



**UiT** The Arctic University of Norway

Faculty of Science and Technology  
Department of Physics and Technology

## **Personalized optimization of blood glucose regulation**

A Diabetes Mellitus case study

Preben Drangsholt

STA-3941 Master's thesis in applied physics and mathematics - 30 stp - July 2022

This thesis document was typeset using the *UiT Thesis L<sup>A</sup>T<sub>E</sub>X Template*.

© 2022 – <http://github.com/egraff/uit-thesis>

# Abstract

Type 1 diabetes (T1D) is a chronic autoimmune disease that leads to insulin deficiency. Consequently the disease will lead to a poor blood glucose (BG) regulation, and in situations of high energy expenditure like exercise, a dysfunctional regulation can cause severe damage or death [5] [14].

Diabetes is responsible for one death every five seconds and financially drains approximately 11% of the global health expenditure [9]. This adversity is subject to a great quantity of research, but is seemingly incurable. Therefore, symptom control treatments are salient. However there is currently no fully individualized autonomous solution to the management of the disease.

In this thesis we present and develop a way of optimizing BG regulation that is fitted to one specific patient. We combine the use of a published mathematical model that models BG as a function of heart rate (HR), insulin injections and carbohydrate (CHO), with artificial intelligence to give the patient a recommended amount of CHO before any given exercise. We also do a statistical optimization of the BG optimization and compare both results.

The results are promising, but additional validation is needed before completely trustworthy conclusions can be made. However, the product is a valuable decision support system that the patient can use, and the methodology presents a way to adapt the product to other patients.



# Acknowledgments

I would like to extend my deepest gratitude to my supervisors, Fred Godtlielsen, Miguel Angel Tejedor Hernandez and Phuong Dinh Ngo. I am deeply honored for this experience, and could not imagine a better project for my thesis.

Thanks to my dear Isabel for your undying support, my family and friends.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgments</b>	<b>iii</b>
<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>viii</b>
<b>List of Abbreviations</b>	<b>x</b>
<b>I Introduction</b>	<b>1</b>
<b>1 Diabetes Mellitus Decision Support System</b>	<b>3</b>
<b>2 Thesis Formalities</b>	<b>5</b>
2.1 Objective . . . . .	5
2.2 Thesis Outline . . . . .	6
<b>II Background Theory and Related Work</b>	<b>7</b>
<b>3 Diabetes Mellitus</b>	<b>9</b>
3.1 Autoimmune diseases . . . . .	11
3.2 Pancreas . . . . .	12
3.3 Bio-dynamics of blood glucose regulation . . . . .	13
3.4 Current treatments of Diabetes Mellitus . . . . .	15
3.5 Breton's physical activity model . . . . .	15
<b>4 Machine Learning</b>	<b>18</b>
4.1 Artificial Neural Network . . . . .	19
4.1.1 Perceptron . . . . .	19
4.1.2 Multilayer Perceptrons . . . . .	20
4.1.3 Error function . . . . .	21

4.1.4	Output of neural network . . . . .	23
4.1.5	Adaptive Moment Estimation . . . . .	23
4.2	Generative adversarial network . . . . .	24
4.3	Reinforcement Learning . . . . .	25
4.3.1	What is reinforcement learning? . . . . .	25
4.3.2	The goal of reinforcement learning . . . . .	26
4.3.3	Policy . . . . .	27
4.3.4	Deep Q-Learning . . . . .	28
<b>5</b>	<b>Statistics and Mathematics</b>	<b>31</b>
5.1	Numerical integration . . . . .	31
5.2	Nonlinear grey-box model . . . . .	32
5.3	Trust-Region-Reflective Least Squares . . . . .	32
5.4	Simulated annealing . . . . .	33
<b>III</b>	<b>Experiments and Methodology</b>	<b>34</b>
<b>6</b>	<b>In-silico BG Simulation</b>	<b>36</b>
6.1	Model setup . . . . .	36
6.2	Workout data from patient . . . . .	39
6.3	Fitting the model . . . . .	40
<b>7</b>	<b>Decision Support Systems</b>	<b>41</b>
7.1	Reinforcement Learning . . . . .	41
7.1.1	Environment elements . . . . .	42
7.1.2	Agent . . . . .	43
7.2	Simulated annealing . . . . .	43
<b>IV</b>	<b>Results and Discussion</b>	<b>44</b>
<b>8</b>	<b>Generative Model</b>	<b>46</b>
8.1	Estimation of the model's parameters . . . . .	46
8.2	Validation . . . . .	52
8.3	Discussion . . . . .	55
<b>9</b>	<b>Food Recommendation</b>	<b>57</b>
9.1	RL . . . . .	57
9.2	Simulated annealing . . . . .	59
9.3	Comparison . . . . .	59
9.4	Discussion . . . . .	60

<b>V Conclusion</b>	<b>63</b>
<b>10 Concluding Remarks</b>	<b>65</b>
10.1 Future Work . . . . .	66
<b>11 Appendix</b>	<b>68</b>



# List of Figures

3.1	Diabetes’s harmful effects on the body [30] . . . . .	10
3.2	Size of the thymus in newborns and adults [24] . . . . .	12
3.3	Relationship of insulin and glucagon hormones [19] . . . . .	14
3.4	Summary of the UVA/PADOVA model [15] . . . . .	16
4.1	Perceptron [31] . . . . .	19
4.2	Pseudocode of adam, a stochastic optimization algorithm adapted from Diederik P. Kingma [12] . . . . .	24
4.3	The agent-environment interaction in a Markov decision process [27] . . . . .	25
4.4	A policy based RL using deep artificial neural network [16] .	29
4.5	Pseudocode of Deep Q-learning, adapted from [17] . . . . .	29
6.1	Data illustration example. . . . .	39
8.1	11 subplots, one for each optimized parameter. Each plot contains a histogram of the parameter and a fitted normal distribution . . . . .	52
8.2	Validation result: A plot of the the event showing the recorded BG history in blue, and the simulation result in orange. . . .	53
8.3	Validation result: A plot of the the event showing the recorded BG history in blue, and the simulation result in orange. . . .	54
9.1	Reward vs. Episodes . . . . .	58
9.2	Average Reward vs. Episodes . . . . .	58
9.3	Simulated Annealing . . . . .	59
9.4	Comparison of results for the optimization of BG regulation .	60

# List of Tables

6.1	In-silico parameters . . . . .	38
8.1	Training result: A plot of the event showing the recorded BG history in grey, and the simulation result in red. The Y axis is BG in $\frac{mg}{dl}$ and the x axis is time in minutes. Three plots of the inputs for the events, HR in <i>bpm</i> , CHO in grams and insulin in units. A list of each optimized parameter and the resulting value. . . . .	47
8.2	Training result: A plot of the event showing the recorded BG history in grey, and the simulation result in red. The Y axis is BG in $\frac{mg}{dl}$ and the x axis is time in minutes. Three plots of the inputs for the events, HR in <i>bpm</i> , CHO in grams and insulin in units. A list of each optimized parameter and the resulting value. . . . .	48
8.3	Training result: A plot of the event showing the recorded BG history in grey, and the simulation result in red. The Y axis is BG in $\frac{mg}{dl}$ and the x axis is time in minutes. Three plots of the inputs for the events, HR in <i>bpm</i> , CHO in grams and insulin in units. A list of each optimized parameter and the resulting value. . . . .	49
8.4	Training result: A plot of the event showing the recorded BG history in grey, and the simulation result in red. The Y axis is BG in $\frac{mg}{dl}$ and the x axis is time in minutes. Three plots of the inputs for the events, HR in <i>bpm</i> , CHO in grams and insulin in units. A list of each optimized parameter and the resulting value. . . . .	50
8.5	Training result: A plot of the event showing the recorded BG history in grey, and the simulation result in red. The Y axis is BG in $\frac{mg}{dl}$ and the x axis is time in minutes. Three plots of the inputs for the events, HR in <i>bpm</i> , CHO in grams and insulin in units. A list of each optimized parameter and the resulting value. . . . .	51



# Abbreviations

**ANN** Artificial neural network

**API** Application programming interface

**BG** Blood glucose in mg/dl

**CGM** Continuous Glucose Monitor.

**CHO** Carbohydrate

**GAN** Generative adversarial network

**HR** Heart rate in beats per minute

**RL** Reinforcement Learning.

**T1D** Type 1 Diabetes Mellitus

## **Part I**

# **Introduction**





# Diabetes Mellitus Decision Support System

T1D is a chronic autoimmune disease that leads to insulin deficiency. The deficiency is caused by the body's own immune system, which destroys the insulin producing cells [5]. Today, autoimmune diseases cannot be cured, meaning that symptom control methods must be used to reduce the impact of the disease and improve the life quality of the patient.

The current solution to T1D is based on keeping the blood glucose at an acceptable level. This can be done manually or semi-manually by the patient, with help and regular guidance from physicians. The regulation process is often expensive and time consuming, and diabetes is responsible for about 11% of the global health expenditure [10]. The regulation process requires external injections of the hormone insulin at the correct time, and often a carefully planned and executed regime of exercise and diet. There exist tools that help with monitoring blood glucose levels and injecting insulin, but there is currently no fully individualized autonomous solution to the management of the disease.

A recent study by the international diabetes federation [10] found that the estimated prevalence of adults was about 10%. The disease is responsible for approximately one death every five seconds [10]. T1D is therefore a major adversity for the human race. In the light of the new processing technology

and science there may be better alternatives than the current solutions. The work to improve the sustainability and autonomy of these solutions and especially automatic individual adaptations is an important contribution and is the primary motivation for this thesis.





# Thesis Formalities

## 2.1 Objective

In short the objective of this thesis is to provide the patient with an optimal amount of CHO at any given situation. It is a known problem for many patients with T1D that it can be problematic to regulate the BG in intense exercises. Henceforth our focus will tackle situations like these.

More specifically, the first objective will be to adapt Breton's physical activity model [4] to include bolus insulin injections, and to refit the model suited to a specific patient. This will be done with safe historical measurements from the subjects continuous glucose monitor (CGM), HR monitor and insulin pump.

Secondly, the personalized model will be used as a base to optimize the subject's CHO intake in a given high intensity exercise. This result can then be evaluated by the subject as a recommendation or to strengthen the subject's food intake regime.

The results will be highly adapted to the specific subject, but can also be seen as a proof of concept and a possible pathway to adaptations for other T1D patients.

## 2.2 Thesis Outline

**Part 1** introduces the thesis by describing the motivation for providing a food recommendation for T1D patients, and states the thesis formalities.

**Part 2** describes the theory in relation to the thesis and the related work that acts as a foundation. This starts by describing Diabetes Mellitus, its cause, impact, current treatments and the bio-dynamics of blood glucose regulation. This is followed by some relevant machine learning theory like artificial neural network (ANN) and reinforcement learning (RL). Lastly some crucial concepts of mathematics and statistics are described followed by some related work.

**Part 3** describes the methodology and setup of our two experiments. Firstly we explain the setup for the generative model and how it was fitted to the patient. Then, two ways of optimization are presented. Lastly, we show how the optimization is calculated by using RL and simulated annealing.

**Part 4** presents and discusses the two conducted experiments.

**Part 5** concludes the thesis and comments on future work.

## **Part II**

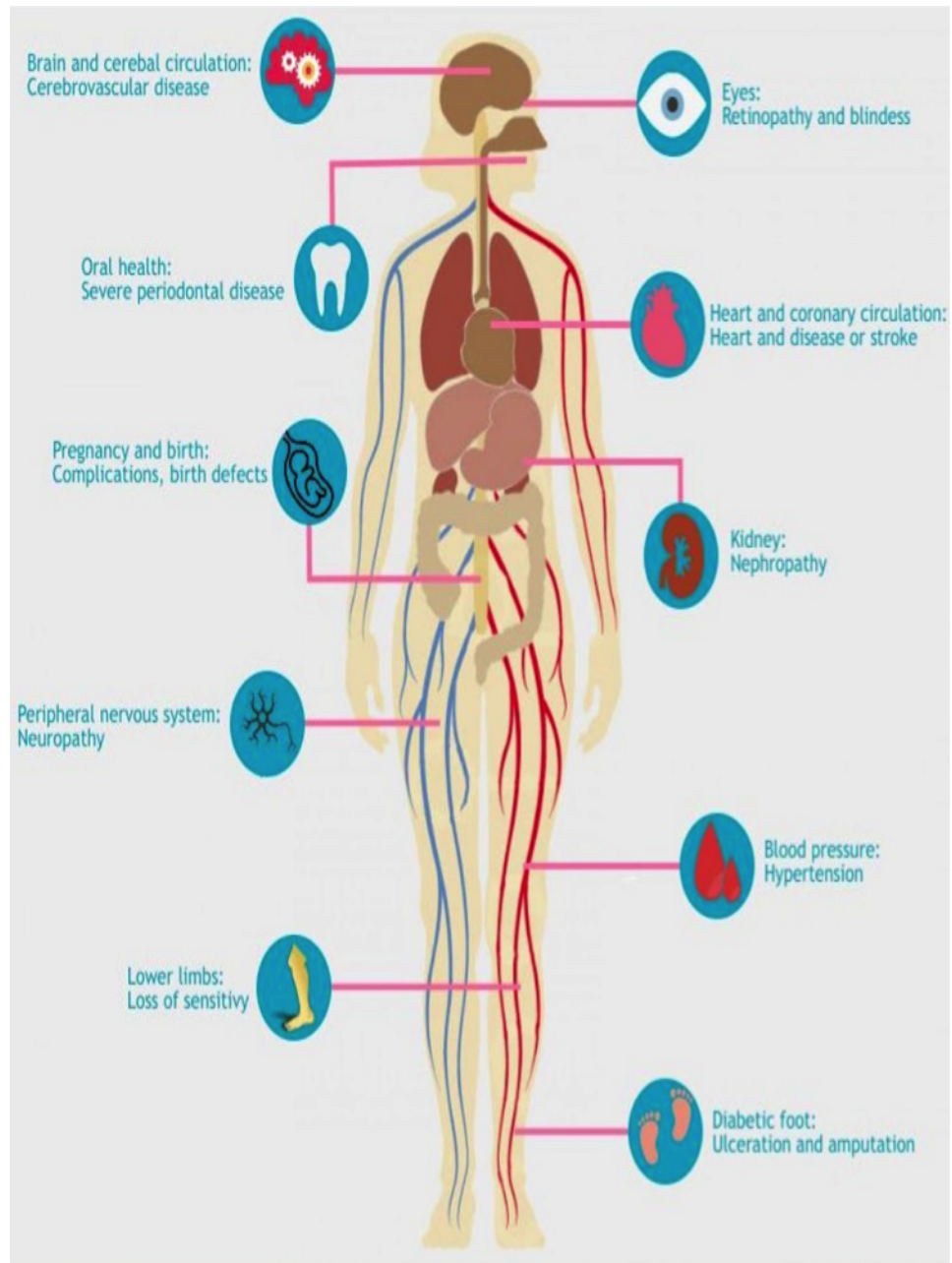
# **Background Theory and Related Work**



# /3

## Diabetes Mellitus

Diabetes Mellitus is a chronic autoimmune disease that is characterized by the subject's inability to correctly regulate the blood glucose levels. In a non-diabetic human, the regulation process is near-perfect due to the bio-mechanics surrounding the production and usage of the hormone insulin (see more in section 3.2). This hormone is produced by the organ pancreas and causes a bio-mechanical reaction that lowers the blood-glucose levels. When this level is not properly regulated, the body will take damage. A BG level that is too low too low (hypoglycemia) can cause unconsciousness and death [14], and when the BG level gets too high (hyperglycemia), the body will experience dysfunctions and organ failures, as shown in Figure 3.1.



