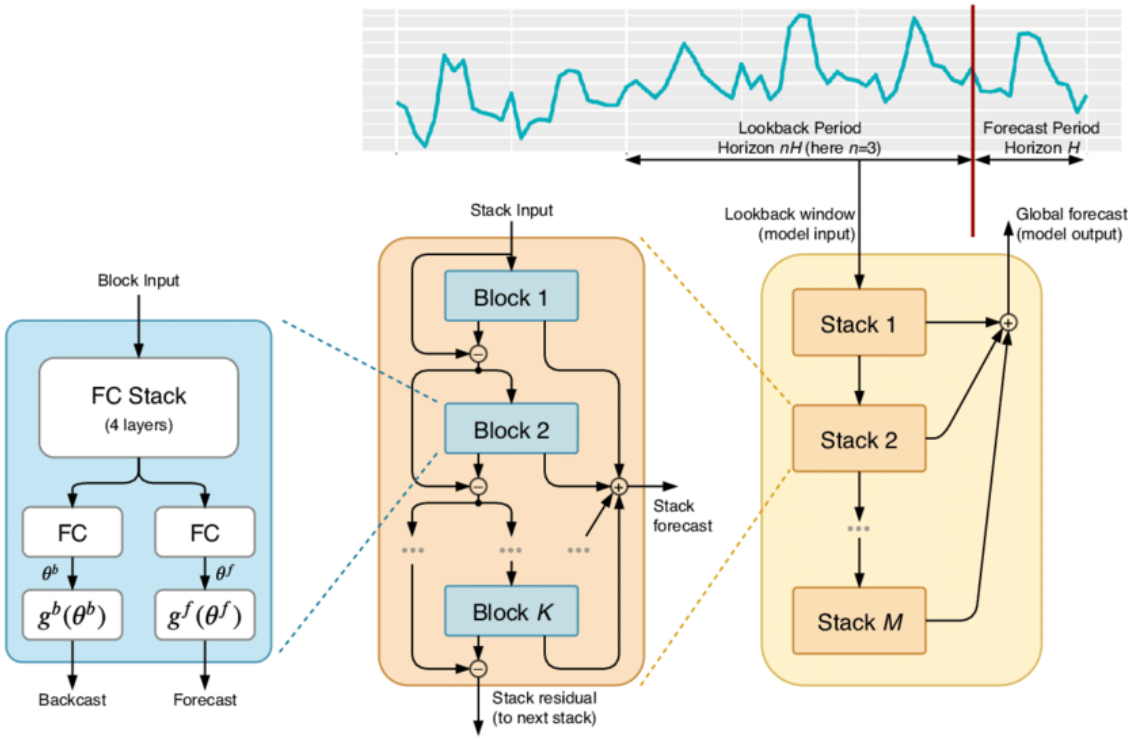

Figure 2.3: A diagram of the original N-BEATS architecture [23].

of the target variable and outputs both forecast (estimates of future values) and backcast (estimates of past values) vectors. Blocks are organized into stacks such that the backcast of the current block is subtracted from its input and fed as input to the next block, whereas the forecast vectors from each block are summed up to provide the overall stack forecast. The stacks themselves are chained in a pipeline where the backcast output of one stack is used as input for the next stack. The overall model forecast is then computed by accumulating the forecasts across all the stacks. The N-BEATS architecture has been shown to achieve state-of-the-art performance on a variety of widely used time series forecasting datasets [23].

2.2 Blood Glucose Prediction

Providing people with T1D the ability to accurately anticipate their impending BGLs would be an incredibly helpful and potentially life changing breakthrough. There has been much research with the goal of making this ambition closer to reality with artificial intelligence, specifically machine learning [5, 22, 27]. While the end goal of the research in this thesis is not to predict future BGLs, there are many parallels between BGL prediction and meal and bolus recommendation. BGL prediction is essentially the converse of meal or bolus recommendation. BGL prediction systems attempt to forecast a BGL in the future given information in the present, while meal or bolus recommendation systems use the desired future BGL (among other factors) given by the user to recommend a meal or bolus to take in the present in order to reach that BGL target. As such, it is important to understand both sides of this problem.

The performance of BGL prediction systems has improved due to advances in machine learning, and more specifically deep learning. Neural networks have allowed researchers to bypass the need for expensive, hand-engineered features. Advances in deep learning such as LSTM networks [14] and the N-BEATS architecture [23] have specifically helped improve performance of such systems due to their ability to process temporal BGL data. It is important to understand what types of techniques have led to improvement in BGL prediction systems, as these same techniques will likely lead to a high level of performance in meal and bolus recommendation systems given the close relationship between the problems.

In [5], a support vector regression model was trained on hand-engineered physiological features based on the raw features extracted from real patient data. This machine learning based approach was shown to outperform three physicians at predicting BGLs 30 and 60 minutes into the future. However, creating these hand-engineered features can be time-consuming, expensive, and require the aid of experts in the field of T1D. Using

deep learning models such as neural networks can eliminate the need for hand-engineered features, as these models can learn relevant features automatically.

In [22], the authors create a neural network model utilizing the LSTM networks described in Section 2.1.1.1 to predict future BGLs. To make these predictions, the models use only raw features from the data, such as a recent history of BGLs, insulin, and the carbohydrate counts from recent meals. One of the three training scenarios the authors outlined in the paper, the "what-if" scenario, is a particularly interesting scenario due to its relationship with the meal and bolus recommendation problems. In this scenario, the BGL is predicted with the aid of "what-if" events, specifically meals and bolus that occur after the current time, but before the end of the prediction window (the current time + 30 or 60 minutes). This scenario is useful as it is essentially predicting BGLs given some future action intended by the user, such as how many carbs will be in a meal that they plan to eat soon, or how much insulin they plan to bolus in the new few minutes. This could provide people with T1D an estimate as to how much a meal or bolus will affect their BGL before taking the action. This is essentially the inverse of the research in this thesis, where instead of answering the question "What would my BGL be if I eat or bolus this much?", this research aims to answer the question "How much do I need to eat or bolus to get my Blood Glucose (BG) to this level?".

Given the state-of-the-art performance of the N-BEATS architecture for time series forecasting tasks Section 2.1.2.1, it is also a natural choice for the BGL prediction task. The authors of [27] proposed three important modifications to the original N-BEATS architecture to make it better suited for this task. The first, and potentially the most important of these modifications, was the introduction of an LSTM into each block to help account for the temporal nature of the blood glucose data. The second proposed modification was the inclusion of additional time series variables, specifically recent carb counts and bolus doses, for each block as context. Without this information, the model

would not be able to learn the effects that carbs and insulin have on a subject's BGLs, which would be a missed opportunity for improvement. Finally, the third proposed modification was restructuring the loss function to aid each block in a stack to learn better intermediate representations. With these improvements, the authors achieved top performance on the task of BGL prediction [2]

2.3 Insulin Recommendation Systems

Providing people with T1D the ability to preview their future BGLs would be of great help in improving the management of their BGLs. However, without the tools to inform a user how to correct their BGL if they are predicted to rise too high or sink too low, the usefulness of BGL prediction systems is fairly limited. Insulin recommendation systems provide the user with recommendations for how much insulin they should take to lower their BGL to the desired target in the near future. There have been a wide range of AI techniques employed to make these insulin recommendation systems such as case-based reasoning [25], K-nearest-neighbors methods [29], and neural networks [31]. Accuracy is paramount with insulin recommendation systems, as recommendations that are too high or too low can lead to hypoglycemia or hyperglycemia, respectively. Both can cause several long-term complications if not hastily corrected. Therefore, it is critical that insulin recommendation systems be as accurate as possible.

The bolus recommendation system introduced in [25] uses case-based reasoning to provide recommendations for the parameters to be used by a bolus calculator. In this system, cases consist of the meal scenario, the solution (the recommended bolus calculator parameters), and the outcome (the postprandial glucose excursion). When making a recommendation, the system searches through past cases to find a case that is similar to the current situation. If one such case is found, the bolus parameters used in the past case are recommended. If these parameters do not produce the desired outcome (i.e.,

the subjects postprandial glucose excursion is too high or too low) the original case's solution is revised by an expert. If no cases similar to the current case are found, a new case is created. This case-based reasoning approach allows the creation of a bolus recommendation system completely personalized to a specific person with T1D. However, this system is not learning how meals or boluses affect a person's BGL. It is recalling the bolus recommendations from past instances of similar situations, which have been revised by a clinical expert. While this can lead to very accurate and safe bolus recommendations, it requires an expert to look over the cases and revise the recommendations by hand on a regular basis. This is an expensive and time-consuming task. This also means that when a unique meal scenario that has not been seen before is encountered, the system lacks the ability to make an intelligent prediction.

