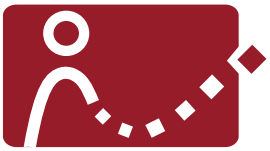
Predicting Blood Glucose Levels of Diabetes Patients



Masterarbeit von Gizem Gülesir
Tag der Einreichung: 22. Januar 2018

1. Gutachter: Prof. Dr. Max Mühlhäuser
2. Gutachter: Sebastian Kauschke

Darmstadt, 22. Januar 2018

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Telekooperation
Prof. Dr. Max Mühlhäuser

Erklärung zur Abschlussarbeit gemäß § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Gizem Gülesir, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Name: Gizem Gülesir Datum: Unterschrift:

Thesis Statement pursuant to § 23 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I, Gizem Gülesir, have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

I am aware, that in case of an attempt at deception based on plagiarism (S38 Abs. 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once.

In the submitted thesis the written copies and the electronic version for archiving are identical in content.

Name: Gizem Gülesir Date: Signature:

Abstract

Time series data is used for modelling, description, forecasting, and control in many fields from engineering to statistics. Time series forecasting is one of the domains of time series analysis, which requires regression. Along with the recent developments in deep learning techniques, the advancement in the technologies personal health care devices are making it possible to apply deep learning methods on the vast amounts of electronic health data. We aim to provide reliable blood glucose level prediction for diabetes patients so that the negative effects of the disease can be minimized. Currently, recurrent neural networks (RNNs), and in particular the long-short term memory unit (LSTM), are the state-of-the-art in timeseries forecasting. Alternatively, in this work we employ convolutional neural networks (CNNs) with multiple layers to predict future blood glucose level of a diabetes type 2 patient. Besides our CNN model, we also investigate whether our static insulin sensitivity calculation model's results have a correlation with basal rate of the patient. We use the static insulin sensitivity data with our prediction model, in order to find out whether it contributes for a better prediction or not. Our experimental results demonstrate that calculated static insulin sensitivity values do not have any correlation with the basal rate. Our convolutional neural network model forecasts multivariate timeseries with multiple outputs including the blood glucose level with a 1.0729 mean absolute error for the prediction horizon of 15 minutes.

Contents

1 Introduction

Diabetes mellitus is a major and increasing global problem [38]. There are around 171 million diabetes patients worldwide according to the study of [38]. This figure is predicted to rise to 336 million by 2030 [51], and it represents 4.4% of the world's population. Short term complications are hypoglycemia and hyperglycemia. The long term complications usually affect the vascular system, central nervous system and organs such as kidneys and eyes. These long term complications usually result in cardiovascular disease, retinopathy, and nephropathy. However, the costly complications can be reduced by managing to keep the blood glucose level between acceptable levels.

The control of the blood glucose level inside safe limits requires continuous monitoring of blood glucose level. The most common method for type 1 diabetics is finger prick test which is taken several times a day. These readings are used in order to adjust the required insulin dose which keeps the blood glucose between certain levels. Besides the blood glucose measurement, there are many other factors influencing the diabetes. Insulin type and dose, diet, stress, exercise, illness, and pregnancy are some of the factors with significant influence on diabetes theraphy. Continuous blood glucose meters (CGMs) and insulin pumps (IPs) provide ease of measuring the blood glucose and better monitoring for the patient. Using the information provided by CGMs, patients can better adjust the injection of exogenous insulin. However, CGMs require user interpretation. A system without the feedback system is called an "Open-loop system". In open-loop systems insulin is delivered in a preprogrammed independent of the amount of glucose measured. This type of glucose monitoring systems are common. Conversely, in closed-loop systems, output of the system is used as a feedback loop. In such system, insulin or other substances are given in response to a measured amount of glucose [42]. A system where these devices can work in closed-loop fashion to maintain a steady blood glucose level with minimal intervention from the patient, can be the next step in diabetes management. However, according to [21] such systems are at research and development stage, but initial clinical trials are promising.

1.1 Scope

The goal of this thesis can be roughly divided into two main categories. First of all, we want to show whether our static insulin sensitivity calculation correlates with the basal rate or not. Secondly, we want to predict future blood glucose levels of the diabetes patient. Additionally, we want to find out whether this newly generated feature, the insulin sensitivity, can contribute to the blood glucose level prediction when used as an additional feature. Therefore, type 1 diabetes is out of the scope of this thesis.

We will be discussing state-of-the-art methods for insulin sensitivity assessment and also our method. Our insulin sensitivity assessment model is a minimal physiological model using few parameters such as, continuous glucose measurement, and bolus insulin value. For blood glucose prediction models we will briefly explain different model structures, also related work, but in this thesis we use neural network approach which is a data-driven model. Even though, RNNs and in particular the LSTM, are the state-of-the-art in timeseries forecasting, we use CNN. Instead of RNN, we use one-dimensional convolutional neural network model using a real dataset

derived from a type 2 diabetes patient. The idea behind applying CNNs to time series forecasting is to learn filters that represent certain repeating patterns in the time series and to use these to forecast future blood glucose values. We belive that the layered structure of CNNs might perform well on noisy time series as in the work of [6]. Since we use the insulin sensitivity feature in our neural network model, the overall architecture of our model is considered as a hybrid model.

1.2 Outline

This thesis is organized in following manner: Chapter 2 describes necessary information required for understanding the disease diabetes mellitus, basic machine learning principles and related terms, artificial neural networks and deep learning. In chapter 3, state-of-the-art has been covered. We also discuss their significance, contribution and downsides. In chapter 4, we present structure of our dataset which is used in our experiments. In chapter 5, we introduce frameworks and libraries used for the experiments. In the next chapter 6, we present our approach for calculation of static insulin sensitivity from the dataset and neural network model for predicting the future blood glucose levels. Besides these, we also evaluate the results derived from these models. In chapter 7 we conclude our work with a forward looking approach.

2 Theoretical Background

In this chapter we will explain; diabetes mellitus disease, why it occurs, how it can be treated and the most important how we can contribute the treatment with our approach in this thesis. In the following section, we discuss why machine learning algorithms are used, especially which types of problems they are capable of solving. How a machine learning algorithm learns and what are the metrics used to evaluate the performance of a machine learning algorithm. We will explain also different types of problems where machine learning can be applied.

2.1 Diabetes Mellitus

Diabetes is a chronic disease due to a malfunction of the pancreas. Pancreas is an endocrine and digestive organ that, in humans, lies in the upper left part of the abdomen. Besides the function of helping the digestion, another function of the pancreas is to produce insulin hormone allowing blood glucose absorption by the body cells. Hence, pancreas also responsible for regulating the blood glucose. Insulin is a hormone released by the beta cells of the pancreas. The most important glucose source is food items that are rich in carbohydrates.

Every cell of the body needs energy to function. Blood glucose can be absorbed by the body cells, only with presence of insulin hormone, which is normally produced by pancreas. This hormone with healthy people, produced just with a right amount, but with diabetes patients, additional insulin injections are needed. Figure 2.1 provides an overview how insulin and glucose metabolism works. In the figure, the carbohydrates inside the food consumed by the patient, provide glucose to the blood stream after it is digested and converted into a smallest building blocks after digestion. Bread unit is used in for estimation of carbohydrate amount inside the food consumed by the patient. One bread unit corresponds to approximately $10g - 12g$ of carbohydrates.

Besides the normal blood glucose and insulin mechanism, there are two more important cases to mention. Firstly, muscles need almost no insulin during the sport in order to make use of the glucose. Secondly, the body has a protection mechanism to deliver quick energy for the cases where blood glucose level is critically low or during a heavily stressed situation. Critical level threshold is 50 mg/dl. Another source of glucose is the glucose store which can release or backup glucose. When the dangerous situation is over, this mechanism works in the opposite direction, retrieving glucose from blood to the glucose store. Another hormone called glycogen controls this glucose release and backup. On the other side, working muscles are the other factor consuming the glucose inside the blood. Since glucose absorption is only possible with an insulin release, for the diabetes patients, an insulin pump which releases insulin into the blood stream is needed. The insulin pump works with a sensor placed under the skin of the patient. The sensor measures the blood glucose level periodically and provides these measurements to the insulin pump. In this way the pump can calculate the required amount of insulin needed for the patient and release it. The rate of insulin released from the pump is called basal rate of the patient. Basal rate varies with each patient and can be programmed with the insulin pump. The dose of insulin released according to basal rate called basal insulin which stabilizes the blood glucose to an optimal value when the patient does not do any physical activity and does not eat anything.

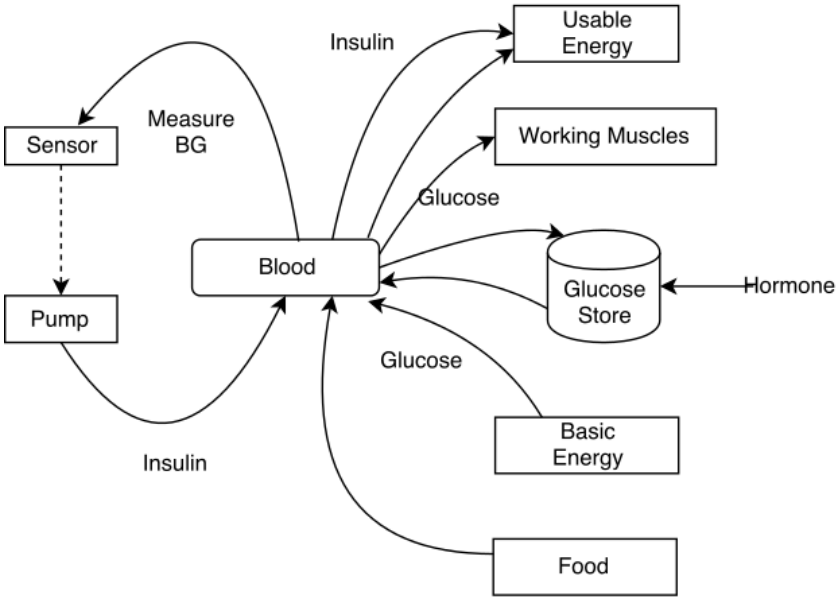


Figure 2.1: Illustration of insulin-glucose system

Disease arises from the reason; either the pancreas cannot produce enough insulin, or the body cannot effectively use the produced insulin. When diabetes is not responded with a suitable treatment, it leads to serious damage to the body's systems. When the blood glucose level is higher than the normal value, it is called hyperglycaemia, conversely low blood glucose is named hypoglycaemia. In the next paragraph, we will discuss two types of diabetes and other related effects in more details.

Type 1 diabetes patients characterized by deficient insulin production, while type 2 diabetes patients by ineffective use of insulin. Majority of the diabetes patients are type 2. In the scope of this thesis, we will be focusing on type 2 diabetes hypoglycemia prediction and prevention using machine learning techniques. For type 2 diabetes patients, controlling the blood glucose, and improving the insulin sensitivity is possible by changing life style and healthy diet. In order to keep the blood glucose within normal level, diabetic person needs to inject certain amount of insulin into the tissue between body skin, and the muscle. If the insulin is not injected, it also could be inhaled with an inhaler or it can be infused with a pump. Insulin amount needed is calculated based on food intake which is considered in terms of bread units. Blood glucose level is highly dependent on injected insulin amount and food intake. Besides food intake, other factors such as exercise, sleep-wake cycles, sleep quality, stress, illness, alcohol are known to be affecting blood glucose mechanism. We summarize the other known effects which are known to be influencing the blood glucose values, in the table 2.1.

2.2 Basics of Machine Learning

A machine learning algorithm which is able to learn from data [17] is used to tackle with tasks, which are too difficult to solve with ordinary programs designed and written by human beings. Machine learning algorithm processes collection of features measured from an object or event. In order to simplify things, we will refer to an experience as E, class of tasks as T, and a performance measure as P. In the context of machine learning tasks, T are, tasks which

Known Effects	Description
Refill Effect	After longer periods (>30 min) of sports, a blood glucose degradation can be observed up to 10 hours after the event. This is caused by a refill of the glucose stores of the body, since the stored glucose is used to provide enough energy for the working muscles.
Dawn Phenomenon	It is an early morning (usually between 2 a.m. and 8 a.m.) increase in blood glucose. It is not associated with nocturnal hypoglycemia (low blood glucose value). The dawn phenomenon can be managed by adjusting the dosage of the basal insulin in the insulin pump.
Chronic Somogyi Rebound	Chronic Somogyi Rebound describes a very high blood glucose value after a hypoglycemia (low blood glucose value). It occurs almost every time when the blood glucose value falls below 50 mg/dl (sometimes even earlier). The high blood glucose value is caused by a panic release of the glucose reserves in the body. This is triggered by hormone like Glucagon, Adrenalin and Cortisol.
Temporal Insulin Resistance	After longer periods without eating carbohydrates, a temporal insulin resistance can occur. However, patients have to inject more insulin to overcome this effect. But mostly patients do not notice this effect, and just wonder why the insulin dose does not work. In order to prevent this effect, the patient should consume small amounts of carbohydrates on a regular basis (3 times a day is suggested).

Table 2.1: Known Effects of Diabetes Mellitus

are too difficult to solve with fixed programs or human beings. When a computer program is learning from experience E with respect to some class of tasks T, and performance measure P; its performance at tasks in T, as measured by P, is called improving with the experience E [26].

Most of the machine learning algorithms can be thought as experiencing the entire dataset during the training phase, in supervised or unsupervised manner. A dataset can be considered as a collection of many examples or data points. In most of the cases, a dataset is a collection of examples, which turns into a collection features by the training. For example; a design matrix can be used to describe a dataset, which contains a collection of different example within every row. In other words, we can represent a dataset with 150 examples, and with 4 features for each example as $X \in \mathbb{R}^150x4$, where $X^{i,1}$ corresponds to the first feature, $X^{i,2}$ is the second feature, and so on. Another way of expressing a design matrix is, defining each example as a vector, each with the same size. However, this is not always possible, especially when the dataset is heterogeneous. For example; a dataset consisting of photographs with different dimensions (width and height) would result into images with different size and number of pixels. Therefore, with the vector representation, it is not possible to represent all the examples. In some cases with the heterogeneous data, instead of using a matrix with m rows to describe a matrix, it is possible to use the representation $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$ which corresponds to a set containing m elements. With this notation, there might any two example vectors $x^{(i)}$ and $x^{(j)}$ different sizes in the dataset exist.

Depending on the dataset, the learning can be categorized as supervised and unsupervised, even though the line between them is not sharp. The other variant of learning algorithm also called semi-supervised learning, which involves some supervision to the unsupervised learning. Machine learning can be used for solving many tasks. We will mention some of these tasks, such as; classification, regression, anomaly detection, and denoising. However, the list can be expanded. In order to accomplish the learning, the desired features from the dataset, cost function, and optimization algorithm are required. Another basic requirement for the machine learning is the performance evaluation, a measure how well the learning has done. In the next section we will be describing these different performance evaluation metrics, and different machine learning tasks and algorithms types.

2.2.1 Performance Measures

In order to evaluate the abilities of a machine learning algorithm, a quantitative measure is needed. This quantitative measure P, is specific to the task T, which is carried out by the learning algorithm. Performance metrics are used for defining, how well the machine algorithm is performing the tasks. Decision process of performance metric is a difficult task, and should be considered closely with the requirements of the application. For different machine learning task categories, there are different metrics with different granularity used for measuring the performance of the model. For example, for a regression task, error rate corresponds well to the desired behavior, on the other hand, accuracy metric is well suited for a classification task. For some problems penalizing the frequent medium sized mistakes, conversely for some others, penalizing rare but large mistakes might fit. During a machine learning algorithm is training using the data points, it is experiencing the training dataset. Evaluation of the learning should be done with a separate test dataset, which is not seen during the training phase. Next section we will be discussing about these different types of machine learning algorithms.

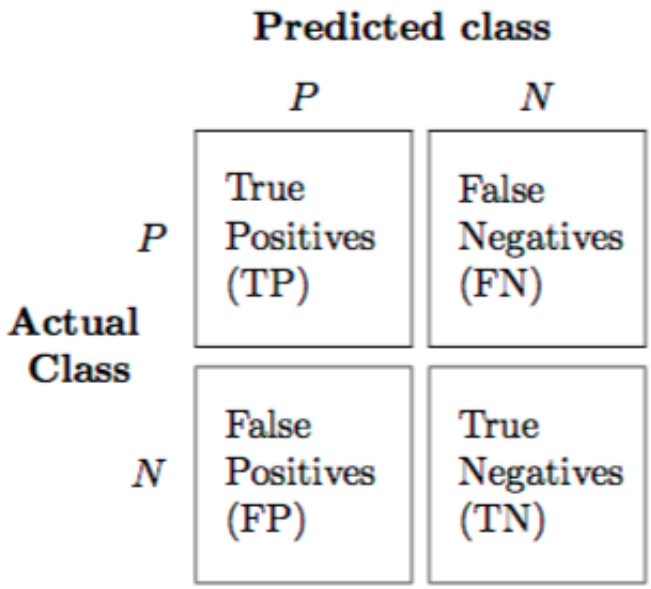


Figure 2.2: Confusion Matrix Layout for Error Visualization

Classification Metrics

One of the performance metrics used for the tasks of classification is accuracy. It is calculated from the proportion of examples, the model gives the correct output to all outputs. Error rate is also used for measuring the accuracy of a machine learning model, where the proportion of examples giving the incorrect output are used for the measurement. In order to introduce the formula of accuracy, we introduce the concepts of sensitivity and specificity which are also known as true positive rate and true negative rate.