**Figure 3.1:** Diabetes's harmful effects on the body [30]

The main types of diabetes are:

- Type 1 Diabetes: The chronic autoimmune disease in focus of this paper. The insulin shortage is caused by a lack of insulin producing cells in the

pancreas, likely due to the immune response that is caused by mistrained lymphocytes T cells (see section 3.1).

- **Type 2 Diabetes:** The most common form of diabetes, occurring in 90% of the cases [21]. This disease is mainly characterized by a developed insulin tolerance due to physical inactivity and excess of body weight. Type 2 diabetes has the same symptoms as T1D, but they are less pronounced.
- **Gestational Diabetes:** This type of diabetes is caused by the body's inability to meet the blood glucose regularization demand of pregnancy. Most commonly, it appears at the second or third trimester, and often disappears after labour.

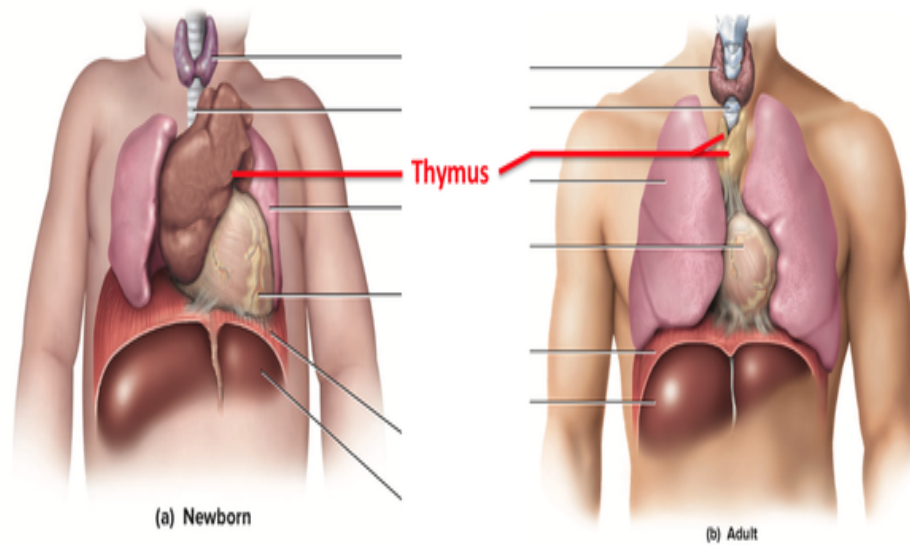
The IDF Diabetes Atlas 10th Edition reported in its Key global findings 2021 that 537 million adults (20–70 years old) are affected by diabetes. That makes up about 10% of the adult world population. In addition, the IDF has predicted that the numbers will increase to about 643 million by 2030 and to 784 million by 2045. It is reported that 81% of adults with diabetes live in a low to middle income country. Diabetes is responsible for one death every five seconds (6.7 million in a year). This major health liability drains approximately 11% of the global health expenditure (996 billion out of 8.8 trillion) [9].

In the following section we will briefly go through the cause of an autoimmune disease and its effect on the human race.

### 3.1 Autoimmune diseases

Thymus is a small organ located in the upper chest area close to the heart. The major assignments of this organ are to foster growth and cellular differentiation, and to accrue new matured T cells to the peripheral T cell pool. The latter part means that the organ is in charge of showing T cells a set of cells that the body does not want to destroy (its own cells) and if any immature T cells that undergo this training fails to ignore the selected test cells, they are promptly destroyed and recycled. If this graduation process fails, the effects can be that some of the units in the peripheral T cell pool have been mistrained and might destroy the body's own cells. This can cause immunodeficiency and autoimmunity [23].

The thymus is known to shrink with age, being at its largest during infancy and mostly gone by the age of 75 (see Figure 3.2). It produces the entire repertoire of T cells by the beginning of puberty and acts as an addition to the endocrine system for the remainder of its lifespan.



**Figure 3.2:** Size of the thymus in newborns and adults [24]

The condition described as an autoimmune disease is an immune response to the body's own cells. In the case of T1D the immune response is when the T cells destroys the insulin producing cells in the pancreas.

A little under 4% of human beings are affected by an autoimmune disease. The most common form of an autoimmune disease is T1D, but there are as many as 80 different documented varieties. The scientific field known as immunology is the study of the body's immune system, and a widely discussed topic is autoimmune diseases. Immunologists who work on T1D focus on the medical and biological aspects of the condition and often collaborate with complex system modeling researchers to increase our knowledge in the field. The goal is to find a solution to autoimmune diseases, which will subsequently result in a general cure for T1D [1].

## 3.2 Pancreas

The pancreas is an organ located in the abdomen. It has two major functionalities in the body. The first and most important objective is to secrete pancreatic liquid containing bicarbonate through the pancreatic duct and depositing it in the duodenum to neutralize/balance the acidity and releasing digestive enzymes to break down nutrients for absorption [32].



The second objective of the organ and an important prerequisite for the next section (Section 3.3), is its endocrinic contribution. The pancreas has the ability to secrete the following hormones: insulin, glucagon and somatostatin. These can be described as the following:

- Insulin has two connected chains (A and B), composed of 51 amino acids (21 and 30) and is produced by beta cells in the pancreas. The hormone stimulates *GLUT<sub>4</sub>* (a glucose transporter), which results in caching/storing the glucose in the body's glucose depository, causing a decrease in the concentration of blood glucose [2].
- Glucagon is composed of 29 amino acids and is produced by alpha cells in the pancreas. The hormone binds to glucagon receptors in the liver causing the liver to start breaking down its available energy storage (called glycogen) into usable energy entities (glucose) and release them into the blood. This process is called glycogenolysis and causes the concentration of blood glucose to increase. If the glucagon receptors are stimulated, but there is a shortage of glycogen, a process called gluconeogenesis will start, which will result in an increase of glycogen [29].
- There are two variants of Somatostatin. One consists of 14 amino acids (somatostatin-14) and the other one of 28 amino acids (somatostatin-28). One of the attributes of this hormone is to decrease the release of various hormones, such as glucagon and insulin. It is known as the inhibiting hormone and acts as a controller in the endocrine system [26].

### 3.3 Bio-dynamics of blood glucose regulation

As mentioned in Section 3, maintaining a good interval of blood glucose (glucose homeostasis) is imperative to allow for proper bio-functionality. Insulin and glucagon are the hormones produced by the pancreas to maintain this tight interval to achieve a state known as normoglycemic. The regulating process is summarized in Figure 3.3.

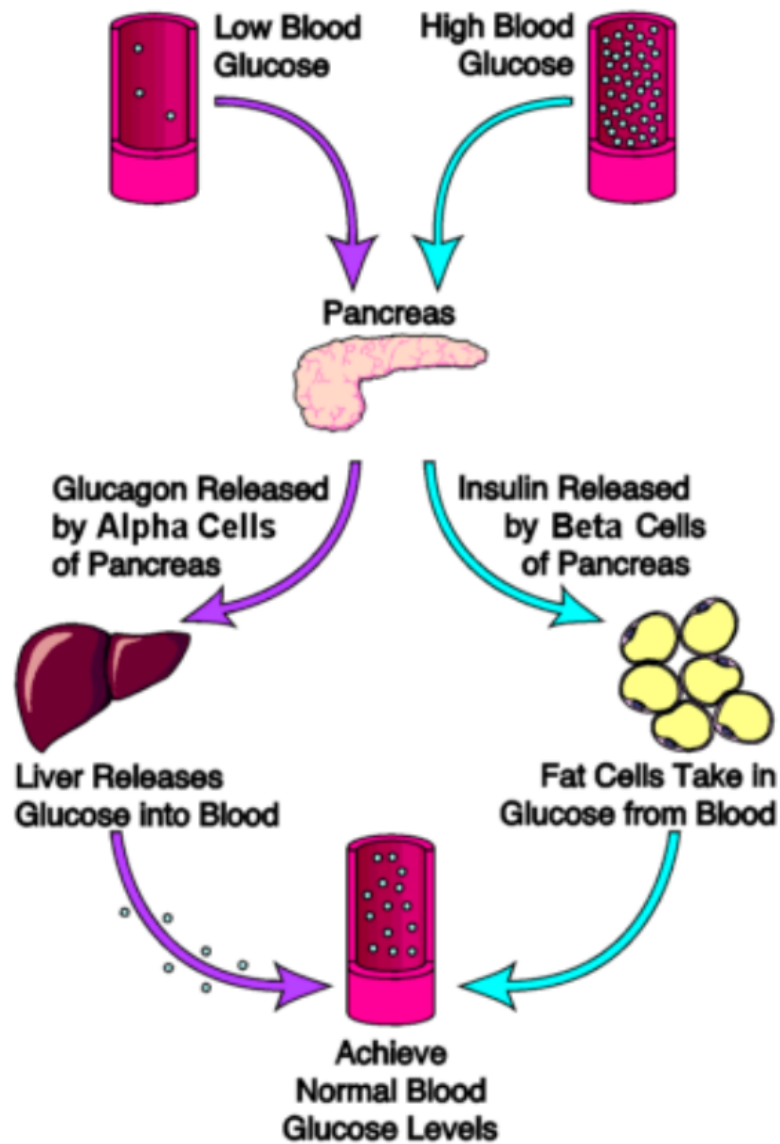


Figure 3.3: Relationship of insulin and glucagon hormones [19]

The narrow interval of glucose homeostasis is generally about  $4\text{--}6 \frac{\text{mmol}}{\text{L}}$  (millimole of glucose per litre of blood). This is equivalent to  $72\text{--}108 \frac{\text{mg}}{\text{dL}}$  (milligram of glucose per decilitre of blood). A value below this level is known as hypoglycemia and a value above  $10 \frac{\text{mmol}}{\text{L}} = 180.18 \frac{\text{mg}}{\text{dL}}$  is known as hyperglycemia [22]. A prolonged hypoglycemic or hyperglycemic state can cause complications, as shown in Figure 3.1.

### 3.4 Current treatments of Diabetes Mellitus

The current treatment of diabetes revolves around keeping the patient in a normoglycemic state. This is mainly done by altering the food intake, exercise and insulin level conditioned on current blood glucose level. Several different methods and apparatuses are developed to help serve this purpose.

Measuring the blood glucose level is mostly done by using a finger-prick or a continuous glucose monitor connected to the body. These tools draw and measure blood from just beneath the skin to gain an estimate of the average glucose levels in the bloodstream. This estimate is not always unbiased as the blood is not drawn directly from the main arteries.

Insulin administration is mostly done by using direct injections with a needle or an insulin pump connected to the body, this includes a fast working insulin injection (bolus) and a more stable long lasting insulin injection (basal).

A regime of diet and exercise is created for the patient by a physician to stabilise the energy consumption and usage. This prevents large fluctuations in the blood glucose level, which facilitates a more streamlined process of regulating the blood glucose levels.

Artificial pancreas is a concept made to mirror the job of the pancreas in a non-diabetic human. This has been done by monitoring and recording food intake, physical activity, stress level and infections to create an algorithm to automatically administer insulin via an insulin pump.

### 3.5 Breton's physical activity model

Breton's physical activity model is a deterministic mathematical model consisting of four differential equations. The model predicts the BG output over time from a set of inputs and parameters. The physical activity model is an extension of UVA/PADOVA [15] – a credible and well-known model that describes BG dynamics. The credibility of UVA/PADOVA has been acknowledged by The United States Food and Drug Administration and has been approved for use in virtual trials to experiment on control systems regarding BG regulation for T1D patients [11] as a substitute for animal testing. The dynamics of the UVA/Pandova model is summarized in figure 3.4.

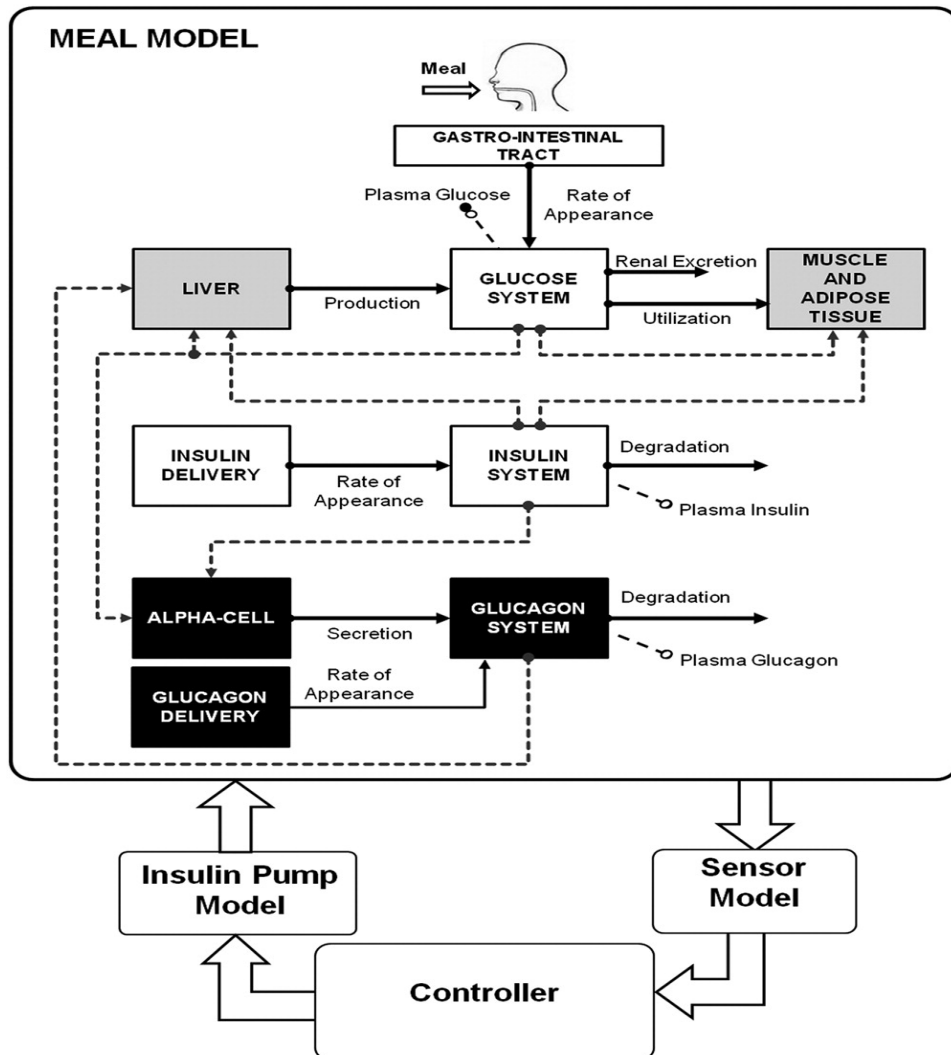


Figure 3.4: Summary of the UVA/PADOVA model [15]

What Breton's model explains is how exercise in the form of heart rate affects the BG output, but this extension has not been validated by real-life trials. The present thesis represents an attempt of testing and validating Breton's model (see section 8.1 for the results. However, as discussed in section 8.3, the experiment is with a small number of measurements and is therefore not a weighty validation.