The authors of [29] introduce an insulin recommendation system that provides a user with recommendations on how to improve their BGLs through adjustments in their insulin routines. This system uses the K-nearest-neighbors machine learning algorithm to provide up to four (out of 12 possible options) weekly recommendations for adjustments to a user's insulin dosages. This is done by looking at BGL, insulin, and physical activity from the user and classifying which types of recommendations they could benefit from. The goal of this system is to maximize the amount of time that a user spends with a BGL in a healthy range while minimizing the amount of time spent in the hypoglycemic range. The system may recommend the user to adjust factors such as their basal insulin rate, their carbs-to-insulin ratio, or their correction factor. While this system offers multiple different types of recommendations, they are adjustments that will have a positive effect on a subject's BGL in a longer-term. As such, it is not as flexible or dynamic as a system that can provide a user with recommendations for real time bolus dosages.

The work in [6] shows that deep learning can be used to personalize the calculation of bolus doses. The authors demonstrated that using a simple feed-forward neural

network to calculate bolus doses can outperform other modern methods for CGM-based personalized bolus calculators. Neural network models can make recommendations that are specific to an individual by not only considering factors such as insulin on board and target BGLs, but also subject-specific parameters such as body weight and insulin sensitivity. While this work shows the usefulness of neural networks for personalizing an individual's bolus recommendations, this approach does not consider any previous history of BGLs, meals, or boluses. Using these features could potentially allow a bolus advisor to make recommendations that are not only personalized for an individual but are personalized for an individual in a specific situation. This would likely result in even better recommendations.

In [31], the authors proposed a deep reinforcement learning based system for recommending bolus dosages for people with T1D. The general approach was to take an action (deliver a bolus) at the time of the meal given the state of the environment (values of various sensors in a glucose control systems) that would maximize the amount of time that a subject's BGL stayed in a healthy range while simultaneously minimizing the amount of time spent in the hypoglycemia range. Since the authors opted to use a reinforcement learning approach, they need an environment in which the models can repeatedly explore various bolus dosages in order to learn a bolus recommendation policy. The ideal environment would be provided by real individuals with T1D. However, this would be incredibly unsafe. Even with safety constraints in place, a bad recommendation could force a subject's BGLs into a hypoglycemic state. Since training on human subjects is infeasible, the authors used a simulator to train their model, namely the UVA/Padova simulator [19]. While simulated data is a more appropriate choice in this situation, it is unclear how well the learned policy would work on data from a real human subject.

The use of insulin recommendation systems can assist people with T1D with managing their BGLs. There are many different approaches and techniques being used to developing

these systems. Similar to BGL prediction systems, advances in artificial intelligence continue to enable the creation of improved systems for insulin recommendations. These systems may eventually allow for people with T1D to avoid hypoglycemia or hyperglycemia through accurate, safe, and reliable insulin recommendations.

3 Deep Learning Models for Carbohydrate and Bolus Recommendations

This chapter will expand on the general problem description from Section 1.1 and introduce in detail the situations commonly encountered in diabetes self-management that could benefit from the research reported in this thesis. Two neural network models will be introduced, as well as a two baseline models, and all will be evaluated extensively on the tasks of carbohydrate and bolus recommendation using examples extracted from the OhioT1DM dataset.

3.1 Recommendation Scenarios

It is assumed that blood glucose levels are measured at 5 minute intervals through a CGM system. It is also assumed that discrete deliveries of insulin (boluses) and continuous infusions of insulin (basal rates) are recorded. Subjects provide the timing of meals and estimates of the number of grams of carbohydrate in each meal. Given the available data up to and including the present (time t), the system aims to estimate how much a person should eat or bolus 10 minutes from now (time $t + 10$) such that their blood glucose will reach a target level τ minutes after that action (time $t + 10 + \tau$). A system that computes these estimates could then be used in the following three recommendation scenarios:

1. **Carbohydrate Recommendations**: Estimate the amount of carbohydrate C_{t+10} to have in a meal in order to achieve a target BG value $G_{t+10+\tau}$.

2. **Bolus Recommendations**: Estimate the amount of insulin B_{t+10} to deliver with a bolus in order to achieve a target BG value $G_{t+10+\tau}$.

3. **Bolus Recommendations given Carbohydrates**: Expecting that a meal with C_{t+20} grams of carbohydrate will be consumed 20 minutes from now, estimate the amount

of insulin B_{t+10} to deliver with a bolus 10 minutes before the meal in order to achieve a target BG value $G_{t+10+\tau}$.

These recommendation scenarios were designed to align with decision-making situations commonly encountered by people with type 1 diabetes. In particular, the corresponding recommendation systems would help an individual to estimate how much to eat or bolus for the purpose of raising or lowering their BGL (scenarios 1 and 2), as well as how much to bolus for a planned meal (scenario 3).

3.2 Recommendation Models

In the following sections, a number of baseline models and neural architectures are described, all implementing the three types of recommendations. The neural architectures use LSTM networks either in a standalone prediction model (Section 3.2.2) or integrated as basic repeating blocks in a deep residual network (Section 3.2.3). The models are trained on examples extracted from the OhioT1DM dataset [20], as explained in Chapter 4. Ideally, to match the intended use of the recommendation scenarios from Section 3.1, training examples should not have any extra meals or boluses in the prediction window $[t, t + 10 + \tau]$. Following the terminology from [22], these examples are called *inertial* examples. However, to benefit from a larger number of training examples, models are also trained and evaluated on a more general class of *unrestricted* examples, in which other bolus or meal events are allowed to appear in the prediction window. Correspondingly, experimental results for both inertial vs. unrestricted examples are presented in Section 5.2.

3.2.1 Baseline Models

Given training data containing time series of blood glucose levels, meals with their carbohydrate intake, and boluses with their corresponding insulin dosages, two baselines are defined as follows:

1. **Global average**: For the carbohydrate recommendation scenario, the average number μ of carbs over all of the meals in the subject's training data is computed and used as the estimate for all future predictions for that subject, irrespective of the context of the example. Analogously, for the bolus and bolus given carbs recommendation scenarios, μ is the average amount of insulin dosage over all boluses in the subject's training data. This is a fairly simple baseline, as it predicts the same average value for every test example for a particular subject.

2. **Time-of-Day (ToD) average**: In this ToD dependent baseline, an average number of carbs or an average amount of bolus insulin is computed for each of the following five time windows during a day:

   - 12am-6am: μ_1 = early breakfast / late snacks.
   - 6am-10am: μ_2 = breakfast.
   - 10am-2pm: μ_3 = lunch.
   - 2pm-6pm: μ_4 = dinner.
   - 6pm-12am: μ_5 = late dinner / post-dinner snacks.

   The average for each ToD interval is calculated over all of the meals or boluses appearing in the corresponding time frame in the subject's training data. At test time, to make a recommendation for time $t + 10$, the ToD interval that contains $t + 10$ is determined and the corresponding ToD average is used as the recommendation.

Given sufficient historical data, the ToD baseline is expected to perform well for individuals who tend to eat very consistently and have regular diets. However, it is expected to perform poorly for individuals who have a lot of variation in their diets.

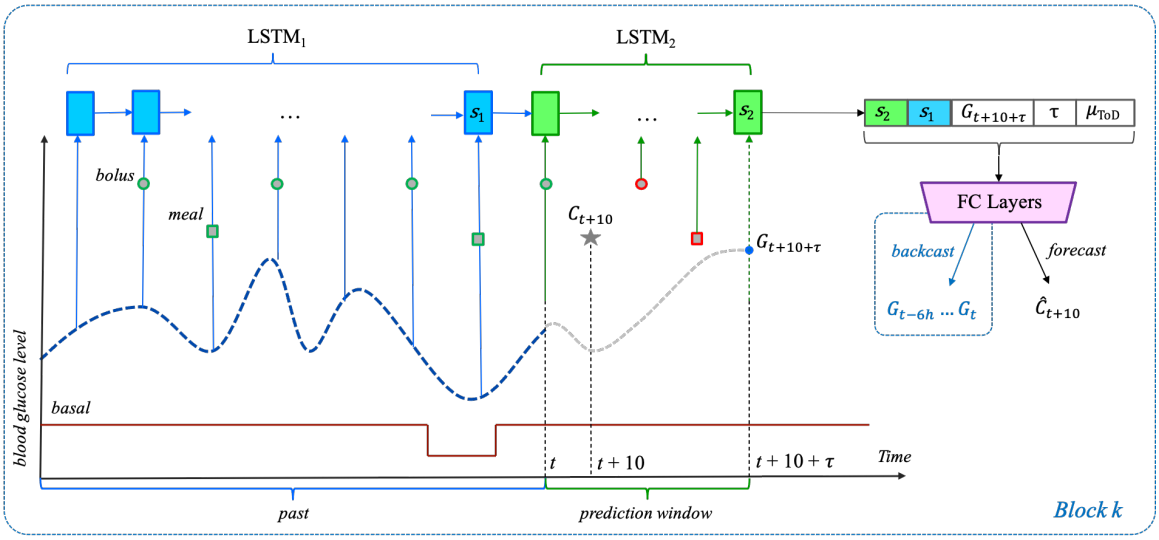

Figure 3.1: The general neural network architecture for the carbohydrate recommendation scenario. The dashed blue line in the graph represents a subject's BGL, while the solid brown line represents the basal rate of insulin. The gray star represents the meal at time $t + 10$. The other meals are represented by squares, and boluses are represented by circles. Meals and boluses with a red outline cannot appear in *inertial* examples, but are allowed in *unrestricted* examples. The blue units in LSTM_1 receive input from different time steps in the past. The green units in LSTM_2 receive input from the prediction window. The purple trapezoid represents the 5 fully connected layers, whereas the output node at the end computes the prediction.