A classifier classifies the outcome into positive and negative, as in the table 2.2 [1]. When the classifier predicts positive and the actual value is also positive or negative, this is called True Positives (TP) and False Positive (FP) respectively. Similarly, when the classifier predicts negative and the actual value is also positive or negative, this is called False Negative (FN), and True Negative (TN) respectively. Accuracy is simply the fraction of the total sample that is correctly identified. In other words, accuracy is the ratio of training examples which produces correct output to the total number of training examples. Therefore, accuracy can be formulated using these terms as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

For the tasks of classification, and classification with missing inputs, accuracy is the metric used to measure the performance of the model. Error rate is another metric also used to derive the same information, other way around. Error rate is the ratio of training examples which produces wrong output to the total number of training examples. It is also referred as the expected 0-1 loss. The 0-1 loss on particular example is 0 if it is correctly classified, and 1 if it is not correctly classified. For some tasks, such as density estimation, none of the performance

[1] Adapted from `https://rasbt.github.io/mlxtend/user_guide/evaluate/confusion_matrix/`

metrics we discussed till now, is suitable. For such tasks, a performance metric which gives a continuous-valued score for each given example should be used. Usually, in order to report the average log-probability, the model itself assigns to some examples.

Regression Metrics

One of the popular metrics for measuring the performance of a regression model, is mean absolute error (MAE). It is also called quadratic loss function. It is widely used in regression tasks as a performance measure. It is expressed with the following formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \hat{y}^{(i)} - y^{(i)}$$

where $\hat{y}$ is the predicted value and y is the original value. The parameter n represents number of data examples. Therefore, it is an average of the absolute errors the prediction and the true value.

Second important metric is Mean Squared Error (MSE), which is also similar to MAE. MSE is always positive value, since it takes the square of the difference between the prediction and the real value. Therefore, it provides the gross idea of the magnitude error, and represented as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)})^2.$$

Another variation of this metric is called Root Mean Squared Error (RMSE), as its name suggests; it takes the square root of MSE. Since this metric takes the square root of MSE, it converts back the unit into its original units of the output variable. Therefore, it provides more meaningful representation. This metric is described with the following formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)})^2}.$$

2.2.2 Types of Machine Learning Algorithms Tasks

In this section we describe different machine learning tasks.

Classification

This type of task, computer program is expected to categorize the input into one of the available categories. The computer program is asked to solve the problem by producing the function $f : \mathbb{R}^n \rightarrow \{1, ..., k\}$ where k is the number of possible class labels. When $y = f(x)$, the learned model is able to assign the input vector x into the correct category k, identified by numeric code y. In other words, learning algorithm must learn a function, which maps the input vector to a

categorical output. Computer program, which can recognize handwritten digits and numbers, is one of the application examples.

More difficult variant of classification is the classification with missing inputs. In this case, computer program might get an input vector with missing measurements. A computer program has to use a learning algorithm, which can learn a set of functions, rather than only a single function. Each function corresponds to classifying x with a different subset, where some of its input is missing. In case of this type of classification problem with n inputs, 2^n different classification functions needed for each set of possible missing inputs, but computer program only has to learn a single function which describes the joint probability distribution.

Regression

In this type of task computer program is expected to predict a numerical value, rather than a class label. A computer program predicting the future number of passengers' based on previous years of passengers' data is a regression problem. Learning algorithm is asked to output a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, similar to classification task, except the format of the output. Similarly, it can be also used to find similarities between two variables. One of the basic regression problems is a linear regression. The goal of the algorithm is to predict a scalar value from the given input vector. Linear regression models' performance can be measured in terms of the squared error produced when test data set given as an input.

Anomaly Detection

In this type of task computer program is expected to point out unusual events or objects from a set. These unusual events do not conform to the expected behavior, and usually referred to as anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities or contaminants in different application domains [7]. One way is, training the machine learning model to be able to reconstruct the original input with minimum error rate. After the model is trained, anomalous input will cause a higher error rate than the usual non-anomalous examples. Anomaly detection has an extensive use in many applications such as fraud detection for credit cards, intrusion detection for cybersecurity domain, fault detection for safety critical systems and military surveillance for enemy activities.

Similar to anomaly detection, novelty detection aims to detect previously unobserved, emergent or novel patterns in the data. The distinction between novel patterns and anomalies, is that the novel patterns are typically incorporated into the normal model after being detected. The solutions for mentioned problems are used often for both anomaly and novelty detection problems.

Denoising

In this type of task, computer program is expected to recover the clean example $x \in \mathbb{R}^n$ from its corrupted version $\tilde{x} \in \mathbb{R}^n$. This task is similar to anomaly detection, except instead of training the model with original data, slightly corrupted version of it if given as input. Learning algorithm learns to derive clean example x from its corrupted version $\tilde{x}$, while predicting the conditional probability distribution $p(x|\tilde{x})$. Error rate is calculated based on the difference between the

Cluster	Genres Included
Cluster 1	Short, Drama, Comedy, Romance, Family, Music, Fantasy, Sport, Musical
Cluster 2	Thriller, Horror, Action, Crime, Adventure, Sci-Fi, Mystery, Animation, Western
Cluster 3	Documentary, History, Biography, War, News
Cluster 4	Reality-TV, Game Show, Talk Show
Cluster 5	Adult

Table 2.2: Outcome of movie clustering , adapted from [43]

produced output, and original version of the input. Therefore, the model is forced to learn the noise and corrupted parts, during the training process [32].

2.2.3 Learning Algorithm Types

In this section, we explain different types of learning algorithms, and the main differences between them. Machine learning algorithms can be roughly divided into 4 categories; unsupervised, supervised, semi-supervised and reinforcement learning algorithms.

Unsupervised Learning

When the algorithm experiences and learns useful features and structure of a dataset; the algorithm is called an unsupervised learning algorithm. Unsupervised learning is used for density estimation, synthesizing, denoising and clustering. There is no instructor or teacher, who shows the machine learning system what to do. The algorithm should find out a way to extract useful properties of the given dataset. In the context of deep unsupervised learning, the entire probability distribution of the dataset is aimed for tasks such as density estimation, synthesis, and denoising. Unsupervised learning algorithms observe several examples of random vector x, and tries to learn the probability distribution $p(x)$, or some other interesting properties of the distribution. There are other unsupervised learning algorithms doing clustering, which is dividing the dataset into many similar examples. For example; clustering movies into different subsets, based on their titles and keywords, using the similarity is one of the applications of clustering. With or without providing the possible outcomes, the possible genres, the algorithm produces different clusters. The table 2.2 has the outcome of movie clustering which carried out in the work of [43].

Supervised Learning

When a machine learning algorithm experiences a dataset, and each example of the dataset has an associated label, the algorithm is called a supervised learning algorithm. Training process

involves observation of several labelled examples acting as a teacher, and showing the algorithm how to learn the target from the given dataset. Label could be a number or a sequence of words. For example, of speech recognition scenario, record of speech would have a sequence of words as a label. Some of the supervised machine learning algorithms are logistic regression, support vector machine (SVM), k-Nearest Neighbors, Naive Bayes, random forest, and linear regression.

For supervised learning, an example contains a label or target and collection of features. Labels can be represented with a numeric code or sequence of words depending on the task. When the dataset contains a design matrix of feature observations X, we also provide a vector of labels y, with y_i providing the label for an example i.

Unlike unsupervised learning algorithm, supervised learning algorithm involves observation of several examples of input vector x, and associated output vector y and given together of this information, algorithm tries to predict y from x during the training usually by estimating $p(y|x)$. Even though there is no instructor or teacher for unsupervised learning, sometimes the line between supervised and unsupervised learning is blurred [17]. Usually these techniques are categorized as semi-supervised learning[54]. We discuss semi-supervised learning in the next section.

Even though supervised and unsupervised learning algorithms are not completely distinct concepts, helps us to categorize the learning algorithms. For example; usually regression, classification, and structure output problems are considered as supervised learning algorithm, while density estimation in support of other tasks is considered as unsupervised learning algorithm according to [17].

Semi-supervised Learning

Another variant of machine learning algorithm including both supervised learning practices by a supervision target and unsupervised fashion, is called semi-supervised learning algorithm. For example; in multi-instance learning algorithm, an entire collection of input examples are labeled as containing or not containing an example of a class, but the individual members of the collection are not labeled.

Majority of the data in real world is unlabeled, and the process of labeling is time consuming task. Therefore, these techniques can use the advantage of both labeled and unlabeled data. There are many machine learning techniques can be used for both of the learning tasks. As an example; an image classifier which uses both supervised and unsupervised methods can be considered [18]. Another example, in the work of [18] a strong Multiple Kernel Learning (MKL) classifier using both the image content and keywords is used in order to score unlabeled images. Afterwards, the techniques; support vector machines (SVM) or least-squares regression (LSR) from the MKL output values on both the labeled and unlabeled images are used.

Reinforcement Learning

Unlike the previous machine learning algorithms, reinforcement learning interacts with the environment during the learning process, besides experiencing the dataset. Interaction with the environment happens within a feedback loop between the experiences and the learning system. Reinforcement learning algorithm does not experience a fixed dataset like other supervised and unsupervised learning algorithms. Popular applications of reinforcement learning are used with

computer programs learning to play games such as chess, Go, Atari [28] and [27]. In this thesis reinforcement learning methods are not used.

2.3 Related Terms

In this chapter, we use the one of the simple machine learning algorithms, in order to explain some related terms of machine learning algorithms. After quick introduction to machine learning algorithm; the linear regression, we explain the terms overfitting, underfitting, regularization, gradient based optimization. As the name suggests, linear regression is a machine learning algorithm, which is used for solving regression problems with a linear function. The algorithm takes a vector $x \in \mathbb{R}$ as an input, and tries to predict the value of scalar $y \in \mathbb{R}$ as an output. The function of linear regression can be written as:

$$\hat{y} = w^T x,$$

where $w \in \mathbb{R}^n$ is a vector of parameters, and $\hat{y}$ is the prediction and the value y is the actual value. The parameters w and x control the behavior of the system. In this equation, w_i is the coefficient that is multiplied by the feature x_i, and the result is summed up together with all the contributions from all feature vectors. If a feature received positive value from the weight vector w_i, the contribution of the feature vector x_i increases on the prediction value $\hat{y}$. When the weight value is negative, the effect of that specific feature decreases on the prediction. Depending on the magnitude value of the weight value, prediction changes. Therefore, if feature's weight is zero, it has no effect on the prediction. For linear regression the task, T is to predict y from x by outputting $\hat{y} = w^T x$.

For the performance measure P, there is a need for a test dataset, which is not used for the training. One way is by computing the mean squared error of the model on the test set. Assuming a model with a design matrix of m example inputs, the inputs of the design represented as $X^{(test)}$, and the vector of regression target represented as $y^{(test)}$. The prediction $\hat{y}^{(test)}$ can be evaluated with the performance measure P as follows:

$$MSE_{test} = \frac{1}{m} \sum_i (\hat{y}^{(test)} - y^{(test)})_i^2.$$

The error decreases when the difference between the prediction $\hat{y}^{(test)}$, and the target $y^{(test)}$ getting closer to the zero. Similarly, another performance metric can be defined in terms of Euclidean distance between the prediction and the target as follows:

$$MSE_{test} = \frac{1}{m} || \hat{y}^{(test)} - y^{(test)} ||_2^2.$$

In the equation above the total error decreases when the Euclidean distance between the prediction and the target increases. Therefore, in order to a machine learning algorithm to function correctly, it should either decrease the mean squared error, or increase the Euclidean distance between the prediction and the target, while it is experiencing the training dataset. To achieve minimization of the mean squared error on the training set MSE_{train}, the algorithm should adapt its weights w accordingly.

Linear regression often used to refer to slightly different model with an additional parameter which is usually called bias b, which makes the new equation:

$$\hat{y} = w^T x + b.$$

The additional parameter b makes the plot of the model still linear, but instead just not passing through the origin. The terminology of the word bias is different from the statistical point of view. It can be thought as the output is biased towards, based on the value of the parameter b.

Linear regression is one of the simplest machine learning algorithms. Therefore, it has limited use for different problem types. In the following sections we will be discussing more sophisticated algorithms and more terms related with machine learning.

2.3.1 Capacity, Overfitting and Underfitting

The ability of a machine learning model performing well on unseen data (test dataset) is called generalization ability of the algorithm. During the training of the model, training error is used as a performance in order to reduce the error rate on the training set. For example, measuring the performance of the test phase of a machine learning algorithm, we have to have certain assumptions about the train and test dataset. First of all, training and test sets should be independent from each other. Another assumption is; both sets should be identically distributed. In other words, both sets should be drawn from the same probability distribution. When these assumptions hold, training error of randomly selected model should be equal to the expected test error of that model, since both are formed from the same data sampling process.

All machine learning algorithms aims to reduce the training error, and the difference between the training and test error. While trying to achieve these goals, the main challenge is to be able to fight with underfitting and overfitting. Capacity of a neural network is one of the important factors of over and under fitting. Capacity of a neural network is directly related with the size of the network. A neural network with more hidden layers and more hidden nodes tend to store more information, therefore they have more capacity according to [49]. When the model has high capacity, it starts to memorize the properties of the training test, instead of learning. This happens when model has high generalization error, also called testing error. An overfitted model performs poorly on unseen test data, while performing well on training data. In this case, the model's capacity is reduced in order to avoid overfitting. On the other hand, when a model has low capacity, it is likely to underfit, which means it can not derive the essential features from the training set. Underfitting takes place when the gap between training and generalization error is too large, which is illustrated in the figure 2.3 in the area after the optimal capacity.

As a summary, machine learning algorithm has the best performance with the right capacity, and also requires less amount of time for the training. Machine learning models with insufficient capacity, fail to solve complex tasks, similarly models with high capacity fails to generalize well and overfit for the tasks.

2.3.2 Regularization

Another method used to control overfitting and underfitting is regularization. It is any modification made to a learning algorithm, in order to reduce its generalization error while not changing the training error [17]. One way to do this for the linear regression algorithm, is modifying the

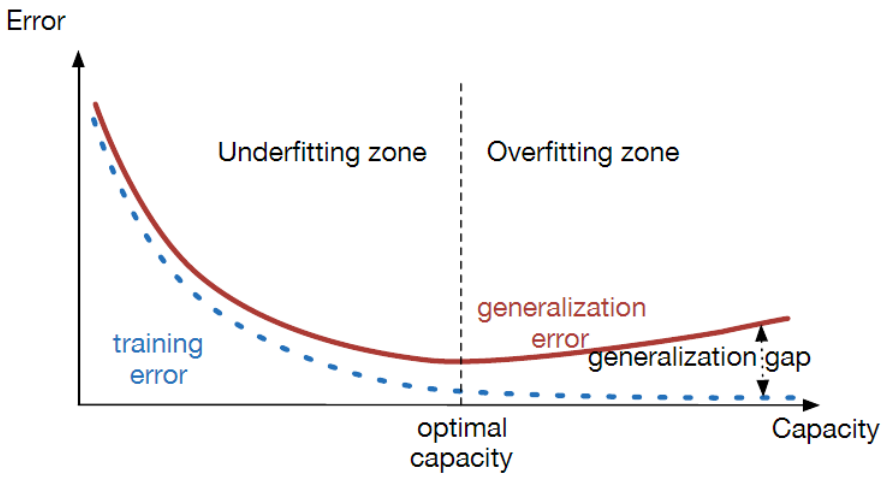


Figure 2.3: Illustration of Overfitting and Underfitting, adapted from [17]

training criterion to include weight decay. Regularization adds a penalty on the different parameters of the model. Therefore freedom of the model is reduced by adding the penalty. As a result of this, the model will be less likely to fit the noise of the training. Therefore regularization improves the generalization abilities of the model. However too much regularization increases the testing error rate, and leaving the model with too few parameters, which would not be able to learn the complex representations from the input.

There are three types of regularization methods; namely $L1$, $L2$ and $L1/L2$. In $L1$ regularization some of the model parameters are set to zero. Therefore those parameters no longer play any role in the model. $L2$ regularization adds a penalty equal to the sum of the squared value of the coefficients. Therefore, L2 regularization will force the bigger parameters for bigger the penalization, while on the smaller coefficients effect penalty is smaller too. For any other machine learning algorithm, there are many ways to design regularization technique. Therefore, there is no regularization form which fits to all of the machine learning algorithms. Regularization is one of the central concerns of the field of machine learning, such as optimization problem.

2.3.3 Gradient Based Optimization

Optimization is a task of minimizing or maximizing a function $f(x)$ by altering x. The function $f(x)$ is called objective function or criterion. For the maximization of $f(x)$ is also referred as cost function, loss function, or error function. We will use the notation x^* to refer the $arg\ min\ f(x)$.

The derivative function is used for minimizing, since it provides the information how to change the x in order to minimize for the function $y = f(x)$ at the point x. The derivative of a function $f(x)$ is denoted as $f'(x)$ or $\frac{d^y}{d^x}$. The figure 2.6 at page 21 provides visual. This derivative function provides the direction information, from the red region where the cost is high, towards which direction to move in order to reach the blue region(where error is minimum). We will discuss gradient based optimization in more details in the section 2.4.2.