Breton's model describes energy consumption  $Y$  as a function of heart rate

with the differential:

$$\frac{dY}{dt} = -\frac{Y}{\tau_{HR}} + \frac{1}{\tau_{HR}}(HR - HR_b)$$

Where  $\tau_{HR}$  is the duration of of the energy consumption, HR is the heart rate and  $HR_b$  is the basal/resting heart rate. The function was designed this way because the literature explains that in a steady state, energy consumption is correlated with oxygen consumption. Oxygen consumption is also correlated with heart rate. Therefore, the energy consumed is modeled as a function of increase in heart rate [4]. Breton's physical activity model uses this relation of heart rate and energy consumption to explain the changes in insulin action Z:

$$\frac{dZ}{dt} = -(f(Y) + \frac{1}{\tau}) \cdot Z + f(Y)$$

where  $\tau$  is the duration of change in insulin action,  $a$  is a parameter and the function  $f$  is:

$$f(Y) = \frac{\left(\frac{Y}{a \cdot HR_b}\right)^n}{1 + \left(\frac{Y}{a \cdot HR_b}\right)^n}$$

The resulting BG calculation adapted from the UVA/PADOVA model with the aforementioned changes is:

$$\frac{dG}{dt} = -p_1 \cdot (G - G_b) - (1 + \alpha \cdot Z) \cdot X \cdot G - \beta \cdot Y \cdot G + \frac{D}{V_G} \quad (3.1)$$

where:

$$\frac{dX}{dt} = -p_2 \cdot X + p_3 \cdot I_c$$

Here, X is the action of insulin in a remote compartment on plasma glucose.  $I_c$  is the bolus insulin injections. G is BG, D is glucose in plasma after CHO intake.  $p_1$ ,  $p_2$ ,  $p_3$ ,  $\alpha$  and  $\beta$  is parameters.

# /4

## Machine Learning

Machine learning is a field of science that encompasses several data processing techniques that links different fields of science like neurology, mathematics, statistics, physics and data science. There is a wide variety of tools and techniques in the field that are used to process data in several different ways, but all of these can be grouped into three category's:

- **Supervised learning:** This is a group of processing methods that uses labeled data to encode patterns. The patterns that is learned from the data is saved in its memory referred to as weights (the details are explained in section 4.1.1 and 4.1.2). After the supervised learning model is subjected to the labeled data and optimized to map the data to its respective labels (this is referred to as training), its memory can be used to map new unlabeled data. Supervised learning is often used to classify data into groups, or perform regression.
- **Reinforcement learning :** This consists of an Actor that is tasked with proposing actions and an environment that explains the outcome of that action. The RL model learns from these interactions because the actor is optimized by a rule-based function called a reward function to propose actions that give more reward. RL is often used in robotics to operate motors or control systems, because in situation like these it is clear what rules to apply for the environment and reward. Essentially, in any case where the outcome of actions can be explained and a reward function can be formalized, RL can be used as an optimization method.

- Unsupervised learning: This is a processing method that processes unlabeled data by using the naturally occurring patterns in the input. It is often used as a data transformation tool or as a method to cluster data in different groups.

## 4.1 Artificial Neural Network

### 4.1.1 Perceptron

Perceptron is a numerical processing method inspired by the biological processing method observed in brains. It consists of inputs and an associated weights. Figure 4.1 illustrates a single perceptron (one single neuron).

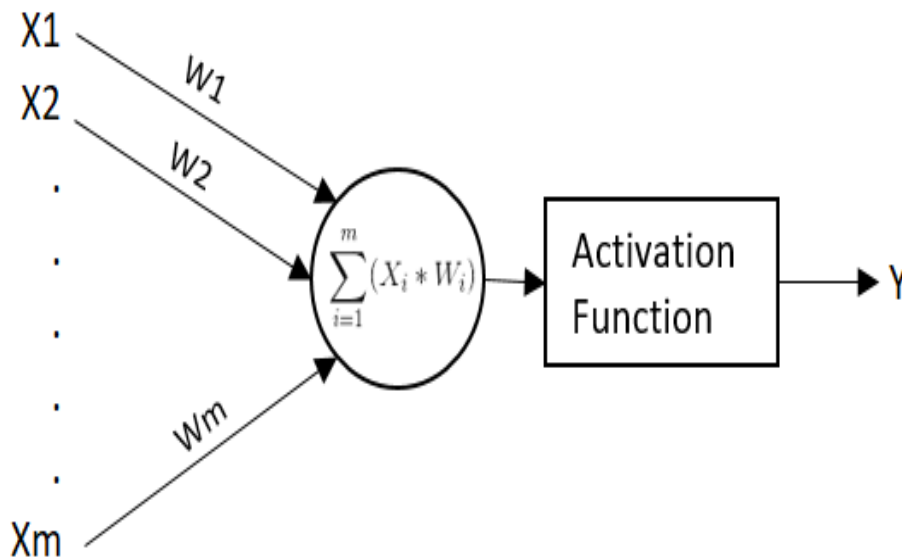


Figure 4.1: Perceptron [31]

The inner product of the inputs can be processed by an activation function to estimate a linear discrimination of the classes based on our labels.

$$y = w^T x$$

Different activation functions have their own attributes and uses. Some are used for regression, and simply outputs any or some values, like a linear function or ReLU. Others are used for classification and forces the value between zero and one, like the logistic function, or between minus one and one in the case

of Tanh [13].

- ReLU  $g(y) = \max(0, y)$
- Logistic  $g(y) = \frac{1}{1+e^{-y}}$
- Tanh  $g(y) = \frac{(e^y - e^{-y})}{e^y + e^{-y}}$

### 4.1.2 Multilayer Perceptrons

A single layer of perceptrons can, as previously mentioned, only estimate a linear discrimination. By stacking several layers, this limitation does not apply. This method is also known as creating a feed forward neural network. All layers except for the input layer and the output layer are called hidden layers. If the network is trained correctly, the weights in the hidden layers will encode the pattern recognition ability to discriminate the data non-linearly. This capacity is proportional to, among other things, the size of the hidden layers and tweaking the length and depth of the hidden layers changes the capacity to encode this information. If this capacity is too large, a problem called overfitting can arise. Likewise, if the capacity is too small, underfitting can occur. The consequence of an overfitted model is that the model is trained to encode the exact pattern of the training data, and an evaluation of unseen data will not discriminate well. The consequence of an underfitted model is that any evaluation has a high error rate due to the model's lack of capturing complex patterns. This problem is notorious in the field and many different techniques have been developed to account for this problem. Some of these techniques will be explored in the following sections [13].

#### 4.1.2.1 Dropout

Dropout is a method used in the training part to generalize the network. This is done by removing neurons in the network with a probability  $p(c)$ . This forces the network not to rely on a specific pattern in the input to fit the associated class, but rather a general pattern which can apply better to unseen cases.

#### 4.1.2.2 Changing size of hidden layers

In all its essence, a neural network with unlimited complexity will find an exact pattern to discriminate or predict every specific training case. Every slightest deviation from the training cases will not fit well to the pattern that the neural



network has learned, which will result in a bad performance in test situations. The purpose of this kind of network is to be able to apply information learned from training cases to accurately discriminate or predict unseen cases.

There are currently no theoretical public descriptions of how to create the optimal network size for a given situation. But large-scale trial-and-error solutions to certain situations have been performed by private companies and research groups. Using the best performing public architectures for a similar input space as a base, followed by adjusting architecture through experiments to achieve a suitable fitting score, is common in the field.

### 4.1.3 Error function

An error function is a function that formally explains the difference between an estimated parameter and the true value of the parameter. Often when we speak about error functions in supervised neural networks it is referred to as a loss function that is a function of the difference between our predicted output and the true label of the input. The loss functions are presented in the following subsections.

#### 4.1.3.1 Mean squared error

Mean Squared Error is a popular loss function used in non-probabilistic regression models. The function is sensitive to outliers and provides a good measurement of the error, as it is based on the mean value. The mean value is the expectation of a normal distribution and is, given a large number of samples, reasonable as explained by the central limit theorem:

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n} \quad (4.1)$$

Where  $\hat{y}_i$  is the predicted output of the last layer in the model (one for every dimension  $i$ ) calculated by an activation function:

$$y = g(w_{n-1}^T y_{n-1}) \quad (4.2)$$

Where  $g$  is an activation function and  $y_{n-1}$  and  $w_{n-1}$  is the outputs and weights of the previous layer (see Sections 3.2 and 3.3 for more details).

### 4.1.3.2 Evidence lower bound

In the case of a probabilistic regression model, our goal is to estimate the posterior distribution of our ground truth (the labels), which is given by Bayes' rule:

$$p_{\theta}(z|x) = \frac{p_{\theta}(z|x) * p_{\theta}(z)}{p_{\theta}(x)} \quad (4.3)$$

Where  $p_{\theta}(z|x)$  is the posterior distribution of the unseen data  $z$  conditioned on our samples  $x$ .  $p_{\theta}(z|x)$  is the likelihood and  $p_{\theta}(z)$  is the prior. In the denominator, the normalizing constant, or the "evidence"  $p_{\theta}(x)$  is found by the integral  $\int p_{\theta}(x|z)p_{\theta}(z)dz$ , which is intractable in higher dimensions. As a result, we cannot estimate the posterior distribution directly.

To get around this we use variational inference to find another distribution  $q_{\phi}(z|x)$  to approximate the true posterior distribution. We use the Kullback–Leibler divergence (measure of dissimilarity between two distributions) to minimize the difference between our approximations and labels, giving us a loss function to optimize:

$$D_{KL}(q_{\phi}|p_{\theta}) = E_{q_{\phi}} \left[ \log \frac{q_{\phi}(z|x)}{p_{\theta}(x|z)} \right] \quad (4.4)$$

However, as previously mentioned,  $p_{\theta}(x|z)$  is intractable. Therefore, it can not be computed, but by factorizing and rewriting the equation into:

$$D_{KL}(q_{\phi}|p_{\theta}) = E_{q_{\phi}} [\log q_{\phi}(z|x)] - E_{q_{\phi}} [\log p_{\theta}(z, x)] + \log p_{\theta}(x) \quad (4.5)$$

We know that  $D_{KL}(q_{\phi}|p_{\theta})$  is positive, which gives us the relationship:

$$\log p_{\theta}(x) \geq -E_{q_{\phi}} [\log q_{\phi}(z|x)] + E_{q_{\phi}} [\log p_{\theta}(z, x)] \quad (4.6)$$

This means that we can minimize the Kullback–Leibler divergence by maximiz-

ing the lower bound of the equation above.

To summarize, we can say that, to approximate the true posterior distribution, we can minimize the dissimilarity between a predicted distribution and the true distribution by maximizing the log evidence lower bound (ELBO) given in equation 8.

#### 4.1.4 Output of neural network

As touched upon in section 4.1.1 the output of the perceptron can be processed by an activation function. By using a squeeze function like logistic or tanh the output is squeezed to a number between 0 and 1. But without any supervision this probability is not useful as it is based on random initialed weights. To train the network we give labels to each data points and process the prediction by using an error function (explained in the previous section). We can then train the network to minimize this error by using gradient decent. This method propagates through the network and calculates the gradients of the weights for all neurons. We can then update each weight by its associated gradient multiplied by a learning rate to nudge the network in the right direction towards a minimum point. There are several different optimizers that use different techniques to achieve the goal of global minimum. One popular technique to avoid getting stuck in a local minimum is to introduce a momentum term in the update function. This serves to push the gradients over potential local minimum points to an increased chance of ending up in a lower gradient minimum [13]. In the next segment we will go through the optimizer that where chosen in this the current thesis.

#### 4.1.5 Adaptive Moment Estimation

Adaptive Moment Estimation (Adam) is a stochastic optimization method that was presented by Diederik Kingma in the paper Adam: A Method for Stochastic Optimization [12]. The algorithm is a combination of the popular optimization methods AdaGrad and RMSProp. It utilizes momentum and adaptive learning rates to achieve a better performance in gradient optimization, and have as a result become a widely popular optimizer in machine learning. In Figure 4.2 is the official pseudocode presented in the paper.

---

**Algorithm 1:** *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation.  $g_t^2$  indicates the elementwise square  $g_t \odot g_t$ . Good default settings for the tested machine learning problems are  $\alpha = 0.001$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  and  $\epsilon = 10^{-8}$ . All operations on vectors are element-wise. With  $\beta_1^t$  and  $\beta_2^t$  we denote  $\beta_1$  and  $\beta_2$  to the power  $t$ .

---

**Require:**  $\alpha$ : Stepsize  
**Require:**  $\beta_1, \beta_2 \in [0, 1)$ : Exponential decay rates for the moment estimates  
**Require:**  $f(\theta)$ : Stochastic objective function with parameters  $\theta$   
**Require:**  $\theta_0$ : Initial parameter vector  
 $m_0 \leftarrow 0$  (Initialize 1<sup>st</sup> moment vector)  
 $v_0 \leftarrow 0$  (Initialize 2<sup>nd</sup> moment vector)  
 $t \leftarrow 0$  (Initialize timestep)  
**while**  $\theta_t$  not converged **do**  
   $t \leftarrow t + 1$   
   $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$  (Get gradients w.r.t. stochastic objective at timestep  $t$ )  
   $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$  (Update biased first moment estimate)  
   $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$  (Update biased second raw moment estimate)  
   $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$  (Compute bias-corrected first moment estimate)  
   $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$  (Compute bias-corrected second raw moment estimate)  
   $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$  (Update parameters)  
**end while**  
**return**  $\theta_t$  (Resulting parameters)

---

**Figure 4.2:** Pseudocode of adam, a stochastic optimization algorithm adapted from Diederik P. Kingma [12]

The algorithm introduces two hyperparameters  $\beta_1$  and  $\beta_2$  that is used to weight a momentum term and a root mean square propagation term, and uses those to update the gradients. This results in a consideration of recent gradients (directions) and a momentum change to gain a better estimate on the gradient updates.

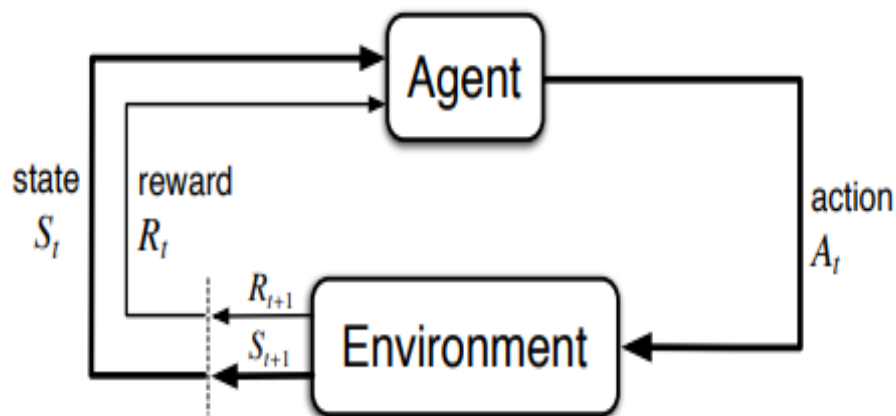
## 4.2 Generative adversarial network

Generative adversarial network (GAN) is a generative model, by that we mean models which models a joint distribution of observed data and the unseen data. This means that we can draw samples from the model that fits within the underlying distribution of the data it have been fitted to. Another example of some popular generative models are an auto-regressive model or an variational autoencoder. GAN consists of two separate artificial neural networks. One network that we call the generator, this is tasked with producing samples  $\mathbf{x} = g(\mathbf{z}; \theta^{(g)})$  ( $\theta$  is a vector). Another network that we call the discriminator is tasked with distinguishing a sample from the generator and our observed data with probability  $d(\mathbf{x}; \theta^{(d)})$ . The two networks are trained together and ties to maximize their own payoffs. The creator of GAN Ian Goodfellow, explains the difficulties of balancing the training, to make sure that neither one of the

networks gets a too big advantage over the other, because this will result in a poor convergence [7]. Goodfellow explains that in an optimal training situation the generator will be able to generate samples that are indistinguishable from the input data.