3.2.2 LSTM Architectures

While simple to compute and use at test time, the two baselines are likely to give suboptimal performance, as their predictions ignore the history of BGL values, insulin (boluses and basal rates), and meals, all of which could significantly modulate the effect a future meal and/or bolus might have on the BGL. To utilize this information, the general LSTM-based network architectures shown in Figure 3.1 for carb recommendation and Figure 3.2 for bolus recommendation were developed. The first component in each architecture is a recurrent neural network instantiated using LSTM cells [14], which is run over the previous 6 hours of data, up to and including the present time t. At each time step (every 5 minutes), this LSTM_1 network takes as input the BGL, the carbs, and the insulin

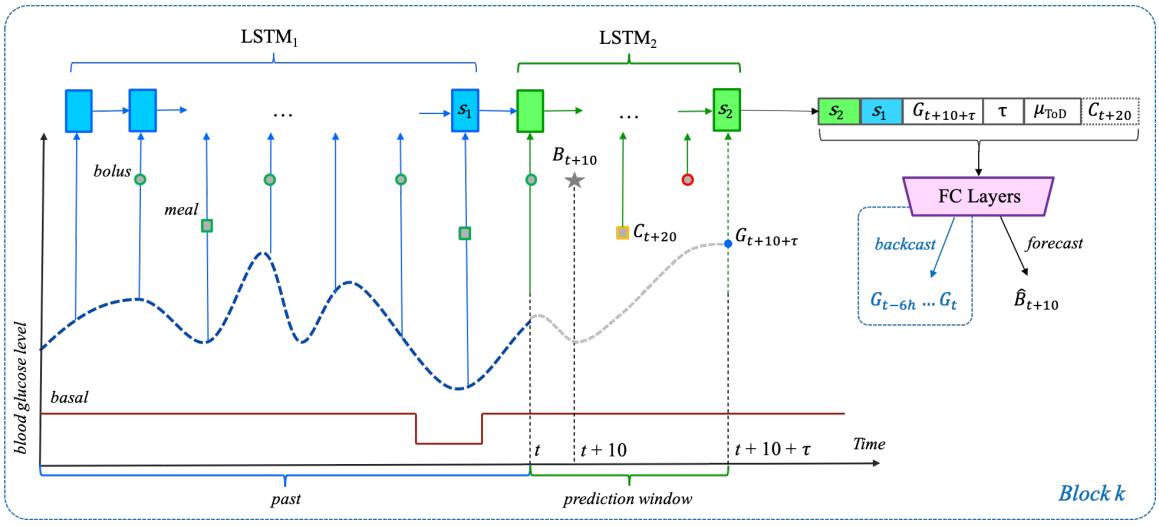

Figure 3.2: The general neural network architecture for the bolus and bolus given carbs recommendation scenarios. The architecture itself is similar to that shown in Figure 3.1. The gray star now represents the bolus at time $t + 10$. For the bolus recommendation scenario, the events outlined in red or orange are not allowed in *inertial* examples. However, in the bolus given carbs scenario, the meal event C_{t+20} shown with the yellow outline is an important part of each example, be it inertial or unrestricted. As such, in this scenario, the dashed C_{t+20} becomes part of the input to the FCN.

dosages recorded at that time step. While sufficient for processing *inertial* examples, the same LSTM cannot be used to process events that may appear in the prediction window $(t, t + 10 + \tau)$ of *unrestricted* examples, because BGL values are not available in the future. Therefore, when training on unrestricted examples, the final state computed by the LSTM_1 model at time t is projected using a linear transformation and used as the initial state for a second LSTM network, LSTM_2, that is run over all the time steps in the prediction window $(t, t + 10 + \tau)$. The final state computed by LSTM_1 (for inertial examples) is appended to the final state computed by LSTM_2 (for unrestricted examples) and is then used as input to a Fully Connected Network (FCN) whose output node computes an estimate of the carbs or bolus insulin at time $t + 10$. In addition to the LSTM final state(s), the input to the N-BEATS contains the following features:

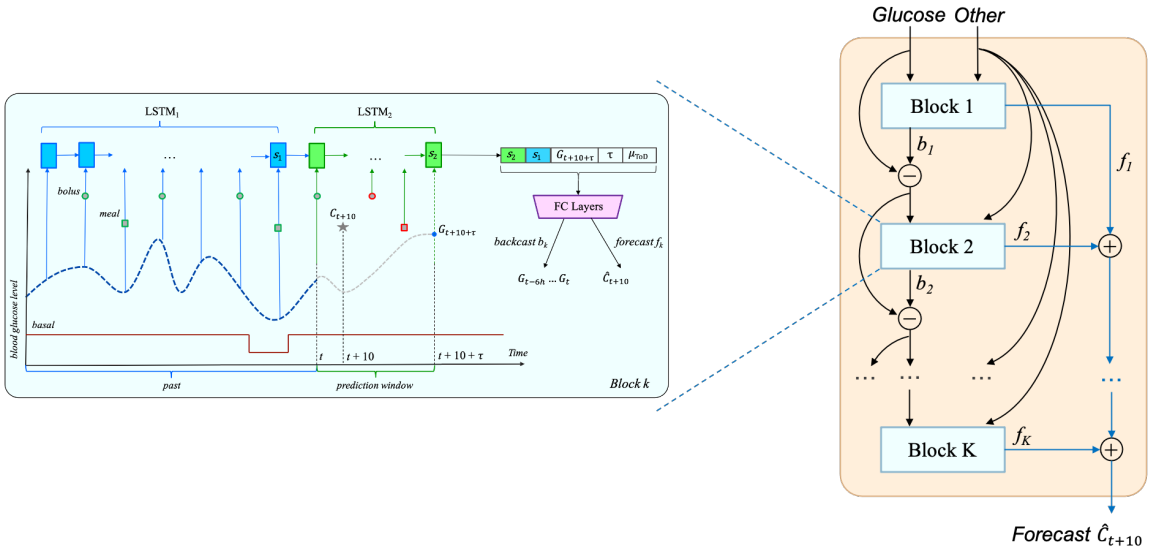

Figure 3.3: The N-BEATS inspired deep residual architecture for carbohydrate recommendation. A similar architecture is used for bolus and bolus given carbs recommendations.

- The target blood glucose level $\tau + 10$ minutes into the future, i.e., $G_{t+10+\tau}$.
- The prediction horizon τ.
- The ToD average for the time frame that contains $t + 10$.
- For the bolus given carbs scenario only, the planned amount C_{t+20} of carbohydrate becomes part of the input, too.

Each LSTM uses vectors of size 32 for the states and gates, whereas the N-BEATS is built with up to 5 hidden layers, each consisting of 64 ReLU neurons, and one linear output node. Note that by using the final state of LSTM_1 to initialize LSTM_2, the latter's final state should theoretically be able to capture any useful information that is represented in the final state of LSTM_1, which may call into question the utility of concatenating the two final states. This architectural decision is nevertheless supported empirically through

evaluations on the validation data, which show improvements in prediction performance when both states are used (Section 5.1.3).

3.2.3 Deep Residual Networks

The N-BEATS architecture described in Section 2.1.2.1 has been shown to obtain state-of-the-art performance on a wide range of time series prediction tasks [23], which suggests that it can serve as a model of choice for BGL prediction, too. However, in BGL prediction, time series of variables other then the primary blood glucose are also available. Correspondingly, the Rubin-Falcone et al. [27] changed the N-BEATS block architecture to also use as input secondary, sparse variables such as meals and bolus insulin, while still backcasting only on the primary forecasting variable, blood glucose. To account for the temporal nature of the input, the fully connected structure of the basic N-BEATS block was replaced with LSTM networks, followed by one fully connected layer whose output was split into the backcast and forecast vector. Additional per-block forecast and backcast loss terms were also added to provide more supervision.

The deep residual network from [27] has been adapted to perform carb or bolus recommendations by using the LSTM-based architecture from Section 3.2.2 to instantiate each block in the stack, as shown in Figure 3.3. Compared to the architecture from [27], the most significant differences are:

1. The use of a chain of two LSTM networks in each block.

2. The inclusion of additional inputs to the fully connected layers, i.e. the target BG level, the time horizon, and the ToD average.