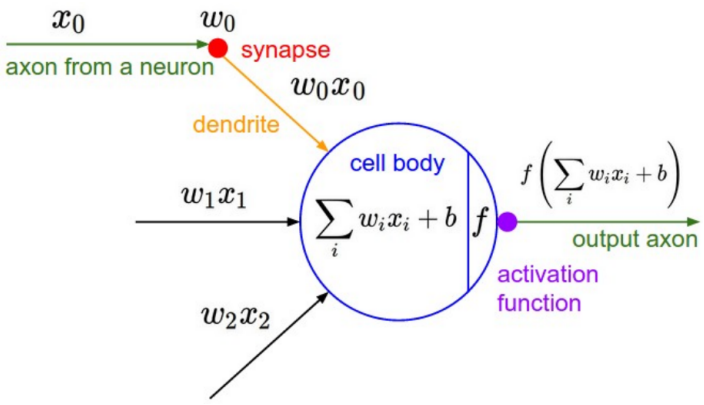


Figure 2.4: Artificial Neuron Model

2.4 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational model which is inspired from biological neurons of human brain [10]. The figure 2.5 [2] on page 20 has typical ANN structure with input, hidden and output layer. ANNs are called networks, because they are typically represented by composing together many functions. A neural network with 3 layers can be considered as computing the formula $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. First layer is $f^{(1)}$, second layer is $f^{(2)}$, and third layer is $f^{(3)}$ of the neural network. ANNs are invented to solve tasks specifically humans are good at, such as pattern recognition and other tasks which does not require to be re-programmed and the tasks involving uncertainty. Therefore, ANNs have potential of solving many real life problems in a wide range of fields. They contain layers of computing nodes called the perceptron doing the nonlinear summation operation. A neuron fires when a particular function of summation of the signals outcome is greater than some threshold value. The neuron in figure 2.4[3] with three incoming connections and the activation function of $\sum_i (w_i x_i + b)$. Nodes within each layer are interconnected by weighted connections, where weights are represented with w_i. Signal strength depends on the weighted connections. Therefore, while some neurons are passing the signal to the next layer, some might drop. ANNs have an input and output layers. The middle layer is known as hidden layer. ANN having at least one hidden layer is capable of solving nonlinear tasks by training with gradient descent methods. ANNs with many hidden layers are called deep neural networks.

Before neural network is trained, its weights are usually initialized by a random distribution function. At this stage ANN is not expected to produce any meaningful result. Input data is fed during the training phase. By the time, over iterations, weights of the neural network are adjusted in order to output according to the desired function. In other words, neural network changes its weights, in order to adapt new environment requirements during its training [13].

During the supervised learning of a neural network, both the input and the correct answer is provided. The input data is propagated forward from input layer to the output neurons. Outcome of propagation is compared with the correct answer. Based on the difference between outcome and the correct answer, weights of the network are adjusted in a way to increase the probability of the network giving correct answer for the future runs with the same or similar

[2] Adapted from `http://cs231n.github.io/neural-networks-1/`

[3] Adapted from `https://towardsdatascience.com/from-fiction-to-reality-a-beginners-guide-to-artificial-`

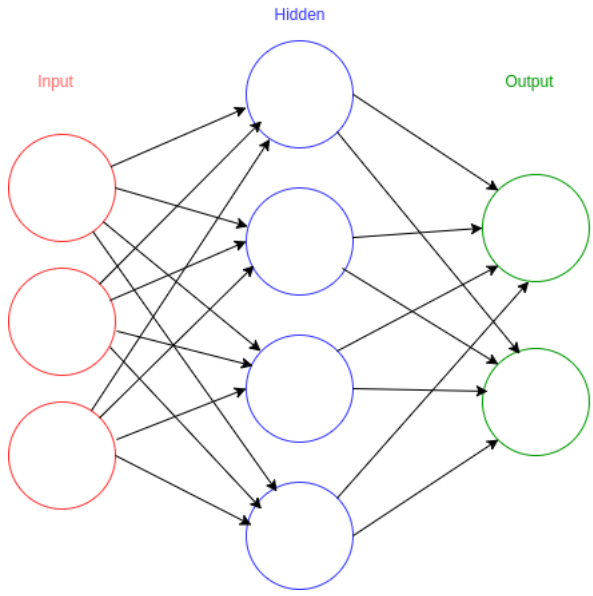


Figure 2.5: Artificial Neural Network Structure

data. For unsupervised learning, no correct answer, but only input data is provided. Network has to evolve by itself according to some structure, which in the given input data. Usually the structure is a some of form redundancy or clusters in the given input data[13]. Another important property of neural network training, as well as all machine learning algorithms is, how well the network generalize after the training. In the case of overlearning set of training input data is called overfitting. Since the network does not generalize well, it performs poorly with the unseen test data. While the number of hidden neurons provides a degree of freedom, on the other hand increases the risk of overfitting, because the network will tend to memorize the data. In order to overcome this problem some weights removed explicitly to increase the generalizability of the neural network. In case of too few hidden layer neurons, network might not perform the desired functionality. Therefore, a compromise should be found while designing neural network.

In the next sections we will be discussing the different types of neural networks and related terms.

2.4.1 Deep Neural Networks

Feedforward neural networks with more than one hidden layers are called deep neural networks or deep-learning networks. Deep neural networks are used to approximate some function $f*$. For classification task, $y = f*(x)$ maps an input x to a category y. The neural network defines a mapping $y = f(x;\theta)$, and learns the value of the parameters θ which would result in the best possible approximation. In deep-learning networks, each layer of nodes trains on a distinct set of features based on the previous layer output. The further hidden layer added to the network, the more complex the features can be recognized, since they aggregate and recombine features from the previous layer. Besides this, training becomes also harder and takes longer time compared to shallow neural networks and perceptrons.

The reason why these neural networks are called feedforward, is the information flows from the input layer direction towards the output layer through the intermediate computations. The

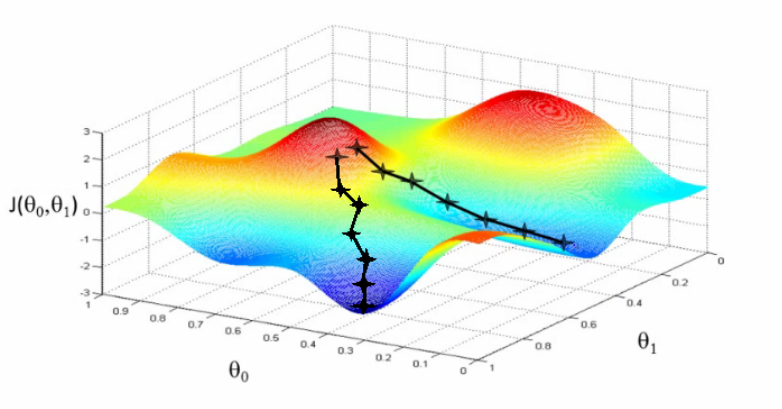


Figure 2.6: Gradient Descent Algorithm Illustration

simplest form of learning algorithm with feedforward neural is called perceptron, input and output layers do not learn the representation of the data, because they are predetermined unlike the hidden units. When feedforward connections have feedback connections, they are called recurrent neural networks. Another type of specialized feedforward neural network is convolutional neural networks, which specifically used for object recognition tasks. We describe some of these deep neural network types in the further sections.

2.4.2 Optimization Algorithm Gradient Descent

Learning of deep neural networks requires gradient computation of many complex functions, similar to any other machine learning algorithm. Besides the gradient algorithm, one needs to specify an optimization procedure, a cost function and a neural network model. The algorithm tries to find the best parameters for network with minimum cost. The figure 2.6[4] represents the cost space. The areas with red color has the highest cost, while blue regions has the lowest cost. The optimization algorithm tries to find the path down from the point of the highest cost to the regions where the cost is lowest. Therefore, y-axis, we have the cost $J(\theta)$ against the parameters $\theta 0$ and $\theta 1$ on x-axis and z-axis respectively. However there are certain challenges with optimization of the algorithms, such as vanishing or exploding gradient problem. Sometimes gradient signal is too small. Therefore, algorithm cannot converge to the global optima. Similarly, when gradient signal is too strong, and bigger steps are taken, again algorithm cannot converge. Since it keeps bouncing around the curvature of the cost valley, overshooting the correct path. It is not always straightforward to find the global optima. Hence, there are many variations of gradient descent algorithm. Some other variations are vanilla gradient descent, gradient descent with momentum, AdaGrad (Adaptive Stochastic (Sub)Gradient) and ADAM (Adaptive Momentum).

Gradient descent algorithms can be roughly classified as two methods; full batch gradient descent algorithm, and stochastic gradient descent algorithm. Full batch gradient descent algorithm the parameters are updated after the use of the full batch of the input data, while with the stochastic gradient descent algorithm, the parameters are updated only after whole dataset is used. We discuss one of gradient algorithm, which is back-propagation algorithm, and its modern generalizations in the next section.

[4] Adapted from `http://cs231n.github.io/neural-networks-3/#sgd`

2.4.3 Backpropagation Algorithm

Backpropagation is learning procedure for networks neurone-like units [39]. The backpropagation algorithm repeatedly adjusts the weights of the connections based on the difference between the actual output vector, and the desired output vector. The algorithm uses the partial derivative of the error of the neural network with respect to each weight [1], in order to minimize the difference between two vectors. The process of modification of each weight, starts from output layer and last hidden layer connection, and propagates through each layer until it reaches the input layer. Therefore, it is called backpropagation algorithm. Using these partial derivatives, and the change of the value of the weight, networks weights are adjusted to reach the local minimal. In other words, if the derivative is positive the error is increasing. When the weight is increasing, then backpropagation algorithm adds a negative value of partial derivative to the weight in order to minimize the error. Similarly, if the derivative is negative, then it is directly added to the weight. The basic idea of the backpropagation learning algorithm is the repeated application of the chain rule to compute the influence of each weight in the network with respect to an arbitrary error function E [37] as follows:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_i}\frac{\partial s_i}{\partial net_i}\frac{\partial net_i}{\partial w_{ij}}$$

where w_{ij} is the weight from the neuron j to neuron i, s_i is the output, and net_i is the weighted sum of the inputs of the neuron i. Once the partial derivatives for each weight is known, the aim of the minimizing the error function is achieved by performing a simple gradient descent:

$$w_{ij}(t+1) = w_{ij}(t) - \epsilon\frac{\partial E}{\partial w_{ij}}(t)$$

The choice of learning rate ϵ, scales the derivative and effects the time needed until convergence is reached. If the learning rate is too small, too many steps are needed to reach an acceptable solution. On the other hand, too big value possibly leads to oscillation, preventing the error to fall below a certain threshold value. In order to cope with this problem another variable called momentum parameter is added. However setting the optimum value of momentum parameter is equally hard as finding the optimal learning rate value.

2.4.4 Activation Function

Feedforward networks have introduced the concept of a hidden layer, therefore activation function is needed in order to compute hidden layer values. In simple words, activation function calculates the sum of weighted inputs from the previous layer of the network, adds the bias value of the neuron to it, and then decides whether the neuron should be fired or not. The most popular activation functions are logistic function $\frac{1}{1+e^{-x}}$, $tanh()$ function, and The Rectified Linear Unit (ReLU) function $f(x) = max(0, x)$. The ReLU makes the network converge faster, and it is resilient to the saturation. Therefore, unlike logistic and hyperbolic tangent function, it can overcome with the vanishing gradient poblem. General idea with activation functions is; their rate of change is greatest at intermediate values, and least at extreme values. Therefore, it makes it possible to saturate a neuron's output at one or other of their extreme values [13]. Second useful property is that they are easily differentiable.

2.4.5 Cost Function

Another important design aspect of deep neural network training is the choice of cost function. A cost function is a measure of how good a neural network did with respect to it's given training sample and the expected output. It is a single value which determines how good the overall neural network scored. Sometimes the term loss function is also used for the cost function, but loss function is a usually a function defined on a data point, prediction, and label to measure the penalty, whereas cost function is more general term. Therefore, loss function is for one training example, while cost function is for the entire training set. Cost function can be defined in terms of neural network's weights, biases, input from the training sample and the desired output of that training sample. Some popular cost functions are mean squared error (MSE), mean absolute error (MAE), and mean absolute percentage error.

2.4.6 Learning Rate

Learning rate determines how quickly or how slowly the parameters of the neural network are updated. Learning rate should be decided upon the architecture, and the nature of the problem. Gradient descent algorithm updates the networks' parameters according to the learning rate decided upon. Therefore, it should be carefully selected. Since small or big of learning rate results in either vanishing or exploding gradient problem, which were previously discussed in the previous section. Usually, one can start with a large learning rate, and gradually decrease the learning rate as the training progresses. This is called adaptive learning rate. The most common learning rate schedulers are time based decay, step decay, and exponential decay [4]. The formula for time based decay is $lr = lr0/(1+kt)$ where lr, k are the hyperparameters, and t is the iteration number.

2.5 Deep Learning

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection, and many other domains such as drug discovery, and genomics. Deep learning discovers intricate structure in large data sets, by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

In the next subsections we will discuss some of the popular deep neural networks, such as; convolutional neural networks, autoencoder and long short-term memory.

2.5.1 Convolutional Neural Networks

Convolutional networks are specialized kind of neural network useful for processing data which is grid-like. Time-series data as a 1-D grid samples, and image data as 2-D grid of pixels can be used with convolutional neural networks. This type of neural network mainly used for image

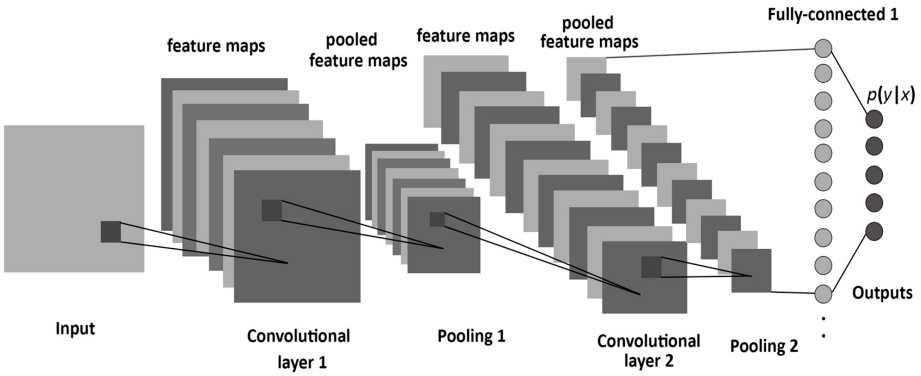


Figure 2.7: Convolutional Neural Network Architecture, adapted from [2]

recognition classification, and employs a mathematical operation called convolution which is a specialized kind of linear operation, in place of general matrix multiplication in at least of the the layer of the network. The second important term about convolutional neural networks is pooling. While convolution and pooling layers together acts as a feature extractors from the input.

A typical convolutional neural network consists of three stages. In the first stage, the network layer performs convolutions in order to produce a number of linear activations. In the next stage which is also referred as detector stage, the produced linear activations run through a nonlinear activation function. At the last stage, pooling function modifies this output. This order of operations can be repeated many times till the output layer.

In the next sections we will be discussing how convolution, ReLU and Pooling works.

Convolution

The primary purpose of convolution operation is to extract features from the input data. The convolution operation preserves the spatial relationship between data points and the small squares of the input data.

In case of an image as an input to the convolutional neural network, the image can be considered as matrix of pixel values. In order to simplify the example we will consider an image with size 5 X 5 whose pixel values are either 0 or 1. Another matrix 3 X 3 which is called filter or kernel or feature detector is slided over the image and it computes the dot products. This matrix is called convolved feature or activation map or feature map. This operation results in different convolved features with every different input image and filter. Sliding the filter over the image captures local dependencies in the input image.

The size of the feature map has three parameters; depth, stride, and zero-padding. Depth of the feature map corresponds to the number of filters used for the convolution operation. Each different filter is used to detect different features. Second parameter stride corresponds to the size of the filter, which is slided over the input matrix. For example, if the stride is 2, means the filters jumps 2 pixel at a time as they are slided. Larger stride results in smaller feature maps. The last parameter zero-padding is used for borders of the input image. While applying the filter, sometimes additional zeroes needed to be added to the border. This procedure is called wide convolution. When zero-padding is not used, the convolution is called narrow convolution.

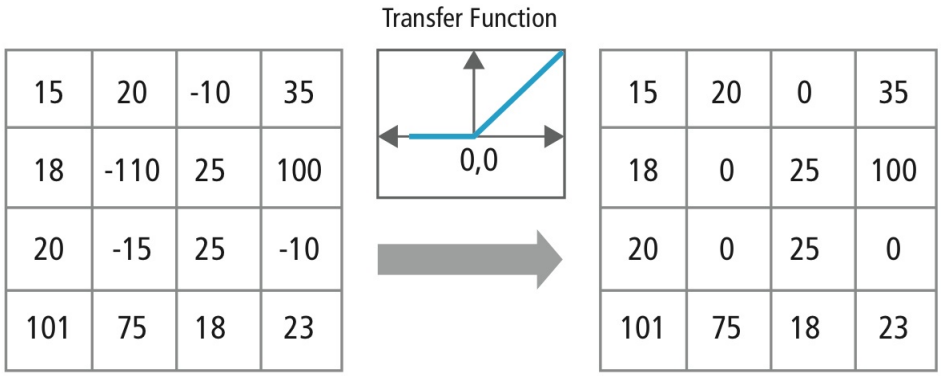


Figure 2.8: ReLU Operation

ReLU (Rectified Linear Unit)

ReLU also called activation operation, is used after every convolution operation in order to introduce non linearity. Since convolution operation is only an element wise matrix multiplication, the function ReLU is used in order to introduce non-linearity without affecting the convolutional layer operations. Because convolutional neural networks should learn mostly non-linear real world data, and convolution is a linear operation. There are other non-linear operations such as tanh and sigmoid function. Earlier these functions are used instead of ReLU, but ReLU has been found to perform better, and make the learning faster in most of the situations. ReLU simply changes the negative activations into 0, while applying the function $f(x) = max(0, x)$ to the incoming data. Figure 2.8 [5] describes the ReLU function with an example.