## 4.3 Reinforcement Learning

Reinforcement learning (RL) is an approach of machine learning that aims to learn from interactions rather than directly learning from labels (supervised learning) or naturally occurring patterns (unsupervised learning).



**Figure 4.3:** The agent-environment interaction in a Markov decision process [27]

We can describe reinforcement in the a Markov decision framework. In figure 4.3 the Agent is the unit of the process that learns and makes decisions/actions based on its learned patterns. The actions at each time step  $A_t$  where  $t \in [0, T]$  interacts with the environment and produces a new state at the next time step  $S_{t+1}$  along with a reward  $R_{t+1}$ . The Agent is again tasked with producing an action based with the goal of maximizing its long term reward [27]. In the next segments we will go through some of the core concepts in RL.

### 4.3.1 What is reinforcement learning?

Reinforcement learning can be described as a science of decision making. This means that we are interested in making a decision based on some input information, and how that decision performs based on our performance critiques. To exemplify we can look at the neuroscience aspect of reinforcement learning,

the brain is presented with its sensory inputs (smell, sight, hearing, taste and touch) and want to take actions like moving a muscle (nervous system outputs) to maximize its dopamine levels. If a human feels cold and moves close to a bonfire this will feel good as the brain will produce dopamine, but if the human moves too close to the heat it might be too warm or even burned and the human will experience the opposite of a good feeling. This interaction of negative and positive rewards based on actions is the core of reinforcement learning.

The aspect of learning from data is what previously mentioned is the main idea behind machine learning, but what does RL bring to the table compared to the two other machine learning methods? Firstly RL negates the use of a supervisor, same as unsupervised learning, there is no supervisor that tells the model how to act or what outputs to learn from. This can often be essential as labels can be expensive and sometimes unobtainable. Secondly RL is a dynamic system, this means that because of its sequential based model it is time dependent and can therefore be used in a time sensitive task, compared to unsupervised learning which only takes into account its current input and performs an association or clustering, an RL system can consider the current output and predict how this will affect future outputs. To exemplify we can consider a portfolio investment system, where the objective is to build a system that finds the best stocks to invest our portfolio in to. A non-dynamic system might see the input stock prices or trends and consider what is a good investment or bad investment based on the current state, but an RL agent have the ability to consider how the stock market will change based on our investment, now this might not have a huge impact for a small investment, but might be important for a substantial large portfolio like the government pension fund of Norway. Thirdly the feedback for an RL system is not instantaneous, the effect of this can vary but in hindsight this delayed feedback property compliments the dynamic system. Delaying the feedback of the system allows the output space to be explored in a more systematic way witch turns out to be an essential part or RL witch we will come back to later.

### 4.3.2 The goal of reinforcement learning

As mentioned above a fundamental quantity of reinforcement learning is the idea of a goal. Since there is no supervisor to guide our neurons we define a reward signal  $R_t \in \mathbb{R}$ . This is a scalar feedback signal for each time step  $t$  that tells our decision algorithm (agent) how well it is doing after each action. Now that the agent achieves a reward number at each time step we can reason that the agents job is to maximise the cumulative reward throughout its trials. This brings us to the reward hypothesis [25]:

[Reward hypothesis] That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).

If we are able to formalize a goal in the means of rewards, we can optimize the actor to achieve that goal. To exemplify, we can describe the goal of winning a chess match as +100 for killing the king. In this example the actor will play the game with only feedback when winning or loosing. It might be smart to give feedback for intermediate actions, like +1 for each chess piece that it kills. The problem with this is that the actor might refuse to kill the king, because it knows that a better outcome for the match is killing all pieces before killing the king. So formalizing the goal is a delicate task, and must be done with great consideration.

If we succeed in defining a reward function that aligns with our goals, how then can we optimize the actor to achieve the best reward possible? As stated in the reward hypothesis, we are looking to archive our goal, by maximization of the expected value of the cumulative sum of a received scalar signal. This means that for each time-step, we have a reward  $R_t$ , resulting in a sequence of rewards  $R_t, R_{t+1} \dots R_n$  where  $n$  is the total amount of time steps. To account for uncertainty in the future (later time steps), we introduce a discount factor  $\gamma$  that weights how much the model should trust its future rewards. In the portfolio investment example that we discussed in the previous section, its not always easy to have a full picture of future outcomes, so to avoid banking all our gold on an uncertain future, we can tune the discount factor to make sure we care mostly about gaining safe rewards at the present (or short in the future). We define this as a return function

$$G_t = R_{t+1} + \gamma R_{t+1} \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (4.7)$$

So to archive our goal we simply need to maximize our return? its not quite that easy, because depending on what state we are in, the future returns may look different, and we are looking to find the actions that gives the most reward given any kind of state. We will explore this in the next segment

### 4.3.3 Policy

In the present thesis we focus on the policy function to map our actions to states, witch in turn will be maximized to gain the most reward. The policy  $\pi(a|s)$  is defined as a probability of taking an action  $A_t = a$  given a present state  $S_t = s$ . The reason to use a stochastic mapping of actions to states is that

this allows for the agent to not always take the action it thinks is the best one. The consequence of this is a natural exploration of seemingly worse actions, that might lead to a resulting better state in the future. But keep in mind the field offers other options like value based methods which we will not go through in the thesis.

As mentioned in section 4.3.2 the expected return is what we are maximizing. A function that measures the expected return of a policy is called the Q-function.  $Q^\pi(s, a)$  is the expected return we get from following the policy  $\pi$  after taking action  $a$  when in state  $s$ . We take advantage of the Bellman optimality equation to find the optimal Q-function:

$$Q^*(s, a) = E[r + \gamma \max_{a'} Q^*(s', a')]$$

Where  $Q^*(s, a)$  is the optimal Q-function,  $r$  is the reward from the action  $a$  and  $\gamma \max_{a'} Q^*(s', a')$  is the discounted maximum Q value from taking the next action  $a'$  from the next state  $s'$

There are several methods to estimate the Q-function that maximises the return, but usually this revolves around finding a set of parameters  $\theta$  that estimates the Q-function  $Q(s, a; \theta)$  because iterating over every sets of actions and states is impractical in complex situations. This method of mapping actions to a Q value and converting the values to a distribution through a squeezing function is called Boltzman Q policy.

In the following section we will go through one popular method of estimating the parameters  $\theta$  called Deep Q-Learning (DQN)

#### 4.3.4 Deep Q-Learning

DQN utilizes an artificial neural network with parameters  $\theta$  as a method to estimate the Q-values  $Q(s, a; \theta)$ . To visualize the process we refer to an illustration by Hongzi Mao [16]



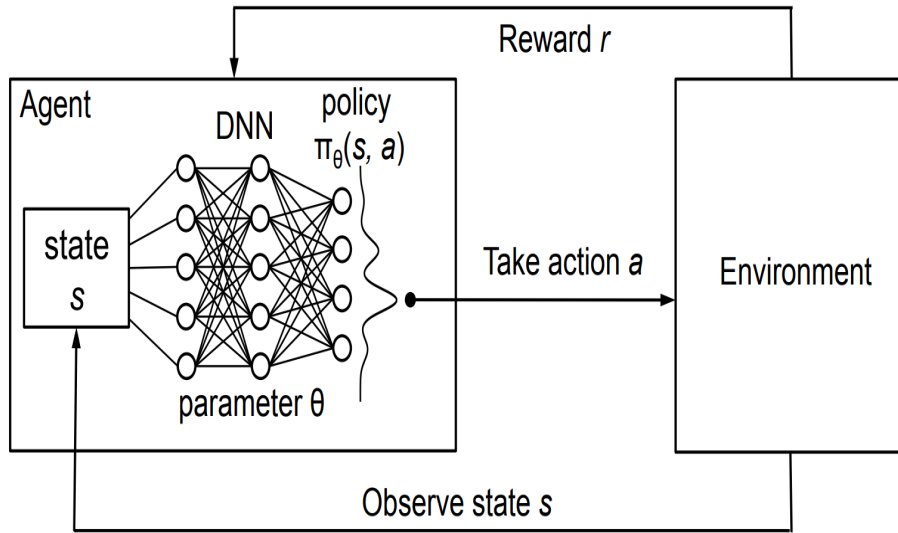


Figure 4.4: A policy based RL using deep artificial neural network [16]

We can see that the network is trained with parameters  $\theta$ , which is used to find the optimal policy. To train the network we refer to Volodymyr Mnih's paper *Playing Atari with Deep Reinforcement Learning*, where the first successful deep q-learning model was presented in 2013.

---

**Algorithm 1** Deep Q-learning with Experience Replay

---

```

Initialize replay memory  $\mathcal{D}$  to capacity  $N$ 
Initialize action-value function  $Q$  with random weights
for episode = 1,  $M$  do
  Initialise sequence  $s_1 = \{x_1\}$  and preprocessed sequence  $\phi_1 = \phi(s_1)$ 
  for  $t = 1, T$  do
    With probability  $\epsilon$  select a random action  $a_t$ 
    otherwise select  $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ 
    Execute action  $a_t$  in emulator and observe reward  $r_t$  and image  $x_{t+1}$ 
    Set  $s_{t+1} = s_t, a_t, x_{t+1}$  and preprocess  $\phi_{t+1} = \phi(s_{t+1})$ 
    Store transition  $(\phi_t, a_t, r_t, \phi_{t+1})$  in  $\mathcal{D}$ 
    Sample random minibatch of transitions  $(\phi_j, a_j, r_j, \phi_{j+1})$  from  $\mathcal{D}$ 
    Set  $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$ 
    Perform a gradient descent step on  $(y_j - Q(\phi_j, a_j; \theta))^2$  according to equation 3
  end for
end for

```

---

Figure 4.5: Pseudocode of Deep Q-learning, adapted from [17]

The gradient decent is preformed by minimizing the loss function at each timestep  $i$ :

$$L_i(\theta_i) = E_{s,a,r,s' \sim p(\cdot)} [(y_i - Q(s, a; \theta_i))^2]$$

where  $y_i = r + \gamma \max_{a'} Q^*(s', a'; \theta_{i-1})$  is the temporal difference target, and the difference  $y_i - Q$  is the error.  $p(\cdot)$  is the behaviour distribution over the transition from the present state to the new state given the action. The differential with respect to the network parameters (weights) is:

$$\Delta_{\theta_i} L_i(\theta_i) = E_{s,a \sim p(\cdot); s' \sim \epsilon} [(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})) \Delta_{\theta} Q(s, a; \theta_i)]$$

This equation is referred to equation 3 in the pseudocode above.

# /5

## Statistics and Mathematics

In this segment we will go through some important mathematical and statistical theory that are relevant for the present thesis.

### 5.1 Numerical integration

Numerical integration is an iterative method of approximating an integral. There are many different numerical integration methods in the field of numerical analysis. One of those methods is the Kunge-Kutta method. We use this one in the thesis, because of its reliability and its straight forwardness.

Rune-Kutta of order four is approximated by:

$$w_{i+1} = w_i + \frac{h}{6}(s_1 + 2s_2 + 2s_3 + s_4)$$

Where h is a step size and  $w_0$  is an initial value.  $s_1..s_4$  is:

$$\begin{aligned}
s_1 &= f(t_i, w_i) \\
s_2 &= f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2}s_1\right) \\
s_3 &= f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2}s_2\right) \\
s_4 &= f(t_1 + h, w_i + hs_3)
\end{aligned}$$

## 5.2 Nonlinear grey-box model

A nonlinear grey-box model estimation is a tool in Matlab's optimization toolbox that estimates coefficients of linear and nonlinear differential equations. In addition to this, it can estimate difference equations and state-space equations. The model formalizes an input-output setup that is compatible with an IDdata object. The model has a wide variety of analytical tools and estimation tools. In the next section we will go through an optimization that was used in the present thesis to estimate parameters in our generative model.

## 5.3 Trust-Region-Reflective Least Squares

Trust-Region-Reflective Least Squares is an optimization method that is widely used in Matlab's Optimization toolbox. The method considers a function  $f(x)$  and approximates a simpler function  $q$  that reasonably reflects the behaviour of  $f(x)$  in a neighbourhood  $N$  around the point  $x$ . This is similar to what we do in variational inference, but instead of approximating a distribution that is equal to the original (incomputable) distribution, we consider a trial sample  $s$  of the new function  $q(s)$  that approximates the numerical neighbourhood  $N$  of our original function  $f(x)$ .

The minimization can be computed by considering trial samples  $s$  and minimizing over  $N$  showed in the trust-region sub-problem:

$$\min_s(q(s), s \in N)$$

A new sample  $s$  can be computed solving the equation for  $s$ :

$$\min\left(\frac{1}{2}s^T Hs + s^T g \text{ such that } \|Ds\| < \Delta\right)$$

Note that  $N$  is an area (usually spherical or ellipsoidal) so  $s^T$  is the transpose of the vector.  $H$  is the symmetric matrix of the second derivatives (Hessian matrix), and  $g$  is the gradients of  $f(x)$  and  $D$  is a diagonal scaling matrix. We denote  $\|Ds\|$  as the second norm of our scaled sample point  $s$ , and  $\Delta$  is some positive scalar.

If  $f(x + s) < f(x)$  the current point is updated as  $x + s$ , this is iterated until convergence. Note that if at some point the iteration problem struggles with finding new updates,  $\Delta$  can be adjusted to increase the search region [28][3].

## 5.4 Simulated annealing

Simulated Annealing is an optimization technique, that stems from a physical process called annealing (which is heating up a solid then cooling it slowly). The technique is a stochastic process where a random candidate (action in our case) is proposed in the neighbourhood of the previous candidate. If the proposal gives a better result (a better return of rewards in our case) it is accepted as the new candidate, however even if it yields a worse result the candidate can still be accepted with a probability of  $\frac{1}{1 + \exp(\frac{\Delta}{T_i})}$ , where  $\Delta$  is the difference between the old action and the new proposed action.  $T_i$  is a temperature parameter that changes through the iteration process with  $T_{i+1} = T_i * 0.95^k$ , where  $k$  is the annealing parameter set to be the same as the current iteration number  $i$  [6].

When  $i$  increases, the acceptance probability of the new actions that are worse decreases. The idea behind this is to avoid local optimums in the earlier iterations to steer towards the global optimum.

Note that there are other options for the acceptance probability distribution and the parameters, the ones that were discussed in this section are used in the final results because they are considered as default by Matlab's global Optimization toolbox [8].

## **Part III**

# **Experiments and Methodology**



# /6

## In-silico BG Simulation

As explained in section 3.5, Breton's physical activity model attempts to predict the BG values of a patient with the given inputs: CHO, HR and insulin. We aim to use this model to predict the BG over a period of several hours with different kinds of inputs. With a sufficient amount of simulations we can then optimize the CHO input (i.e. give a food recommendation) so that the predicted BG values will be as safe as possible for a given workout.