3. While backcasting is still done for blood glucose, forecasting is done for carbs or bolus, depending on the recommendation scenario.

While Oreshkin et al. [23] used 30 blocks and Rubin-Falcone et al. [27] used 10 blocks, the validation experiments for the recommendation tasks showed that the most effective deep residual architecture uses only up to 5 blocks, depending on the recommendation scenario (Section 5.1.3).

4 The OhioT1DM Dataset for Recommendation Examples

To evaluate the proposed recommendation models, training and test examples are created using data collected from 12 subjects with type 1 diabetes that is distributed with the OhioT1DM dataset [21]. The 12 subjects are partitioned in two subsets as follows:

1. **OhioT1DM 2018**: This is the first part of the dataset, containing data collected from 6 patients. It was used for the 2018 Blood Glucose Level Prediction (BGLP) challenge [1].

2. **OhioT1DM 2020**: This is the second part of the dataset, containing data collected from 6 additional patients. It was used for the 2020 BGLP challenge [2].

Time series containing the basal rate of insulin, boluses, meals, and BGL readings were collected over 8 weeks, although the exact number of days varies from subject to subject. Insulin and BGL data was automatically recorded by each subject's insulin pump. Meal data was collected in two different ways. Subjects self reported meal times and estimated carbs via a smartphone interface. Subjects also entered estimated carbs into a bolus calculator when bolusing for meals, and this data was recorded by the insulin pump.

4.1 The Bolus Wizard

To determine their insulin dosages, the subjects in the OhioT1DM study used a bolus calculator, or Bolus Wizard (BW), which was integrated in their insulin pumps. They used it to calculate the bolus amount before each meal as well as when using a bolus to correct for hyperglycemia. To use the BW, a subject enters their current blood glucose level and, if eating, their estimated number of grams of carbohydrate. To calculate a recommended insulin dosage, the BW uses this input from the subject, plus the amount of active insulin the subject already has in their system, along with the following three pre-programmed, patient-specific parameters:

1. The carb ratio, which indicates the number of grams of carbohydrate that are covered by a unit of insulin.

2. The insulin sensitivity, which tells how much a unit of insulin is expected to lower the subject's blood glucose level.

3. The target blood glucose range, which defines an individual's lower and upper boundaries for optimal blood glucose control.

All three parameters may vary, for the same individual, throughout the day and over time[1]. Given this input and these parameters, the BW calculates the amount of insulin the subject should take to maintain or achieve a blood glucose level within their target range. The calculation is displayed to the subject as a recommendation, which the subject may then accept or override.

Based on the inputs and the patient-specific parameters described above, the BW uses a deterministic formula to calculate the bolus amount before each meal. As such, when trained in the bolus given carbs recommendation scenario, there is the risk that the deep learning models introduced in Section 3.2 might simply learn to reproduce this deterministic dependency between bolus and carbs, while ignoring the target BG level that is used as input. However, this is not the case in our experimental settings, for the following reasons:

- The machine learning models do not have access to any of the three patient-specific parameters above, which can change throughout the day and over time, and which are set based on advice from a health care professional.

- The BW uses a fixed target BG range depending on the time of day, whereas the target in the recommendation scenarios is a more specific BG level, to be achieved at a specific time in the near future.

[1] https://www.medtronicdiabetes.com/loop-blog/4-features-of-the-bolus-wizard

- The amount of insulin calculated by the BW is only a recommendation, which is often overridden by subjects. An analysis of the OhioT1DM dataset was conducted in order to count the number of times the amount of insulin that was actually delivered was different from the BW recommendation. The analysis revealed that, of all the times that the BW was used, its recommendation was overridden for about a fifth of the boluses. Furthermore, there are subjects in the dataset who often did not use the BW (540 and 567), or who chose to not use the BW at all (596).

Therefore, the machine learning models will have to go beyond using solely the carbohydrate amount in the intended meal. In order to fit the bolus recommendation examples, they will need to learn the impact that a bolus has on the target BG level for the specified prediction horizon, taking into account the amount of carbohydrate in the meal as well as the history of carbs, insulin, and BG levels. This data driven approach to bolus recommendation relieves the physician from the cognitively demanding task of regularly updating parameters such as the carb ratio and the insulin sensitivity, which often requires multiple fine tuning steps. In contrast, any relevant signal that is conveyed through the carb ratio and insulin sensitivity is expected to be learned by the machine learning models from the data.

4.2 Pre-processing of Meals and BG Levels

While exploring the data, it was observed that self-reported meals and their associated boluses were in unexpected temporal positions relative to each other. For many meals, patients recorded a timestamp in the smartphone interface that preceded the corresponding bolus timestamp recorded in the insulin pump. This was contrary to what was recommended to the subjects by their physicians, which was to bolus shortly before the meal, and no more than 15 minutes prior to the meal. This discrepancy is likely due to subjects reporting incorrect meal times in the smartphone interface.

To correct the meal events, the data provided to the BW was used in a pre-processing step that changed the timestamp of each meal associated with a bolus to be exactly 10 minutes after that bolus. For these meals, the number of carbs provided to the BW was used, which is likely to be more accurate than the estimate provided by the subject through the smartphone interface. To determine if the self-reported meal event was associated with a bolus having non-zero carb input, the meal that was closest to the bolus was searched within one hour before or after it. In case there were two meals that were equally close to the bolus, the one for which the number of carbs reported in the smartphone interface was closest to the number of carbs entered into the BW was selected. If no self-reported meal was found within one hour of the bolus, it was assumed that the subject forgot to log their meal on the smartphone interface. As such, a meal was added 10 minutes after the bolus, using the amount of carbs specified in the BW for that bolus. Ablation results reported in Section 5.1.2 show that this pre-processing of meal events leads to significantly more accurate predictions, which further justifies the pre-processing.

All gaps in BGL data are filled in with linearly interpolated values. However, examples that meet any of the following criteria are filtered out:

1. The BGL target is interpolated.
2. The BGL at present time t is interpolated.
3. There are more than 2 interpolated BGL measurements in the one hour of data prior to time t.
4. There are more than 12 interpolated BGL measurements in the 6 hours of data prior to time t.

4.3 Mapping Prototypical Recommendation Scenarios to Datasets

According to the definition given in Section 3.1, the carbohydrate recommendation scenario refers to estimating the amount of carbohydrate C_{t+10} to have in a meal in order to achieve a target BG value $G_{t+10+\tau}$. This is done by using the history of data up to and including the present time t. However, many carbohydrate intake events C_{t+10} are regular meals, which means that they are preceded by a bolus event at time t. Since in the carbohydrate recommendation scenario, the cases where the subject eats in order to correct or prevent hypoglycemia are particularly of interest, two separate datasets for carbohydrate prediction were created:

1. Carbs$^{(\pm b)}$: this will contain examples for all carbohydrate intake events, with $(+b)$ or without $(-b)$ an associated bolus.

2. Carbs$^{(-b)}$: this will contain examples only for carbohydrate intake events without $(-b)$ an associated bolus.

Most of the Carbs$^{(-b)}$ examples are expected to happen in one of three scenarios: (1) when correcting for hypoglycemia; (2) before exercising; and (3) when having a bedtime snack to prevent nocturnal hypoglycemia. Given that they are only a small portion of the overall carbohydrate events, Section 5.2 presents the results for both Carbs$^{(\pm b)}$ and Carbs$^{(-b)}$ recommendation scenarios.

Furthermore, mirroring the two bolus recommendation scenarios introduced in Section 3.1, the following dataset notation is used:

1. Bolus$^{(\pm c)}$: this will contain examples for all bolus events, with $(+c)$ or without $(-c)$ an associated carbohydrate intake.

2. Bolus$^{(+c)}$: this will contain examples only for the bolus events with $(+c)$ an associated carbohydrate intake.

The three major recommendation scenarios introduced in Section 3.1 can then be mapped to the corresponding datasets as follows:

1. **Carbohydrate Recommendations**: Estimate the amount of carbohydrate C_{t+10} to have in a meal in order to achieve a target BG value $G_{t+10+\tau}$.

    - Carbs$^{(-b)}$, inertial: this reflects the prototypical scenario where a carbohydrate intake is recommended to correct or prevent hypoglycemia.

2. **Bolus Recommendations**: Estimate the amount of insulin B_{t+10} to deliver with a bolus in order to achieve a target BG value $G_{t+10+\tau}$.

    - Bolus$^{(\pm c)}$, inertial: this reflects the prototypical scenario where a bolus is recommended to correct or prevent hyperglycemia. Because in the inertial case a carb event cannot appear after the bolus, this could also be denoted as Bolus$^{(-c)}$.