Pooling

Pooling layer also called downsampling layer, combines the outputs of neuron clusters at one layer into a single neuron in the next layer. Pooling function is also called subsampling or downsampling, which reduces the dimensionality of each feature map while leaving only the most important features. A pooling function in the figure 2.9 [6] replaces the output of the previous operation at a certain location, with a summary statistics of the nearby outputs. This is called spatial pooling, and it can be max, min, average or sum pooling. In case of max pooling, based on the defined spatial neighborhood, the largest element from the rectified feature map within that window is taken. Similarly, average pooling calculates the average value of the cluster of neurons at the previous layer. Max pooling has been shown to work better for most of the situations.

In particular pooling function makes the input representations (feature dimension) smaller and more manageable. Since the number of parameters in the network is reduced, it controls the overfitting. Network becomes more invariant to the small transformations, distortions since max/average value is taken in a local neighborhood. This provides higher accuracy while detecting the objects in images, independent of where they are located in the image.

[5] Adapted from https://www.embedded-vision.com/platinum-members/cadence/embedded-vision-training/documents/pages/neuralnetworksimagerecognition

[6] Adapted from https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

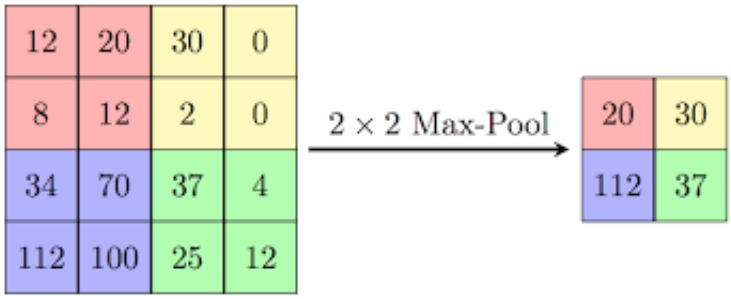


Figure 2.9: Illustration of Max Pooling Operation

Fully Connected Layer

The fully connected layer of convolutional neural network is a traditional Multi Layer Perceptron (MLP). Since every neuron within a layer is connected to the every other neuron in the next layer, this type of neural network is called "Fully Connected". The high-level features of the input are used at fully connected layer, in order to classify the input into various classes. At the output of this layer, there are same number of neurons as the number of possible outcomes based on the training dataset. The Softmax function takes a vector of arbitrary real-valued scores, and squashes it to a vector of values between zero and one, in a way that sum of all probabilities are one.

Dropout Layers

Dropout Layer's main functionality is avoiding overfitting problem. Overfitting problem occurs when the network performs poorly with unseen examples, although it performs well with training data. This layer drops out some random set of activation simply by setting to zero. The idea of dropout is simplistic in nature. In other words, some random connections of the neural network becomes redundant, and therefore network is forced to learn the features, instead of memorizing the training data. This makes sure that the network does not get too fit to the training data [44]. Therefore, this layer is only used during training, and not during the testing.

2.5.2 AutoEncoder

Autoencoder is a type of neural network, which attempts to represent the input by trying to copy its input to its output. It consists of two parts; an encoder function $h = f(x)$, and a decoder function $r = g(h)$, which produces a reconstruction. Autoencoder model is forced to learn the data by approximately, rather than perfectly copying the input. In other words, they are forced to learn only the most important information from the input data. Figure 2.10 [7] has the diagram of an autoencoder, where X is the input, and $\hat{X}$ is the reconstructed output. In this way autoencoders generalizes on the training data, and retrieves useful properties of the dataset. Autoencoders traditionally used for dimensionality reduction, or feature learning. Modern autoencoders are used for generative modeling. Similar to traditional feedforward neural networks, autoencoders can be trained using minibatch gradient descent, following gradients computed by back-propagation. Another learning algorithm recirculation may be used for train-

[7] Adapted from `https://iksinc.wordpress.com/2016/07/07/autoencoders/`

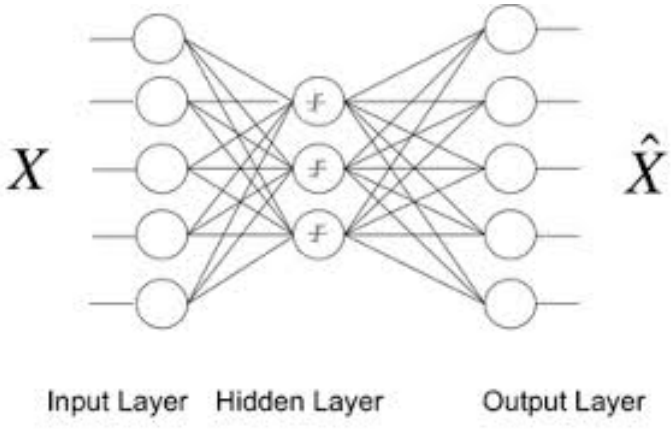


Figure 2.10: Diagram of an Autoencoder

ing, which is an algorithm based on comparing the activation of the neural network with the original input and reconstructed input.

Autoencoders can be mainly categorized into undercomplete and regularized autoencoders. Undercomplete autoencoders tries to capture useful features of the input data, while trying to copy the input at output layer. When autoencoder's code dimension is less than the input dimension, the autoencoder is called undercomplete. On the other hand, this type of autoencoders, would fail to learn the salient features of the input data, in case of too high capacity. Second type of autoencoder is regularized autoencoders, which can also has subtypes of sparse, denoising and contractive autoencoders. Unlike undercomplete autoencoders, regularized autoencoders are able to learn useful features even though the model is overcomplete, and nonlinear. Instead of just keeping the encoder and decoder shallow and the code size small, the use of loss function helps the model to overcome the shortcoming of undercomplete autoencoders. These models are good at learning high-capacity, overcomplete encodings of the input without a need for a regularization of the encodings.

One of the example of regularized autoencoders is sparse autoencoders whose training criterion involves a sparsity penalty, in addition to the reconstruction error. Common use case for sparse autoencoders is classification tasks. Sparsity penalty can be thought as a regularization term added to a feedforward network which copies the input to the output, in other words a feedforward neural network with unsupervised learning objective. The second one is called denoising autoencoder which adds the penalty to the cost function, and learns from input data by changing the reconstruction error term of the cost function. Denoising autoencoder tries to correct the corrupted input rather than trying to copy at the output layer.

2.5.3 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a specific type of recurrent neural network type for processing sequential data introduced by [20]. LSTM works well with long sequences and sequences with variable length as well. RNNs and LSTMs are successfully applied to sequence prediction and classification problems. The LSTM has a special structure containing the memory block, allowing them to store the temporal state of the network [40]. Input gate allows the input information flow structure within each memory block, output gate controls the flow of activations into the next layers of the network. The LSTM networks has gained importance due to its

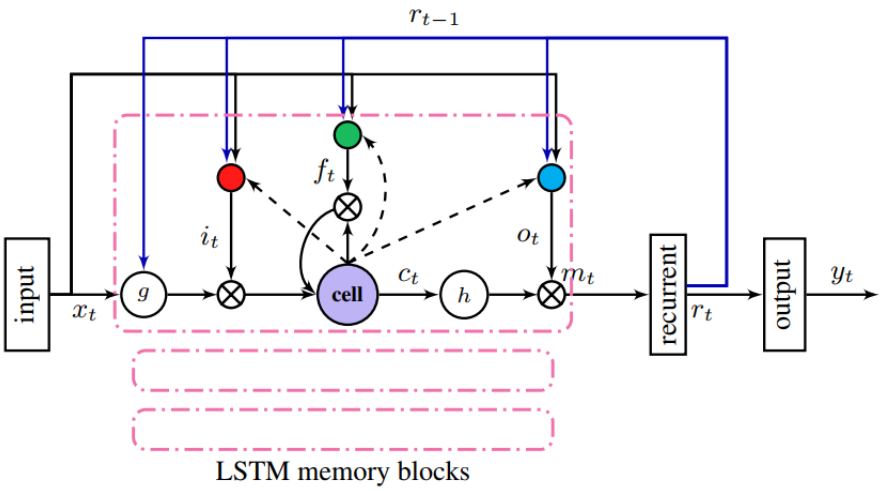


Figure 2.11: Diagram of a LSTM cell, adapted from [40]

superior performance, compared to regular recurrent neural network for overcoming the vanishing gradient problem. Thanks to LSTM's special units which are called memory blocks in the recurrent hidden layer, it can recall or forget an input from the multiple time steps before. The LSTM memory blocks contain memory cells with self-connections which can store the temporal state of the network. Besides this these connections there are special multiplicative units which controls the flow of information. Input gate controls the flow of the input activations into the memory cell. Output gate controls the output flow of cell activations into the rest of the network. The figure 2.11 shows the input gate as i_t, output gate o_t, and forget gate f_t. The earlier architecture of LSTM block contains an input, and output gate. Afterwards a forget gate is added to the memory block [14]. This addition prevented LSTM models to process continuous input streams that are not segmented into subsequences. Therefore, forget gate works as a scaler for the input, before it adapts the input according to the internal state of the cell, while forgetting or resetting the cell's memory. Lastly, modern LSTM architectures contain peephole connections between its internal and the gates in the same cell. In this way, it can learn precise timing of the outputs [15].

3 Related Work

In this section, the three main model categories, a brief review of existing ANN approaches to blood glucose level (BGL) prediction studies, and insulin sensitivity assessment models are presented. Due to differences of dataset and architectures used in the studies, it is hard to make direct comparison between them. Therefore, we will be providing dataset details and learning algorithm types used along with the architecture of the models.

In the next section we continue with presenting our static insulin sensitivity calculation method, and show the comparison of the static insulin sensitivity with basal rate.

3.1 Blood Glucose Prediction Models

In the study of [34] blood glucose prediction models, techniques are discussed within three categories, namely; physiological models, data driven models and hybrid models. Since physiological models are time consuming, and require previous knowledge, less time consuming models take advantage of machine learning techniques. On the other hand, data-driven models completely rely on non-physiological formulations. Hybrid models take the simplest forms of physiological models to process meal, and insulin metabolism. This outcome fits into the data-driven model to predict the future blood glucose level.

Since there is a compromise between accuracy of the prediction and prediction horizon, majority of the studies use the prediction horizon which varies between 15 min to 120 min. The most common prediction horizon is 30 min. Generally increased prediction horizon leads to a model with the less prediction capability.

Prediction of continuous blood glucose value is more popular than classified outcome. In other words, decision is; whether the model should learn to predict continuous values of BG which is the regression problem, or the model should map the outcome into some predefined classes which is a classification problem.

3.1.1 Physiological Models

Physiological models are used to simulate the BG metabolism in the form of compartment models used to describe the directly immeasurable processes. This type of models requires previous knowledge about the characteristics of insulin and glucose metabolism. Hence, several physiological parameters have to be adjusted before the simulation. This approach usually has submodels for digestion, and absorption dynamics of the intake carbohydrates, exercise, and insulin absorption feeding the main model which outputs the prediction of the BG level. Figure 3.1 has an overview structure of physiological models. In the diagram, meal intake (CHO) data used as an input to the model which simulates digestive system of human body. The next model uses exercise data in order to simulate exercise behavior of human body. The last model uses insulin intake for simulating the insulin mechanism of human body. The outputs from all these three models used as an input to the main physiological model in order to predict the blood glucose level.

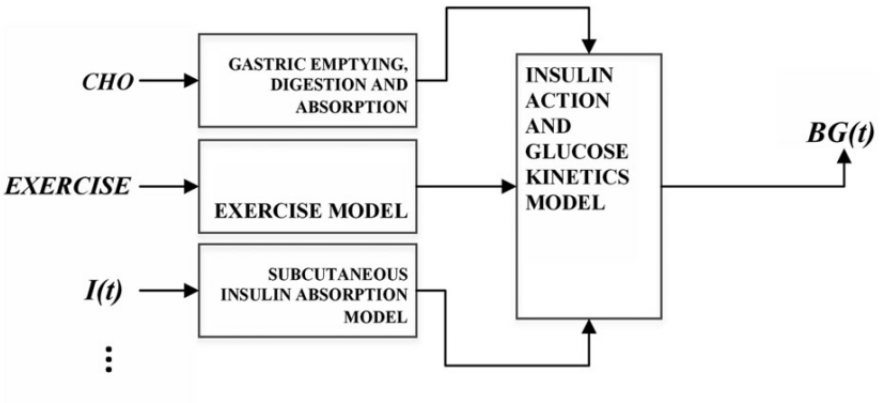


Figure 3.1: Diagram of physiological models, adapted from [34]

Physiological models also further divided into two categories based on the complexity. One of them has few actions and parameters, in order to capture the most important characteristics of glucose and insulin metabolism; so called minimal models. Unlike minimal models, the second type comprehensive models involve all the present information in the physiological system. Hence, comprehensive models allows simulating the metabolism of diabetic patient, and also retrieving the response. The most popular proposals for this model in literature are; [22], [9], and [5].

3.1.2 Data-Driven Models

Neural networks (NN) and Autoregressive Models (AR) are common examples of non-physiological models which are known also as data driven models. Non-physiological models only depend on CGM data, insulin, food intake, and few other inputs. Even though many models use more additional inputs, [24] suggests to use just cgm data as an input. There are other studies such as [53] also discuss about not to including the carbohydrate, and injected insulin information. Data-driven models use many machine learning algorithms; such as artificial neural network models, genetic algorithm models, grammatical evolution models, multi model approaches, Gaussian mixture models (GMM), regularized learning, reinforcement learning, random forest, Kalman filters, and support vector models. Figure 3.2 has an overview structure of data-driven models. In the diagram meal intake (CHO), continuous glucose measurement data (CGM), and insulin intake (I) are directly used as an input to the data driven model. The model gives output the predicted the blood glucose value.

Since there are so many techniques are available, it makes possible to experiment by mixing different techniques, and to increase accuracy of the model. Therefore, identifying a single technique as a most popular one is not possible. However the most popular models are divided roughly into five categories in the work of [34], which are namely; autoregressive and autoregressive with moving average (ARMA), other approaches(ARX-ARMAX), ANN technique, multi-model/mixed techniques, and others with AR-ARMA with additional external inputs (ARX-ARMAX). Models based on AR and ARMA use time series in order to derive future glucose concentration as a linear function of previous cgm values. ARX-ARMAX also considers external signals such as meal information and insulin plasma.

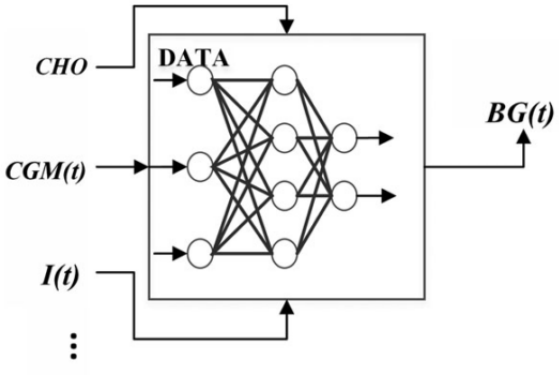


Figure 3.2: Diagram of data-driven models, adapted from [34]

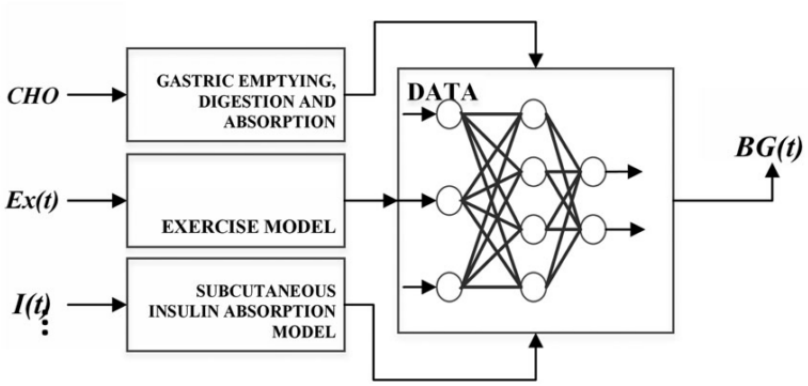


Figure 3.3: Diagram of hybrid models, adapted from [34]

3.1.3 Hybrid Models

Hybrid models usually use a physiological model followed by a data-driven model, in order to predict blood glucose value or the indication of hypo/hyperglycemia risk. Data-driven model learns the relationship between the physiological model's outcome and future outcomes. Resulting output could be either the number of classes or predicted blood glucose value. Since hybrid models use physiological model, this type of model takes advantage of meal information and insulin absorption mechanism. Hence, increasing the accuracy of the prediction of the overall model. Figure 3.3 has an overview structure of hybrid models, which is mixture of physiological model followed by a data driven model with exact inputs, and the final output of blood glucose level prediction.