The problem with using general/population estimations of the parameters is explained in section 3.4. We attempt to estimate the parameters using data from one patient to make sure that the model is as personalized as possible. This means that the results of the model is only applicable to this specific patient. The results of the model is only applicable for one patient. The methodology in the thesis can be applied to any other patients.

### 6.1 Model setup

The model was set up in regards to [18], but some changes were made to generalize it so that the patient could use the food recommendation for any type of workout. The dynamics of the model are as follows:



$$\frac{dG}{dt} = -p_1 \cdot (G - G_b) - (1 + \alpha \cdot w \cdot Z) \cdot X \cdot G - \beta \cdot Y \cdot G + \frac{U_G}{V_G} \quad (6.1)$$

$$\frac{dY}{dt} = \left( -\frac{Y}{\tau_{HR}} + \frac{HR(t)}{\tau_{HR}} \right) \quad (6.2)$$

$$\frac{dD_1}{dt} = A_G \cdot D(t) - \frac{D_1}{\tau_G} \quad (6.3)$$

$$\frac{dD_2}{dt} = \frac{D_1}{\tau_G} - \frac{D_2}{\tau_G} \quad (6.4)$$

$$\frac{dX}{dt} = -p_2 \cdot X + p_3 \cdot I_c \quad (6.5)$$

$$\frac{dZ}{dt} = -\left( \frac{f(Y)}{\tau_{in}} + \frac{1}{\tau_{ex}} \right) \cdot Z + f(Y) \quad (6.6)$$

$$\frac{dx_1}{dt} = (-k_{21} \cdot x_1) + u(t) \quad (6.7)$$

$$\frac{dx_2}{dt} = k_{21} \cdot x_1 - (k_d + k_a) \cdot x_2 \quad (6.8)$$

$$\frac{dI_c}{dt} = \frac{k_a}{V_G \cdot BW} \cdot x_2 - k_e \cdot I_c \quad (6.9)$$

The 3 inputs to the model are set to be time series of heart rate  $HR(t)$ , bolus insulin injections  $u(t)$  and carbohydrates intake  $D(t)$ . Both  $u(t)$  and  $D(t)$  have an associated integral scaling parameter ( $I_{scale}$  and  $CHO_{scale}$ ) to make sure that the area under the curve is scaled correctly.

The main addition to Breton's model is a the description of insulin kinetics after an injection.  $\frac{dx_1}{dt}$ ,  $\frac{dx_2}{dt}$  and  $\frac{dI_c}{dt}$  is a model described in [20]. This model explains the delayed effect when injecting insulin.

Table 6.1 lists all 23 parameters, their initial values, units and if they are set to be free or fixed in the optimization.

**Table 6.1:** Table of parameters used in generative model and their population estimates from [18].

Parameter	Value	Unit	Free/Fixed
$p_1$	0.0040	$\frac{1}{min}$	Free
$p_2$	0.028	$\frac{1}{min}$	Free
$p_3$	$8.6779 \cdot 10^{-4}$	$\frac{ml}{min^2 \cdot \mu U}$	Free
$G_b$	80	$\frac{mg}{dl}$	Fixed
$\tau_{HR}$	5	min	Fixed
$HR_b$	70	bpm	Fixed
$\tau_G$	40	min	Fixed
$A_G$	0.8	dimensionless	Fixed
$\beta$	$8.31 \cdot 10^{-5}$	$\frac{1}{bpm}$	Free
$\alpha$	1.3381	dimensionless	Free
n	4	dimensionless	Fixed
a	0.1	dimensionless	Fixed
$\tau_{in}$	1	min	Fixed
$\tau_{ex}$	600	min	Fixed
$V_G$	$2.028 \cdot 60$	$\frac{dl \cdot kg}{kg}$	Fixed
$CHO_{scale}$	dimensionless	min	Free
w	49.9972	dimensionless	Fixed
$k_{21}$	0.0013	$\frac{1}{min}$	Free
$k_d$	0.0247	$\frac{1}{min}$	Free
$k_a$	$1.5654 \cdot 10^{-5}$	$\frac{1}{min}$	Free
$k_e$	0.0357	$\frac{1}{min}$	Free
BW	60	kg	Fixed
$I_{scale}$	$15 \cdot 18$	dimensionless	Free

The values was set to be free or fixed as a result of experimentation, unfortunately because of computational constraints we could not optimize all unknown parameters.

The model was set up with all initial values and numerically integrated over the duration of the simulation as explained in section 5.1. The BG values were evaluated every five minutes and the resulting BG time series were stored along with the inputs for further processing.

## 6.2 Workout data from patient

To personalize the model, we gathered BG measurements, bolus insulin injections and CHO from the patients CGM. We also gathered the heart rate history during exercise from the patients heart rate monitor. The heart rate before the exercise was assumed to be stable at  $70bpm$ .

The patient was then subjected to high intensity workouts and the aforementioned information was gathered. All of the data points were gathered from the same type of workout, with different inputs and initial BG values.

In Figure 6.1 we show an example of what the CGM's information looks like (mentioned above). Note that the CHO as you can see in the bottom row in orange, was registered manually, therefore it may be prone to some uncertainty. This explicit example is not during a registered exercise so it was not used in the parameter fitting procedure, this is showed for illustrative purposes only.



Figure 6.1: Data illustration example.

The problem with the data that were gathered is that it is only from safe situations, so we do not have any information about what would happen in unsafe situations as this would be highly unethical. The main idea was that by using the safe data we are able to capture the behaviour of the blood glucose dynamics and apply this to simulate situations that are unsafe.

### 6.3 Fitting the model

Firstly, a Matlab file was made where inputs, parameters and the patient data were defined. A new Matlab file was then created defining the model explained in section 6.1. The two files were combined in a Nonlinear grey-box model explained in section 5.2. The parameters were defined as free or fixed, and the free parameters were set with a lower bound value of 0.0000000000001.

The nonlinear grey-box model was made using matlab's `nlgreyest` function with max iterations set to 150 and optimization algorithm set to trust-region-reflective least squares (explained more in detail in section 5.2, 5.3 and 8.1)

This was repeated for the entire training data set and every estimation was stored and used to estimate a normal distribution for every free parameter. The resulting goodness-of-fit measure was calculated as a performance measure.



# Decision Support Systems

## 7.1 Reinforcement Learning

The custom environment was set up with the generative model explained in section 6.1. Observation space is a continuous space  $\in [0, 400] \frac{mg}{dl}$ . This means that the state can be any BG value in this interval. The initial state is set to the initial BG value at the start of the simulation. To generalize the model some noise is added to this initial value.

The action space is defined as a discrete space  $\in [0, 80]g$ . This upper value is set because it would be unreasonable to consume more carbs than this, as it would result in an extreme hyperglycemic state for the patient for the types of workouts that are accounted for in the thesis.

The environment consists of, among other things, the required elements for openAI's API gym. This is done for the purpose of an easy implementation of the agent through the API. The required elements include an initializing function, a step function, a render function, a reward function and a reset function. We will go through these parts in the following section, a more detailed description can be found in openAI's documentation and in section 4.3.

### 7.1.1 Environment elements

The initialization function defines the state and action spaces using gym's built in functions. Any additional information is initialized, like simulation length, indexations, parameters and storage arrays for the inputs and outputs.

The step function is defined as a computation of responses for each step/action in the environment. It takes an action as an input and calculates the next state. This is done through the model that was explained in section 6.1. The model is fed with the correct values of input through the correct indexations, and a numerical integration is performed throughout the simulation. For each step, a reward is calculated, and the state, reward, done and info is returned.

The render function is empty as there was no need for any rendering during the training. Usually this function is called to visualise changes through the training. To exemplify, a simulation of a chess board with the corresponding actions and rewards.

The reward function outputs a reward for any given state. This was adapted from a recent paper that explored a risk analysis of BG regulation [18] on virtual patients. This function was chosen to use as a starting point for a performance measure, with plans to expand/tweak by collaboration with medical experts:

$$r(BG) = \begin{cases} -10 & BG \leq BG_{hypo-} \\ \exp\left(\frac{\log(19.157)}{BG_{hypo}}BG\right) - 19.157 & BG \in [BG_{hypo-}, BG_{hypo}] \\ \frac{1}{36}BG - 2 & BG \in [BG_{hypo}, BG_{ref}] \\ -\frac{1}{72}BG + \frac{5}{2} & BG \in [BG_{ref}, BG_{hyper}] \\ -5 & BG \geq BG_{hyper} \end{cases}$$

Where:

- $BG_{ref} = 108 \frac{mg}{dL}$  Reference blood glucose
- $BG_{hyper} = 180 \frac{mg}{dL}$  Hyperglycemia blood glucose
- $BG_{hypo} = 72 \frac{mg}{dL}$  Hypoglycemia blood glucose
- $BG_{hypo-} = 54 \frac{mg}{dL}$  Severe hypoglycemia blood glucose

The reset function outputs a state and resets all variables to their initial values.

In addition it applies noise in the initial condition.

### 7.1.2 Agent

The agent used is deep q-learning, as explained in section 4.3.4. The policy function is BoltzmanQPolicy, as explained in section 4.3.3. The agent is model-based using a small fully connected neural network with two hidden layers having 24 neurons each. The input neurons are the same as the amount of states, and the number of output neurons is the same as actions i.e. 80.

## 7.2 Simulated annealing

To optimize the CHO input to make sure the BG value will be as close as possible to normoglycemic, a simulated annealin optimization was preformed. This was done by using the Global Optimization Toolbox from matlab.

The reward function was set up as a function of CHO intake. The reward function then computes a simulation with the given CHO input and computes the reward. A lower and upper bound was set to be  $\in [0, 80]$  and the `simannealbnd` function was called to optimize the CHO input.

## **Part IV**

# **Results and Discussion**



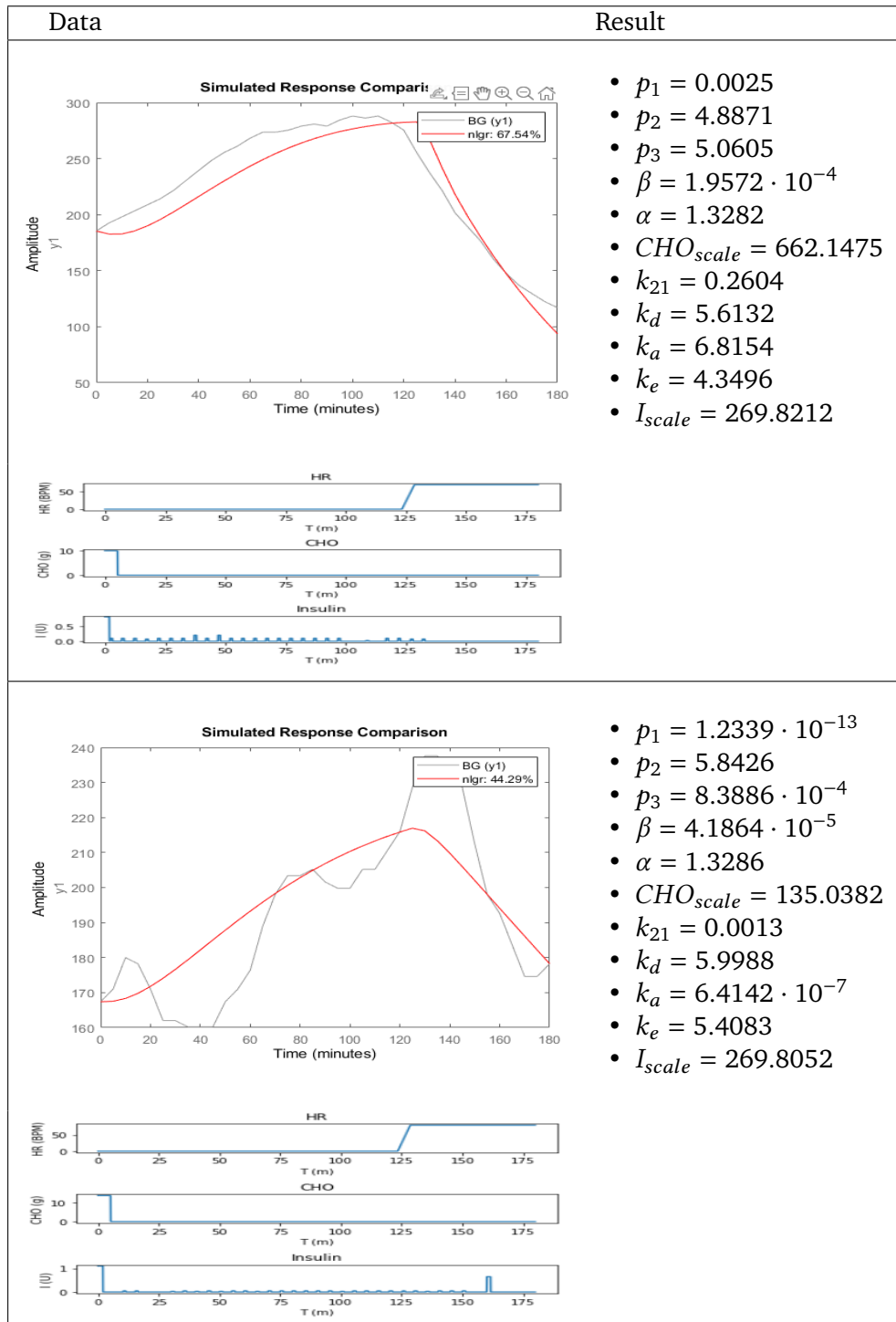


# / 8

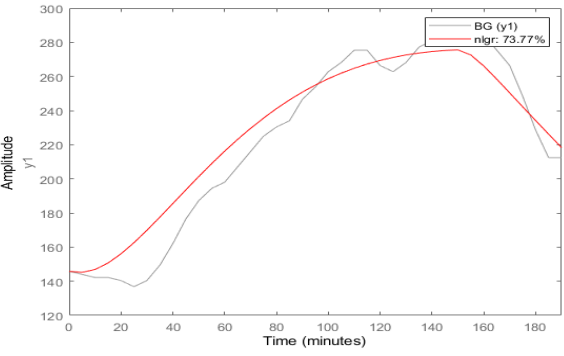



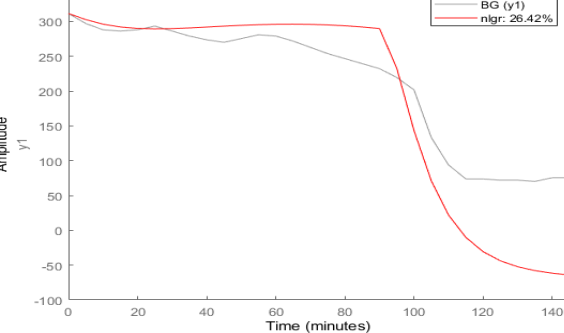
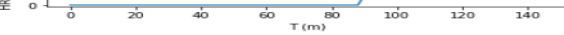
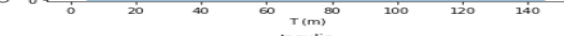