3. **Bolus Recommendations given Carbohydrates**: Expecting that a meal with C_{t+20} grams of carbohydrate will be consumed 20 minutes from now, estimate the amount of insulin B_{t+10} to deliver with a bolus 10 minutes before the meal in order to achieve a target BG value $G_{t+10+\tau}$.

    - Bolus$^{(+c)}$, inertial: this reflects the prototypical scenario where a bolus is recommended before a meal.

4.4 Carbohydrate and Bolus Statistics

Table 4.1 shows the number of carbohydrate events in each subject's pre-processed data, together with the minimum, maximum, median, average, and standard deviation for the number of carbs per meal. Overall, the average number of carbs per meal is between 22 and 69, with the exception of subjects 570 and 544 whose meal averages and standard

deviations are significantly larger. Table 4.2 shows similar statistics for boluses and their dosages, expressed in units of insulin. Overall, the number of boluses is more variable than the number of meals. There is also a fairly wide range of average bolus values in the data, with subject 567 having a much higher average than other subjects. It is also interesting to note that subject 570, who had the largest average carbs per meal, had more than twice the number of boluses than any other subject while at the same time having the lowest average bolus. Subject 570 also used many dual boluses, which were not used as prediction labels because the scope of the project covers only recommendations for regular boluses.

4.5 From Meals and Bolus Events to Recommendation Examples

In all recommendation scenarios, the prediction window ranges between the present time t and the prediction horizon $t + 10 + \tau$. For the carbohydrate or bolus recommendation scenarios, the meal or the bolus is assumed to occur at time $t + 10$. For the bolus given carbs scenario, the bolus occurs at time $t + 10$ and is followed by a meal at time $t + 20$, which matches the pre-processing of the meal data. For evaluation purposes, τ is set to values between 30 and 90 minutes with a step of 5 minutes, i.e, $\tau \in \{30, 35, 40, ..., 90\}$ for a total of 13 different values. As such, each meal/bolus event in the data results in 13 recommendation examples, one example for each value of τ. While all 13 examples use the same value for the prediction label, e.g., B_{t+10} for bolus prediction, they will differ in terms of the target BG feature $G_{t+10+\tau}$ and the τ feature, both used directly as input to the FC layers in the architectures shown in Figures 3.1 and 3.2. For the bolus given carbs scenario, the 13 examples are only created when there is a meal that had a bolus delivered 10 minutes prior. Due to the way the data is pre-processed, it is guaranteed that if a meal had a bolus associated with it, the bolus will be exactly 10 minutes before the meal.

Table 4.3 shows the number of *inertial* examples for 5 prediction horizons, as well as the total over all 13 possible prediction horizons. Table 4.4 shows the number of

Table 4.1: Per subject and total meal and carbohydrate per meal statistics: Minimum, Maximum, Median, Average, and Standard Deviation (StdDev). Carbs$^{(\pm b)}$ refers to all carbohydrate intake events; Carbs$^{(-b)}$ refers to carbohydrate intakes without a bolus. Statistics are shown for the 2018 subset, the 2020 subset, and for the entire OhioT1DM dataset.

			Carbs Per Meal				
Subject	Carbs$^{(\pm b)}$	Carbs$^{(-b)}$	Minimum	Maximum	Median	Average	StdDev
559	215	83	8.0	75.0	30.0	35.5	15.5
563	225	28	5.0	84.0	31.0	33.8	18.0
570	174	39	5.0	200.0	115.0	106.1	41.5
575	297	122	1.0	110.0	40.0	40.0	22.0
588	268	73	2.0	60.0	20.0	22.7	14.6
591	264	60	3.0	77.0	28.0	31.5	14.1
2018 Total	1443	405	1.0	200.0	33.0	41.5	32.7
540	234	14	1.0	110.0	40.0	50.2	29.8
544	206	41	1.0	175.0	60.0	68.7	36.3
552	271	25	3.0	135.0	26.0	36.7	29.3
567	207	5	20.0	140.0	67.0	67.0	21.5
584	233	44	15.0	78.0	60.0	54.6	11.6
596	300	277	1.0	64.0	25.0	25.1	14.0
2020 Total	1451	406	1.0	175.0	42.0	48.2	29.5
Combined Total	2894	811	1.0	200.0	39.0	44.9	31.3

unrestricted examples. Since the same number of unrestricted examples are available for every prediction horizon, only the totals are shown. The only exceptions would be if an event was near the end of a subject's data and the prediction horizon $t + 10 + \tau$ goes past the end of the dataset for some value of τ.

Table 4.2: Per subject and total boluses and insulin units statistics: Minimum, Maximum, Median, Average, and Standard Deviation (StdDev). Bolus$^{(\pm c)}$ refers to all bolus events; Bolus$^{(+c)}$ refers to bolus events associated with a meal. Statistics are shown for the 2018 subset, the 2020 subset, and for the entire OhioT1DM dataset.

			Insulin Per Bolus				
Subject	Bolus$^{(\pm c)}$	Bolus$^{(+c)}$	Minimum	Maximum	Median	Average	StdDev
559	186	132	0.1	9.3	3.6	3.7	1.9
563	424	197	0.1	24.7	7.8	8.0	4.2
570	1,345	132	0.2	12.1	1.3	1.8	2.1
575	271	175	0.1	12.8	4.4	4.1	3.0
588	221	195	0.4	10.0	3.5	4.3	2.3
591	331	204	0.1	9.4	2.9	3.1	1.8
2018 Total	2,758	1,035	0.1	24.7	1.9	3.5	3.4
540	521	220	0.1	11.4	2.0	3.0	2.8
544	264	149	0.7	22.5	5.0	6.5	4.9
552	426	246	0.1	16.0	2.8	3.9	3.3
567	366	202	0.2	25.0	11.4	12.0	5.8
584	311	188	0.1	16.2	9.1	7.3	3.1
596	230	0	0.2	7.6	3.3	3.0	1.5
2020 Total	2,118	1,169	0.1	25.0	4.0	5.8	5.0
Combined Total	4,876	2,204	0.1	25.0	2.9	4.5	4.3

For the carbohydrate and bolus given carbs recommendation scenarios, the gap between the number of *inertial* and *unrestricted* examples is not very large, as most examples qualify as inertial examples. However, in the bolus recommendation scenario, there is a very sizable gap between the number of inertial vs. unrestricted examples. This is because a significant number of boluses are associated with meals, and since these meals are timestamped to be 10 minutes after the bolus, the result is that a bolus at time $t+10$ will

Table 4.3: *Inertial* (*I*) examples by recommendation scenario and prediction horizon. Carbs$^{(\pm b)}$ refers to all carbohydrate intake events; Carbs$^{(-b)}$ refers to carbohydrate intakes without a bolus.

	Carbs$^{(\pm b)}$ recommendation				Carbs$^{(-b)}$ recommendation			
Horizon	Training	Validation	Testing	Total *I*	Training	Validation	Testing	Total *I*
$\tau = 30$	1,192	340	331	1,863	265	53	40	358
$\tau = 45$	1,156	334	321	1,811	255	51	40	346
$\tau = 60$	1,121	318	315	1,754	243	50	40	333
$\tau = 75$	1,057	301	293	1,651	226	44	34	304
$\tau = 90$	975	279	278	1,532	200	40	31	271
All 13 horizons	14,343	4,103	4,007	22,453	3,100	620	486	4,206

	Bolus$^{(\pm c)}$ recommendation				Bolus$^{(+c)}$ recommendation			
Horizon	Training	Validation	Testing	Total *I*	Training	Validation	Testing	Total *I*
$\tau = 30$	461	160	143	764	856	267	271	1,394
$\tau = 45$	416	142	124	682	833	259	258	1,350
$\tau = 60$	368	124	104	596	816	253	249	1,318
$\tau = 75$	303	102	96	501	790	243	243	1,276
$\tau = 90$	271	90	86	447	743	234	229	1,206
All 13 horizons	4,732	1,606	1,423	7,761	10,514	3,269	3,249	17,032

be associated with a meal at time $t + 20$. Therefore, for preprandial boluses at $t + 10$, the meal at time $t + 20$ will prohibit the creation of inertial recommendation examples, because by definition inertial examples do not allow the presence of other events in the prediction window $(t, t + 10 + \tau)$.

Table 4.4: *Unrestricted* (U) examples by recommendation scenario, also showing, in the last column, the total number of non-inertial ($U - I$) examples. Carbs$^{(\pm b)}$ refers to all carbohydrate intake events; Carbs$^{(-b)}$ refers to carbohydrate intakes without a bolus.