The most popular hybrid prediction model for modeling the meal/glucose absorption is the Dalla Man meal model together with the Lehmann and Deutsch model. Berger's model is the most popular model, when the information about the insulin therapy is used.

3.2 Insulin Sensitivity Assessment Models

A person who is insulin-insensitive needs more insulin in order to lower the blood glucose level than a healthy insulin-sensitive person. In other words, a person who has insulin resistance is

more insensitive to the insulin. Within this thesis, we will be using the term insulin sensitivity rather than insulin resistance.

According to [19], hyperinsulinemic euglycemic clamp (HEC) is well known reliable standard for the measurement of insulin sensitivity. However this method is expensive in terms of time and money. Therefore, other methods for quantification of insulin sensitivity using the oral glucose tolerance test (OGTT) are proposed in order to develop insulin sensitivity indices. In the work of [19], these indices are grouped into two, namely; indices calculated by using fasting plasma concentrations of insulin and triglycerides, and indices calculated by using plasma concentrations of insulin and glucose obtained during 210 minutes of a standard (75 g glucose) OGTT. Homeostasis Model Assessment (HOMA-IR), and Quantitative Insulin Sensitivity Check Index (QUICKI) examples of the group where indices calculated by using fasting plasma concentrations of insulin and triglycerides. Within the next sections, we will briefly talk about different models used for insulin sensitivity calculation.

3.2.1 HOMA-IR (Homeostasis Model Assessment-Insulin Resistance)

This method was introduced by [25], and used to assess insulin resistance and beta-cell function from basal (fasting) glucose and insulin concentrations. HOMA calculates the insulin sensitivity by simple mathematically-derived nonlinear equations, using the relationship of glucose and insulin dynamics. Simplified version of this equation uses fasting blood samples, and divides the insulin-glucose product by a constant.

3.2.2 QUICKI (Quantitative Insulin Sensitivity Check Index)

QUICKI is a variation of HOMA equations which derives mathematical transformation of fasting blood glucose and plasma insulin concentrations empirically. [19] states that; QUICKI uses the fasting values of insulin (microIU/ml) and fasting plasma glucose (mg/dl) concentrations as in HOMA calculation. It is considered to be providing estimation of insulin sensitivity for obese and diabetic subjects better than HOMA-IR.

3.2.3 FGIR (Fasting Glucose/Insulin Ratio)

[48] discusses that FGIR is correlated with the insulin sensitivity index calculated from the frequently sampled iv glucose tolerance test with tolbutamide using the minimal model computer program.

3.2.4 Glucose Clamp Technique

This technique is also quantifying method insulin secretion and resistance.[11] and [23] use glucose clamp technique in their studies. It is used to measure either how well an individual metabolizes glucose, or how sensitive an individual is to insulin. Hyperglycemic clamp and hyperinsulinemic clamp are the two main techniques used. The hyperglycemic clamp requires maintaining a high blood glucose level by perfusion or infusion with glucose. This way it measures how fast beta-cells respond to glucose. The hyperinsulinemic clamp requires insulin

perfusion or infusion in order to maintain a high insulin level in order to assess how sensitive the tissue is to insulin. The hyperinsulinemic clamp is also called euglycemic clamp, meaning a normal blood glucose level is maintained.

3.3 ANN Approaches

In this section, we will review some of the existing ANN approaches. There are studies doing experiments different types of data (real or synthetic), different models (machine learning or simulation). We also mention the downsides of the different approaches when mentioned in the study.

In the work of [41], Elman recurrent ANN is used to predict future blood glucose values. The network is designed and trained using MATLAB. It consists of 95 neurons in the hidden layer. The network uses the input parameters including insulin, diet, exercise, BG levels, and other factors. A recurrent type of neural network was chosen for its superior performance in time series prediction problems. For the training and testing the ANN, clinical data were sourced from two patients. This data is derived by taking regular BG level readings using finger prick blood tests. Besides this, the diary entries recording information is also used. The problem mentioned with this study was the lack of data due to the less number of patients willing to participate in the study. The accuracy of the meal data records done by patients are also mentioned to be not accurate enough.

In an another work done by [29] a simulation model is used. The model was based on two submodels. One of the model is a three compartmental model describing short- and long-acting insulin effects on the glucose-insulin metabolism of a Type-1 diabetes patient. Another model, the compartment model for glucose absorption from the gut is used. Similar to [41] a recurrent ANN is used and trained using a real-time recurrent learning algorithm. The data from the carbohydrate metabolism model used as inputs to make the blood glucose level prediction. Dataset consists of Type-1 diabetes patient's record which contains 69 days of measurements. The measurements include blood glucose levels at breakfast, lunch, dinner and bedtime, insulin injections, and food intake. The results were quoted as an RMSE between the short term prediction of the ANN, and the actual data derived from the patient. This derived data is not used during ANN training. In the paper it is suggested that usage of additional patient information would improve the model performance. This additional data is data relating to sex, age, other diseases, and years of diabetes. Because this type of data helps to describe the patient more specifically and more in detail. In addition to these the data relating, the physical exercise was also considered as an important factor affecting the accuracy of the blood glucose prediction model.

In the other work of [31], compartment model and neural network system similar to their work reported in [29], is used. This model was used to predict blood glucose level of four children with Type-1 diabetes. The output got from the compartment model (for insulin intake and glucose absorption from the gut) was provided to an artificial neural network model, along with the most recent blood glucose level. In this work, recurrent artificial neural network is used. The network trained using a real-time recurrent learning algorithm and backprogation algorithm which is a feed-forward variety. The blood glucose level interstitial readings used in this experiment. This data is taken from the CGM measurements which is done every 5 minutes over a period of days. Besides the cgm readings, insulin and meal intake data were also recorded. The RMSE was used for the comparison of the predicted blood glucose level of

the artificial neural network model and the actual data record of the patient for that day. The patient data used for comparison is not used during the training of the neural network. For every patient, a corresponding artificial neural network model is used. Both the real-time recurrent, and the feedforward type of neural networks showed the similar performance. However real-time recurrent neural network is preferred due to its better weight adaptation capability when new input is supplied.

Artificial neural network with compartment model is also used in the work of [52] which is developed in previously mentioned [29], and [31]. This model is able to advise an insulin dose additionally. In other words, it is closed-loop system which combines the artificial neural network model with the compartment model in order to predict the short-term blood glucose level. The predicted blood glucose level is used as an input to another nonlinear predictive controller which gives the recommended level of insulin doses its output. The data provided to the model is derived from a mathematical model, which is intended to describe patients with Type-1 diabetes, with a sampling rate of 3 minutes. According to the results of the study, the developed closed-loop system can control blood glucose levels when realistic meal intakes are along with used.

In the next study we are looking into [35], which used artificial neural networks generated from the NeuroSolutions software package for predicting the blood glucose level for a duration of a 50 to 180 minutes. These generated neural networks are time-lagged feed-forward type, and are trained using the backpropagation algorithm. The dataset is derived from 18 patients with Type-1 diabetes. Patients were using devices capable of continuous glucose measurement at a sampling rate of 1 to 5 minutes over a period of 3 to 9 days. Besides the cgm measurements electronic diary is used in order to record the insulin dosages used by the patients. In addition to this, an electronic diary is used by the patients to record insulin dosages, carbohydrate intake, hypo/hyperglycaemic symptoms, lifestyle (activities and events), and emotional states. For the training of the networks datasets from 17 patients, for the testing dataset from the 18th patient are used. Results are evaluated with a performance metric mean absolute difference (MAD%) between the ANN prediction and the output from the CGM. According to the results, study reports that the predictions are more accurate for normoglycaemic and hyperglycaemic blood glucose level ranges than those in the hypoglycaemic range. The lack of hypoglycaemic events in the training dataset is stated as a possible reason in the study. Because of the few examples of such events, neural network predicts hypoglycaemic blood glucose range poorly. Another result obtained from the study is, increase in predictive length results in a decrease in predictive accuracy.

In another study done by [36], a feed-forward, fully connected multilayer perceptron with 3 layers is used. Blood glucose measurements within 20 minutes is used as an input which sourced from 15 patients wearing continuous glucose monitors. The glucose monitors measures the blood glucose with sampling frequencies ranging from 1 to 5 minutes. The training sets included both hypoglycaemic and hyperglycaemic events. Three ANNs were trained to predict 15, 30, and 45 minutes into the future, for a 24-hour period. The results were reported as an RMSE of the difference between the ANN prediction and the data from the patient which was not used during the training. The RMSE error was 0.56, 1.00, and 1.5 mmol/L for the prediction horizon of 15, 30, and 45 minutes respectively. In the study it is mentioned that, only using past cgm data as the input to the ANN, is a limiting factor of the performance of the neural network prediction. Therefore, it was suggested that besides the cgm data, additional input data, such as meal intake and insulin dosage, could improve the model's prediction performance.

Another study done by [8], used Marquardt algorithm to train an artificial neural network. The model makes blood glucose level predictions based on expected insulin doses, carbohydrate intake, and exercise. In the study, a mobile phone application takes input data in order to predict future glucose levels. The application intends to help to patient to adjust the carbohydrate intake, and maintain steady normoglycemia. Besides these, application is capable of storing previous blood glucose level measurements, insulin doses, carbohydrate intakes, and exercise in its own database and sending it via 3G connection to an endocrinologist. The paper states that, this feature allows the endocrinologist to better control the blood glucose level of a particular patient. Future work by the group will involve the development of the network to predict glucose levels 1 and 2 hours after the consumption of a meal, as well as predicting the amount of insulin necessary to control the blood glucose level of the patient.

In this section we summarized different approaches for predicting blood glucose level with artificial neural networks. The other studies also with similar approaches, which are not mentioned in this section are [21], [46], [47], [30], [33], [16], [50], [12], [45] and [3].

4 Data Resources

The data set we use provides the main interesting data for the diabetes therapy. Most of the patients uses the insulin pump which produces data giving information about continuous glucose measurement, bolus value, basal value, exercise information, hearth rate, and location. In the next sections, we discuss features of OpenDiabetesVault Framework and data resource for our static insulin sensitivity calculation and neural network model.

4.1 OpenDiabetesVault Framework

The OpenDiabetesVault framework allows gathering, importing, processing, exporting and visualizing. It supports several devices for data import. Medtronic CSV, Freestyle Libre CSV, Google Fit CSV, Sony Smartband 3 SWT12 database dumps are supported for data import. Every importer works in two stages; parsing and interpretation. A parser reads values from one data point and converts it into the internal data format. The interpreter annotates data points and does very basic filtering on a data series. It also adds semantic implications (e.g. when a pump suspend is read, the interpreter would add a basal value 0). Data can be exported to our csv data format. Also, a compressed (deflate), and signed format is available odv which is basically a zip archive with the csv file and signature file. Furthermore, slicing information which is produced by the DataSlicer can be exported as csv for the plotting script. For the visualization of the data the odv-CSV data, the plotting script using matplotlib from a separate project can be used.

4.2 Vault Database

bgValue: Provides the blood glucose values. Blood glucose value is measured out of freshly harvested blood from a finger. It can be considered as the ground truth however the measurement tolerances are very low.

cgmValue: Provides the continuous glucose measurement (cgm) data from a sensor placed on the skin. The sensor uses the cell liquids to measure the glucose level. Naturally this value can not be considered as the same as the blood glucose value, since the system calculates this value from the cell liquid. To achieve the close value with cgm, the different systems use different adjustment methods. Some are self calibrating, some are manually calibrated and provide a raw value, displayed in the data set as cgmRawValue. Besides this, all systems have in common, that measured values have a time delay of around 15 to 30 minutes. The values are mostly reliable, but sometimes the calibration is not accurate. For this reason, the cgmValue should be rated by the given blood glucose level.

cgmAlertValue: Provides cgm values, which have attracted some attention by the patient. The different systems have different mechanisms to provide current values to the patients. Some use audio alert systems and other have to be manually triggered to output the current value.

bolusCalculationValue: Provides blood glucose values for the insulin dose calculation. This is usually a copy of the bgValue or the cgmValue. The patient knows this value for sure.

basalValue: Provides the basal insulin value. Basal insulin value is the insulin amount required for the basic energy delivery system which is the energy usable for the cells. This corresponds

to the insulin amount to stabilize the blood glucose to an optimal value when the patient does not do any physical activity and does not eat anything. This value is empirically generated and might differ within every patient. Therefore, it can be considered partly correct. Some technical details regarding the status are provided at basalAnnotation. The values are accurate in time and amount.

bolusValue: This value provides the bolus insulin value for every meal containing carbohydrates. Patients with diabetes should take corresponding amount of insulin for their every carbohydrates containing meal intake. The amount of the insulin should be taken by the patient, is calculated by the insulin pump with a programmable translation rule which is determined empirically. Thus, this value should not be considered completely correct. Even though there are many insulin types, we do not consider all of them. We only consider the one called "rapid-acting" type. It has no instant effect, rather after 15-20 minutes. Its effect almost linearly decreases over 3 hours.

bolusAnnotation: Bolus insulin can be applied as a bulk portion or as a square wave over a given time. How the insulin is applied is stored in, with the given time in minutes for square wave types. The amount and time point of the value is perfectly accurate.

mealValue: It provides the data for consumed carbohydrates by the patient. Therefore, patient has to use the pump to calculate the needed insulin. Patient might calculate this value before, after or both before and after the meal. Thus, this value should be considered with +/- 20 minutes time drift. In particular the meal intake for fixing the hypoglycemia (low blood glucose) value are not completely accurate since the value of the meal is estimated by the patient, rather than the insulin pump. In case of constant wrong estimation, the pattern can be also considered as reliable for the specific patient.

pumpAnnotation: It provides some technical information about the insulin pump. If the data varies from the usual flow, that is an indication of a technical issue usually. Some pump events are pump rewind, pump fill, pump fill interpreter, pump suspend, pump unsuspend, pump reservoir empty and pump unknown error.

exerciseTimeValue: It provides the information the exercise duration in minutes. It is measured by the Google activity framework reverse geocoding. It also allows manual entry from the patient. The resource used for exercise time value is stored in exerciseAnnotation. The possible entry values are manual, other, walk, bicycle, and run. The correctness of data is not really critical, since the exact influence of exercise information on diabetes is not completely known and it varies within every patient. Therefore, exercise entries with a duration lower than 20 minutes can usually be ignored.

heartRateVariabilityValue: It provides heart rate variability value which is measured by a Sony Smartband 2. This value can be considered accurate. Until now it is not proven whether this information relates with diabetes related data or not. Apart from the heart rate variability data, stress is a known factor which has effect on diabetes therapy which is proposed by Sony.

sleepAnnotation: It provides information whether the patient is awake or sleeping. It is measured by Sony Smartband 2, and is nearly correct, but boundaries between sleep and awake cycles might be different from the actual by +/-20 minutes.

locationAnnotation: It provides location categories which is based on the measurement from the Google Maps reverse geocoding with including automatic tracking. Possible location categories are; transition (the patient is in move between transitions), home (the patient is at home), work (the patient is at work), food (the patient is at a location for eating), sport (the

patient is at a location for sport), other (the patient is not in transition but no other category is recognized).

4.3 Static Insulin Sensitivity Calculation

In the previous chapter, other methods for calculating insulin sensitivity are discussed. In this section we will present our technique for assessment of static insulin sensitivity using the dataset derived from the vault database. Our calculation roughly consists of 3 main stages; finding time series with the bolus event, discarding the series which does not comply to our criteria and finally calculating the static insulin sensitivity value. Figure 4.1 has detailed steps of our model.

We begin with determining the all bolus events. Based on the location of the bolus event, we retrieve sub time series containing events 2 hours before and after. This process is done for every bolus event detected. Later, we check each sub time series whether they contain another bolus event within maximum half hour afterwards from the original time stamp of the bolus event. If such bolus event is found, then these found bolus event values are summed up and saved back to our bolus event value. If found bolus events are not in this half hour range the sub timeseries is discarded.

In the second stage of our calculation, the cut time series are further filtered. Each timeseries is checked whether it includes any meal, exercise or bolus sqare event. The timeseries containing any of these events are again discarded. We continue with the remaining set, and look for the first cgm event after our bolus event's time stamp with a predefined delay. We used 15 minutes for the predefined delay. This delay is used due to the reaction time of the insulin injection. Finally we save the last cgm measurement belonging to this time series. Insulin sensitivity factor is calculated as follows:

$$\frac{firstcgm - lastcgm}{bolusvalue}.$$

If the difference $firstcgm - lastcgm$ is positive, the value is valid for static insulin sensitivity calculation, otherwise again it is discarded. The valid differences are divided by bolus amount, which gives us the static sensitivity factor.