## Generative Model

### 8.1 Estimation of the model's parameters

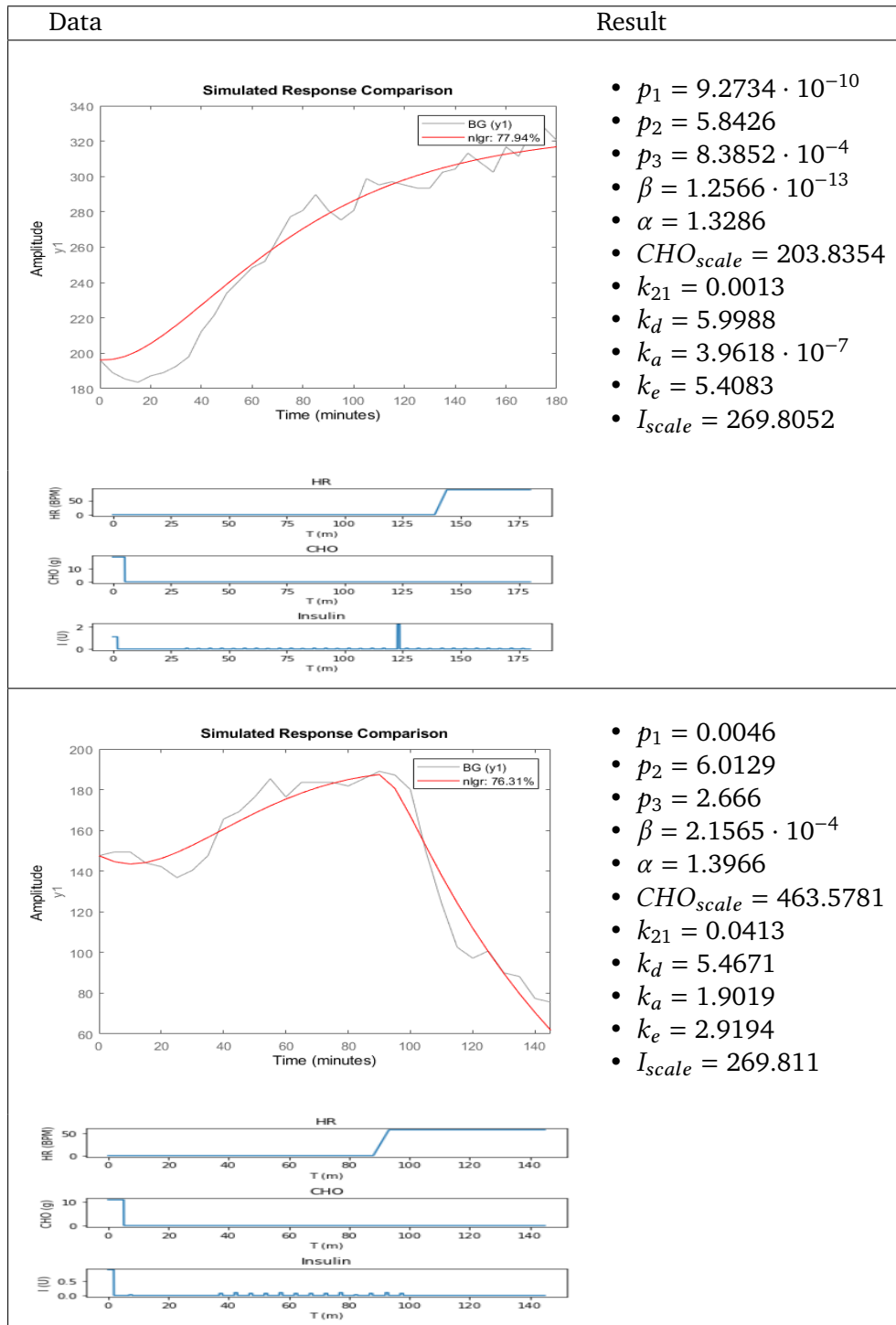
In this section we present the optimization Tables for the free parameters in the generative model. In the left coulomb of the tables we have plots of the optimisation, comparing the recorded event (grey graph) and the simulated event (red graph). Under the simulation we present the inputs of the event, three time series of HR, CHO and bolus insulin injections. In the right coulomb of the figures we list the optimized parameters from that event. Lastly we present a figure of the resulting distributions from all of the episodes.



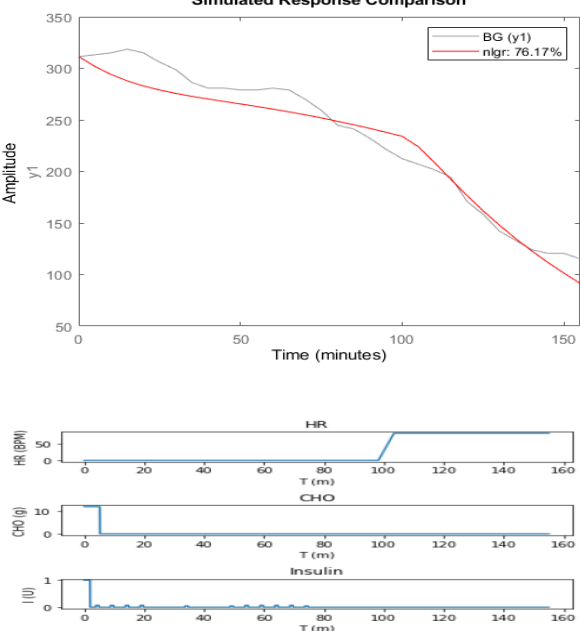
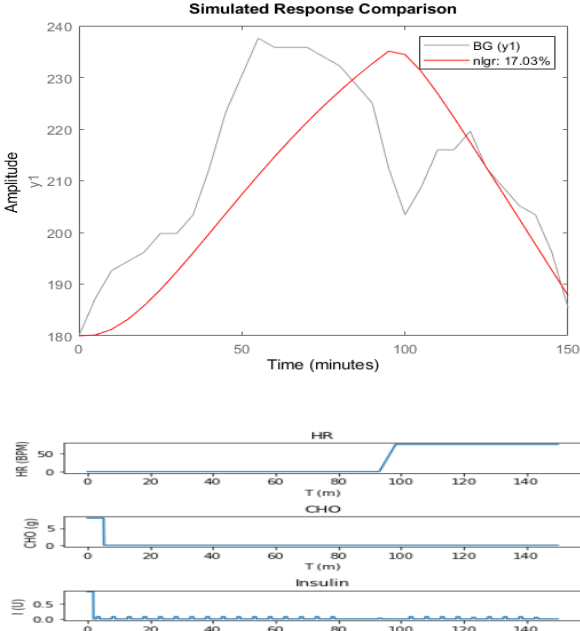
**Table 8.1:** Training result: A plot of the event showing the recorded BG history in grey, and the simulation result in red. The Y axis is BG in  $\frac{mg}{dl}$  and the x axis is time in minutes. Three plots of the inputs for the events, HR in  $bpm$ , CHO in grams and insulin in units. A list of each optimized parameter and the resulting value.

Data	Result
<p data-bbox="430 424 998 459"><b>Simulated Response Comparison</b></p>  <p data-bbox="430 849 990 883"><b>HR</b></p>  <p data-bbox="430 929 990 964"><b>CHO</b></p>  <p data-bbox="430 1010 990 1044"><b>Insulin</b></p> 	<ul style="list-style-type: none"> <li>• <math>p_1 = 0.0015</math></li> <li>• <math>p_2 = 5.8426</math></li> <li>• <math>p_3 = 8.0001 \cdot 10^{-4}</math></li> <li>• <math>\beta = 5.8203 \cdot 10^{-5}</math></li> <li>• <math>\alpha = 1.3286</math></li> <li>• <math>CHO_{scale} = 281.7148</math></li> <li>• <math>k_{21} = 0.0228</math></li> <li>• <math>k_d = 5.9988</math></li> <li>• <math>k_a = 0.0059</math></li> <li>• <math>k_e = 5.4083</math></li> <li>• <math>I_{scale} = 269.8052</math></li> </ul>
<p data-bbox="430 1125 998 1159"><b>Simulated Response Comparison</b></p>  <p data-bbox="430 1549 990 1584"><b>HR</b></p>  <p data-bbox="430 1630 990 1664"><b>CHO</b></p>  <p data-bbox="430 1710 990 1744"><b>Insulin</b></p> 	<ul style="list-style-type: none"> <li>• <math>p_1 = 0.0056</math></li> <li>• <math>p_2 = 5.8426</math></li> <li>• <math>p_3 = 0.0095</math></li> <li>• <math>\beta = 0.0011</math></li> <li>• <math>\alpha = 1.3286</math></li> <li>• <math>CHO_{scale} = 361.2663</math></li> <li>• <math>k_{21} = 0.0091</math></li> <li>• <math>k_d = 5.9988</math></li> <li>• <math>k_a = 10</math></li> <li>• <math>k_e = 5.4083</math></li> <li>• <math>I_{scale} = 269.8052</math></li> </ul>

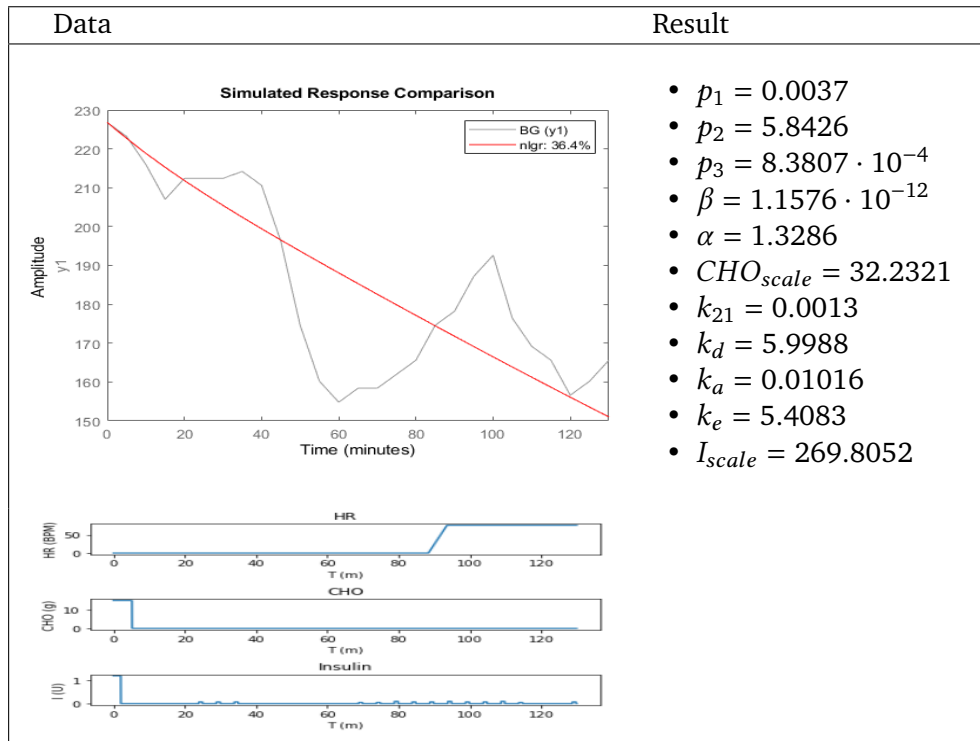
**Table 8.2:** Training result: A plot of the event showing the recorded BG history in grey, and the simulation result in red. The Y axis is BG in  $\frac{mg}{dl}$  and the x axis is time in minutes. Three plots of the inputs for the events, HR in *bpm*, CHO in grams and insulin in units. A list of each optimized parameter and the resulting value.



**Table 8.3:** Training result: A plot of the event showing the recorded BG history in grey, and the simulation result in red. The Y axis is BG in  $\frac{mg}{dl}$  and the x axis is time in minutes. Three plots of the inputs for the events, HR in  $bpm$ , CHO in grams and insulin in units. A list of each optimized parameter and the resulting value.

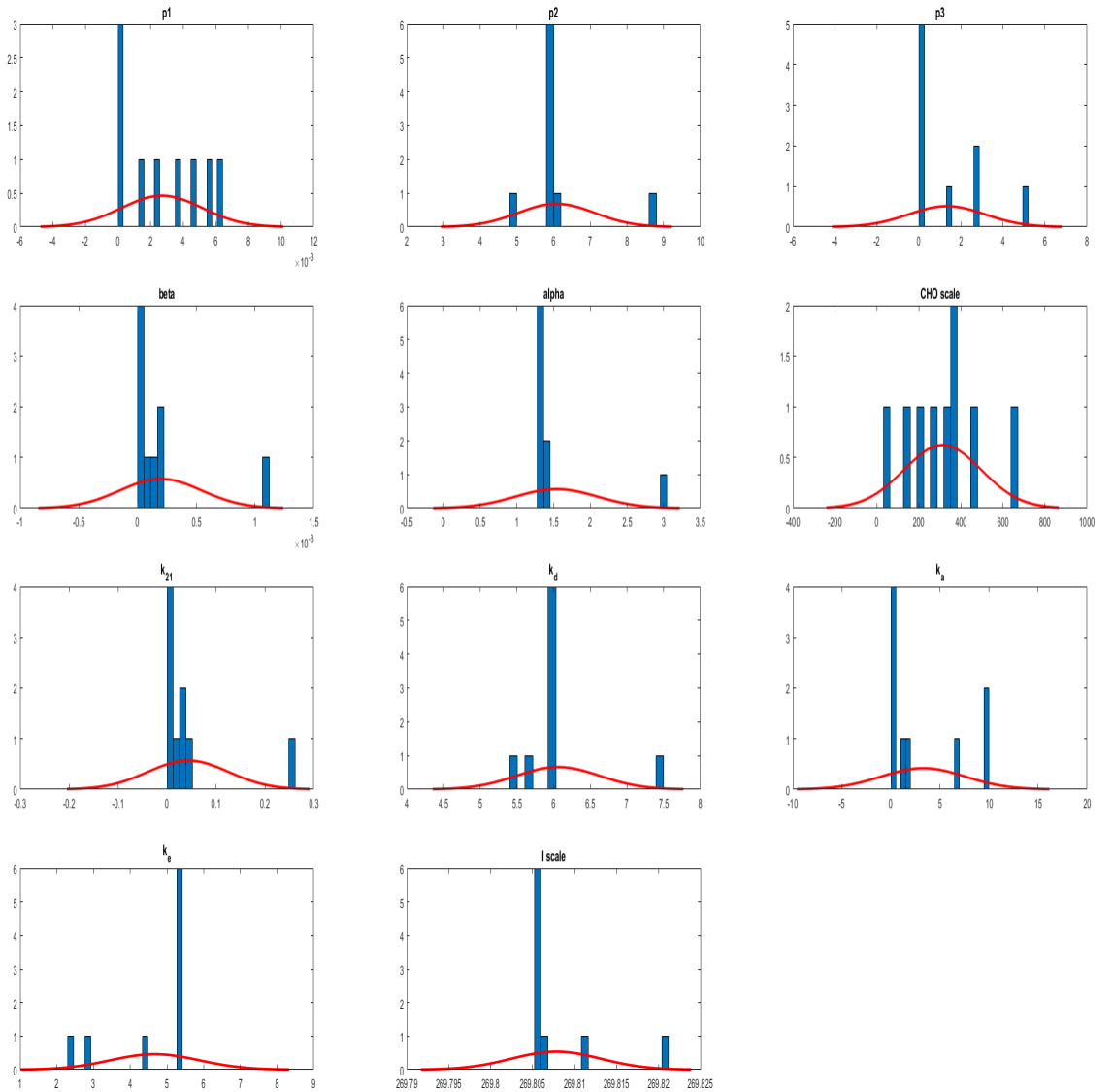
Data	Result
<p style="text-align: center;"><b>Simulated Response Comparison</b></p> 	<ul style="list-style-type: none"> <li>• <math>p_1 = 0.0063</math></li> <li>• <math>p_2 = 5.8236</math></li> <li>• <math>p_3 = 2.7522</math></li> <li>• <math>\beta = 1.1401 \cdot 10^{-4}</math></li> <li>• <math>\alpha = 1.4548</math></li> <li>• <math>CHO_{scale} = 351.112</math></li> <li>• <math>k_{21} = 0.0266</math></li> <li>• <math>k_d = 5.9916</math></li> <li>• <math>k_a = 9.6366</math></li> <li>• <math>k_e = 5.3872</math></li> <li>• <math>I_{scale} = 269.8056</math></li> </ul>
<p style="text-align: center;"><b>Simulated Response Comparison</b></p> 	<ul style="list-style-type: none"> <li>• <math>p_1 = 1.569 \cdot 10^{-5}</math></li> <li>• <math>p_2 = 8.7319</math></li> <li>• <math>p_3 = 1.4717</math></li> <li>• <math>\beta = 4.9324 \cdot 10^{-5}</math></li> <li>• <math>\alpha = 3.0282</math></li> <li>• <math>CHO_{scale} = 322.4433</math></li> <li>• <math>k_{21} = 0.0262</math></li> <li>• <math>k_d = 7.4817</math></li> <li>• <math>k_a = 1.1944</math></li> <li>• <math>k_e = 2.3279</math></li> <li>• <math>I_{scale} = 269.8064</math></li> </ul>

**Table 8.4:** Training result: A plot of the event showing the recorded BG history in grey, and the simulation result in red. The Y axis is BG in  $\frac{mg}{dl}$  and the x axis is time in minutes. Three plots of the inputs for the events, HR in *bpm*, CHO in grams and insulin in units. A list of each optimized parameter and the resulting value.