Scenario	Training	Validation	Testing	Total U	Total $U - I$
Carbs$^{(\pm b)}$	17,937	5,106	4,943	27,986	5,533
Carbs$^{(-b)}$	4,140	853	624	5,617	1,411
Bolus$^{(\pm c)}$	19,640	6,279	6,136	32,055	24,294
Bolus$^{(+c)}$	12,052	3,784	3,816	19,652	2,620

5 Experimental Evaluation

This chapter will introduce the methods used to train and evaluate the models described in Chapter 3 for the tasks of carbohydrate and bolus recommendations, according to the three scenarios detailed in Section 3.1. The experimental methodology is introduced, and experimental results are reported and discussed.

5.1 Experimental Methodology

For each of the 12 subjects in the dataset, their time series data is split into three sets, as follows:

- *Testing*: the last 10 days of data.
- *Validation*: the 10 days of data preceding the testing portion.
- *Training*: the remainder of the data, around 30 days.

The blood glucose, carbs, and insulin values are all scaled to be between $[0, 1]$ by using maximum and minimum values computed over training data. When computing the performance metrics at test time, the predicted values are scaled back to the original range. The neural architecture is trained to minimize the mean squared error between the actual event (meal or bolus) value recorded in the training data and the estimated value computed by the output node of the fully connected layers in the LSTM models, or by the accumulated forecasts in the N-BEATS architecture. The Adam [16] variant of gradient descent is used for training, with the learning rate and mini-batch size being tuned on the validation data. In an effort to avoid overfitting, dropout and early stopping with a patience of 10 epochs are used in all experiments.

Before training a personalized model for a specific subject, a generic model is first pre-trained on the union of all 12 subjects' training data. The generic model is then fine tuned separately for each individual subject, by continuing training on that subject's training data

only. The pre-training allows the model parameters to be in a better starting position before fine tuning, allowing faster and better training. The learning rate and batch size are tuned for each subject on their validation data. For each subject, the results are aggregated over 10 models that are trained with different seedings of the random number generators.

The metrics used to evaluate the models are Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE). Two scores are reported for each of the LSTM-based and N-BEATS-based recommendation models:

1. The ⟨**model**⟩**.mean** score calculates the average RMSE and MAE on the testing data across the 10 models trained for each subject, and then averages these scores across all subjects.

2. The ⟨**model**⟩**.best** score instead selects for each subject the model that performed best in terms of MAE on the validation data, out of the 10 models trained for that subject. The RMSE and MAE test scores are averaged over all subjects.

Two sets of models were trained for each recommendation scenario: a set of models was trained and evaluated on *inertial* examples and a set was trained and evaluated on *unrestricted* examples.

5.1.1 Subject Selection for Testing in Each Recommendation Scenario

While using both the 2018 and 2020 subsets of the OhioT1DM Dataset [20, 21] provides us with data from 12 total subjects, not all 12 can be used in each scenario, due to insufficient examples in their respective development or test subsets. The subjects whose data was used or not at test time are listed below for each scenario, together with a justification:

- *Carbs*$^{(\pm b)}$ *Recommendation*: Subjects 567 and 570 were left out at test time. Subject 567 had 0 meal events in the testing portion of their data. Subject 570 frequently used

dual boluses; as such, there were very few inertial examples for this subject at all. Of the few inertial examples that were available, 0 were in the testing or validation portions of the data.

- *Carbs*$^{(-b)}$ *Recommendation*: Due to the limited number of examples for this scenario, models were trained and evaluated only for the subjects whose data contained at least 50 carb events with no associated bolus. These are subjects 559, 575, 588, and 591. While subject 596 also had a sufficient number of carb events, it was discovered that all carbohydrate inputs for their BW were 0. As a consequence of this missing data, it cannot be determined which boluses were used for BGL correction, and which were used to cover meals. Therefore, subject 596 cannot be used in this scenario.

- *Bolus*$^{(\pm c)}$ *Recommendation*: Subjects 544 and 567 were left out at test time. Subject 544 had few inertial examples overall, and 0 in the validation portion of the data. This is because the vast majority of bolus events in their data was used in conjunction with a meal. Similar to the carbohydrate recommendation scenario, subject 567 was not used in this scenario because of the lack of meal events in their test data. The missing meal data would make the bolus recommendation results for this subject unrealistic and also indistinguishable between the inertial and unrestricted cases.

- *Bolus*$^{(+c)}$ *Recommendation*: Subjects 567, 570, and 596 were left out at test time. As explained for other scenarios above, subject 567 had 0 meals in the test portion of their data. For subject 570, there were 0 inertial examples in the test portion. As explained for the Carbs^{-b} recommendation scenario, due to missing BW data, for subject 596 it cannot be determined which boluses were used for BGL correction, and which were used to cover meals, so their data cannot be used in this scenario, either.

Irrespective of which subjects are used at test time, the data from all 12 patients is used for pre-training purposes in each recommendation scenario. Furthermore, the set of subjects stays consistent between the inertial and unrestricted cases for any given recommendation scenario.

5.1.2 Evaluating the Impact of Pre-processing of Meals

To determine the utility of the pre-processing of meals procedure introduced in Section 4.2, N-BEATS-based models were trained and evaluated for the carbohydrate recommendation scenario Carbs$^{(\pm b)}$ using the original data vs. using the pre-processed data. When training on pre-processed data, two development results are reported in Table 5.1: when evaluating on all the pre-processed meals in the development data (pre^{+}) vs. evaluating only on meals that were not added during pre-processing (pre^{-}). The results show that in both cases the pre-processing of meals leads to statistically significant improvements in RMSE and MAE. Pre-processing of meals also benefits the bolus recommendation scenario, as shown in Table 5.2. These results can be seen as further evidence of the fact that the meal timestamps recorded in the smartphone interface are unreliable and that meal times should instead be anchored to the bolus timestamps recorded by the BW, as done in the pre-processing procedure.

5.1.3 Tuning the Architecture and the Hyper-parameters

Table 5.3 show the results of the LSTM- and N-BEATS-based models, with vs. without using the final state produced by the LSTM_1 component as input to the fully connected network. The results show that using the final state from LSTM_1 directly as input leads to a substantial improvement for the carbohydrate recommendation scenario Carbs$^{(\pm b)}$, while maintaining a comparable performance for the bolus recommendation scenario. Consequently, in all remaining experiments the architecture is set to use the final state of LSTM_1 as input to the Fully Connected (FC) layers.

Table 5.1: Results with pre-processing of meals (pre) vs. original raw data for meal events (raw), for the carbohydrate recommendation scenario Carbs$^{(\pm b)}$ on unrestricted examples. pre^{+} refers to using all pre-processed meals (shifted original meals and added meals), whereas pre^{-} does not use meals added by the pre-processing procedure. The symbol † indicates a p-value < 0.03 when using a one-tailed t-test to compare against the results without pre-processing (raw).

	Pre-processing			
	Train	Devel	RMSE	MAE
N-BEATS.mean	raw	raw	13.42	10.32
	pre^{+}	pre^{-}	†9.38	†6.59
	pre^{+}	pre^{+}	†**8.84**	†**6.16**
N-BEATS.best	raw	raw	12.32	9.28
	pre^{+}	pre^{-}	†8.48	†5.90
	pre^{+}	pre^{+}	†**8.12**	†**5.53**

Table 5.2: Results with pre-processing of meals (pre) vs. original raw data for meal events (raw), for the Bolus$^{(\pm c)}$ recommendation scenario on unrestricted examples. All meals (shifted or added) are used for the pre-processed data. The symbol † indicates a p-value < 0.01 when using a one-tailed t-test to compare against the results without pre-processing (raw).

	Pre-processing	RMSE	MAE
N-BEATS.mean	raw	1.85	1.41
	pre	†**1.30**	†**0.92**
N-BEATS.best	raw	1.81	1.32
	pre	†**1.22**	†**0.84**

In the original N-BEATS model of Oreshkin et al. [23], the backcast and forecast outputs of each block are produced as the result of two separate fully connected layers. In the block architecture shown in Figures 3.1, 3.2, and 3.3 however, the *FC Layers*

Table 5.3: Performance of the LSTM- and N-BEATS-based models, with (+) and without (−) the final state s_1 of LSTM_1 as part of the input to the FC Layers.

LSTM.mean		RMSE	MAE
$\text{Carbs}^{(\pm b)}$	$-s_1$	10.14	7.56
	$+s_1$	**8.99**	**6.57**
$\text{Bolus}^{(\pm c)}$	$-s_1$	**1.33**	**0.97**
	$+s_1$	1.41	1.03

N-BEATS.mean		RMSE	MAE
$\text{Carbs}^{(\pm b)}$	$-s_1$	10.27	7.58
	$+s_1$	**8.84**	**6.16**
$\text{Bolus}^{(\pm c)}$	$-s_1$	1.33	**0.85**
	$+s_1$	**1.30**	0.92

component uses just one final fully connected layer to produce both backcast and forecast values. The results in Table 5.4 show that, overall, using a joint final layer is competitive or better than using separate layers.

Table 5.4: N-BEATS-based model results, with a *separate* vs. *joint* final fully connected layer for computing backcast and forecast values.