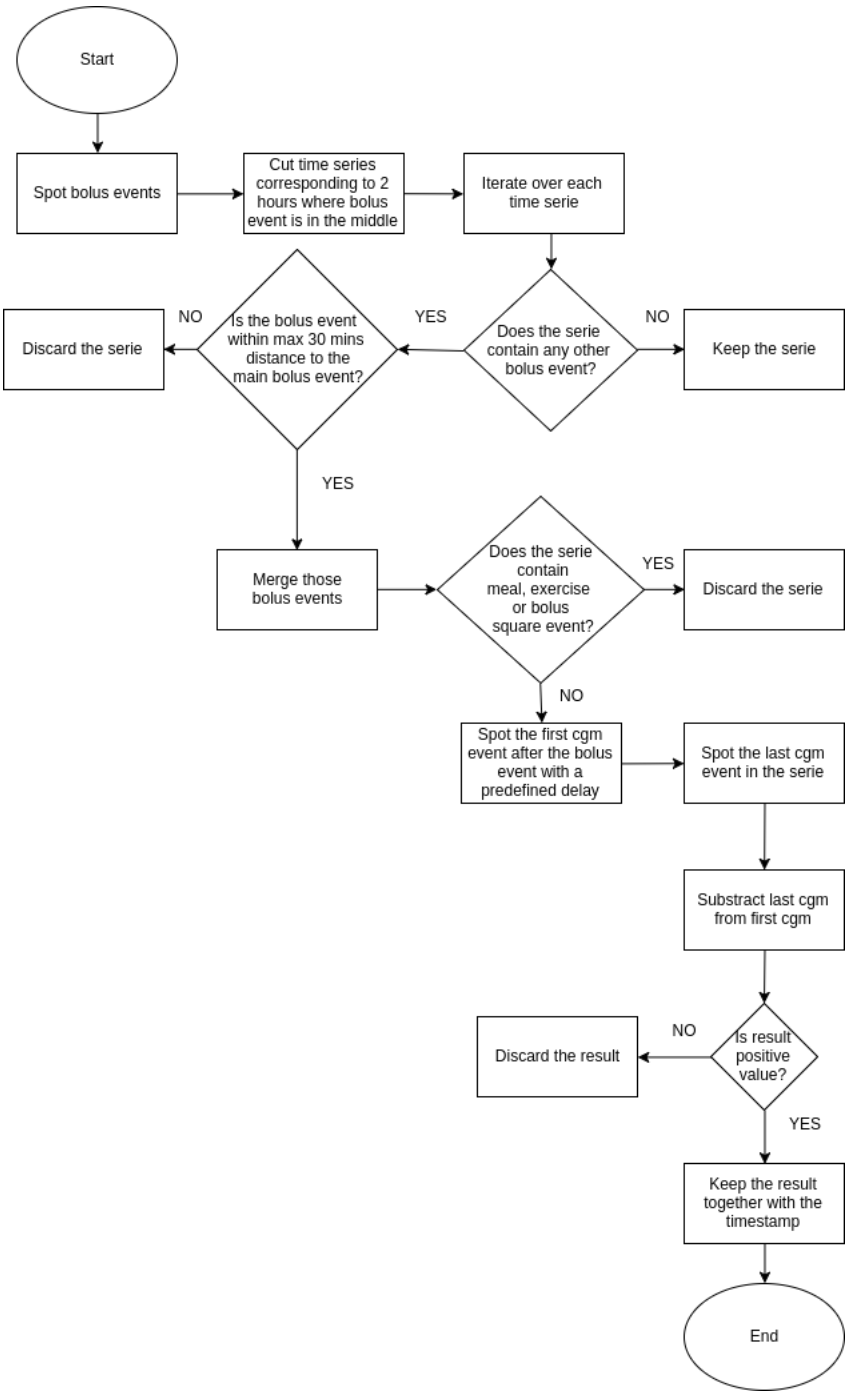


Figure 4.1: Static Insulin Sensitivity Calculation Flowchart

5 Experiment Setup

We now briefly describe our experiment setup. The aim of the experiments can be roughly divided into two main categories. First of all, we aim to show whether our static insulin sensitivity calculation correlates with the basal rate or not. Secondly, we want to find out whether this newly generated feature can contribute to blood glucose level prediction when used as an additional feature. Figure 5.1 shows the architecture of our model. In the next chapter, we explain how we find an answer for these two main questions.

Static insulin sensitivity factors together with corresponding timestamp which is calculated over the vault database is used for basal rate verification. For every hour of a day there are multiple sensitivity values retrieved. The count of data points for each hour of the day can be seen in the top of the plot in the figure 6.7 which are aligned with the hours of the day. For the plots we used matplotlib.

Our overall prediction model is a hybrid model consisting of a physiological model and a data-driven model. The static insulin sensitivity calculation model is a physiological model since it obtains values using predefined assumptions and based on pure mathematical formula. Artificial neural network model is type of data-driven model. Our data driven model uses input features from vault database and the static insulin sensitivity factor. Our neural network model is a one-dimensional convolution neural network. For this model, python Keras library with tensorflow backend is used.

Time series data retrieved from vault database preprocessed and simplified before it is used as an input to our convolutional neural network model. In the next section, we explain how the preprocessing of the database is done.

5.1 Preprocessing of Dataset

The time series are preprocessed before used in the prediction model. The features are used either used as discrete values or one-hot encoded. We summarize the procedure in the table 5.1. The column Relevance is used to group the similar features. The column Sub types contains

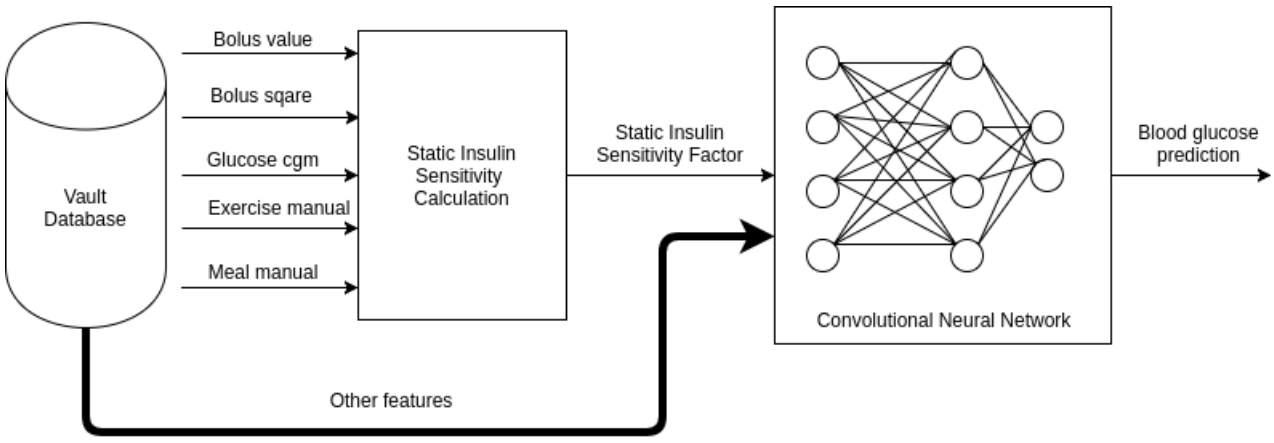


Figure 5.1: Architecture of our hybrid model

similar features according to the relevance. If the features in the sub types are merged into another new field, its name is given under the column Merge Info. Otherwise, the time series are not changed (mentioned with -). The column Usage is used in order to state whether the time serie is used as it is (discrete), one-hot encoded or not used in our model at all (mentioned with -).

The features basal profile, basal manual, and basal interpreter are merged into new time series called basal. Meal bolus calculator and meal manual are merged into another feature called meal. Sleep light, sleep rem and sleep deep are merged under new field called sleep. The other features; bolus normal, bolus sqare, glucose cgm and dm insulin sensitivity are used as it is. Glucose cgm alert, cgm sensor finished, cgm sensor start, cgm connection error, cgm calibration error, pump rewind, pump prime, pump suspend, pump autonomous suspend, pump unsuspend and sleep are converted into one hot encoded vector before it is used as input. The rest of the time series ignored and not used in our model.

The input data is parsed as a matrix with total number of 3990264 rows and 5 columns. Each column has different features and rows are increasing with the same direction of time step. We divided our dataset into two sets; training and test set, where training set corresponds to 0.5, test set 0.25 and validation set 0.25 of dataset.

Relevance	**Sub types**	**Merge Info**	**Usage**
Basal	basal profile, basal manual, basal interpreter	basal	discrete
Meal	meal bolus calculator, meal manual	meal	discrete
Sleep	sleep light, sleep rem, sleep deep	sleep	one-hot encoded
Bolus	bolus normal, bolus sqare	-	discrete
Glucose	glucose cgm	-	discrete
Glucose	glucose cgm raw, glucose cgm calibration, glucose bg manual, glucose bolus calculation, glucose elevation 30	-	no
DM Insulin Sensitivity	dm insulin sensitivity	-	discrete
CGM	glucose cgm alert, cgm sensor finished, cgm sensor start,cgm connection error, cgm calibration error	-	one-hot encoded
CGM	cgm time sync	-	no
Pump	pump rewind, pump prime, pump suspend, pump autonomous suspend, pump unsuspend	-	one-hot encoded
Pump	pump fill, pump fill interpreter, pump no delivery, pump untracked error, pump reservoir empty, pump time sync, pump cgm prediction	-	no
Exercise	exercise manual, exercise other, exercise walk, exercise bicycle, exercise run,	-	no
Heart rate	heart rate, heart rate variability	-	no
Stress	stress	-	no
Location	loc transition, loc home, loc work, loc food, loc sports,loc other, other annotation	-	no

Table 5.1: Summary of preprocessing of Vault Database

6 Experiments

In this chapter we mainly describe experiments done with our prediction model. Firstly, we explain our prediction model in more detail. We will compare our experiment results which were run with different parameters. We want to know whether static insulin sensitivity calculation value improves the overall model prediction or not. In order to find out that, we trained our convolutional neural network model, with and without static insulin sensitivity feature, while keeping the rest of the parameters of the network fixed. In the following paragraphs, we describe our experiments in which we trained our model with different window sizes, number of epochs, and different prediction horizons. We make two types of forecasting; multivariate one-step and multivariate multiple-step. In the next section, we explain the CNN model in more detail.

6.1 Convolutional Neural Network Model

We use convolutional neural network from sequential Keras model consisting of two convolutions and two pooling layers. We apply is one dimensional convolution operation. Unlike image recognition tasks where input data is three dimensional, our timeseries data is two dimensional. The difference between one dimensional and two dimensional convolution operation is; one dimensional filter's height is fixed to the number of input time series, and it can only slide along the window dimension. Since the input time series do not have any spatial or ordinal relationship between them, we are looking for patterns that are invariant with respect to the subsets of the time series. We use deep neural network since a neural network without a hidden layer is only sufficient for modeling a linear function. Inside a deep convolutional network, each hidden layer is responsible for extracting some set of useful features. These features are used by the next layer hierarchically and becomes more abstract in the next layer. We aim our network model to learn right filters capable of deriving such abstract features of the glucose metabolism from the input data. Therefore, we choose a convolutional neural network with two hidden layers. In order to introduce non-linearity ReLU operation is applied after convolution. Each convolutional layer is followed by a pooling layer, which reduces the dimensionality of the input.

We want to capture the meaningful features of blood glucose and insulin metabolism dynamics of a diabetes patient via the CNN model for better forecasting. Meal intake is one of the most important factors affecting the blood glucose level. Therefore, we want our model to learn characteristics of patients blood glucose and insulin metabolism dynamics within each meal time. In order to achieve this we have to divide the input data in smaller sets which corresponds to the three main intervals of a day; morning, evening, and night. Therefore, we have to decide upon meaningful window size and batch size. Window size is the number of time series values used to predict the next one. Batch size is the value used for the number of inputs taken by the network to before it updates its internal parameters. For deciding upon the meaningful window size and batch size for our prediction, we had to make certain assumptions.

First of all, we assume that the patient is having meal three times a day on a regular basis. Secondly, we roughly estimate the number of cgm measurements as 600 per day. If our dataset did not have missing values and if each measurement was done with the same time stamp, every column (corresponding to different feature) would be synchronized. Therefore, we would not

need to interpolate each column of the dataset. In this scenario window size of 200 would be enough to represent morning, evening and night periods of a day.

Unfortunately in our case, our dataset has missing values, and no column is synchronized with each other which makes the decision process more difficult. In order get rid of the missing values and make our training possible, we had to interpolate each column of the dataset. Consequently, in our case, we need to approximate the number of time steps which corresponds to 200 continuous measurement. Due to missing values of each column we decided upon much bigger window size would be required.

While with a larger batch there is a significant degradation in the quality of the model, as measured by its ability to generalize. While smaller batch size allows more frequent updates after each propagation, it allows network to train faster. On the other hand the smaller batch size causes the gradient descent algorithm to estimate the gradient less accurate. In this case, network might not able to converge to a global optimum. Considering these factors we decided to use 2000 for the batch size.

6.1.1 Filter Length and Number of Filters

For our convolutional neural network, we had to decide on the parameters such as; filter length, number of filters, and network depth. Smaller kernel size produces more feature maps and allows model to perform more accurate. Additionally, it results in faster training. We have 4 features in total. Considering these factors, we use kernel size of 4 in our first convolutional layer. Kernel size is also called number of filters. Number of filters are directly related with the number of columns of the output shape of the neural network. Bigger the filter length allows to gather more information, but also this might lead overfitting due to more parameters. Therefore, smaller filter lengths which are stacked in the network has more advantage than having one large filter. Considering this factor, we used filter length of 5. Filter length effects the number of rows of the output shape of each convolutional layer. The figure 6.1 shows the architecture of a CNN with 4 filters, each has length of 5. Since the filter with length 5 slides over the input with dimensions (None, 50, 4), one step at a time and produces output with (None, 46, 4). In this figure "None" keyword represents the number of samples which can be given as input to the network with the dimensions (50, 4), which can be considered as a matrix with 50 rows, and 4 columns. Here each column has one value from each feature; bolus sqare, glucose cgm, meal bolus calculator and basal. The row number 50 corresponds to the input window size. Similarly, second convolutional layer takes the input with dimensions (None, 23, 4) and produces output with dimensions (None, 19, 4) after applying the filters with length 5.

Each pooling layer, does the operation of max pooling and reducing the dimension x of the input by factor of 2 where the input has a shape of (z, x, y). This layer takes two inputs from the input vector, discards the smaller value and passes the greater value to the next layer. As it can be seen from the figure 6.1, each max pooling layer reduces the input with shape (None, 46, 4) and (None, 19, 4) by factor 2 and outputs with dimensions (None, 23, 4) and (None, 9, 4) respectively.

We use ReLU as an activation function. The input data given to this layer as an input has a shape of length of window size and number of input series. A pooling layer which downsamples the output from this layer by using max pooling operation is used. The following layer, another convolutional layer with the same parameters is used. Similarly, this convolutional layer is followed by another maxpooling layer. Finally, a dense layer, another layer which consists of

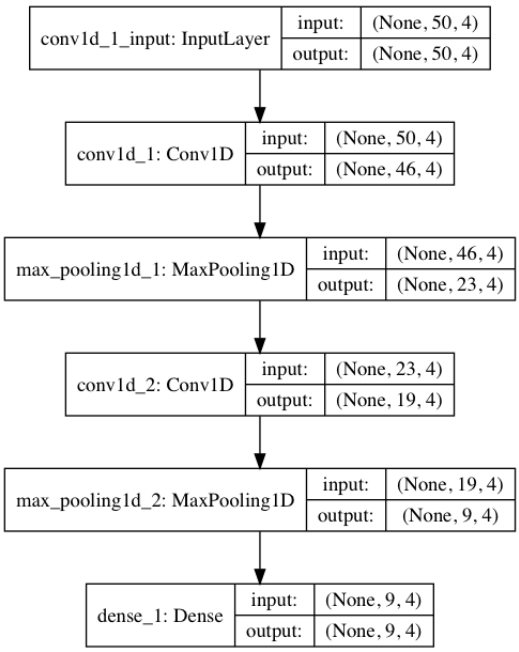


Figure 6.1: Multistep Prediction CNN Model with input window size 50 and predicted timesteps 9

perceptrons is used. This layer has a linear activation function and outputs the predicted values. The linear activation function is the simplest activation function, this function $f(x) = x$ simply passes the input without any modification.

The model is evaluated with error metric MAE and a loss function in terms of MSE. We use these metrics for comparing different trained models. The errors in the tables correspond to the multiple input-multiple output model. We preferred 10 epochs for the training with longer window size. Each epoch means that the training algorithm does a complete pass over the entire dataset total 10 times. In the next sections, we discuss the prediction results and performances.

6.2 Multivariate One-Step Prediction

In this section, we summarize the results from our CNN model which predicts one step future value. We also compare one-step prediction results of two different CNN models trained with and without static insulin sensitivity value. Therefore, we use window size of 50, 100, and 500. We used 10 epochs and batch size of 2000. This showed us that increasing the window size of one-step prediction model, does not contribute to the performance of the prediction, on the contrary increases the training time considerably. Experiments with different window sizes of 50, 100 and 500 are summarized in the tables 6.1, 6.2, and 6.3. During the training, number of epochs is fixed to 10. The best performance is achieved with the window size of 100.