**Table 8.5:** Training result: A plot of the event showing the recorded BG history in grey, and the simulation result in red. The Y axis is BG in  $\frac{mg}{dl}$  and the x axis is time in minutes. Three plots of the inputs for the events, HR in  $bpm$ , CHO in grams and insulin in units. A list of each optimized parameter and the resulting value.

Figure 8.1 present a fitted normal distribution for each of the parameters and the histogram of the values from the training events that was presented above.



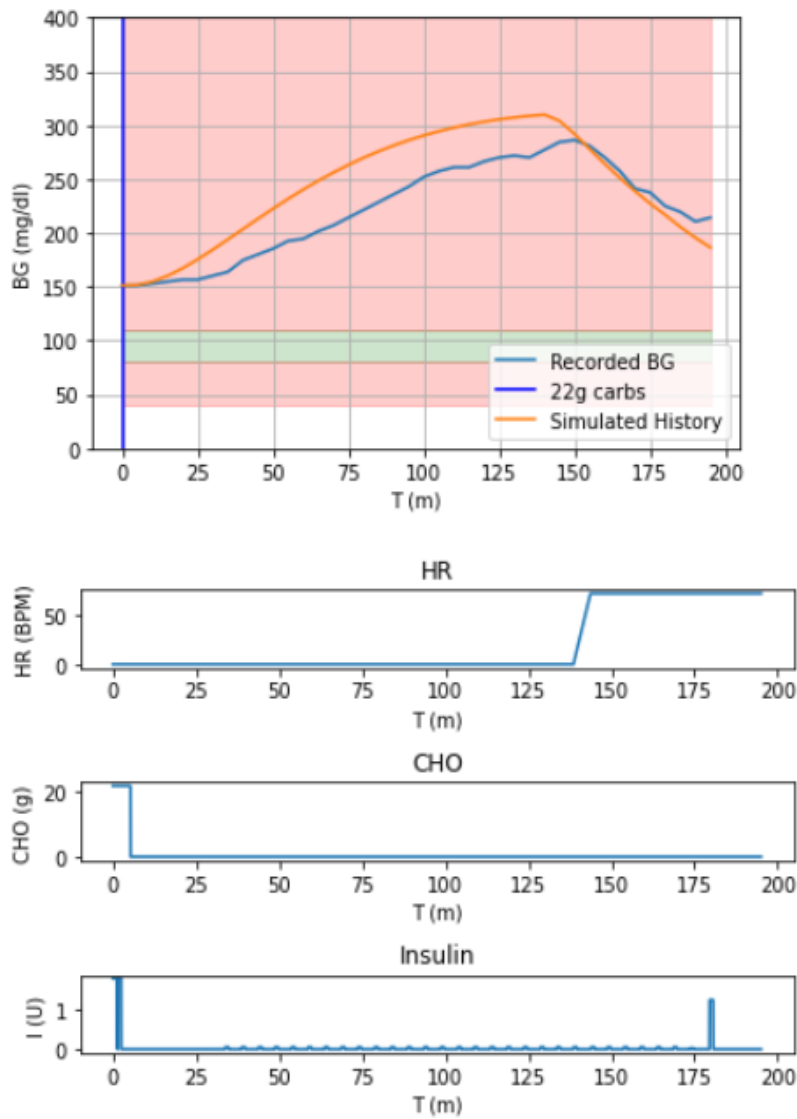
**Figure 8.1:** 11 subplots, one for each optimized parameter. Each plot contains a histogram of the parameter and a fitted normal distribution

## 8.2 Validation

In this section we present the validation result of the generative model. Out of all 11 excesses events two were randomly chosen to be used for validation, the remaining 9 were used in training showed in the previous section. In Figure 8.2 and 8.3 there are tree highlighted areas, the green are is the



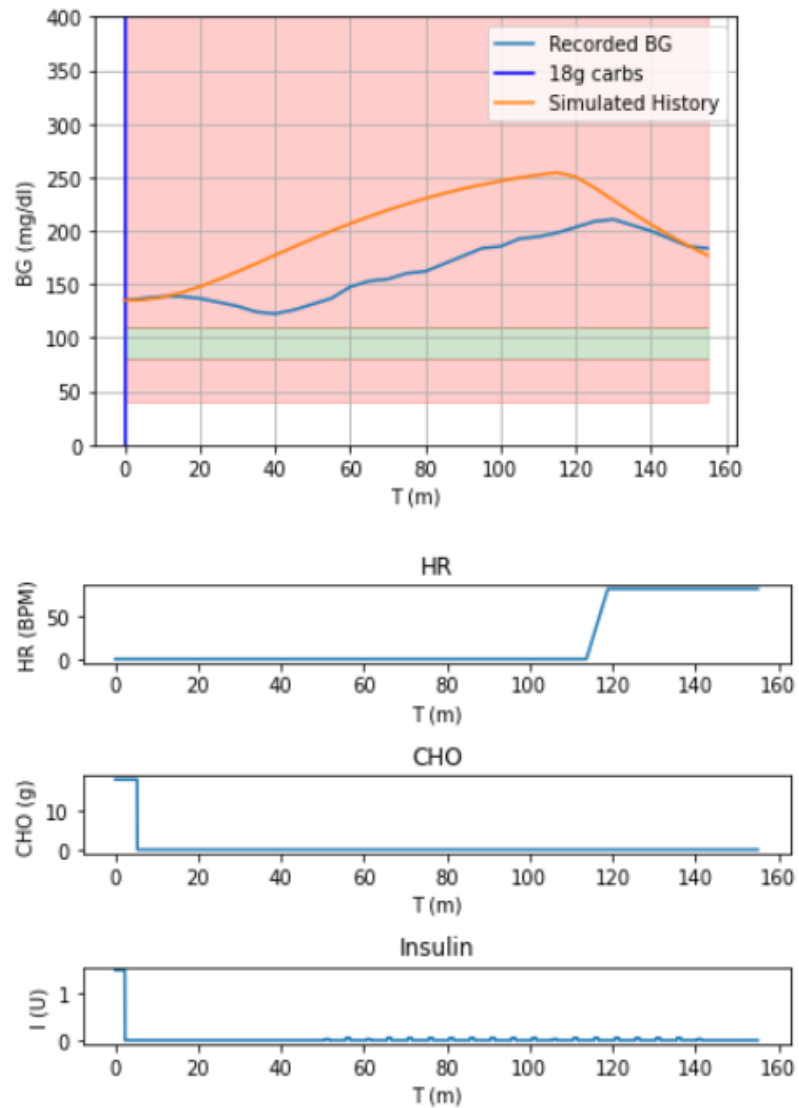
normoglycemic interval and the red areas over and under are hyperglycemic and hypoglycemic ranges respectively. The orange graphs are the simulated BG history in  $\frac{mg}{dl}$  using the expectation of the normal distribution for each parameter shown in Figure 8.1. The blue graphs are the recorded BG in  $\frac{mg}{dl}$  for the recorded history of the excesses event with the inputs for the events, HR in *bpm*, CHO in grams and insulin in units.



**Figure 8.2:** Validation result: A plot of the the event showing the recorded BG history in blue, and the simulation result in orange.

The resulting goodness of fit measure between the predicted values and the

true values is  $R^2 = 0.497$ .



**Figure 8.3:** Validation result: A plot of the the event showing the recorded BG history in blue, and the simulation result in orange.

The resulting goodness of fit measure between the predicted values and the true values is  $R^2 = -1.4715$ .

## 8.3 Discussion

As we can see in figure 8.2 and 8.3 the simulations seem to follow the general behaviour of the label, by increasing the BG in response to the CHO intake in the beginning of the simulation and decreasing as a response of the increase in HR at the end of the simulations. The magnitude of peak is closer in figure 8.2 and misses by approximately  $40 \frac{mg}{ml}$  in figure 8.3. Both ends of the simulations are a close to the respective label which indicates that the energy expenditure part of the simulation is captured by the model. In Figure 8.3 we see a small dip in BG shortly after the meal intake (approximately at 30 to 50 minutes), this dip is not reflected in the simulated BG values. The most likely reason for this is that the model is not able to properly capture the effect of the insulin bolus.

One problem with the results is that there are only two validation data points. This is because there are few total data points that were gathered resulting in few validation points. This is not optimal as it is hard to account for outliers in the result. In addition to this, having only 9 data points to fit the model on is not great because the resulting parameter distribution shown in figure 8.1 may not give the optimal expectation.

It is important to address one of the major assumptions in this thesis. It is assumed that the patient is at rest  $HR = 70$  between eating and workout. This was done because there was no heart rate data collected in this time period. It is easy to imagine that this might not always be the case, for instance if the patient happens to be running to the bus or similar situations. This will cause the resulting data point to introduce a bias in the fitting procedure, or a bad performance measure in a validation data.

The last point to make about the data is that the CHO inputs have been manually registered by the patient. This adds additional uncertainty in the generative model which permutes through the optimization part.

Even though there were some problems with the data, they do not seem to explain all of the validation error. It is clear that model fitting is having problems (can be seen in table 8.4 and 8.5) with large variety. This is likely due to the insulin behaviour not being captured well by the model. This might be a computation problem, or an inherent problem with the model.

It is hard to tell if there is some underlying problem with the model, or if there are too much uncertainty in the data. One possible way to figure this out would be to perform a large-scale test with enough computational power to run every training data until convergence. This would be a very resource heavy test, so perhaps it's better to look for other alternatives.

One of the more promising emerging/forthcoming generative tools is the generative adversarial model explained in section 4.2. This might be a viable option in this situation, because it is not restricted to relying on specific equations with parameters to explain the behaviour. This might be imperative in a complicated situation like this because other factors like hormonal patterns or cycles are not accounted for in the differential equations.

A large benefit one would have to sacrifice by switching to a neural network is the explainability and interpretability of the model, as factoring the model into individual bits like energy expenditure, insulin behaviour, etc., would be a challenge in such a network. With that being said, there is forthcoming research in that department that could prove to contradict this argument.

With all this in mind, the model does prove to provide a general explanation of the BG dynamics. With a very cheap fine-tuning, one could argue that this is the best option in this scenario.

# /9

## Food Recommendation

The food recommendation will be given in the same situation as in the validation data in figure 8.2, this is to gain a reference point (the food intake that the patient actually had). The methodology works for any kind of situation as long as the inputs are corrected accordingly.

### 9.1 RL

The agent was set to train for 300 episodes with a learning rate of 0.001. Figures 7.1.1 and 9.2 shows the reward and average reward over the training period. The resulting action was 7g of CHO.

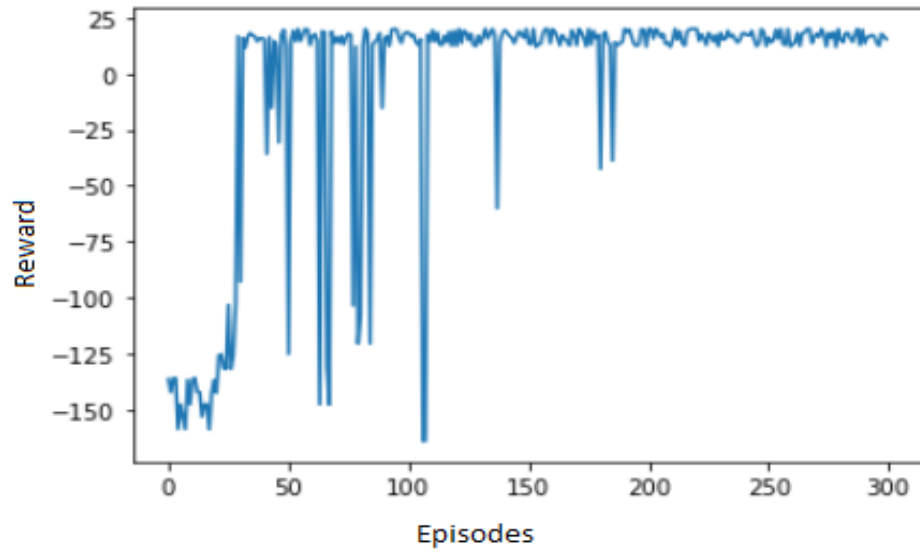


Figure 9.1: Reward vs. Episodes

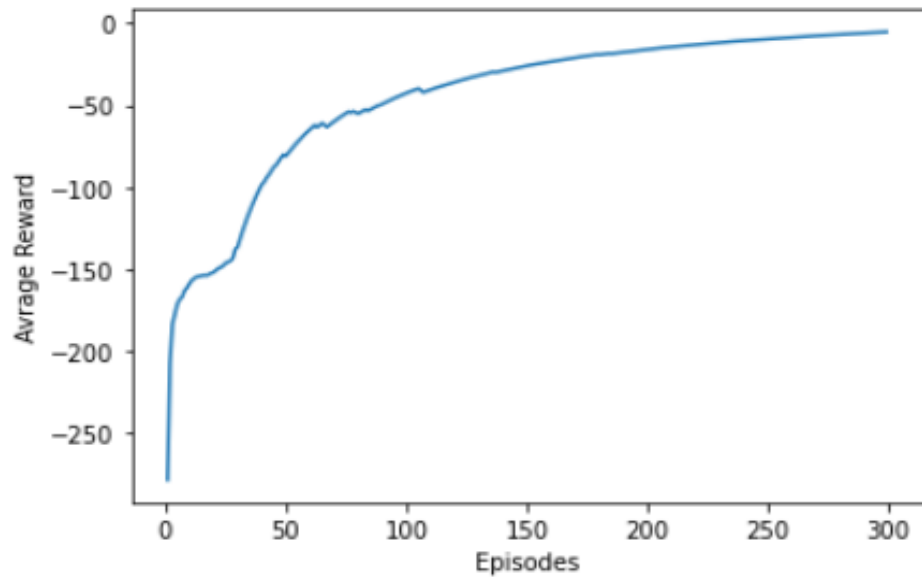


Figure 9.2: Average Reward vs. Episodes

## 9.2 Simulated annealing

The simulated annealing optimization converged after 185 seconds on an action of 10.8369, figure 9.3 shows the output information.

```
>> Simulated_anneal
Optimization terminated: change in best function value less than options.FunctionTolerance.

x =

    10.8369

fval =

    22.9528

exitFlag =

     1

output =

  struct with fields:

    iterations: 866
    funccount: 868
    message: 'Optimization terminated: change in best function value less than options.FunctionTolerance.'
    rngstate: [1x1 struct]
    problemtype: 'boundconstraints'
    temperature: 0.3609
    totaltime: 185.6994
```

**Figure 9.3:** Simulated Annealing

## 9.3 Comparison

Both results are plotted in figure 9.4 together with the 18g of CHO that the patient consumed in this scenario.