N-BEATS.mean		RMSE	MAE
$\text{Carbs}^{(\pm b)}$	*separate*	**8.77**	6.48
	joint	8.84	**6.16**
$\text{Bolus}^{(\pm c)}$	*separate*	1.32	0.94
	joint	**1.30**	**0.92**

For each prediction scenario, the hyper-parameters for both the LSTM-based and N-BEATS-based models were tuned on development data. The inertial and unrestricted models are tuned independent of each other. The learning rate was tuned by monitoring the learning curves, using values between 0.0002 [27] and 0.1. After multiple experiments, a fixed learning rate of 0.001 was observed to give the best results on development data in all scenarios. The number of blocks in N-BEATS, the number of FC layers in the LSTM, and the dropout rate were then tuned in that order. The number of N-BEATS blocks was

selected from {1, ..., 10}, the number of layers was selected from {1, 2, 3, 4, 5}, whereas the dropout rate was tuned with values from {0, 0.1, 0.2, 0.3, 0.4 0.5}. The tuned values are shown in Table 5.5 for the LSTM models and Table 5.6 for the N-BEATS models. Overall, the LSTM-based models worked best with only 2 or 3 fully connected layers in all scenarios, whereas the N-BEATS-based models worked best with 4 or 5 fully connected layers. The tuned number of blocks in the N-BEATS-based models varied between 3 and 5, depending on the scenario and the unrestricted vs. inertial case. The tuned dropout rates varied a lot between scenarios for the LSTM-based models, with rates ranging from 0 to 0.5, whereas the tuned rates for N-BEATS-based models varied between 0.2 and 0.5.

Table 5.5: Tuned hyper-parameters for the LSTM-based models.

		Hyper-Parameters	
Scenario	Examples	FC Layers	Dropout
Carbs$^{(\pm b)}$	Inertial	3	0.1
	Unrestricted	3	0.1
Bolus$^{(\pm c)}$	Inertial	3	0.0
	Unrestricted	2	0.3
Bolus$^{(+c)}$	Inertial	2	0.2
	Unrestricted	2	0.5

The size of the LSTM state was tuned to 32, whereas the size of each fully connected layer was tuned to 64, which is substantially smaller than the hidden size of 512 used in the original N-BEATS model [23]. For the carbohydrates without bolus scenario Carbs$^{(-b)}$, due to the much smaller number of examples, the number of units in the LSTM networks and fully connected layers were reduced by a factor of 2. The same hyper-parameters that

Table 5.6: Tuned hyper-parameters for the N-BEATS-based models.

		Hyper-Parameters		
Scenario	Examples	Blocks	FC Layers	Dropout
$\text{Carbs}^{(\pm b)}$	Inertial	5	2	0.3
	Unrestricted	3	3	0.3
$\text{Bolus}^{(\pm c)}$	Inertial	5	4	0.2
	Unrestricted	4	4	0.2
$\text{Bolus}^{(+c)}$	Inertial	5	4	0.5
	Unrestricted	3	5	0.2

were tuned on the general carbohydrate recommendation scenario $\text{Carbs}^{(\pm b)}$ were used for $\text{Carbs}^{(-b)}$.

5.2 Experimental Results

Tables 5.7 and 5.8 show the results for the two baselines and the two neural architectures: the LSTM-based (Figures 3.1 and 3.2) and the N-BEATS-based (Figure 3.3). Across all scenarios and for both example classes, the neural models outperform both baselines, often by a wide margin. Furthermore, the N-BEATS-based models outperform their LSTM-based counterparts across all evaluations with inertial examples, which are the ones with the most practical utility. In general, there is little difference between the best model scores and the average model scores, which means that the model performance is relatively stable with respect to the random initialization of the network parameters.

For the prediction of carbohydrates without an associated bolus scenario $\text{Carbs}^{(-b)}$, the improvement brought by the two neural models over the two baselines was less substantial, which is to be expected for two reasons. First, the baselines do much better in this scenario than in the more general carbohydrate recommendation scenario $\text{Carbs}^{(\pm b)}$ because most

Table 5.7: Results for the Carbs$^{(\pm b)}$ and Carbs$^{(-b)}$ recommendation scenarios, for both classes of examples. The simple † indicates a p-value < 0.05 when using a one-tailed t-test to compare against the baseline results; the double ‡ indicates statistical significance for comparison against the baselines as well as against the competing neural method; the ↑ indicates significant with respect to the Global Average baseline only.

	Inertial		Unrestricted	
Carbs$^{(\pm b)}$ recommendation	RMSE	MAE	RMSE	MAE
Global Average	20.90	17.30	20.68	17.10
ToD Average	20.01	15.78	19.82	15.68
LSTM.mean	11.55	7.81	10.99	7.40
LSTM.best	10.95	7.50	10.50	7.31
N-BEATS.mean	‡**9.79**	‡**6.45**	10.34	7.04
N-BEATS.best	9.92	6.56	†**10.07**	†**6.75**

	Inertial		Unrestricted	
Carbs$^{(-b)}$ recommendation	RMSE	MAE	RMSE	MAE
Global Average	15.92	13.71	14.66	12.19
ToD Average	15.55	13.45	14.27	11.93
LSTM.mean	14.02	11.47	14.70	12.27
LSTM.best	13.75	10.92	14.94	12.57
N-BEATS.mean	**13.76**	**11.42**	↑**13.69**	↑**11.09**
N-BEATS.best	14.52	11.78	14.17	11.47

of the carb intakes are relatively small, e.g. hypo correction events where subjects are advised to eat a fixed amount of carbohydrate. Second, and most importantly, the number of training carbohydrate events and their associated examples in the Carbs$^{(-b)}$ scenario is

Table 5.8: Results for the Bolus$^{(\pm c)}$ and Bolus$^{(+c)}$ recommendation scenarios, for both classes of examples. The simple † indicates a p-value < 0.05 when using a one-tailed t-test to compare against the baseline results; the double ‡ indicates statistical significance for comparison against the baselines as well as against the competing neural method; the ↑ indicates significant with respect to the Global Average baseline only.

	Inertial		Unrestricted	
Bolus$^{(\pm c)}$ recommendation	RMSE	MAE	RMSE	MAE
Global Average	2.40	2.13	2.84	2.30
ToD Average	2.21	1.86	2.71	2.17
LSTM.mean	1.75	1.35	1.53	1.10
LSTM.best	1.70	1.30	1.50	1.05
N-BEATS.mean	†**1.56**	‡**1.20**	†**1.49**	1.04
N-BEATS.best	1.65	1.26	1.51	†**1.03**

	Inertial		Unrestricted	
Bolus$^{(+c)}$ recommendation	RMSE	MAE	RMSE	MAE
Global Average	3.00	2.35	3.04	2.39
ToD Average	2.87	2.21	2.90	2.25
LSTM.mean	1.02	0.73	**1.00**	0.73
LSTM.best	0.94	0.67	†**1.00**	†**0.72**
N-BEATS.mean	0.89	0.65	1.11	0.82
N-BEATS.best	†**0.85**	†**0.61**	1.06	0.78

much smaller than in the Carbs$^{(\pm b)}$ scenario (Table 4.1), which makes ML models much less effective.

In all experiments reported so far, one model was trained for all prediction horizons, using the value of $\tau \in \{30, 35, ..., 90\}$ as an additional input feature. This global model

was then tested on examples from all prediction horizons. To determine if transfer learning happens among different prediction horizons, for each value of $\tau \in \{30, 45, 60, 75, 90\}$ at test time, the performance of the globally trained model and the performance of a model trained only on examples for that particular prediction horizon are compared, using inertial examples for both. The inertial case was chosen for this experiment because it corresponds better to the intended use of a carbohydrate or bolus recommendation system. Furthermore, only the N-BEATS-based model is used for these experiments because of its better performance in the inertial case. The results in Table 5.9 show transfer learning clearly happening for the carbohydrate recommendation $\text{Carbs}^{(\pm b)}$ and bolus given carbs recommendation $\text{Bolus}^{(+c)}$ scenarios, where the models trained on all prediction horizons outperform those trained only on a specific prediction horizon when evaluated on that prediction horizon. For the bolus recommendation scenario $\text{Bolus}^{(-c)}$ (i.e. $\text{Bolus}^{(\pm c)}$ inertial) the results were mixed, with transfer learning being clear only for the short $\tau = 30$ time horizon. Transfer learning results for the $\text{Carbs}^{(-b)}$ scenario are not calculated due to the lack of a sufficient number of training examples for each prediction horizon.

Table 5.9: Comparison between models trained on all prediction horizons vs. one prediction horizon τ, when evaluated on the prediction horizon τ. The symbol † indicates a p-value < 0.05 when using a one-tailed t-test to compare against the one prediction horizon results.