We extracted 1000 time steps long actual blood glucose level, predicted without static insulin sensitivity. Behavior of the predictions can be seen in the figure 6.2. In this plot is also can be seen, how two predictions are close to each other and the actual value. However one-step prediction does not give insight about the future behavior of blood glucose level, even though prediction error is very low. Hence, we made another model which is capable of multiple

CNN Model	loss	MAE	val_loss	val_MAE
with Static Insulin Sensitivity	1.8199	0.6687	10.0022	1.4774
without Static Insulin Sensitivity	2.2711	0.7879	12.0884	1.6721

Table 6.1: Comparison of Multivariate One-Step Prediction with and without Static Insulin Sensitivity value (window size 50, number of epoch 10, batch size 2000)

CNN Model	loss	MAE	val_loss	val_MAE
with Static Insulin Sensitivity	1.4143	0.6199	8.0298	1.4363
without Static Insulin Sensitivity	1.3647	0.5179	8.5898	1.2114

Table 6.2: Comparison of Multivariate One-Step Prediction with and without Static Insulin Sensitivity value (window size 100, number of epoch 10, batch size 2000)

step forward forecasting capability, with given several (window size) measurements of previous blood glucose levels.

6.3 Multivariate Multiple-Step Prediction

In this section, we summarize the result of the multi-step prediction CNN model. We trained four different models with window sizes 50, 100 and predicted future steps 9, 19, 22, 44. Each models' performance result is summarized in table 6.4. As we increase the predicted time-steps, prediction error increases. The results in table 6.4 shows the prediction error of total four features; bolus sqare, glucose cgm, meal bolus calculator and basal.

For the visualization of the predictions, we made a plot from each model $1-4$ in the table 6.4. In the figures, the input timeseries and the following timesteps are chosen from separate test set, which is not used during the training. When there are many missing values in the dataset after the preprocessing, the values are not varying much. Therefore, it makes the prediction unrealistic. Even though such test timeseries gives us prediction much closer to the actual values, we are interested in the part which is closer to the real life situation. Hence, the portion which we used for the illustration of the prediction is especially taken from dataset where there is not much interpolation.

In the figures 6.3, 6.4, 6.5, and 6.6 predicted time-steps are shown in red. The model with 9 time-step prediction window, approximately corresponds to 15 minutes of prediction horizon. The second with 19 predicted time-steps, corresponds to 25 minutes of prediction horizon. Third model with 22 predicted time-steps corresponds to 35 minutes of prediction horizon, and the last model with 44 timesteps corresponds to 1 hour of prediction horizon approximately. For the approximation of prediction horizons from the timesteps, we counted each blood glucose

CNN Model	loss	MAE	val_loss	val_MAE
with Static Insulin Sensitivity	1.7289	0.7293	9.8728	1.5246
without Static Insulin Sensitivity	3.2356	1.0809	19.3925	1.9252

Table 6.3: Comparison of Multivariate One-Step Prediction with and without Static Insulin Sensitivity value (window size 500, number of epoch 10, batch size 2000)

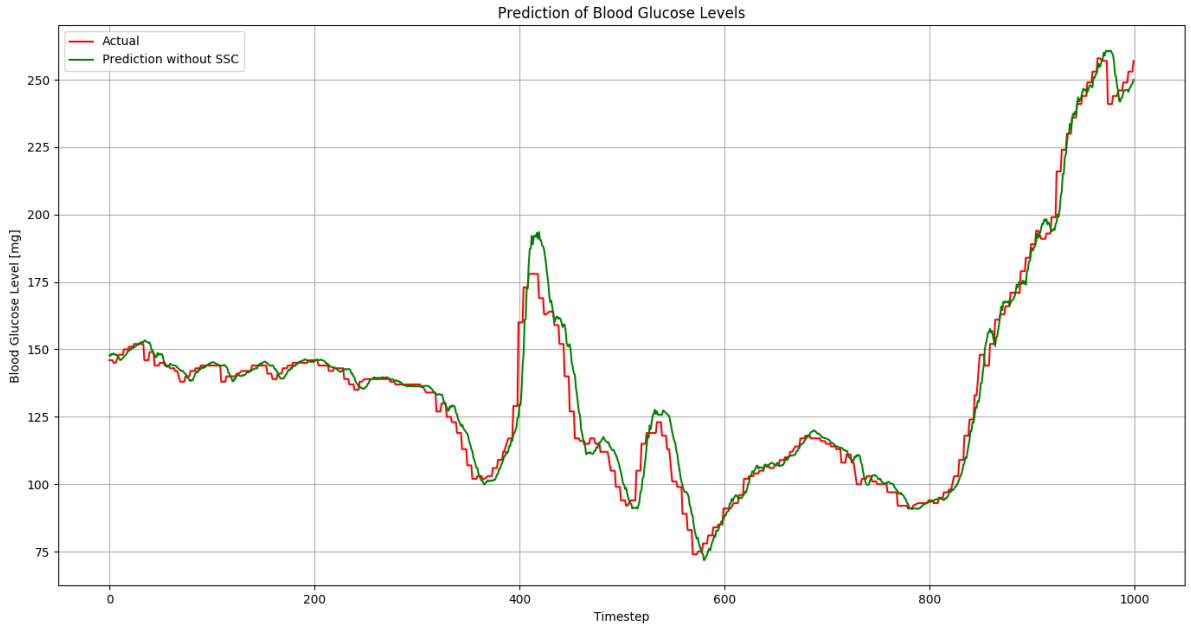


Figure 6.2: Plot for Actual and Predicted Blood Glucose Levels of 1000 time steps

Model Number	Input Window Size	Time Steps Predicted	loss	MAE	val_loss	val_MAE
1	50	9	10.3113	1.0729	70.4468	2.7346
2	50	19	12.064	1.1515	83.2704	3.0644
3	100	22	29.5713	1.3807	213.7033	4.3906
4	100	44	40.0762	1.6637	280.4602	5.3311

Table 6.4: Performance Evaluation of Multivariate Multi-Step Prediction with different prediction time steps (number of epoch 10, batch size 2000)

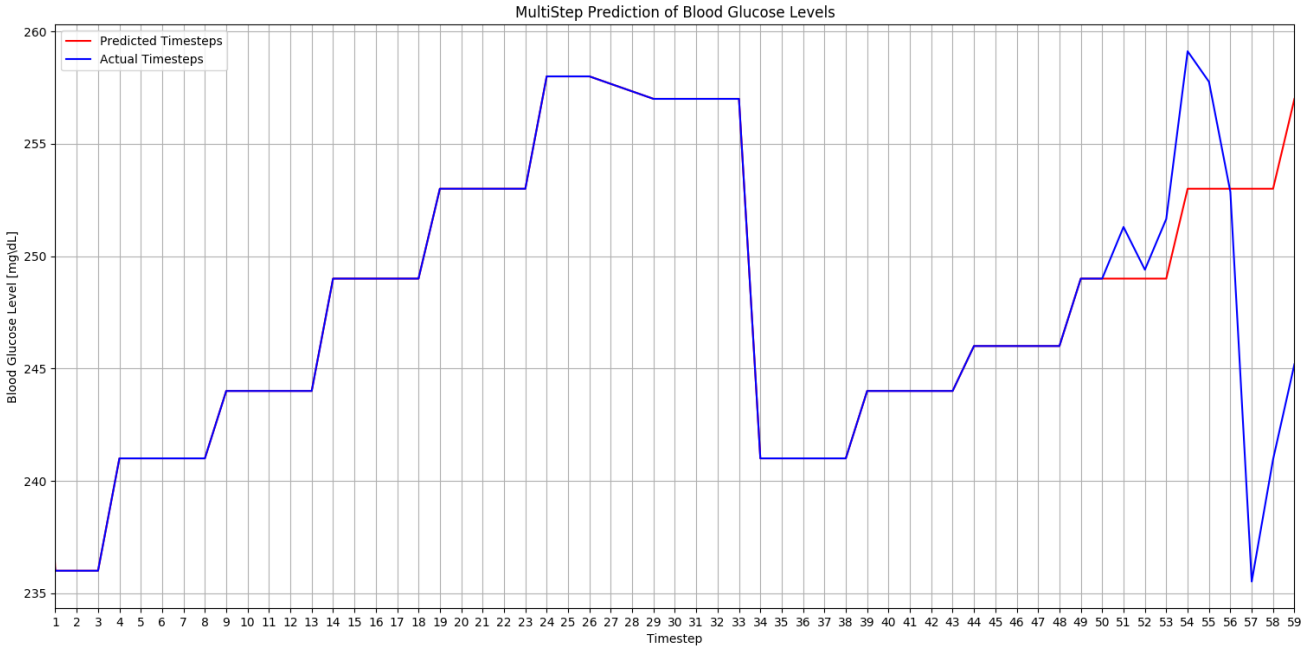


Figure 6.3: Model 1: Input window size 50, and 15 minutes prediction horizon (time-steps 9)

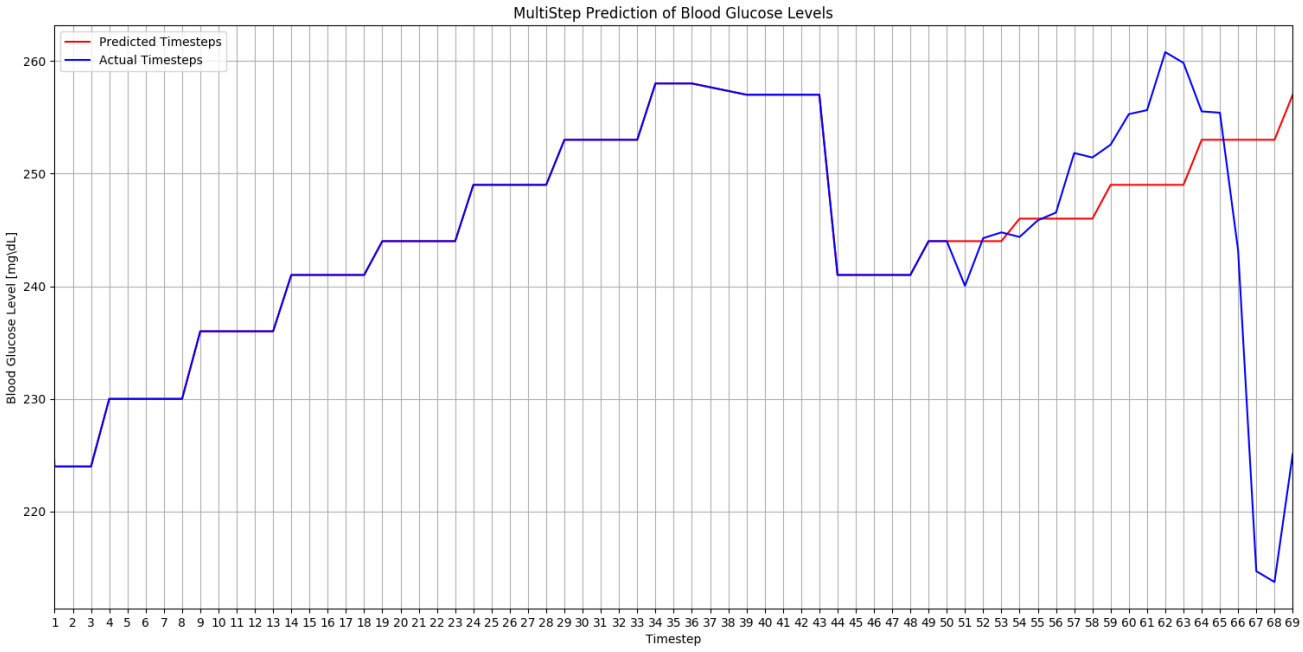


Figure 6.4: Model 2: Input window size 50, and 25 minutes prediction horizon (time-steps 19)

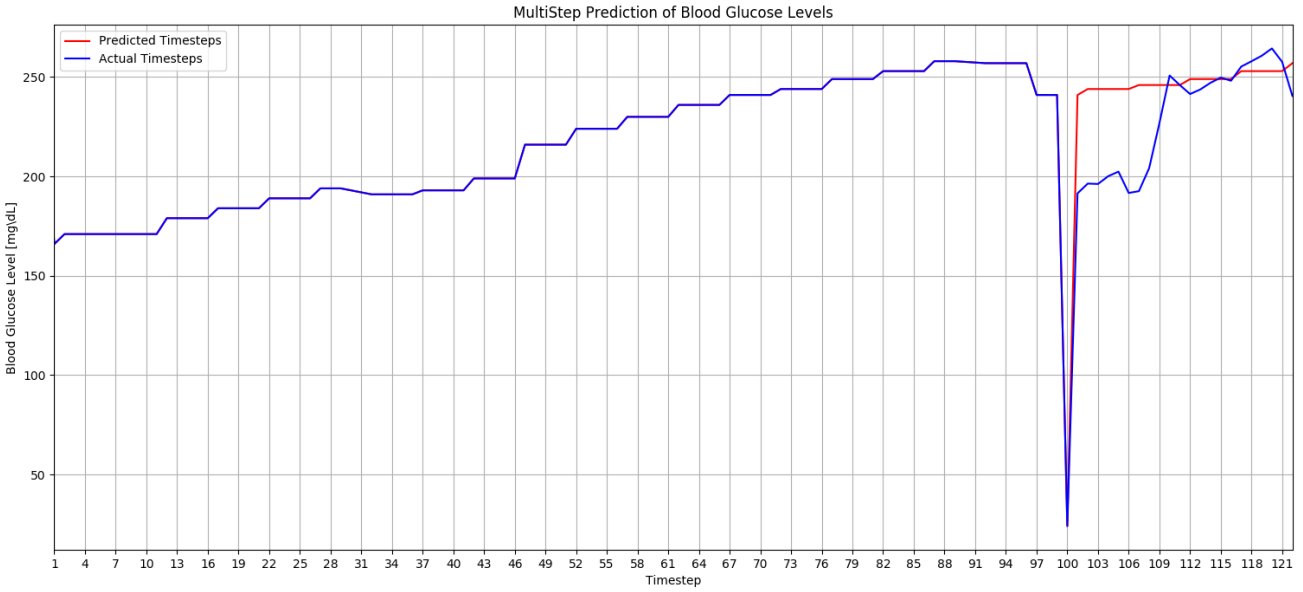


Figure 6.5: Model 3: Input window size 100, and 35 minutes prediction horizon (time-steps 22)

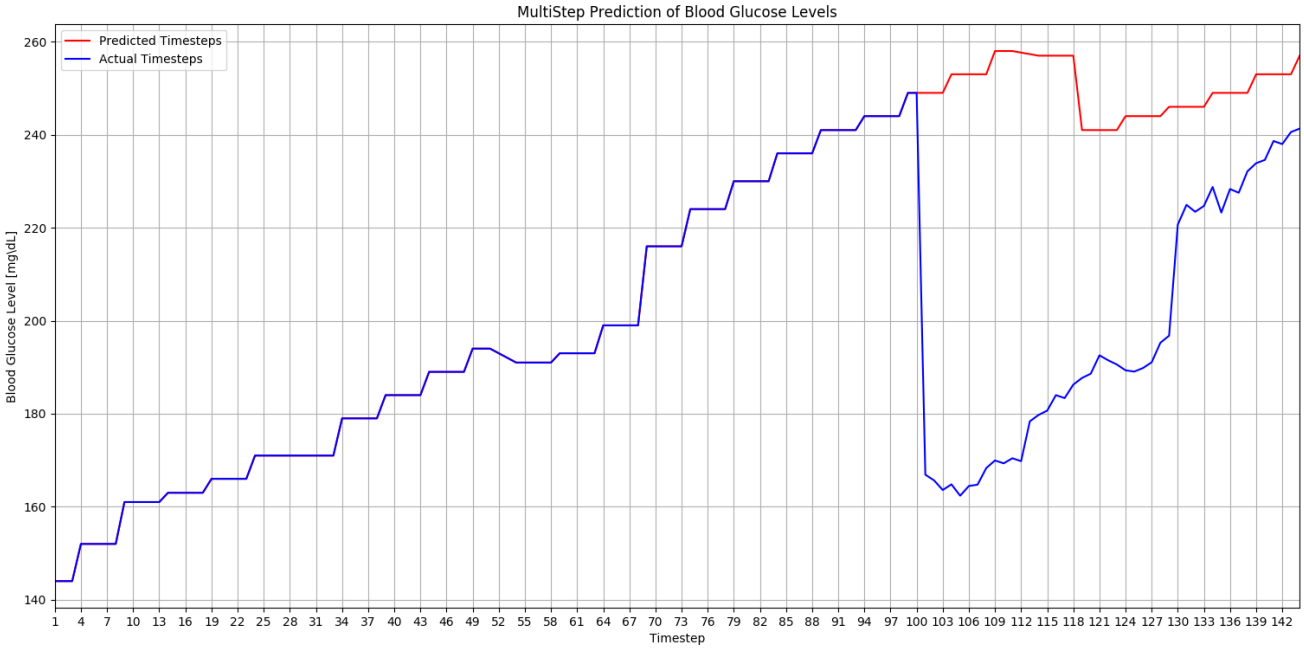


Figure 6.6: Model 4: Input window size 100, and 1 hour prediction horizon (time-steps 44)

measurement with minimum 1.0 mg/dL difference in the timeseries as 5 minutes. According to the results, we conclude that the model performs worse where the blood glucose value drops sharp, compared to other places where blood glucose level is more stable. In the next section we discuss about the static insulin sensitivity calculation model's results, and we evaluate the relation of it with the basal rate. We also evaluate our overall model considering this relation.