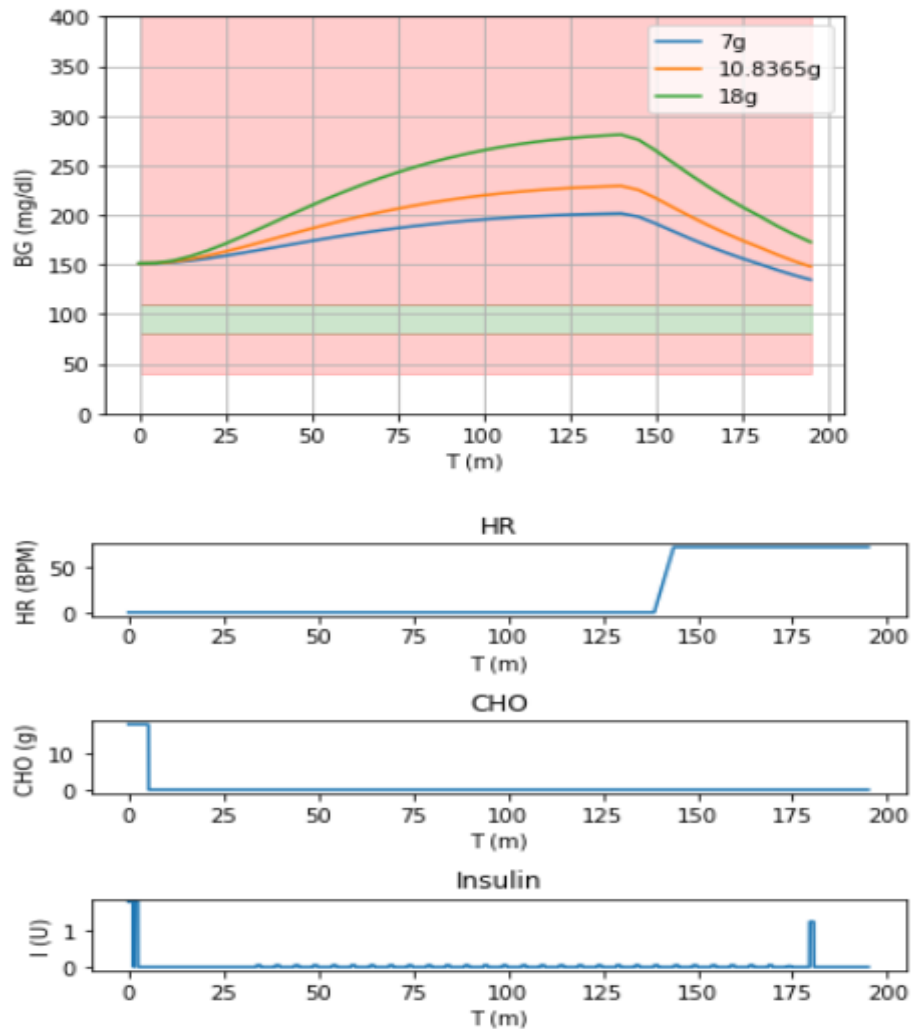


Figure 9.4: Comparison of results for the optimization of BG regulation

## 9.4 Discussion

The RL training converged at around 200 episodes as you can see in figure 9.1. This took approximately 10 hours with a conventional computer, which is a reasonable amount of time to train such a network. The actor which consists of an artificial network of one hidden layer with 24 neurons converged to an action of 7g in the validation example resulting in a reward of 24.78. This keeps the BG above the green zone in figure 9.4. This is because the reward function explained in section 7.1.1 is designed to heavily penalize BG values



under  $72 \frac{mg}{dL}$  which would result in a hypoglycemic state.

One thing to note is that using only one action makes this a multi-armed bandit problem. This is well known in RL theory, and is considered as relatively easy for the agent to solve. This is important to keep in mind when considering a potential expansion of the decision support system, as having many potential meals and/or optimizing insulin injections may require a significantly increase in computational power.

The simulated annealing optimization terminated after 185 seconds because of convergence. This was very fast compared to the RL, making it practical for the patient to use. The optimal intake of CHO according to the simulated annealing is 10.8365 grams, resulting in a reward of 22.83.

The RL achieved a higher reward and gained therefore a better result. This is reasonable since the respective computation time was so different. With this in mind the simulated annealing optimization received close to the same reward by using only a fraction of the time. When the patient will be using this system, and inputs his or her current BG value and the exercise profile it will be unpractical to wait several hours for a result, so in hindsight the favour would fall with the simulated annealing algorithm. A possible solution to this would be to generalize the RL by training with many different exercises.

An important question to ask is why none of the result fall in to the normoglycemic range (the green zone). This is largely dependent on the design of the reward function, as the relation between the rewards for each value will determine the overall optimization. The design of this should fall mainly on medical experts as numerically rating the relation between hyperglycemic and hypoglycemic conditions is beyond the scope of this thesis and require medical expertise.



## **Part V**

# **Conclusion**



# /10

## Concluding Remarks

From the experiments conducted in this thesis we have attempted to personalize Breton's physical activity model to explain the BG dynamics of an individual T1D patient. Furthermore, we have made two variations of a decision support system that were tasked with optimizing the CHO intake before a given exercise. In this section, we summarize the results from the experiments and present some final comments. Furthermore, we will present some remarks on the usage of the decision support system that we have build for the patient. Lastly, we propose future work and directions related to the present thesis.

The results from the BG dynamics model show that the behaviour of the patient is modeled with minor error, this can be seen in Figure 8.2 and 8.3. It was discussed that because there were few data, the validation results from two exercise scenarios were not sufficient to draw any solid conclusions. Nevertheless, the validation that was performed shows promising results and points towards a working methodology. The fact that the BG dynamics of the patient were accurately modeled (with some error), without exposing the patient to dangerous situations is a remarkable result because, by applying this knowledge to potentially dangerous situations, we can recommend actions that reduce or even remove the danger.

We are using recorded data from a real patient to solve a very important problem, to avoid hypoglycemia during physical activity. The two decision support systems produced were compared in the same validation example as the generative model. The RL and simulated annealing recommended almost

the same amount of CHO intake, which indicates that both of the decision support systems are viable. It was discussed that the major factors that separate these two are the computation time (a few minutes for the simulated annealing and a few hours for the RL) and the expandability of the two systems (the RL can easily be expanded to include several meals and an expansion for the simulated annealing is impractical).

The two food recommendation systems are easy to use for the patient, as they only require the patient to choose between exercise profiles and to input the starting BG level. Importantly, this is only an educated recommendation for CHO intake and the patient still has to use her own judgement and common sense when utilising the product.

## 10.1 Future Work

Two food recommendation systems for a patient with T1D have been constructed, and a methodology for adaptations to other patients has been presented. Due to resource and time constraints, there was only few exercise events to validate the results on, and several aspects were not included. We present a proposal for further work in regards to the present thesis:

- Collect more data, and do further validation.
- Expand the BG regulation systems to include insulin as an optimization (this point can be very dangerous and must be researched with great care).
- Experiment and compare different models, both regarding the optimization and the generative model.

This concludes the thesis ✓



**/ 11**

## **Appendix**



# Bibliography

- [1] *Autoimmune Disease*. National Stem Cell Foundation. URL: <https://nationalstemcellfoundation.org/glossary/autoimmune-disease/> (visited on 07/08/2022).
- [2] Rudy Bilous and Richard Donnelly. “Handbook of Diabetes, Excerpt# 9: Management of Type 1 Diabetes.” In: ().
- [3] Mary Ann Branch, Thomas F. Coleman, and Yuying Li. “A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems.” In: *SIAM Journal on Scientific Computing* (July 25, 2006). Publisher: Society for Industrial and Applied Mathematics. DOI: 10.1137/S1064827595289108. URL: <https://epubs.siam.org/doi/10.1137/S1064827595289108> (visited on 07/12/2022).
- [4] Marc D. Breton. “Physical Activity—The Major Unaccounted Impediment to Closed Loop Control.” In: *Journal of Diabetes Science and Technology* 2.1 (Jan. 1, 2008). Publisher: SAGE Publications Inc, pp. 169–174. ISSN: 1932-2968. DOI: 10.1177/193229680800200127. URL: <https://doi.org/10.1177/193229680800200127> (visited on 07/01/2022).
- [5] *Diabetes type 1*. NHI.no. URL: <https://nhi.no/sykdommer/hormoner-og-naring/diabetes-type-1/type-1-diabetes/> (visited on 11/03/2021).
- [6] Geof H. Givens and Jennifer A. Hoeting. *Computational Statistics*. Google-Books-ID: bCJx53VQS7IC. John Wiley & Sons, Nov. 6, 2012. 496 pp. ISBN: 978-0-470-53331-4.
- [7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Google-Books-ID: Np9SDQAAQBAJ. MIT Press, Nov. 18, 2016. 801 pp. ISBN: 978-0-262-03561-3.
- [8] *How Simulated Annealing Works - MATLAB & Simulink - MathWorks Nordic*. URL: <https://se.mathworks.com/help/gads/how-simulated-annealing-works.html> (visited on 07/12/2022).
- [9] “IDF Diabetes Atlas, 10th edn. Brussels, Belgium.” In: (2021).
- [10] “IDF Diabetes Atlas, 10th edn. Brussels, Belgium.” In: *International Diabetes Federation* (2021).
- [11] Sami S. Kanderian, Stuart A. Weinzimer, and Garry M. Steil. “The Identifiable Virtual Patient Model: Comparison of Simulation and Clinical Closed-Loop Study Results.” In: *Journal of Diabetes Science and Technology* 6.2 (Mar. 1, 2012). Publisher: SAGE Publications Inc, pp. 371–

379. ISSN: 1932-2968. DOI: 10.1177/193229681200600223. URL: <https://doi.org/10.1177/193229681200600223> (visited on 07/06/2022).
- [12] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980 [cs]. URL: <http://arxiv.org/abs/1412.6980> (visited on 07/08/2022).
- [13] Konstantinos Koutroumbas and Sergios Theodoridis. *Pattern Recognition*. Academic Press, Nov. 26, 2008. 981 pp. ISBN: 978-0-08-094912-3.
- [14] *Low blood sugar (hypoglycaemia)*. nhs.uk. Section: conditions. Oct. 19, 2017. URL: <https://www.nhs.uk/conditions/low-blood-sugar-hypoglycaemia/> (visited on 07/06/2022).
- [15] Chiara Dalla Man et al. “The UVA/PADOVA Type 1 Diabetes Simulator: New Features.” In: *Journal of Diabetes Science and Technology* 8.1 (Jan. 1, 2014). Publisher: SAGE Publications Inc, pp. 26–34. ISSN: 1932-2968. DOI: 10.1177/1932296813514502. URL: <https://doi.org/10.1177/1932296813514502> (visited on 07/06/2022).
- [16] Hongzi Mao et al. “Resource Management with Deep Reinforcement Learning.” In: *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. HotNets ’16. New York, NY, USA: Association for Computing Machinery, Nov. 9, 2016, pp. 50–56. ISBN: 978-1-4503-4661-0. DOI: 10.1145/3005745.3005750. URL: <https://doi.org/10.1145/3005745.3005750> (visited on 07/09/2022).
- [17] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. Dec. 19, 2013. DOI: 10.48550/arXiv.1312.5602. arXiv: 1312.5602 [cs]. URL: <http://arxiv.org/abs/1312.5602> (visited on 07/10/2022).
- [18] Phuong Ngo et al. “Risk-Averse Food Recommendation Using Bayesian Feedforward Neural Networks for Patients with Type 1 Diabetes Doing Physical Activities.” In: *Applied Sciences* 10.22 (Jan. 2020). Number: 22 Publisher: Multidisciplinary Digital Publishing Institute, p. 8037. ISSN: 2076-3417. DOI: 10.3390/app10228037. URL: <https://www.mdpi.com/2076-3417/10/22/8037> (visited on 06/09/2022).
- [19] James Norman. “Normal Regulation of Blood Glucose.” In: (2016).
- [20] Gianluca Nucci and Claudio Cobelli. “Models of subcutaneous insulin kinetics. A critical review.” In: *Computer Methods and Programs in Biomedicine* 62.3 (July 1, 2000), pp. 249–257. ISSN: 0169-2607. DOI: 10.1016/S0169-2607(00)00071-7. URL: <https://www.sciencedirect.com/science/article/pii/S0169260700000717> (visited on 07/13/2022).
- [21] World Health Organization et al. *Definition, diagnosis and classification of diabetes mellitus and its complications: report of a WHO consultation. Part 1, Diagnosis and classification of diabetes mellitus*. Tech. rep. World Health Organization, 1999.
- [22] Pia V Röder et al. “Pancreatic regulation of glucose homeostasis.” In: *Experimental & molecular medicine* 48.3 (2016), e219–e219.

- [23] Hans-Reimer Rodewald. “Thymus organogenesis.” In: *Annu. Rev. Immunol.* 26 (2008), pp. 355–388.
- [24] Ken Saladin. “What is a peculiar feature of the thymus.” In: (2021).
- [25] David Silver. *Lectures on Reinforcement Learning*. URL: <https://www.davidsilver.uk/teaching/>. 2015.
- [26] *somatostatin | biochemistry | Britannica*. URL: <https://www.britannica.com/science/somatostatin> (visited on 07/06/2022).
- [27] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [28] *Unconstrained Nonlinear Optimization Algorithms - MATLAB & Simulink - MathWorks Nordic*. URL: <https://se.mathworks.com/help/optim/ug/unconstrained-nonlinear-optimization-algorithms.html#f3137> (visited on 07/12/2022).
- [29] Donald Voet, Judith G Voet, and Charlotte W Pratt. *Fundamentals of biochemistry: life at the molecular level*. John Wiley & Sons, 2016.
- [30] *When Going to the Bathroom Becomes Scary*. MIT Comparative Media Studies/Writing. June 2, 2017. URL: <https://cmsw.mit.edu/stephen-truong-diabetes-when-going-bathroom-becomes-scary/> (visited on 07/01/2022).
- [31] Gamil Yassin. “Build Simple AI .NET Library - Part 3 - Perceptron.” In: (2017).
- [32] Barbara Young, Phillip Woodford, and Geraldine O’Dowd. *Wheater’s functional histology E-Book: a text and colour atlas*. Elsevier Health Sciences, 2013.