		Carbs$^{(\pm b)}$ recommendation											
		τ = 30		τ = 45		τ = 60		τ = 75		τ = 90		Average	
	Trained	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
N-BEATS.mean	One τ	**9.74**	6.72	10.24	6.89	10.06	6.85	10.52	7.19	9.82	6.73	10.08	6.88
	All τ	9.96	**6.57**	**9.98**	**6.56**	**9.84**	**6.50**	†**9.55**	†**6.30**	**9.37**	**6.22**	**9.74**	**6.43**
N-BEATS.best	One τ	9.92	6.70	10.39	6.90	10.21	6.88	10.62	7.18	9.92	6.66	10.21	6.86
	All τ	**9.84**	**6.50**	**9.94**	**6.56**	**10.02**	**6.57**	**9.76**	†**6.34**	**9.43**	**6.08**	**9.80**	**6.41**

		Bolus$^{(-c)}$ recommendation											
		τ = 30		τ = 45		τ = 60		τ = 75		τ = 90		Average	
	Trained	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
N-BEATS.mean	One τ	1.82	1.42	**1.57**	**1.24**	1.51	1.24	**1.37**	**1.10**	1.40	1.17	1.53	1.23
	All τ	**1.75**	**1.33**	1.61	**1.24**	**1.47**	†**1.17**	1.38	**1.10**	**1.28**	†**1.03**	**1.50**	†**1.17**
N-BEATS.best	One τ	1.77	1.37	**1.54**	**1.21**	**1.51**	**1.23**	**1.38**	**1.10**	**1.34**	**1.11**	**1.51**	**1.20**
	All τ	**1.72**	**1.28**	1.75	1.33	1.58	**1.23**	1.45	1.12	1.44	1.13	1.59	1.22

		Bolus$^{(+c)}$ recommendation											
		τ = 30		τ = 45		τ = 60		τ = 75		τ = 90		Average	
	Trained	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
N-BEATS.mean	One τ	0.98	0.73	0.91	0.69	0.91	0.69	0.95	0.74	0.93	0.72	0.94	0.71
	All τ	**0.95**	**0.68**	**0.87**	**0.65**	**0.86**	**0.65**	†**0.87**	†**0.65**	†**0.86**	†**0.64**	†**0.88**	†**0.65**
N-BEATS.best	One τ	**0.94**	0.69	0.91	0.69	0.92	0.68	0.93	0.71	0.91	0.70	0.92	0.69
	All τ	**0.94**	**0.66**	**0.84**	†**0.62**	†**0.82**	†**0.59**	†**0.82**	†**0.61**	†**0.83**	†**0.61**	†**0.85**	†**0.62**

6 Conclusion

This thesis has introduced a general LSTM-based neural architecture, composed of two chained LSTM networks and a fully connected network, with the purpose of training models for making recommendations with respect to any type of quantitative events that may impact blood glucose levels, in particular, carbohydrate amounts and bolus insulin dosages. A deep residual N-BEATS-based architecture was also developed, using the chained LSTM networks as a component in its block structure. Experimental evaluations show that the proposed neural architectures substantially outperform a global average baseline as well as a time of day dependent baseline, with the N-BEATS-based models outperforming the LSTM-based counterparts in all evaluations with inertial examples. The trained models are shown to benefit from transfer learning and from a pre-processing of meal events that anchors their timestamps shortly after their corresponding boluses. Overall, these results suggest that the proposed recommendation approaches hold significant promise for easing the complexity of self-managing blood glucose levels in type 1 diabetes. Potential future research directions include investigating the proposed pre-processing of carbohydrate events for blood glucose level prediction and exploring the utility of the two neural architectures for recommending exercise.

References

[1] Bach, K., Bunescu, R., Farri, O., Guo, A., Hasan, S., Ibrahim, Z. M., Marling, C., Raffa, J., Rubin, J., and Wu, H., Eds. *Proceedings of the 3rd International Workshop on Knowledge Discovery in Healthcare Data* (Stockholm, Sweden, 2018), CEUR-WS.org.

[2] Bach, K., Bunescu, R., Marling, C., and Wiratunga, N., Eds. *Proceedings of the 5th International Workshop on Knowledge Discovery in Healthcare Data* (Santiago de Compostela, Spain, 2020).

[3] Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks 5*, 2 (1994), 157–166.

[4] Boutayeb, A., and Chetouani, A. A critical review of mathematical models and data used in diabetology. *BioMedical Engineering OnLine 5*, 43 (2006).

[5] Bunescu, R., Struble, N., Marling, C., Shubrook, J., and Schwartz, F. Blood glucose level prediction using physiological models and support vector regression. In *Proceedings of the IEEE 12th International Conference on Machine Learning and Applications (ICMLA)* (Miami, Floria, USA, 2013), IEEE, pp. 135–140.

[6] Cappon, G., Vettoretti, M., Marturano, F., Facchinetti, A., and Sparacino, G. A neural-network-based approach to personalize insulin bolus calculation using continuous glucose monitoring. *Journal of Diabetes Science and Technology 12* (2018), 265–272.

[7] Centers for Disease Control and Prevention. Manage Blood Sugar. https://www.cdc.gov/diabetes/basics/what-is-type-1-diabetes.html, 2020. Accessed: 2021-02-10.

[8] Centers for Disease Control and Prevention. What is diabetes? https://www.cdc.gov/diabetes/basics/diabetes.html, 2020. Accessed: 2020-10-06.

[9] Gers, F., Schmidhuber, J., and Cummins, F. Learning to forget: Continual prediction with LSTM. *Neural Computation 12* (10 2000), 2451–71.

[10] Glorot, X., and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (2010), JMLR Workshop and Conference Proceedings, pp. 249–256.

[11] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems 28*, 10 (2016), 2222–2232.

[12] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[13] He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision* (2016), Springer, pp. 630–645.

[14] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural Computation 9* (12 1997), 1735–1780.

[15] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017).

[16] Kingma, D. P., and Ba, J. L. Adam: A method for stochastic optimization. In *Third International Conference for Learning Representations (ICLR)* (San Diego, California, 2015).

[17] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25* (2012), 1097–1105.

[18] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature 521*, 7553 (2015), 436–444.

[19] Man, C. D., Micheletto, F., Lv, D., Breton, M., Kovatchev, B., and Cobelli, C. The UVA/PADOVA type 1 diabetes simulator: New features. *Journal of Diabetes Science and Technology 8* (2014), 26–34.

[20] Marling, C., and Bunescu, R. The OhioT1DM dataset for blood glucose level prediction. In *The 3rd International Workshop on Knowledge Discovery in Healthcare Data* (Stockholm, Sweden, 2018).

[21] Marling, C., and Bunescu, R. The OhioT1DM dataset for blood glucose level prediction: Update 2020. In *The 5th International Workshop on Knowledge Discovery in Healthcare Data* (Santiago de Compostela, Spain, 2020). in press.

[22] Mirshekarian, S., Shen, H., Bunescu, R., and Marling, C. LSTMs and neural attention models for blood glucose prediction: Comparative experiments on real and synthetic data. In *Proceedings of the 41st International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2019)* (Berlin, Germany, 2019).

[23] Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. N-BEATS: neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations (ICLR)* (2020).

[24] Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning* (2013), PMLR, pp. 1310–1318.

[25] Pesl, P., Herrero, P., Reddy, M., Xenou, M., Oliver, N., Johnston, D., Toumazou, C., and Georgiou, P. An advanced bolus calculator for type 1 diabetes: System architecture and usability results. *IEEE Journal of Biomedical and Health Informatics 20*, 1 (2016), 11–17.

[26] Plis, K., Bunescu, R., Marling, C., Shubrook, J., and Schwartz, F. A machine learning approach to predicting blood glucose levels for diabetes management. In *Modern Artificial Intelligence for Health Analytics: Papers Presented at the Twenty-Eighth AAAI Conference on Artificial Intelligence* (2014), AAAI Press, pp. 35–39.

[27] Rubin-Falcone, H., Fox, I., and Wiens, J. Deep residual time-series forecasting: Application to blood glucose prediction. In *The 5th International Workshop on Knowledge Discovery in Healthcare Data* (Santiago de Compostela, Spain, 2020), pp. 112–116.

[28] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1–9.

[29] Tyler, N., Mosquera-Lopez, C., Wilson, L., Dodier, R., Branigan, D., Gabo, V., Guillot, F., Hilts, W., Youssef, J. E., Castle, J., and Jacobs, P. An artificial intelligence decision support system for the management of type 1 diabetes. *Nature Metabolism 2* (2020), 612–619.

[30] Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE 78*, 10 (1990), 1550–1560.

[31] Zhu, T., Li, K., Kuang, L., Herrero, P., and Georgiou, P. An insulin bolus advisor for type 1 diabetes using deep reinforcement learning. *Sensors 20*, 18 (2020).