6.4 Evaluation of Models

We used two methods to verify the applicability of calculated static insulin sensitivity factors. First, we compare our finding with the basal rate. The figure 6.7 shows the boxplot of sensitivity values for every hour of a day. There are multiple values retrieved for each hour of a day. We removed outlier values from the plot. The red step line represents the median of each sensitivity boxplot. For readability purpose basal rate boxplots are not shown in the plot, instead lower and upper quartile of basal rate are represented with blue step lines.

According to the figure 6.7 we do not see correlation between the static insulin sensitivity values and the basal rate. In order to have a concrete results we calculated the correlation values. Correlation coefficient is used to measure the strength of a linear association between two variables. The value $C(X,Y) = 1$ means a perfect positive correlation and the value $C(X,Y) = -1$ means a perfect negative correlation between variables X and Y. For the correlation calculation following formula is used where X and Y represents the data of two different variables:

$$C(X,Y) = \frac{\sum_i (x_i - \bar{x})(yi - \bar{y})}{\sqrt{\sum_i (xi - \bar{x})^2}\sqrt{\sum_i (yi - \bar{y})^2}}$$

The correlation coefficients which are in the table 6.5 shows us that there is neither positive nor negative correlation between average insulin sensitivity factor and average basal rate and

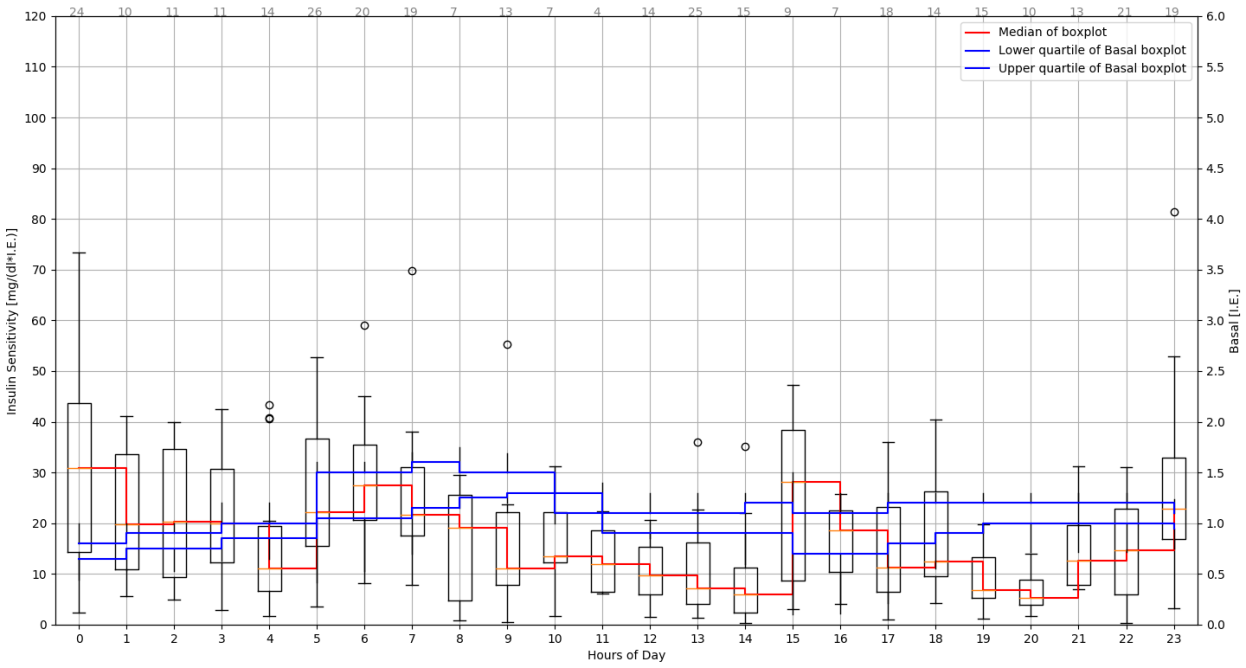


Figure 6.7: Boxplot for Static Insulin Sensitivity Verification with Basal Rate

C(X,Y)	Correlation coefficient
C(Avg Basal Rate, Avg Insulin Sensitivity)	-0.0683446
C(Median Sensitivity, lower quartile of Basal Rate	-0.2455060
C(Median Sensitivity, upper quartile of Basal Rate)	-0.0497927
C(Median Sensitivity, median Basal Rate)	-0.2438207

Table 6.5: Correlation coefficients for Insulin Sensitivity and Basal Rate Verification

between median insulin sensitivity and upper quartile of basal rate. However there is some negative correlation between median insulin sensitivity and lower quartile of basal rate, and between median sensitivity and median basal rate.

In the experiments with one-time step prediction, we got similar results with different window sizes. While prediction with window size 100 has better performance without the static insulin sensitivity feature, the other predictions with window size 50 and 500, we see that the model with the static insulin sensitivity feature performed better. Besides these, it is hard to conclude about the contribution of this additional feature, since the error rates are very close to each other. Besides, we see constant increase in the error rate of models when larger prediction horizon is used. On the other hand increase in the input size is decreasing the error rate.

We calculated the MAE and MSE of predicted blood glucose levels individually using the package of Scikit Learn metrics.mean squared error and metrics.mean absolute error functions. Regression error metrics MAE and MSE of predicted Blood Glucose levels are summarized in the table 6.6.

Model Type	MAE	MSE
Prediction with SSC	2.236405	35.746923
Prediction without SSC	2.533498	36.845206

Table 6.6: Error rates of Blood Glucose prediction

7 Conclusion

In this thesis, we presented a novel approach for static insulin sensitivity calculation in order to make a better prediction of blood glucose levels of diabetes patients. With this approach we aim to minimize the risk of hyperglycemia, and consequently reducing the harmful complications of the disease. In our experiments, we have verified that our static insulin sensitivity calculation neither correlates with basal rate, nor brings significant improvement to the blood glucose level prediction model, when used as an additional feature. This clearly indicates that the findings of the two experiments are compatible with each other.

7.1 Summary

Throughout the thesis, we explained necessary background information for understanding the disease diabetes mellitus, machine learning basics together with artificial neural networks, and deep learning. Afterwards, we covered state-of-the-art methods used in the literature regarding blood glucose prediction with different approaches. We categorized these models into three types; Physiological Models, Data-Driven Models, and Hybrid Models. We introduced data resources used in the experiments, and how the experiment is setup.

We have done our experiments with the calculation of static insulin sensitivity with a personal clinical data from a diabetes patient. We used basal rate to verify the correctness of our novel approach for static insulin sensitivity. The result of correlation coefficients of the average of basal rate versus average insulin sensitivity clearly shows us that there is neither negative nor positive correlation between the basal rate and static insulin sensitivity. Afterwards, we used a one dimensional convolutional neural network to predict future blood glucose levels of the diabetes patient. We made multivariate one time-step, and multivariate multiple time-step predictions. We made another experiment to find out the effect of static insulin sensitivity model, when it is used together with our neural network model. We trained two neural networks with and without static insulin sensitivity values along with patient's data. Our experiments results showed us there is no significant performance difference between two models.

7.2 Future Work

We discuss the possible ideas for further research, which we did not have time to attempt during our experiments due to time or scope reasons.

Firstly, quality of the dataset could be enhanced. Other methods for dealing with missing data could be tried. We have interpolated missing value within each column/feature. For example within the parts where too many fields are missing, complete row can be removed. However this might result in a much smaller dataset and consequently might lead to a poor generalization of the model. Another technique for dealing with missing values is replacing the missing fields with mean value. Other methods such as replacing with zero or a dummy variable would not be suitable in our case. Close inspection of dataset characteristics would be helpful for the decision process, which is also time consuming task.

We mention another possible improvement with preprocessing of the dataset. We used the input data without applying any normalization. When there is huge difference between the independent features in the training dataset, the feature with huge range dominates the other features. In order to avoid this, and allow each feature to contribute to the model fairly, input features are scaled. Applying normalization sometimes reduces the training time and also improves the convergence of the optimization algorithm. Some normalization techniques are Z-score normalization (standardization) and Min-Max scaling. However, there is no certain rule that normalization definitely contributes to the performance of the model. Therefore, it should be decided according to the nature of the dataset.

We did not change the number of filters of the CNN, and the depth of the network, and the filter length during the experiments in order to truly observe the effect of static insulin sensitivity feature. In order to achieve better predictions, these parameters can be tuned further. We only used the features; bolus sqare, insulin sensitivity, glucose cgm, meal bolus calculator, and basal in our experiments. Even though we mentioned binary alert features such as cgm and pump related in the section 5.1, we did not use them in our experiments. These values can be included to the model as a future work.

List of Figures

List of Tables

Bibliography

[1] Ajith Abraham. Artificial neural networks. *Handbook of measuring system design*, 2005.

[2] Saleh Albelwi and Ausif Mahmood. A framework for designing the architectures of deep convolutional neural networks. *Entropy*, 19(6):242, 2017.

[3] Golnaz Baghdadi and Ali Motie Nasrabadi. Controlling blood glucose levels in diabetics by neural network predictor. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 3216–3219. IEEE, 2007.

[4] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[5] Richard N Bergman, Y Ziya Ider, Charles R Bowden, and Claudio Cobelli. Quantitative estimation of insulin sensitivity. *American Journal of Physiology-Endocrinology And Metabolism*, 236(6):E667, 1979.

[6] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[8] Kevin Curran, Eric Nichols, Ermai Xie, and Roy Harper. An intensive insulinotherapy mobile phone application built on artificial intelligence techniques. *Journal of diabetes science and technology*, 4(1):209–220, 2010.

[9] Chiara Dalla Man, Robert A Rizza, and Claudio Cobelli. Meal simulation model of the glucose-insulin system. *IEEE Transactions on biomedical engineering*, 54(10):1740–1749, 2007.

[10] Judith E. Dayhoff and DeLeo' James M. Artificial neural networks. *cancer*, 2001.

[11] Ralph A DeFronzo, Jordan D Tobin, and Reubin Andres. Glucose clamp technique: a method for quantifying insulin secretion and resistance. *American Journal of Physiology-Endocrinology And Metabolism*, 237(3):E214, 1979.

[12] A Karim El-Jabali. Neural network modeling and control of type 1 diabetes mellitus. *Bioprocess and biosystems engineering*, 27(2):75–79, 2005.

[13] Colin Fyfe. Artificial neural networks and information theory. *University of Paisley*, 2000.

[14] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[15] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[16] G Gogou, N Maglaveras, BV Ambrosiadou, D Goulis, and C Pappas. A neural network approach in diabetes management by insulin administration. *Journal of Medical Systems*, 25(2):119–131, 2001.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[18] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Multimodal semi-supervised learning for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 902–909. IEEE, 2010.

[19] Manish Gutch, Sukriti Kumar, Syed Mohd Razi, Kumar Keshav Gupta, and Abhinav Gupta. Assessment of insulin sensitivity/resistance. *Indian journal of endocrinology and metabolism*, 19(1):160, 2015.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] Roman Hovorka. The future of continuous glucose monitoring: closed loop. *Current Diabetes Reviews*, 4(3):269–279, 2008.

[22] Roman Hovorka, Valentina Canonico, Ludovic J Chassin, Ulrich Haueter, Massimo Massi-Benedetti, Marco Orsini Federici, Thomas R Pieber, Helga C Schaller, Lukas Schaupp, Thomas Vering, et al. Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. *Physiological measurement*, 25(4):905, 2004.

[23] A Leturque, AF Burnol, P Ferre, and J Girard. Pregnancy-induced insulin resistance in the rat: assessment by glucose clamp technique. *American Journal of Physiology-Endocrinology And Metabolism*, 246(1):E25–E31, 1984.

[24] Yinghui Lu, Andrei V Gribok, W Kenneth Ward, and Jaques Reifman. The importance of different frequency bands in predicting subcutaneous glucose concentration in type 1 diabetic patients. *IEEE Transactions on Biomedical Engineering*, 57(8):1839–1846, 2010.

[25] DR Matthews, JP Hosker, AS Rudenski, BA Naylor, DF Treacher, and RC Turner. Homeostasis model assessment: insulin resistance and β-cell function from fasting plasma glucose and insulin concentrations in man. *Diabetologia*, 28(7):412–419, 1985.

[26] Tom M Mitchell. Does machine learning really work? *AI magazine*, 18(3):11, 1997.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[29] SG Mougiakakou, K Prountzou, and KS Nikita. A real time simulation model of glucose-insulin metabolism for type 1 diabetes patients. In *Engineering in Medicine and Biology*

Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the, pages 298–301. IEEE, 2006.

[30] Stavroula G Mougiakakou and Konstantina S Nikita. A neural network approach for insulin regime and dose adjustment in type 1 diabetes. *Diabetes technology & therapeutics*, 2(3):381–389, 2000.

[31] Stavroula G Mougiakakou, Aikaterini Prountzou, Dimitra Iliopoulou, Konstantina S Nikita, Andriani Vazeou, and Christos S Bartsocas. Neural network based glucose-insulin metabolism models for children with type 1 diabetes. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 3545–3548. IEEE, 2006.

[32] Timo Nolle, Alexander Seeliger, and Max Mühlhäuser. Unsupervised anomaly detection in noisy business process event logs using denoising autoencoders. In *International Conference on Discovery Science*, pages 442–456. Springer, 2016.

[33] Erik Otto, Christopher Semotok, Jan Andrysek, and Otman Basir. An intelligent diabetes software prototype: predicting blood glucose levels and recommending regimen changes. *Diabetes technology & therapeutics*, 2(4):569–576, 2000.

[34] Silvia Oviedo, Josep Vehí, Remei Calm, and Joaquim Armengol. A review of personalized blood glucose prediction strategies for t1dm patients. *International journal for numerical methods in biomedical engineering*, 2016.

[35] Scott M Pappada, Brent D Cameron, and Paul M Rosman. Development of a neural network for prediction of glucose concentration in type 1 diabetes patients. *Journal of diabetes science and technology*, 2(5):792–801, 2008.

[36] Carmen Pérez-Gandía, A Facchinetti, G Sparacino, C Cobelli, EJ Gómez, M Rigla, Alberto de Leiva, and ME Hernando. Artificial neural network algorithm for online glucose prediction from continuous glucose monitoring. *Diabetes technology & therapeutics*, 12(1):81–88, 2010.

[37] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE, 1993.

[38] Gavin Robertson, Eldon D Lehmann, William Sandham, and David Hamilton. Blood glucose prediction using artificial neural networks trained with the aida diabetes simulator: a proof-of-concept pilot study. *Journal of Electrical and Computer Engineering*, 2011:2, 2011.

[39] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[40] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[41] William Sandham, Dimitra Nikoletou, David Hamilton, Ken Paterson, Alan Japp, and Catriona MacGregor. Blood glucose prediction for diabetes therapy using a recurrent artificial

neural network. In *Signal Processing Conference (EUSIPCO 1998), 9th European*, pages 1–4. IEEE, 1998.

[42] Julio V Santiago, Anton H Clemens, William L Clarke, and David M Kipnis. Closed-loop and open-loop devices for blood glucose control in normal and diabetic subjects. *Diabetes*, 28(1):71–84, 1978.

[43] Dharak Shah, Saheb Motiani, and Vishrut Patel. Movie classification using k-means and hierarchical clustering.

[44] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[45] Chan Wai Ting and Chai Quek. A novel blood glucose regulation using tsk-fcmac: A fuzzy cmac based on the zero-ordered tsk fuzzy inference scheme. *IEEE Transactions on Neural Networks*, 20(5):856–871, 2009.

[46] Zlatko Trajanoski and Paul Wach. Neural predictive controller for insulin delivery using the subcutaneous route. *IEEE Transactions on Biomedical Engineering*, 45(9):1122–1134, 1998.

[47] Volker Tresp, Thomas Briegel, and John Moody. Neural-network models for the blood glucose metabolism of a diabetic. *IEEE Transactions on Neural networks*, 10(5):1204–1213, 1999.

[48] Patricia Vuguin, Paul Saenger, and Joan Dimartino-Nardi. Fasting glucose insulin ratio: a useful measure of insulin resistance in girls with premature adrenarche. *The Journal of Clinical Endocrinology & Metabolism*, 86(10):4618–4621, 2001.

[49] Aosen Wang, Hua Zhou, Wenyao Xu, and Xin Chen. Deep neural network capacity. *arXiv preprint arXiv:1708.05029*, 2017.

[50] Wei Wang, Zheng-Zhong Bian, Lan-Feng Yan, and Jing Su. A novel individual blood glucose control model based on mixture of experts neural networks. *Advances in Neural Networks-ISNN 2004*, pages 203–219, 2004.

[51] Sarah H Wild, Gojka Roglic, Anders Green, Richard Sicree, and Hilary King. Global prevalence of diabetes: estimates for the year 2000 and projections for 2030: response to rathman and giani. *Diabetes care*, 27(10):2569–2569, 2004.

[52] Konstantia Zarkogianni, Stavroula G Mougiakakou, Aikaterini Prountzou, Andriani Vazeou, Christos S Bartsocas, and Konstantina S Nikita. An insulin infusion advisory system for type 1 diabetes patients based on non-linear model predictive control methods. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 5971–5974. IEEE, 2007.

[53] Chiara Zecchin. Online glucose prediction in type-1 diabetes by neural network models. 2014.

[54] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4, 2006.